ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

MATHEUS LYRA DA SILVA

A MODULAR ARCHITECTURE FOR THE EMULATION OF SCALABLE EDGE APPLICATIONS

Porto Alegre
2024

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
POLYTECHNIC INSTITUTE
COMPUTER SCIENCE GRADUATE PROGRAM

A MODULAR ARCHITECTURE FOR THE EMULATION OF SCALABLE EDGE APPLICATIONS

MATHEUS LYRA DA SILVA

Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science.

Advisor: Prof. César A. F. De Rose

Porto Alegre
2024

Ficha Catalográfica

D111m Da Silva, Matheus Lyra

A modular architecture for the emulation of scalable edge applications / Matheus Lyra Da Silva. – 2024.
96.
Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Cesar Augusto Fonticielha De Rose.

1. Edge Computing. 2. Internet of Things. 3. Industry 4.0. 4. Emulation Architecture. I. De Rose, Cesar Augusto Fonticielha. II. Título.

MATHEUS LYRA DA SILVA

A MODULAR ARCHITECTURE FOR THE EMULATION OF SCALABLE EDGE APPLICATIONS

This Master Doctoral Thesis has been submitted in partial fulfillment of the requirements for the degree of Ph. D. In Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 26, 2024.

COMMITTEE MEMBERS:

Prof. Dr. Fabiano Passuelo Hessel (PPGCC/PUCRS)

Prof. Dr. Leonel Pablo Carvalho Tedesco (PPGCC/UNISC)

Prof. Dr. Mario Antonio Ribeiro Dantas (PPGCC/UFJF)

Prof. Dr. César Augusto Fonticielha De Rose (PPGCC/PUCRS - Advisor)

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001. This work has been partially supported by the project “GREEN-CLOUD: Computação em Cloud com Computação Sustentável” (#16/2551-0000 488-9), from FAPERGS and CNPq Brazil, program PRONEX 12/2014. Also, this work was achieved in cooperation with HP Brasil Indústria e Comércio de Equipamentos Eletrônicos LTDA. using incentives of Brazilian Informatics Law (Law nº 8.2.48 of 1991).

This study was also supported by the Ministry of Science, Technology, and Innovations, with resources from Law No. 8.248, dated October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex, in cooperation with HP Brasil Indústria e Comércio de Equipamentos Eletrônicos LTDA. using incentives of Brazilian Informatics Law (Law nº 8.2.48 of 1991).

UMA ARQUITETURA MODULAR PARA A EMULAÇÃO DE APLICAÇÕES DE BORDA ESCALÁVEIS

RESUMO

Na última década, a área de cloud computing transformou a indústria de tecnologia da informação, permitindo a execução de serviços com a disponibilidade de recursos computacionais sob demanda. Seu novo paradigma trouxe modelos de custo mais flexíveis e entre outras coisas, viabilizou o processamento de grandes volumes de dados. Nos últimos anos, outras tecnologias como a internet das coisas e a indústria 4.0 surgiram e fomentaram um aumento ainda maior na produção de dados, o que fez com que a indústria e a academia buscassem por novas abordagens para o processamento e armazenamento de dados. O foco desses estudos tem sido a redução da latência inerente ao modelo centralizado e a área de computação de borda vem sendo vista como uma possível solução. A modelagem de soluções eficientes dentro desse contexto deve levar em conta quais componentes da aplicação devem ser alocados na borda para melhorar efetivamente o desempenho e reduzir o custo de execução. Para tanto, surge a necessidade de ferramentas que viabilizem o projeto e experimentação, reproduzindo cenários específicos de computação de borda. Nesta tese, contribuímos para o estado da arte da area de computação de borda e exploramos as possibilidades de emular cenários considerando aspectos de escalabilidade e diferentes infraestruturas, permitindo experimentação e design com resultados realistas. Propomos uma arquitetura de emulação modular para prototipagem de aplicações de borda escaláveis. A arquitetura proposta é adaptável a diversas condições. Permitimos a medição de cenários específicos considerando a pegada computacional dos dispositivos de borda e exploramos as possibilidades de avaliar diferentes comportamentos de aplicações de borda. Além disso, validamos um protótipo com um caso de uso de agricultura inteligente, no qual os resultados da experimentação de escalabilidade apontaram 200 como o número máximo observado de dispositivos de borda que uma única instância de *Edge Broker* pode suportar nas restrições aplicadas e configurações de testbed.

Palavras-Chave: Computação de Borda, Internet das Coisas, Indústria 4.0, Arquitetura de Emulação.

A MODULAR ARCHITECTURE FOR THE EMULATION OF SCALABLE EDGE APPLICATIONS

ABSTRACT

In the last decade, cloud computing transformed the information technology industry, allowing companies to execute many services that require on-demand availability of computational resources with more flexible provisioning and cost models, including the processing of already growing data volumes. But in the past few years, other technologies such as internet of things and the digitized industry known as industry 4.0 have emerged, increasing data generation even more. The large amounts of data produced by user devices and manufacturing machinery have made both industry and academia search for new approaches to process all this data. Alternatives to the cloud centralized processing model and its inherent high latencies have been studied and edge computing is being proposed as a solution to these problems. In this context, to design efficient solutions, it is necessary to consider which application components must be allocated at the edge to effectively improve performance and reduce the cost of execution, which raises the urge for tools that enable the design and experimentation over specific edge computing scenarios. In this thesis, we contribute to the state-of-the-art of edge computing and explore the possibilities of emulating edge scenarios considering scalability aspects and different infrastructures, allowing experimentation and design with realistic results. We propose a modular emulation architecture for prototyping scalable edge applications. The proposed architecture is adaptable to many different conditions. We enable measurements of specific scenarios considering the computational footprint of edge devices and explore the possibilities of evaluating different behaviors of edge applications. Furthermore, we validate a prototype with a smart agriculture use-case, in which the results of the scalability experimentation pointed 200 as the observed maximum number of Edge Devices that a single Edge Broker instance can support on the applied restrictions and testbed configurations running this implementation.

Keywords: Edge Computing, Internet of Things, Industry 4.0, Emulation Architecture.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CONTENTS

1. INTRODUCTION

In recent years, millions of mobile phones, vehicles, sensors, and other devices are continuously producing large amounts of data [1, 2]. Internet of Things (IoT) has become a popular research subject and is very present in our daily lives [3]. The term IoT addresses a wide range of devices and systems that exchange information with each other directly and eventually communicate with data centers [4]. Besides the explosive growth in demand for high bandwidth services due to the millions of user devices, manufacturing industries have evolved in their methods of making products, collecting information, and driving further intelligent actions during the manufacturing process [5]. Industry 4.0, as it is called, connects the IoT with manufacturing techniques to enable systems to share information, analyze it, and use it to guide intelligent actions, which tends to be even more intense due to the expected advances in the 5G technology for the next few years, since 5G promises to hand very high capacity of IoT devices [5]. These areas introduced a massive jump in the scale of data production [4], and an important subset of these devices involves real-time control, particularly for industrial applications [2].

Cloud computing has an important role regarding IoT applications, due to its capacity to process large amounts of data, management, scalability, and configurability [6]. Data processing, in the IoT context, is usually based on the conventional cloud computing architecture, where all data is uploaded to centralized servers, that after computation, send the results back to the original devices [4]. To support IoT applications, significant capacity for computation and storage is required. However, due to the great amount of data and message exchanges, large-scale applications may suffer while attempting to communicate with centralized servers [4]. Therefore, edge computing proposes that the infrastructure must contemplate servers located closer to the end user than cloud servers usually are. The geographic proximity from user devices provides faster response, which is an important feature since IoT applications tend to be delay-sensitive.

Despite edge computing servers having less computation power than cloud servers, they still tend to provide better Quality of Service (QoS) [7] and show the way to potentially leverage the areas of cloud computing and IoT. Having its concepts focus on important metrics, such as delay, high bandwidth, latency cost, and power consumption this area is becoming an attractive research topic. Along with that, many use case scenarios emerge, such as smart agriculture, healthcare, mobile applications, big data, video analytics, connected vehicles, etc [3]. However, scalability is still one of the biggest challenges inherited from the IoT area. It is crucial to know the limits and how much stress an application can take. For IoT applications, to get those numbers, companies have to spend large amounts of money to design Proof of Concepts (PoCs) to test their solutions directly in a real environment. Several expensive materials, such as Printed Circuit Boards (PCBs), gateways, and batteries, must be installed just to validate the behavior of simple routines in large-scale conditions. The problem with this practice can be easily spotted when a designed PoC does not work as expected. All efforts invested might be wasted because of an issue that could be easily tracked if there were better developing and experimentation environments.

The complexity of edge infrastructures in terms of their distributed nature, heterogeneous environments, resource constraints, latency, and bandwidth requirements, as well as integration with cloud services, makes experimentation, testing, validation, tuning, and evaluation difficult tasks. We might take smart agriculture as an example. In this field, the applications are usually subject to many weather conditions and might have inherent delays in message exchange. Those conditions are usually hard to reproduce in the development phase and even in a PoC with many physical devices, require the team to wait for a specific climatic condition to reproduce such a testing scenario. This situation leads to the necessity of simplifying these processes and providing cost-effective ways for application prototyping and testing. Smart agriculture's relationship to scalability is intricately tied to the widespread deployment of thousands of devices in agricultural fields. The adoption of IoT technologies in agriculture has led to the integration of a multitude of sensors, actuators, and smart devices across farming landscapes. Smart agriculture relies on an extensive network of sensors to collect data on various environmental parameters, such as soil moisture, temperature, humidity, and crop health. With thousands of these sensors distributed throughout fields, scalability becomes crucial to manage and process the vast amount of data generated. This involves tailoring agricultural practices based on localized data from individual devices, optimizing resource use, and minimizing environmental impact. Several factors contribute to the importance of scalability in the context of smart agriculture. A few points would be sensor proliferation, real-time decision-making, and precision farming practices [8, 9, 10].

Virtualization has already been proven as a valuable asset in the past when applied to cloud simulation with CloudSim [11] and the work that came after that. However, simulation, while a valuable tool in various fields, exhibits disadvantages when compared to emulation. Simulations often suffer from limited realism due to approximations and models, leading to inaccuracies in predicting outcomes, especially in complex systems. In contrast, emulation provides a more realistic representation by directly using real hardware and software components, offering greater accuracy, improved handling of complexity, and better support for low-level interactions. Considering the nuances of the edge environments, we see emulation as a bright path for these multiple scenarios. In this work, we discuss the use of edge emulators as an alternative for the scalability evaluation of IoT solutions. We present an emulation architecture that, by relying on the presence of real-world components, tends to provide more realistic results compared to simulation tools and can be an effective alternative to developing optimized edge applications without having to build the entire architecture in the real world.

The main contributions of this work are as follows:

- A systematic mapping where we analyzed 219 papers to provide an overview of the edge computing state-of-the-art related to the IoT and industry 4.0. This systematic mapping revealed a notable research gap concerning emulation architectures, leading to the formulation of the proposal outlined in this thesis;
- We analyzed the main differences between the use of simulation and emulation with a particular focus on edge applications, weighing the pros and cons, which led us to invest in a modular emulation architecture for edge computing applications.
- An architecture for edge emulation aiming to create realistic edge applications within clustered environments providing modularity specifically tailored to evaluate the scalability of edge architectures in IoT scenarios that might accurately replicate complex real-world conditions and promote a comprehensive understanding of system behaviors;
- A prototype implementation of the proposed edge emulation architecture highlighting its modular and easily expandable aspects, enabling the design, experimentation, scalability stress over, and testing of diverse edge environments, as well as the usage of pre-programmed popular IoT protocols and standards;
- The validation of our tool with a use case in the area of smart agriculture where the emulation tool has been applied to evaluate the scalability of the system, generating accurate metrics for resource utilization, and traffic control optimization.

1.1 Document Organization

The remainder of this thesis is organized as follows. Chapter 2 summarizes the concepts used in this study. Chapter 3 presents a systematic mapping, analyzing edge computing research efforts and boundaries to the mentioned related areas and possible future directions. Chapter 5 advocates for a paradigm shift in the approach of edge application development to the usage of emulation. Furthermore, the chapter presents our thesis proposal. Chapter 6 is dedicated to the experimentation and results analysis. Chapter 4 presents related work in the literature. Finally, Chapter 7 offers conclusions, contributions, and directions for future research.

2. BACKGROUND

In this section, we present the most important concepts to understand the remainder of this study. Considering that this analysis addresses multiple cloud problems and current solutions for data processing are usually entirely cloud-based, firstly, we describe the cloud computing area. Furthermore, we briefly present the definitions of IoT, industry 4.0, edge computing, and smart agriculture.

2.1 Cloud Computing

Since its beginning, cloud computing has been seen as a promising technology, transforming a large part of the IT industry. According to the NIST definition, “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, application, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [12]. The term cloud computing refers not only to applications but also to the infrastructure, where both can be offered as a service [13]. More precisely, regarding the service characteristics, the literature uses the following service models to categorize cloud services: Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Data storage as a Service (DaaS) [14]. Furthermore, according to distribution and physical location, cloud computing environments are classified into four distinct models, namely: private cloud, public cloud, hybrid cloud, and community cloud [15].

The advantages provided by cloud computing include the capacity to process large amounts of data, management, easy access to high-performance computing, storage infrastructure, massive scalability, and specifiable configurability, while hiding the complexity of IT infrastructure management from its users [16]. Due to the mentioned features, cloud computing became financially attractive, since users can centrally manage a pool of storage and computing resources, rather than owning and managing on-premise infrastructures, as cited in [17].

Cloud computing has presented itself as a solution to alleviate the tension between IT service supplies and business demands, which is a real challenge nowadays. Cloud infrastructure is a technological option that makes it feasible for many companies to maintain services without having to own an expensive infrastructure [18]. Cloud computing constantly deals with multiple processing and access challenges [6]. In recent years, due to the increase in data production, one of the problems that have always been present in the cloud computing context, the access peaks, has become a more prominent challenge. Several strategies attempt to solve this problem. A few studies present approaches and tools that intend to predict anomalous situations. Detecting and analyzing potential anomalous performances in cloud computing systems is essential for avoiding losses to customers and ensuring the efficient operation of the systems. To this end, a variety of automated techniques have been developed to identify anomalies in cloud computing [19], [20], [21]. Despite the attempts to predict anomalies, due to the massive increase in data production in the past few years, it seems that reducing data transmission and reducing data processing in the cloud itself, would be the most suitable approach to be pursued. Even with the advances brought by cloud computing, such an amount of data led to the rise of new paradigms such as fog and edge computing, aiming at decreasing network latency by extending cloud services to the edge of the network.

2.2 Internet of Things and Industry 4.0

The term IoT addresses a scenario where ordinary devices can be uniquely identified and connected to the internet [22]. In recent years with the physical size and price reduction, these heterogeneous devices are becoming more and more present in people's daily lives. The high connectivity level that IoT provides to the IT industry, brings a whole new segment to be explored. Companies may now consider real-time data to improve the interaction between users and applications. However, significant challenges remain to be solved from a technological and business perspective.

At a strategic level, existing business models may have to be adapted or re-defined, and at an operational level, challenges related to hardware and software requirements, and even to the offered services, start taking place [23]. The highly connected environments, as well as the great amount of data produced by IoT devices, are also creating new discussions inside the academia. The IoT research goals are pointed to lightweight protocols, energy efficiency, cognition, security, identification, data, connectivity, and miniaturized devices [24].

The IoT is present in many different areas in multiple ways. Inside the manufacturing industry, the IoT is known as the Industrial Internet of Things (IIoT), which is also recognized as the fourth industrial revolution or industry 4.0. As cited in [25], the first industrial revolution was represented by mechanical production plants based on water and steam power at the end of the 18th century. The second started with the symbol of mass labor production based on electrical energy at the beginning of the 20th century. The third started with automatic production based on electronics and internet technology in the 1970s. And now, the fourth industrial revolution, with digital manufacturing, network communication, computer, and automation technologies [26].

The industry 4.0 can be seen as the integration of complex physical machinery and devices with networked sensors and software, or as a new level of value chain organization and management across the life cycle of products [25]. The understanding of what exactly industry 4.0 represents for the academia and manufacturing industry is still not fully known. However, there are many expectations around this area and what it will become in a few years from now. Clearly, among other things, this new industrial era will be responsible for a digital transformation of the traditional industry. Furthermore, the areas of IoT and industry 4.0 have introduced a new scale of data production. Uploading all produced data to centralized servers started to cause performance degradation in delay-sensitive applications, and data exchange started to represent higher costs. These motives are making the established cloud paradigm reach its limits and pushing to the exploration of new solutions based on upcoming computational paradigms.

2.3 Edge Computing

Applications based on IoT devices and smart industrial devices, or simply edge devices, are usually associated with high connectivity and massive data production. In order to handle location awareness and the low latency expected to support these delay-sensitive applications, new strategies besides usual cloud services, became a necessity. Once the IoT and industry 4.0 are increasing data production, companies are trying new strategies to handle the large amount of data expected for the next few years. Cloud companies are starting to readapt to keep quality of service, having low-latency response as one of the most important goals. The state-of-the-art presents a series of experiments and academic research that places edge computing in an essential position to make data processing cheaper and also utilize the current cloud infrastructure.

Edge computing shifts computational data, applications, and services from cloud servers to the network's edge. Content providers and application developers can leverage edge computing systems to provide users with services nearby. The distinctive features of edge computing include high bandwidth, ultra-low latency, and real-time access to network information, benefiting multiple applications [27]. Edge computing brings services responsible for processing and storage to the network edge, intending to be geographically closer to the edge devices, which allows edge nodes to reduce bandwidth consumption and network latency. Those services, usually offered by cloud servers, in the edge computing paradigm, are placed in edge nodes, also referred to as cloudlets, micro datacenters, or fog nodes [28, 4].

The future of edge computing is characterized by the integration of 5G networks, fostering rapid data transfer and minimal latency. As edge computing becomes central to the IoT, its adoption will grow across industries and areas such as smart agriculture, enhancing efficiency, and localized data processing. Moreover, the convergence of artificial intelligence with edge computing is reshaping digital landscapes, enabling real-time decision-making at the edge devices themselves. Industries are poised to leverage edge computing for sector-specific applications, from healthcare's remote patient monitoring to manufacturing's predictive maintenance. Emphasizing decentralized infrastructure, future developments will focus on robust security measures, standardization efforts, and seamless integration with cloud computing to create a versatile computing environment addressing the evolving needs of technology. Due to the challenge of experimenting and designing edge applications in real-world scenarios, studies focused on building edge computing simulators and emulators have been published. These frameworks have the potential to simplify this process and provide fast and scalable tools for addressing real-world applications. In Chapter 3 we present a study that aims to map the knowledge about edge computing that led us to the proposal in the area of edge emulation later presented.

2.4 Smart Agriculture

Smart or precision agriculture represents a paradigm shift in traditional farming practices by seamlessly integrating advanced technologies and sophisticated information management tools [29]. At its core, this innovative approach harnesses the power of technologies such as the IoT, sensors, artificial intelligence (AI), drones, and data analytics to revolutionize the agricultural landscape. Sensors are responsible for various tasks such as sensing soil, temperature, weather, light sensing, and moisture. Similarly, devices carry out numerous control functions, including node discovery, device identification, and naming services. All these operations are executed by devices controlled by a microcontroller. The control process is managed by a remote device or computer connected through the Internet [30]. A few trends and challenges of IoT-based smart agriculture are farm management systems, tracking and tracing, crop management, water management, irrigation management, soil management, weather management, precision farming, nutrient management, waste management, and livestock monitoring [31].

In a smart agriculture system, precision is key, as farmers can employ data-driven strategies to optimize the allocation of resources. This includes targeted and efficient use of water, fertilizers, and pesticides, leading to minimizing waste and environmental impact. By leveraging digital technologies, smart agriculture not only enhances operational efficiency but also contributes to sustainable farming practices. The result is a more resilient and productive agricultural sector that meets the challenges of an evolving global landscape while promoting environmental stewardship. The fusion of technology and agriculture in smart farming signifies a transformative era in food production, aligning with the growing need for resource efficiency, environmental sustainability, and increased food security. Due to its intricate requirements, such as reduced latency, bandwidth efficiency, reliability, data security concerns, scalability, and customization capabilities, smart agriculture is a compelling use case for testing edge computing applications. Therefore, in the proposal presented in this thesis, we employ this domain as our chosen validation use case.

3. STATE OF THE ART IN EDGE COMPUTING AND ITS INTERPLAY WITH IOT AND INDUSTRY 4.0

This systematic mapping was developed as the first contribution of our study. Accepted in the Journal of Grid and Utility Computing 2021 edition by the title Edge Computing and its Boundaries to IoT and industry 4.0: A Systematic Mapping Study [1], this study aims to map edge computing state-of-the-art. This study provides an overview of the edge computing research area and identifies the quantity, research types, and results available in the literature. As secondary goals, this study identifies possible research gaps and viable approaches for future efforts. Our initial objectives with this study can be defined as follows:

- To identify edge computing state-of-the-art research;
- To identify the boundaries between the areas of edge computing and IoT;
- To identify the relation between the areas of edge computing and industry 4.0;
- To identify challenges, literature gaps, and possible future directions.

3.1 Systematic Mapping Methodology

This section presents the methodology used to conduct the systematic mapping presented in this chapter, and how the objectives above guide the study processes.

3.1.1 Research Questions

The current work was conducted based on the protocol for systematic mapping studies presented in [32]. In the following sections, we present how each stage of our study was planned. The questions below were defined to be answered throughout this systematic mapping:

- What are the boundaries between the areas of edge computing, internet of things, and industry 4.0?
- What is the impact and challenges of edge computing on the internet of things and industry 4.0?

[1] https://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijguc

The research questions developed for this study are based in our initial motivations and intents to expand the general understanding of the scientific efforts in the edge computing area. Therefore, we expose where this new technology has been used and the adversities that it can solve.

3.1.2 Research Strategies

For the research strategy, we defined a string to be used over the academic databases. This process was carried out as follows. First, for each research question, synonyms related to the objective of the question were searched, and new logical clauses were specified, one for each research question using these synonyms. The strings were defined based on three major search terms: "Edge computing", "Internet of Things", and "Industry 4.0". Our initials search strings based on the mentioned terms are presented below:

- “Edge computing" AND (“Internet of Things" OR IoT).
- “Edge computing" AND “Industry 4.0".

Finally, in the creation of the search string, the clauses originating from the search questions were combined by logical OR operators, so the results would include at least one of the synonyms. This new clause was then combined with the initial clause using the AND operator, searching for the intersection of the major terms in the string. This ensures that at least one of the main terms and phrases of the research clauses must be present in the results. The final string is the following:

“Edge computing" AND (“Internet of Things" OR IoT) AND “Industry 4.0".

In order to collect a significant sample of scientific studies we searched in two relevant databases for the research area, namely IEEE and ScienceDirect. The search was then carried out by applying the criteria of the search string in the title, abstract, or keywords fields. This result was stored to be used in the classification process specified in Section 3.1.3. In future work, we intend to expand the search phase also to other databases.

3.1.3 Selection Strategy

We defined the criteria for inclusion and exclusion for the results of our study. Such action was taken, aiming to keep only the relevant articles. To address the state-of-the-art studies, the following inclusion criteria were applied to titles and abstracts:

- Result is a scientific article;

- Terms fulfill the search string;
- Results published in the time frame 2016 to 2020.

The excluded results were not analyzed in the research. The following exclusion criteria were applied to titles and abstracts:

- Result is not a scientific article;
- Result is not in English;
- The Search Strings are not in the abstract or the article keywords;
- Researchers do not have access to the result;
- The study is not a primary study;
- The article address another research area;
- Result presents summaries of conferences/editorials or guidelines/templates for conducting mapping studies.

3.1.4 Literature Sources and Studies Selection

In this section, we analyze the primary studies obtained with the classification criteria and research questions outlined previously. In our first search, based on the string earlier mentioned, we found 81 studies in the IEEE database and 138 studies in the ScienceDirect database. Considering the total results, the first phase obtained 219 studies. In order to classify the search results, we applied the inclusion and exclusion criteria. We conducted the classification based on titles and abstracts, as well as full-text reading for ambiguous situations.

The classified studies resulted in 54 from IEEE and 36 from ScienceDirect. The acceptance score for the former database is equal to 67%, and for the latter 26%. Table 3.1

Table 3.1 – Distribution of primary studies by research method.

Database	**Study**
IEEE	[33],[34],[35],[36],[37],[38],[39], [40],[41]
	[42],[43],[44],[45],[46],[47],[48], [49], [50]
	[51],[52],[53],[54],[55],[56],[57], [58], [59]
	[60],[61],[62],[63],[64],[65],[66], [67], [68]
	[69],[70],[71],[72],[73],[74], [75],[76],[77]
	[78],[79],[80],[81],[82],[83],[84],[85],[86]
ScienceDirect	[87],[88],[89],[90],[91],[92], [93],[94],[95]
	[96],[97],[98], [99],[100],[101],[102],[103],[104]
	[105],[106],[107], [108],[109],[110],[111], [112], [113]
	[114],[115],[116], [117],[118],[119],[120]
	[121], [122]

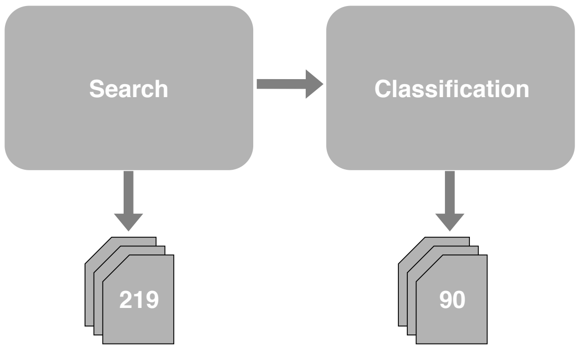

Figure 3.1 – Results obtained from the search process.

presents the results of the classification process in detail. The reason for the high score of excluded studies, in our perception, is that most of the database results were related only to a single term of the research string. Figure 3.1 presents a summary of the number the total numbers regarding the first and second phases.

3.2 Data extraction and Classification

As we present in Section 3.1.4, from the classification process, we obtained 90 studies. In order to better understand the distribution of such studies regarding the research area, we conducted a full-text reading phase where the main goal was then to group the studies by the main contribution area mentioned throughout the study, the results are shown in Table 3.2.

Due to the interdisciplinary nature of the analyzed studies, some of them could fit into more than a single category. However, we intend to generalize the domains and classify each study in a single area. Therefore, to conduct the classification process, we focus on the main contribution. It is noteworthy that in order to associate the different terms used in each study, we decided to unify related subjects into single categories. Therefore, terms such as distributed processing and processing were both considered processing studies. Similarly, specific research areas were considered in more general ways, such as blockchain studies are considered security studies. Studies with more applied approaches, such as farming and education, despite being research areas with very specific challenges, were also classified more generally regarding the research area. Furthermore, the monitoring category addresses monitoring, analytics, and anomaly detection studies. The network category addresses network, delay-sensitive, and caching studies. Moreover, the resource allocation category addresses resource allocation, management, resource provision, and placement studies.

Table 3.4 presents the summary of the classification previously mentioned, grouping the studies by research areas and main contributions. In the analyzed studies, we may

Table 3.2 – Studies classification

General		Processing		Devices, Hardware		Security		Monitoring		Network		Energy consumption		Resource allocation	
Study	Classification	Study	Classification	Study	Classification	Study	Classification	Study	Classification	Study	Classification	Study	Classification	Study	Classification
[60]	Overview	[51]	Application Development	[35]	Evaluation	[52]	Design	[39]	Methodology	[34]	Architecture	[61]	System	[43]	Application Development
[72]	Overview	[64]	Application Development	[41]	Architecture	[59]	Overview	[33]	Application Development	[36]	Architecture	[65]	Application Development	[57]	System
[75]	Architecture	[69]	Architecture	[98]	System	[67]	Model	[42]	Architecture	[37]	Methodology	[94]	System	[63]	Framework
[77]	Architecture	[78]	Architecture			[71]	Overview	[47]	Application Development	[38]	Algorithm	[108]	Analysis	[66]	Methodology
[90]	Architecture	[88]	Implementation			[89]	Overview	[49]	Framework	[40]	Architecture			[70]	Model
[91]	Framework					[110]	Architecture	[50]	Application Development	[46]	Architecture			[73]	Architecture
[102]	Architecture					[113]	Architecture	[53]	Application Development	[48]	Architecture			[80]	Policy
[103]	Framework					[115]	Architecture	[81]	Application Development	[54]	Protocol			[82]	Method
[105]	Architecture					[44]	Design	[83]	Architecture	[55]	Architecture			[85]	System
[106]	Methodology							[84]	Methodology	[56]	Architecture			[92]	Architecture
[111]	Protocol							[87]	System	[58]	Implementation			[96]	Design
[112]	Architecture							[97]	Framework	[62]	Architecture			[101]	Architecture
[114]	Architecture							[99]	Overview	[68]	Architecture			[104]	Algorithm
[118]	Design							[100]	Methodology	[74]	Method			[116]	Algorithm
[121]	Overview							[107]	System	[76]	Framework				
[45]	Overview							[117]	System	[79]	Algorithm				
										[86]	Framework				
										[93]	Architecture				
										[95]	Middleware				
										[109]	Plataform				
										[119]	Evaluation				
										[120]	Architecture				
										[122]	Design				

notice a higher concentration in three specific areas. Delay-sensitive is the category with higher effort concentration, representing 26% of the articles. This category is also composed of network and caching studies. The second category that stands out is the category composed of monitoring, analytics, and anomaly detection works, representing 18% of works. The third category with high concentration is the resource allocation category. Also composed of management, resource provision, placement, and orchestration studies this category represents 16% of the analyzed studies. Regarding the main contribution, most articles present new architectures, which may be related to the novel characteristics of the

Table 3.3 – studies years distribution of selected over the years.

	Database		
Year	**ScienceDirect**	**IEEE**	**Total**
2016	0	2	2
2017	1	4	5
2018	5	17	22
2019	21	26	47
2020	9	5	14
Total	36	54	90

edge computing area. Table 3.3 presents the distribution of the publications over the years since 2016. In general, it is possible to observe how the area has been gaining more visibility within the scientific community.

Table 3.4 – Summary of classification.

	General	Processing	Devices, hardware	Security	Monitoring, analytics, anomaly detection	Delay-sensitive, network, caching	Energy consumption	Resource allocation, management, resource provision, placement, orchestration	Total
Application, system, middleware		3	1		8	2	2	3	**19**
Architecture, design, model, method	8	2	1	6	3	12		6	**38**
Framework, plataform	3				3	3		1	**10**
Protocol, policy						1		1	**2**
Evaluation, analisys			1			1	1		**3**
Methodology	1				1	1		1	**4**
Algorithm						3		2	**5**
Overview	4			3	1				**8**
Simulator							1		**1**
Total	**16**	**5**	**3**	**9**	**16**	**23**	**4**	**14**	90

The scientific community is still attempting to understand the most efficient ways to utilize and provide edge services closer to the end-user devices and manufacturing industries using the infrastructure inherited from cloud computing. We may also notice some general studies that address more than a single specific area and present solutions that cover edge computing in multiple scenarios. Those also represent 18% of the covered studies. Both IoT and industry 4.0 areas are quite concerned about the same correlated areas, such as latency, data processing, security, and other examples earlier mentioned. However, the IoT attention is generally focused on better QoS and the improvement of the user experience. Industry 4.0 studies, in general, focus on the optimization of a certain manufacturing process, management, product improvement, or even to provide complete information about the final product, considering the entire manufacturing life-cycle. Edge computing seems to be a solution for many problems to be faced by IoT and industry 4.0. In most cases, related to data processing, monitoring, and delay-sensitive applications.

3.2.1 Research Questions Analysis

In this section, we perform an analysis of the primary studies obtained through the classification criteria and research questions previously outlined. The answers to the stated search questions, according to the analysis of the primary studies selected are as follows:

What are the boundaries between the areas of edge computing, Internet of Things, and Industry 4.0?

As presented in previous sections, IoT and industry 4.0 are strongly related areas. In fact, industry 4.0 addresses the improvement of industrial machinery using the recent achievements of IoT, which led to some early studies using the term Industrial IoT (IIoT). In both cases, the massive data generation is a subject to closely follow.

This greater amount of data brings new possibilities such as more specialized services, analytics, manufacturing control, etc. On the other hand, delay-sensitive applications are those that suffer the most. Lightweight protocols, better systems architectures, methodologies, specialized frameworks, and new algorithms may be a solution to communicate and process such a heavy workload and handle real-time applications without losing QoS. This concern may be observed in Table 3.4, with the higher number of studies of this nature. These problems are generally presented on the “Things" side, which may be inside the manufacturing industry, smart cities, or even closer to the end-user with wearable gadgets. With the higher number of devices connected to the network and data traffic, plenty of security problems also need to be handled. Over the analyzed studies, we noticed that blockchain was the most recurring security solution applied to this scenario. We consider four main distribution groups in the general study area: IoT devices, edge servers, cloud infrastructure, and the network. Notably, network studies may focus on between the IoT environment and the edge server, mostly targeting delay-sensitive applications. Also, network studies may target the in-between the edge server and the cloud, mostly addressing the reduction of cloud services utilization.

Another problem related to how to handle such a heavy workload is that the centralized services offered by cloud computing may not be so efficient and might be too expensive in some scenarios. This condition is what starts the discussion about where should be located the edge node and how much data should be processed in those nodes in order to improve the application performance with minimal costs. The attempts to solve these problems, in general, are studies focused on the edge nodes and the cloud architecture itself. It seems that the role of the edge node has been already understood. However, how it is going to be accomplished and change the current applications and services is still an open issue. Studies of this nature focus on solutions that in terms of location, will be at the edge node, the cloud infrastructure, or in the in-between. The large number of sensors and devices connect to the network producing data that allow monitoring systems to be more assertive and also influence strategic decisions, such as the usage of historical data and machine learning applications to predict user behaviors and product line necessities.

Edge computing is the area considered to be between the end-user devices and the cloud environment. For that reason, defining the boundaries between these areas and which equipment it addresses may be quite intricate. At the edge of the network, with low processing power, the IoT devices, which may also be called edge devices, will regularly communicate with the edge nodes. The communication protocols used for that matter are an object of study itself. Reasonable options to be considered are the protocols already proposed for related projects in industry and academia. One example is the Next Generation Service Interface (NGSI), an open standard interface defined by the Open Mobile Alliance (OMA) focused on the development of IoT and smart city platforms [123]. The edge nodes are meant to have much more processing power than the IoT devices. However, there is not really a lower and upper limit to how much it is.

From the edge node to the cloud, another concept that may be misunderstood is related to the difference between fog and edge computing. Conducting this work, we have noticed an uncertainty in the terms used to refer to and define the area itself. In the literature, the definitions of the terms “fog computing" and “edge computing" most of the time are hard to distinguish. We notice that some papers have difficulties in defining which of the two fit in. Some, for example, mention both fog computing and edge computing in the keywords, but in the paper, just one is actually used. This uncertainty makes database searches not return some of those studies as a result. According to Qinglin et al. [124], the difference between fog computing and edge computing is that fog computing relies on interconnection capabilities among nodes, whereas edge computing runs in isolated edge nodes. The edge devices provide the input data for edge computing. Meanwhile, the fog computing layer is carried out in the data transmission network, in which the top layer is the cloud, where big data can be stored and analyzed.

What is the impact and challenges of edge computing on the internet of things and industry 4.0?

The main goal of edge computing is to bring processing closer to the end user. Out of the centralized cloud servers, the applications hosted at the edge bring the possibility of faster response considering IoT devices over the cities and the digital revolution inside the manufacturing industry. A big question to be answered in the next years is how the combination of the IoT, industry 4.0, and edge computing will change the traditional cloud services that became very well established in the past decade.

The classification conducted in this work may already present some of the challenges to be faced by the combination of the mentioned areas. In our perception and in accord with the analysis performed the challenges can be named as follows:

- Security and data privacy;
- Common architectures and protocols;
- Frameworks and middleware for remote management and data analytics;

- Network and connections constraints;
- Hardware constraints;
- To integrate with the current systems reusing the current infrastructure.

Nowadays, it is not possible to implement any new technology without considering data security and privacy matters. As discussed in previous sections, the edge computing environment involves constant and massive data exchange. Problems of the security and privacy domain could be considered, in this scenario, IoT or industry 4.0 concerns. However, as edge computing is strongly related to these areas, its systems will certainly have to consider ways to protect private information as well as handle malicious activities. With a large number of connected devices, managing large-scale infrastructures, requires not only a good view of all devices but also easy ways to securely connect new equipment without creating security vulnerabilities.

Edge computing applications consider at least three location levels in their environments. The first is the data source, the second is the cloudlet location, and the third, is the cloud computing infrastructure. To develop a real application, in this scenario, there are different integration challenges. The point is not only, the integration itself, but to make it dynamically and efficiently, providing good time response and user experience. Furthermore, there must be considered the usage of patterns to handle the heterogeneity of different data sources and infrastructure platforms that will host those applications. Another related challenge is to have a complete visual of the connected devices. Centralized monitoring of all devices in the manufacturing industry, for example, is a desirable feature for any large-scale system. Furthermore, the possibility to control and send tasks to them would also be a helpful feature.

The purpose of this study is to determine and characterize the state-of-the-art of the edge computing area, considering industry 4.0 and IoT perspective. On the other hand, by looking at the state-of-the-art, we may have a proper direction for future studies. From the discussion of the results obtained from the classification and analysis, along with the results presented in Table 3.4, it is possible to point out some of the current literature gaps, which may give us a good perspective of opportunities for the next few years. We may consider literature gaps, categories that were not found in our research, but also categories with a short number of publications. In our perception, some notable gaps in the literature addressing the edge computing area are:

- Scalability;
- Specialized Devices.
- Processing;

- Energy consumption;
- Storage;

As previously presented, data processing at the edge is the integration of the IoT devices and the edge nodes. We found a few studies that focus on the analysis and improvement of these processes, specifically focused on collecting data from vehicles in an autonomous way, so the data, after being extracted from the vehicle sensors, can be mined for interactions in order to provide customer feedback and tracking. In a general sense, there are many scenarios and similar opportunities. Edge computing considers computational environments with hundreds of devices connected to an edge node. Furthermore, the number of edge nodes may also be large. For that matter, service orchestration with high scalability and reliability are potential challenges to be faced. The scalability at that point can be seen as a literature gap, as well as further studies, focused on the usage of processing power and energy consumption of the edge nodes. Addressing the scalability and processing studies, the usage of Docker-based environments, and running its containers associated with other open technologies such as Kubernetes for the container orchestration, may be an appropriate approach, making it easier to handle such heavy workloads. However, since its environments handle heavy workloads, studies focused on the energy consumption of these environments are becoming even more necessary.

Computational power and storage are two of the biggest concerns of cloud-based applications. Cloud infrastructures usually offer different on-demand services that attempt to minimize the companies' costs. Edge nodes may be a potential solution for some of those problems since it will enable to computing of some data before uploading to the cloud system. Reducing the data flow between edge nodes and the cloud is a very present intention in the experiments analyzed in this study. However, it is still necessary to define how much and for how long this data will remain stored in the edge nodes and used for additional operations. Furthermore, to Identifying the sweet spot for the location of the cloudlet, as well as which tasks the edge should perform without being a solution attached to a specific problem, is a point to be better understood and explored in the next few years.

3.3 Mapping Conclusion

In this Chapter, we conducted a systematic mapping study focused on the intersection of edge computing with IoT and industry 4.0. Utilizing a research string we analyzed 219 studies classified from two academic databases, namely, IEEE and ScienceDirect. Applying our selection criteria, we performed a classification phase and selected 90 studies to map into a summary categorization based on their field of study and research method. Based on these results, we answered research questions related to the boundaries between the areas of edge computing, IoT, and industry 4.0 and the impact and challenges of edge computing on these other areas. In this systematic mapping, we notice that the edge computing area is strongly related to the IoT, industry 4.0, and cloud computing areas. Where, according to the literature, industry 4.0 may also be called industrial IoT [125, 126].

The summary of our classification process provides an overview of the studies in these fields and evidences a main concentration area of recent studies focused on architectures, new designs, models, and methods applied for delay-sensitive applications, networks, and caching. Another representative concentration area was focused on applications, systems, and middleware applied for monitoring, analytics, and anomaly detection. With these results in hand, we clearly defined the boundaries between these areas and the impact edge computing is having on IoT and industry 4.0. Furthermore, we analyzed some of the challenges that will be faced in the years to come. We strongly believe that the results of this systematic mapping will help to better understand how these areas are related and can leverage from each other, and also the challenges that will be faced in their further development, especially regarding the edge computing area. The areas of IoT, industry 4.0, cloud, and edge computing have common challenges caused by the high data production. We understand that due to the complexity of edge computing environments, there is a research gap in the development of emulation tools that allow proper experimentation in the intersection of these areas.

Our systematic mapping revealed significant challenges in collecting and processing data, as well as integrating IoT devices into edge computing systems. The existing gaps in the state-of-the-art highlighted in this chapter emphasizes the current limitations of the area. The research to tackle the edge gaps requires experimentation over complex scenarios, which implies a high financial cost. Creating a physical setup for testing and validating concepts to accomplish these points involves substantial expenses related to acquiring hardware, infrastructure, and ongoing operational maintenance. This paves the way for exploring other alternatives for software development such as the usage of digital environments, which in contrast to the real environments, allow the generation of intricate scenarios without the necessity of a tangible and resource-intensive infrastructure. Creating a development tool focusing on software development of edge applications would make it possible for the area to go further in fulfilling the stat-of-the-art gaps and pair with the ongoing topics such as analysis and improvement of the integration processes of IoT devices and edge nodes and the development of scalable edge applications. This understanding took center stage, shaping the trajectory of the subsequent developments in this work. The next chapter goes further into the use of digitized environments as an alternative for edge system development.

4. RELATED WORK

Many studies have been previously conducted on building edge computing simulators and also emulators due to the challenge of experiment and design application in real-world scenarios. These frameworks have the potential to simplify this process and provide fast and scalable tools for addressing real-world applications. In this section, we first present existing experimental tools for edge environment emulation. Then, we outline the design objectives of the proposed Edge classifier.

EmuFog [127] is a framework focused on enabling the design and emulation of Fog computing infrastructures based on real-world scenarios. The authors aimed to provide a test environment for fog computing through EmuFog. Using an emulated network, the EmuFog framework focuses on enabling researchers to design specific network topologies based on user cases, considering algorithms for fog topology enhancement and its extensible nature. The proposed emulator is built on top of an extension of the Mininet [128] emulator called MaxiNet [129]. Since, EmuFog focus is to allow developers to design the Fog infrastructure topologies. This process is made in two steps. The first step is to build a network topology considering switches/routers. The authors mentioned the possibility of employing network topology generators or importing topology datasets. The second step is to place and customize the placement of Fog nodes in the generated network topology to be fed into MaxiNet for emulation. Furthermore, the authors mention the intention to augment EmuFog by Embedding mobility models for clients and Fog nodes such as cars or drones.

EmuEdge [130] is a hybrid emulator aiming to reproduce networks close to reality through tuning and replaying network traces. The proposed hybrid design of EmuEdge enables on-demand degrees of realism by supporting both physical and emulated nodes. EmuEdge focuses on on-demand degrees of realism, which is considered by the authors a key step to reproducible edge computing experiments. The authors claim the EmuEdge to be an emulator designed specifically for hybrid edge computing systems from both computation and network perspectives that augments Mininet-alike systems with better isolation and heterogeneity support. Furthermore, the authors claim that the on-demand degrees of realism on EmuEdge is a key step to reproducible edge computing experiments, is also regarded as a hybrid extension of Mininet that fills the gap on the computation plane.

Aming at bridging the functionalities of existing emulators for data centers and mobile networks, namely srsLTE [131] for the mobile network and Mininet [128] to emulate an SDN-based data center network. OpenLEON [132] proposes a platform that enables experimentation and prototyping in a Multi-Access edge computing context, where the computing service must be closer to the end user since locality and low-latency requirements are essential. The authors claim the OpenLeon platform satisfies the requirement of maintaining high fidelity to incorporate network topologies that faithfully reproduce on a smaller scale those of data center networks and the current LTE stack to guarantee correct handling of the mobile traffic. Furthermore, the authors claim that the platform also satisfies the requirement of allowing the user to emulate different data center topologies and to extend the functionalities of a mobile network.

Fogbed [133] uses a desktop approach to provide a framework and toolset integration enabling the deployment of fog nodes under different network configurations. The proposed framework is composed of a fog emulation system based on Mininet and Docker open-source software and allows the testing of fog components with third-party systems through standard interfaces. Fogbed extends the Mininet framework to allow the usage of Docker containers as virtual nodes. The authors claim that these features allow for the emulation of real-world cloud and fog infrastructures. It is also mentioned the possibility of changing resource limitations for a container such as CPU time and memory available at runtime.

In Celestial [89] the authors address the exploration of LEO (Low Earth Orbit) edge computing possibilities and the evaluation of arbitrary software systems within a precise runtime environment with cost-efficient scalability. They introduce Celestial, a virtual testbed for LEO edge based on microVMs, adept at emulating individual satellites, their movements, and ground station servers with realistic network conditions in an application-agnostic manner. The authors emphasize the necessity of a testbed for LEO edge to enable the testing and evaluation of software by systems researchers, application developers, and platform designers on Earth. The paper concludes by highlighting Celestial's potential for future research and development in LEO edge computing, especially in addressing challenges related to state management, resource allocation, and request routing.

Table 4.1 – Comparison between related work and our proposal.

Name	**Research Topic**	**Language**
EmuFog	Network Topology	Kotlin
Emuedge	On-demand degrees of realism on network	Pyhon
OpenLEON	Integration between preexistent cloud and mobile emulators	Shell
Fogbed	Third-party system integration	Pyhon
Celestial	LEO Edge Emulation	Python/Go
Our proposal	Infrastructure Configuration	NodeJS

During our research, we identified a promising path within the field of edge computing through emulation. Our interest became to enable the prototyping of edge applications in a modular way that could favor experimentation considering constraints on computational resources, focusing on IoT equipment cases, and could be easily expanded in terms of its components, operational rules, and dataflow. We understand that these characteristics are explored in a certain way by related work. However, they have different focuses and do not directly address these same topics as in this work. Thus, our focus concentrated on proposing an architecture that can have different implementations. Similarly to the emulators mentioned above, our research focuses on the design, experimentation, and testing of specific environments with accurate metrics. All of the mentioned work considers the edge as an extension of the cloud and contemplates cloud integration. Another common aspect is that the five of them integrate with the Mininet network emulator. Throughout the work, we present our prototype, which can then be considered an emulator. This implementation concerns instrumental aspects such as initiating and terminating communication between components and functions related to data measurement and visualization. Still, the experiments demonstrated with our prototype also point to differences from other works, as they lead to conclusions regarding consumption limits and component behavior that can be reached through the emulator. Table 4.1 presents a comparison of the edge emulators mentioned above.

Our work differs from the mentioned studies due to the proposal of a modular emulation architecture that focuses on prototyping scalable edge applications. We explore the possibility of emulating realistic edge computing scenarios within clustered environments and docker-friendly applications. Due to being an architecture and not a tool, our proposal may have different implementations and adaptations and also presents data visualization features. We validate our implementation by experimenting with a smart agriculture use case, demonstrating that emulating edge computing applications in clustered environments enables the recreation of real-world scenarios conditions. Moreover, we validate that our tool may be used as an aid to the examination of performance dynamics and resource utilization patterns in a digitized edge emulator environment contributing to a comprehensive understanding of real-world application behavior.

5. A MODULAR ARCHITECTURE FOR THE EMULATION OF SCALABLE EDGE APPLICATIONS

Cloud computing focuses on delivering reliable and scalable infrastructures for hosting Internet-based application services. However, quantifying aspects such as the performance of scheduling and allocation policy for different application and service models under varying loads, energy performance (power consumption, heat dissipation), and system size might be a challenging problem. Therefore, aiming to simplify this process, research on cloud computing has proposed simulators such as CloudSim [11], which intends to provide a single application that models specific environments and behaviors, enabling experimentation of emerging cloud computing infrastructures and management services. Despite the advance of cloud computing, due to the massive volume of data overcome in recent years, the paradigms of fog and edge computing have been proposed, where cloud services are extended to the edge of the network, decreasing network latency. Likewise in the cloud, these new paradigms present the same problems regarding experimentation throughout its environments. Therefore, new simulators have been proposed to tackle these problems. Simulators such as iFogSim, an evaluation platform that enables the simulation and measurement of the impact of resource management techniques in latency, network congestion, energy consumption, and costs. However, despite great contribution, some behavior aspects through models still end up based on assumptions and simplifications, which might lead to non-realistic results [130].

In Chapter 3 we discuss the boundaries between the areas of edge computing, IoT, and industry 4.0, considering the impact and challenges of edge computing on these specific areas. The complexity of edge infrastructures leads to the necessity of frameworks that allow experimentation, testing, validation, tuning, and evaluation. Simulators are able to provide a cost-effective way for application prototyping. Emulators, on the other hand, tend to provide more realistic results, since the application is not deployed in a single machine running throughout a single application, but relies on the presence of aspects such as real network delay. Therefore, in this thesis, we intend to simplify these processes by providing an accurate way to emulate edge computing environments. This chapter describes the problem statement and thesis proposal.

5.1 Motivation

The systematic mapping presented in Chapter 3 led our research to a pivotal point in our study. After identifying the literature gaps we notice a lack of tools to assist in system development specifically for edge computing scenarios. Consequently, the next step was to determine whether the path our research would trace to be over simulation or emulation. This chapter advocates for a paradigm shift in the approach of edge application development by endorsing the utilization of development tools to create alternatives to expensive real implementations. Simulation and emulation have evolved into powerful tools that can replicate intricate real-world scenarios with a high degree of accuracy. Such tools have as key advantages the ability to mimic real-world conditions, reduce time-to-market, and provide a versatile testing environment. As the capabilities of these tools continue to advance, they offer a compelling solution to the financial and logistical challenges associated with software development. Below we present some findings on the state-of-the-art edge simulators and emulators. Also, we present the distinctions between the two concepts.

5.1.1 Simulation of Edge Architectures

The paradigms of edge and fog computing extend the cloud services to the edge of the network. Applications of these segments intend to solve delay-sensitive problems where minimizing the latency and maximizing the throughput are essential conditions. To this end, evaluation platforms that enable the performance quantification of management policies were required [134]. Considering simulators as frameworks focused on experimenting and testing specific environments running in a single application, we have found a series of edge simulators that claim to provide cost-effective ways for edge environment prototyping. The list below presents the simulators we found:

- iFogSim [134];
- Sphere [135].
- EdgeCloudSim [136];
- FogNetSim++ [137];
- FogExplorer [138];
- YAFS [139];
- RECAP Simulator [140];
- FogTorchPI [141];
- Sleipnir [142].

Despite the significant contributions of simulators in the mentioned problems, such applications do not consider physical aspects of deployment across the network, and due to the high level of abstraction over simulated environments, the models for their devices, protocols, and traffic generation patterns may raise fidelity concerns, opening paths to the possibility of undesired non-realistic results [143, 130].

5.1.2 Emulation of Edge Architectures

In general, research on edge computing has the intention to tackle the gap between cloud infrastructure and the time response demands brought by areas such as IoT and industry 4.0. Emulation is an alternative that enables experimentation and design of edge and fog infrastructures, providing more realistic results since it considers real-world elements and has fewer abstractions in comparison with the simulators mentioned in the previous section. The list below presents the emulators we found, a more detailed explanation about each one of them may be found in Chapter 4:

- EmuFog [127];
- EmuEdge [130].
- OpenLEON [132];
- Fogbed [133];

The intricacy of edge infrastructures poses challenges when it comes to conducting experiments, tests, validations, fine-tuning, and evaluations. This situation highlights the importance of frameworks that make these processes simpler. While simulators can assist with certain aspects of application prototyping cost-effectively, emulators, reliant on real-world components, tend to yield more authentic results. They emerge as a compelling alternative for the development of optimized edge applications, especially when dealing with substantial data volumes.

5.1.3 Practical Advantages in the Usage of Emulation of Edge Architectures

In the landscape of modern technological development, digital environments stand out as a strategic advantage over traditional systems. Firstly, the cost factor plays a pivotal role. Establishing a physical setup for testing and validating concepts incurs substantial expenses related to acquiring hardware, infrastructure, and operational maintenance. Digital environments, on the other hand, enable the creation of complex scenarios without the need for a tangible, resource-intensive infrastructure. This not only results in significant cost savings but also expedites the testing and iteration processes, allowing for a more agile and responsive development cycle. Moreover, unlike traditional implementations that may be constrained by physical limitations and fixed configurations, digital setups provide the freedom to create a diverse range of conditions and scenarios. This versatility is invaluable for researchers and developers seeking to explore the full spectrum of possibilities within their domain. It allows for rapid prototyping and the ability to test under various conditions, ultimately fostering a more comprehensive understanding of the system's behavior without the constraints of a fixed physical infrastructure, offering a strategic alternative by combining cost-effectiveness with unparalleled flexibility, making them a compelling choice for proof of concepts and experimental developments in diverse fields.

The distinction between simulation and emulation lies in their approaches to replicating real-world scenarios. While both are valuable, emulation incorporates tangible hardware components along with the virtual environment. This integration results in a more accurate representation of the physical world, capturing nuances such as hardware-specific behaviors and dynamic interactions that may be critical in certain domains. Emulation stands out for its ability to bridge the gap between the virtual and the real, offering a level of precision that simulations may struggle to achieve. By mimicking the behavior of actual hardware, emulators provide results that closely align with those observed in practical implementations. This tangible integration enhances the realism of the emulation process, making it particularly advantageous in fields where hardware nuances significantly impact the reliability and performance of systems. In summary, the distinction between simulation and emulation lies in their depth of replication, and the added fidelity of emulation often renders it the superior choice when aiming for a more accurate and realistic representation of real-world scenarios. In Chapter 5, we go further into the subject of edge emulation and present our proposal.

5.1.4 Emulation of Edge Architectures Requirements

Considering the replicability of edge computing scenarios, which are associated with large amounts of data, latency, and costs-related issues, we propose here an architecture for edge emulation focused on analyzing data transfers and scalability aspects, which have been accepted at the 9th IEEE World Forum on Internet of Things (WF-IoT 2023) by the title "A versatile emulator tool to evaluate the scalability of Edge architectures in IoT scenarios". We enable edge experimentation and testing in order to identify the most efficient allocation of system components considering an edge-to-cloud environment, mitigating metric abstractions present within simulated environments. Due to the intense data exchange of such applications, we consider the edge environment within containers in cluster infrastructures with a cloud connection. Furthermore, our proposal mirrors the low processing power of edge devices and their bandwidth limitations, such as industrial machinery, by limiting the container's capacity and network interfaces to resemble such devices. The main points considered are:

- Edge infrastructure emulation;
- Application and environment prototyping;
- Performance measurements (data exchange, processing, memory, disk);
- Scalability experiments;
- Device capacity definition;
- Different message exchange protocols and configurable payloads;
- Different configurations regarding edge/cloud visibility.

Proposal Objectives

To provide a guide to our work, the following objectives were defined:

- Emulate edge computing applications within clustered environments, replicating the specific conditions of real-world scenarios;
- Promote a comprehensive understanding of a real-world application behavior by analyzing performance dynamics, and resource utilization patterns of an edge emulator digitized environment.

5.2 A New Modular and Scalable Edge Emulator Architecture

A recurrent topic for edge computing solutions that helps in understanding the area domain is to reduce the volume of data sent to the cloud [28]. A dataflow from edge-to-cloud that is not pre-processed at the edge and due to that, contains additional data that is not relevant to the application at any point, might represent a waste of money or even add a layer of delay in the entire application, leading at least, to a higher volume of paid data traffic and storage. An example of that would be a monitoring logs application in supermarket PoS (Points of Sale) represented in Figure 5.1. In general, the volume of data in edge applications might be seen as one of the main causes of scalability problems, along with the message exchange itself between edge devices. In this case, regarding edge-to-cloud communication, the worst-case scenario would take all generated logs to be analyzed in the cloud, represented in scenario 1. An optimization would consider the possibility of sending to the cloud only relevant data or even just tags that represent previously mapped errors, as represented in scenario 2. Edge computing applications might consider N-layers of devices that interact and orchestrate work independently of each other and, if necessary, have to consolidate results.

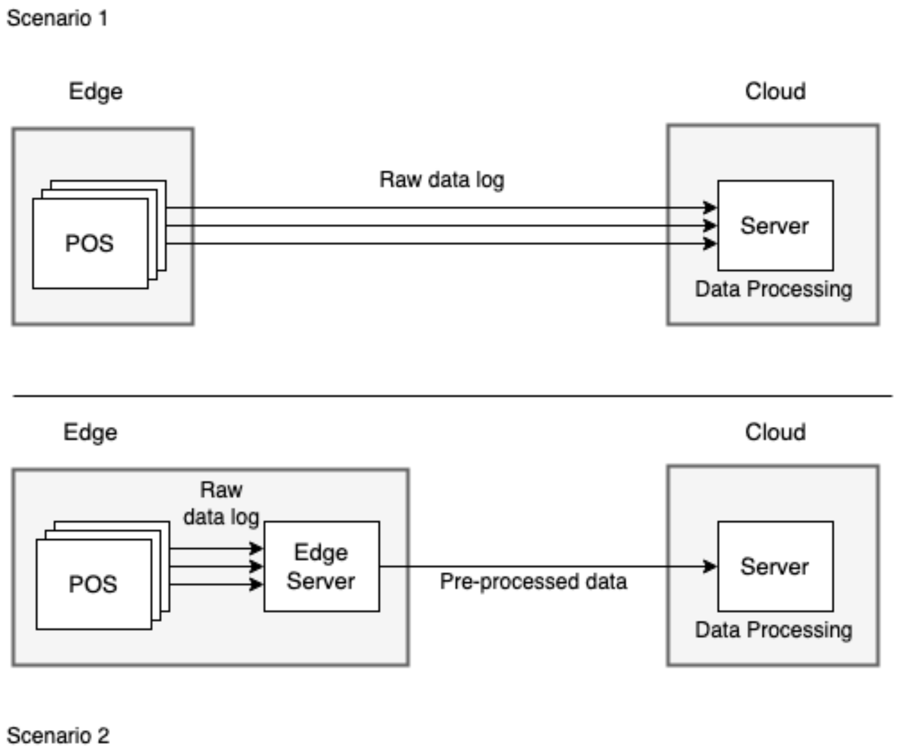

Figure 5.1 – POS data exchange.

Similar to the example of supermarket point-of-sale presented above, many edge computing applications need to deal with issues related to the volume of edge-to-cloud data. However, the problem becomes even more significant when the application doesn't only consolidate data, but has a higher volume of message exchange between the elements at the edge. This dataflow typically originates from IoT devices with limited processing capabilities, often operating under constrained network conditions. In our understanding, the adoption of a strong architectural definition in the early stages of software development is fundamental and can promote the anticipation of the inherent challenges, leading to continuous optimization. Early identification of potential bottlenecks provides a unique opportunity to make design decisions when they are still economically viable, avoiding significant reengineering costs in advanced development stages. Ultimately, a well-defined architecture not only strengthens the testing process but also provides a clear and actionable insight into software performance, enabling continuous and efficient evolution throughout the entire development cycle. The importance of a well-defined architecture for testing goes beyond mere technical formality, playing a crucial role in enhancing development processes. With a carefully outlined architectural structure, the development and test phases might be conducted more efficiently and effectively, enabling the early identification of potential limitations for the application.

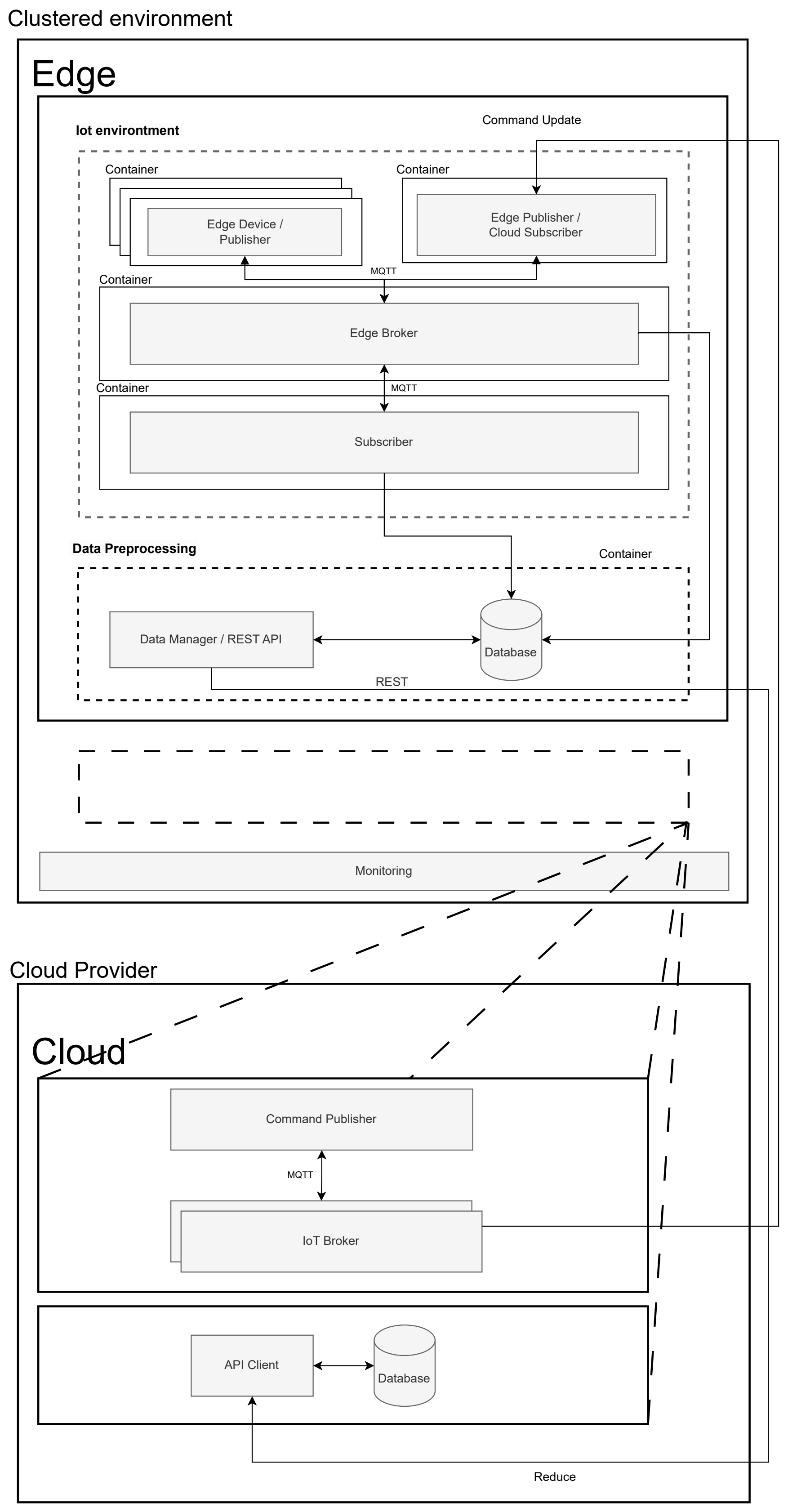

Figure 5.2 – Modular Emulation Architecture for Scalable Edge Applications.

In Figure 5.2 we present the Modular Emulator Architecture for Scalable Edge Applications [1] we propose in this thesis. The understanding of our architecture starts with the two resource allocations considered. First, there is the clustered environment, which we assume to be an on-premise cluster running container images. These elements represent the edge components and create a communication layer that will start the dataflow throughout the network. Once we have the edge elements within containers in a cluster environment allows a series of paths to be explored, such as resource constraints regarding memory, CPU, and disk. Furthermore, traffic control can also be explored throughout the network. By having these nodes on containers, the messages sent will still go through the network, and each container might be deployed in a different domain if necessary. The second part considered is the cloud domain. For these elements, allocation placement is meant to be at a cloud provisioning service. These same containers could also be deployed at the edge environment if network constraints were applied to generate the resource limitations.

At the edge, the message exchange between the emulated edge devices is based on a publish/subscribe protocol called Message Queue Telemetry Transport (MQTT). The MQTT protocol is also used in the communication from the cloud to the edge side in the scenario where the cloud does not know the edge device clients. Message exchange plays a critical role in edge applications, and the architecture addresses this by outlining the flow of communication between different elements. MQTT protocol is adopted for efficient and lightweight messaging between nodes. Publishers establish connections with brokers to send messages, while subscribers connect and subscribe to specific topics. The initial handshake between these elements incurs additional traffic, the volume of which depends on the number of instances instantiated. This handshake process can potentially become a bottleneck, particularly when dealing with a large number of instances.

To emulate large amounts of data, we consider a component that connects with the containers as a publisher, and its role is to represent IoT sensors. As mentioned above, the messages are generated with specific topics to be consumed by their subscribers. More specifically, the component produces data representing, for instance, readings from humidity, temperature, sound, and fluidity sensors that generate data considering a configurable interval. Algorithm 5.1 presents the pseudo-code for the MQTT publisher to generate IoT device parameters data. Sensor data transmission represents a core aspect of edge applications, and the architecture includes provisions for managing this flow. Sensor readings are generated at the "Edge Device/Publisher" component, mimicking the behavior of IoT sensors. These readings are then propagated through the MQTT broker to subscribers. The architecture acknowledges the potential for bottlenecks, particularly with centralized edge brokers handling all incoming and outgoing data.

The architecture presents subscriber components that interact with N brokers to collect messages at the edge. A broker is responsible for receiving and propagating mes-

[1] https://github.com/mmatheuslyra/EdgeExperiment

Algorithm 5.1 Device data publisher pseudo-code

```
Initialization
EdgeDevices $\leftarrow [Device_0, Device_1, ..., Device_n]$
$PublishTopic \leftarrow [flow, temperature, humidity, sound]$
$MQTTClient \leftarrow Connection$
while Emulator is active do
    Wait Interval
    Topic $\leftarrow Random(PublishTopic)$
    $Message \leftarrow Random(EdgeDevices)$
    $MQTTClient.Publish(Topic, Message)$
end
```

sages from generated IoT devices to their respective subscribers. The remaining components are the data manager, responsible for sending pre-processed data to the cloud using a Representational State Transfer (REST) protocol, and the cloud client. The choice of using REST is due to the intention to maintain communication between edge and cloud using a protocol that supports larger payloads compared to MQTT, which focuses on devices that usually have an intense throughput but with lighter payloads. Thereby, cloud communication assumes an aspect closer to an Application Programming Interface (API) and uses a protocol with the appropriate purpose [144]. Regarding the implementation details on the visibility to the cloud, there are two different ways edge devices might be exposed, which implies the usage of different communication protocols. One might reproduce a scenario where the cloud has the addressees of edge devices and might freely communicate with it. The second should consider an environment where edge device addresses are not exposed to the cloud. To contemplate the second scenario where only edge components know the cloud address but the cloud component does not know the address of each element at the edge, in our proposal, a client using the MQTT protocol connects to the broker allocated in the cloud. The cloud publishes specific topics that are used to propagate commands to the edge devices.

Through these elements, our architecture represents a minimal structure for an edge ecosystem. The well-defined responsibilities among the mentioned nodes, as well as the existence of two different allocation blocks, add a degree of modularity to the system that enables various modifications in the IoT elements interactions. Within these responsibilities, the edge block has components focused on creating the dataflow that we consider fundamental among the IoT elements, which we nominated as the IoT environment inside the edge block. In addition to this, within this same edge block, we have elements that play the role of data storage and processing within the block called data processing. This entire edge block was designed to operate independently of the cloud, as we understand this as a situation that may eventually occur in real scenarios, where edge and cloud may lose their access paths from time to time. The interactions between edge and cloud elements establish fundamental communications that allow data traffic analysis. Furthermore, hosting the cloud block in a cloud provider, allows developers to estimate financial costs through the experimentation and performance of its elements.

Edge computing has several limitations, such as resource consumption, network, processing, etc. Therefore, through docker resource configuration, we can set specific limits for each container and network controller of the emulated IoT application, reproducing the conditions of a real scenario. These configurations can be easily changed if the user wants to experiment with different requirements. Furthermore, the area faces problems in estimating performance against hardware contingencies, which once more advocates the usage of emulation tools since this is a phenomenon hard to reproduce in simulation. As previously mentioned, within a simulated environment, most of the metrics are abstracted, which can distance the final result from a real-world scenario. Through emulation, resources, and infrastructure have different degrees of realism, where real elements can be added keeping metrics such as network latency equal to the real world. Such experimentation tools that consider data traffic between devices and explore its limitations in terms of processing power, different communication protocols, and visibilities between elements might contribute to deciding which part of an application should remain on the edge or in the cloud. Existing emulators have specific strengths that diverge from our goals, as later presented in Chapter 4. Therefore, we intend to contribute to the area of edge computing enabling the development of optimized applications with efficient device distribution considering an edge/cloud environment.

We believe this architecture addresses a few challenges identified in the early stages of our research. This thesis paves the way for software development of digital environments over edge emulation. The components presented in the architecture, despite being generic in terms of their functions, play fundamental roles in an edge environment. These components might be used to create an intense dataflow based on message exchanges throughout the network, data persistence, and summarization to the cloud. As a result, they create a dataflow that can be used as assistance in the development of other applications, as we validate in the following chapters. By considering modular elements, we understand that each node can be used as a generic case, replicated multiple times, altered, restricted, or even entirely replaced. Our proposal does not restrict the implementation but allows it to take different facets, and if the basic aspects of the architecture are kept, an edge computing environment will be achieved. Therefore, implementing such architecture might be used to mimic specific conditions, develop PoCs, validate application behaviors over extremely restricted scenarios, verify the maximum number of clients supported simultaneously, and more.

5.3 Emulation of Resource Constraints

In this section, we go into the intricate landscape of resource constraints, scrutinizing pivotal dimensions. Understanding and effectively managing these constraints in the pursuit of optimized system performance ensures the seamless operation of applications in the face of dynamic and often challenging computing scenarios. This section delves into the nuanced challenges posed by limitations in general resources and the intricacies of traffic control and networking. By dissecting these aspects, we aim to glean insights that not only deepen our understanding of resource limitations but also pave the way for strategic solutions that enhance the robustness and efficiency of computing systems in the face of resource constraints.

5.3.1 Motivation in the usage of Resources constraints

The significance of resource constraints within emulators cannot be overstated. CPU and memory constraints are especially crucial for mirroring the processing limitations that applications would encounter in current hardware environments. This constraint not only adds a layer of realism to the emulation but also forces applications to adapt to the available computational resources, allowing researchers and developers to assess how software behaves under different processing conditions. In the digitalization of edge scenarios, restricting these resources is fundamental.

Furthermore, understanding and imposing these constraints is essential for evaluating the scalability of applications. By creating scenarios where memory and CPU resources are limited or fluctuate, researchers can gain insights into how applications handle varying computational workloads. This is particularly pertinent in the context of edge computing and distributed systems, where these resources might be very restricted depending on the hardware. Which makes an even more pivotal role in determining the feasibility and efficiency of deploying applications across diverse computational environments. The strategic imposition thus becomes a valuable asset or even a necessity for not only gauging the performance of applications but also for optimizing them to operate effectively within the confines of real-world computational limitations. In essence, the careful consideration of resource constraints contributes to the creation of more reliable and adaptive software solutions that can navigate the complexities of diverse hardware environments.

5.3.2 CPU constraints

In this thesis, CPU constraints refer to limitations or restrictions defining the amount of CPU resources that a system, process, or application can utilize. In general, these constraints are utilized to manage and allocate resources to concurrent tasks and can take various forms, such as setting maximum usage limits, defining priorities for access, or restricting usage to specific cores or intervals. In the context of optimizing applications, CPU constraints are particularly relevant for managing performance and resource utilization within a shared computing environment. By defining such limitations, users can ensure fair allocation of resources, prevent monopolization, and optimize overall system performance. These constraints may be applied in applications using different tools, one example is Docker virtualization, where restrictions can be placed on the amount of CPU resources that containers can utilize. These constraints are essential for managing performance and resource allocation. There are several aspects to consider regarding CPU constraints in Docker, such as shares, quota, and period.

Applying restrictions over processing power aids in emulating applications by facilitating fair resource allocation, mimicking specific requirements, isolating performance, providing predictability, testing scalability, and optimizing costs. Each emulated application might receive a balanced share, preventing resource monopolization and ensuring consistent performance across multiple applications running concurrently. This ensures that even under heavy loads or CPU-intensive tasks, the performance of other emulated applications remains unaffected. Moreover, CPU constraints enable developers to predict application behavior accurately under different resource scenarios, allowing for more reliable testing and development. Additionally, by gradually adjusting parameters, developers can assess application scalability and identify potential bottlenecks. Therefore, the usage of CPU constraints helps optimize costs by ensuring efficient resource utilization, thereby minimizing unnecessary expenses associated with over-provisioning.

5.3.3 Networking and Traffic Control

Research in the IoT area has been mentioning network-related problems as one of the main challenges for a long time. As we can see in early research from more than a decade ago, such as "The Internet of Things: A survey" [145], where it was already discussed the pursuit to ensure scalability considering the incredibly high number of nodes, along with the differences and new possibilities brought by IPv6. In the research of virtual systems, while trying to create accurate metrics for the emulated environments, the network conditions are still a main concern, as we might see in one of the related works, such as

CELESTIAL [146], going through different mechanisms to achieve that goal on the realm of emulation of low-Earth orbit (LEO) edge.

Traffic control plays a decisive role in shaping the efficacy and fidelity of emulated environments. Emulators strive to replicate real-world conditions, and in doing so, they must mimic the intricate dynamics of data traffic within networks. The importance of traffic control in emulators lies in its ability to go further in network behavior, considering variables such as latency, packet loss, and bandwidth constraints. By integrating traffic control mechanisms, emulators can go through diverse network scenarios, from optimal conditions to challenging and congested environments. This nuanced approach ensures that the emulation process extends beyond mere software functionality, providing a realistic representation of how applications perform in diverse network conditions. The ability to create variations in network behavior allows researchers to scrutinize how applications respond under different levels of stress, paving the way for the identification and resolution of potential bottlenecks. For instance, by having real messages sent between nodes, in addition to the traffic control elements, it becomes possible to create edge scenarios with severe restrictions that would happen only under bad weather conditions.

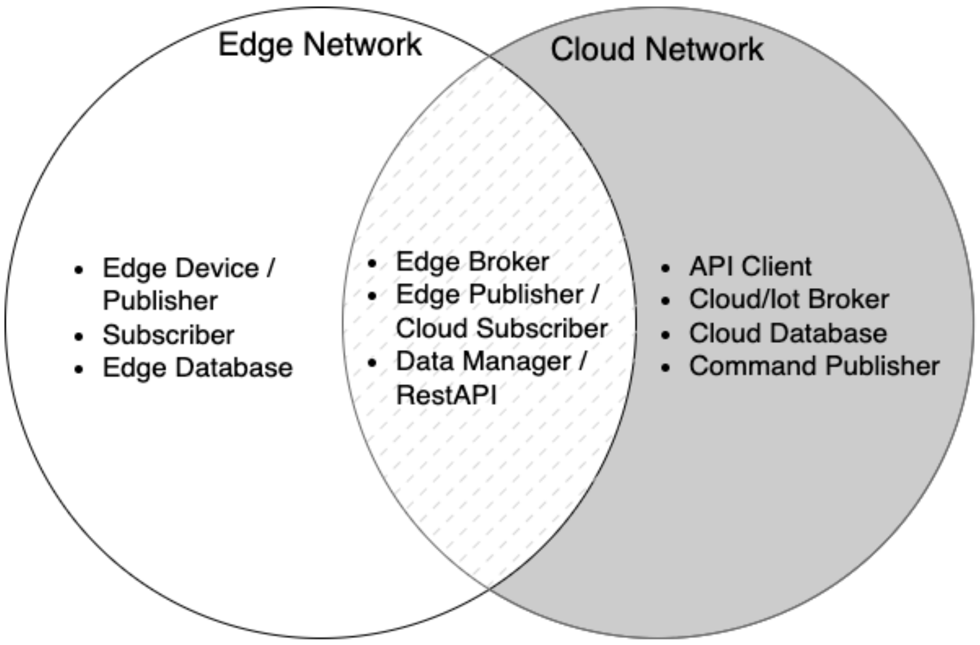

Figure 5.3 – Containers network distribution.

Image 5.3 presents the distribution of architecture containers introduced in Section 5.2 between two networks, Edge and Cloud. This distribution makes it possible to exercise two main points. The visibility of the elements between each other and the traffic control mentioned above. Regarding the experiments presented in the next chapter. In initial tests, only visibility aspects of the network distribution were considered. In the second battery of tests, conducted in the same testbed, traffic control was the specific topic.

5.4 Edge Emulator Prototype Implementation

In this chapter, we presented a Modular Emulation Architecture for Scalable Edge Applications where we consider different nodes spread throughout the network to create a digital environment capable of representing the nuances of an edge environment. We developed a prototype for the architecture considering the elements presented in Figure 5.2. Regarding the implementation details of our prototype, the overall programming language utilized was NodeJS. That applies to edge and cloud nodes, as well as the implementation of MQTT and REST communications. Each node runs in docker containers with resource constraints on the image configurations. Docker was also used to establish traffic control in the independent edge and cloud networks. Since the nodes in the implementation do not run in a single application but are distributed in independent containers, it allows each node to be allocated in different hosts. This is also a characteristic that might be used to achieve scalability goals, by instantiating new containers building up over the proposed architecture.

The proposed architecture considers a great volume of message exchange which is intricate to the nature of an edge application. To understand that overall interactions between the elements might be challenging. Therefore, Figure 5.4 represents the flow of the main message exchanges between the elements. The components that communicate through MQTT, start with the protocol handshake, which will occur for subscribers and publishers towards the broker as represented in the "MQTT Handshake" frame. The difference between them is that publishers only need to connect with the broker to start sending messages on a given topic, the subscribers need to connect and subscribe to a topic list of interest. These initial protocol messages might be considered as additional traffic, and the volume of messages in this handshake will depend on how many instances were created. For example, in Figure 5.4 the publishers and subscribers at the edge are represented only once, but in an experiment, they could be instantiated as many times as necessary. With just a few containers this traffic would certainly not be a problem and could even be disregarded. However, having a more expressive number of instances could make this a bottleneck.

After the protocol establishes communication, the elements start to send the sensors reading messages. These data are generated in the "Edge Device/Publisher" component, which implements Algorithm 5.1. Representing IoT sensors, this component might be considered the initial point of the readings dataflow. The volume of data generated at this point can be used to create bottlenecks in the environment. Especially, considering that there is a centralization in the edge broker, since all publishers and subscribers rely on this element to send and receive data. These interactions might also be considered as a loop, in the overall communications, as represented in the "Main Loop" frame of the diagram. Since the publisher will continue sending messages with the sensors reading topics, these messages will be propagated by the broker as long as the containers are running. Also, there is

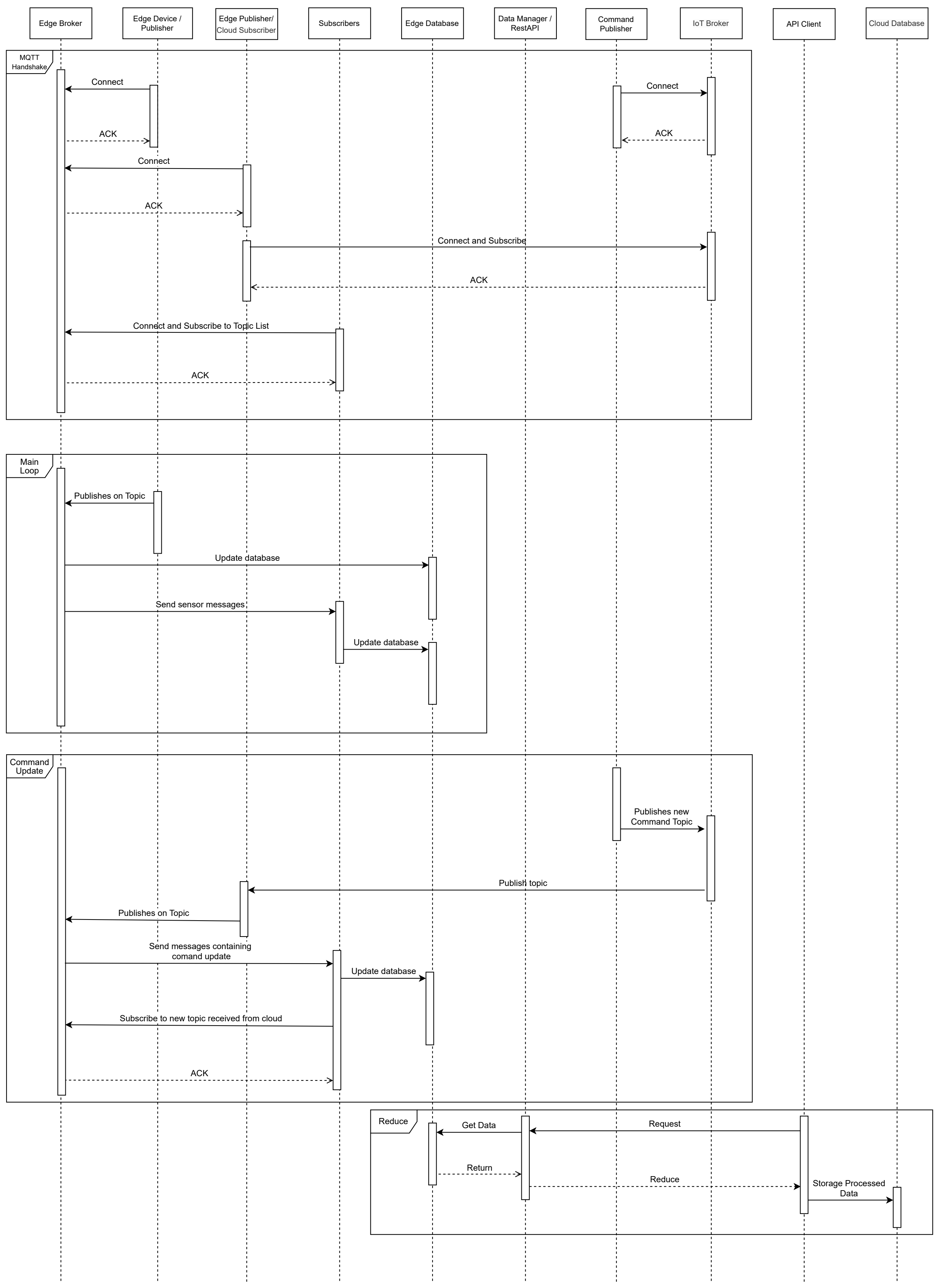

Figure 5.4 – Sequence Diagram of the Edge Emulator Architecture Implementation Prototype containing the main messages exchanges.

no actually limit to the number of instances for publishers and subscribers instances. Which might be used to increase the traffic along with parameter changes in the publisher algorithm.

By default, the subscribers have a list of interests, containing the topics that the elements will be subscribed to. However, the architecture suggests an additional mechanism, creating an MQTT connection with a cloud broker to update the edge subscriber topics. Therefore, even without knowing all the edge devices, it is possible to have control over communication from the cloud. This mechanism appears in the chart through the interactions between the "IoT broker" and the "Edge Publisher / Cloud Subscriber". These interactions are represented in the "Command Update" frame. Another pivotal point is the interaction between the "Data Manager" and the cloud "API Client", represented in the "Reduce" diagram frame. Using REST communication, the API client sends a request to the Data Manager, which processes the data and sends a response to the cloud. This mechanism allows the reduction in the data amount that is sent to the cloud. Therefore, the requirement for cloud resources to be processed and stored is also reduced.

For traffic control, there would also be different ways to implement it. In the related work, we found the usage of the native Linux Traffic Control (TC) and other tools such as mininet. For our prototype, we chose to go with Docker's networking resources. In Docker, traffic control is implemented through the manipulation of network settings and policies. Docker provides a range of networking options, such as bridge networks and user-defined networks, allowing users to configure traffic control parameters. For instance, rate limiting, and bandwidth constraints can be set on Docker networks to control the flow of data between containers. Moreover, Docker allows users to leverage the inherent delay features within network settings to introduce latency, mimicking real-world network conditions.

5.4.1 Edge Emulator as a Tool: Monitoring Layer

To enhance usability and create a more user-friendly environment„ we developed features containing a few orchestration scripts focused on trying to centralize the usage of the emulator's internal calls, and the steps to create the environment such as building and starting containers. Therefore, the user can start using the tool just by sending commands and with no need to actually change any parameter.

Furthermore, the architecture described in Chapter 5.2 predicts a monitoring layer. In our implementation, we added real-time chart generation with Python. We created a feature to start chart plots considering the containers proved to be the overall bottlenecks in the experiments further described in Chapter 6. Figure 5.5 presents an example of real-time charts for resource consumption, representing the usage of CPU, Memory, Network Rx, and

Network Tx considering four different containers. Further behavior analysis with different examples might be found the following chapters.

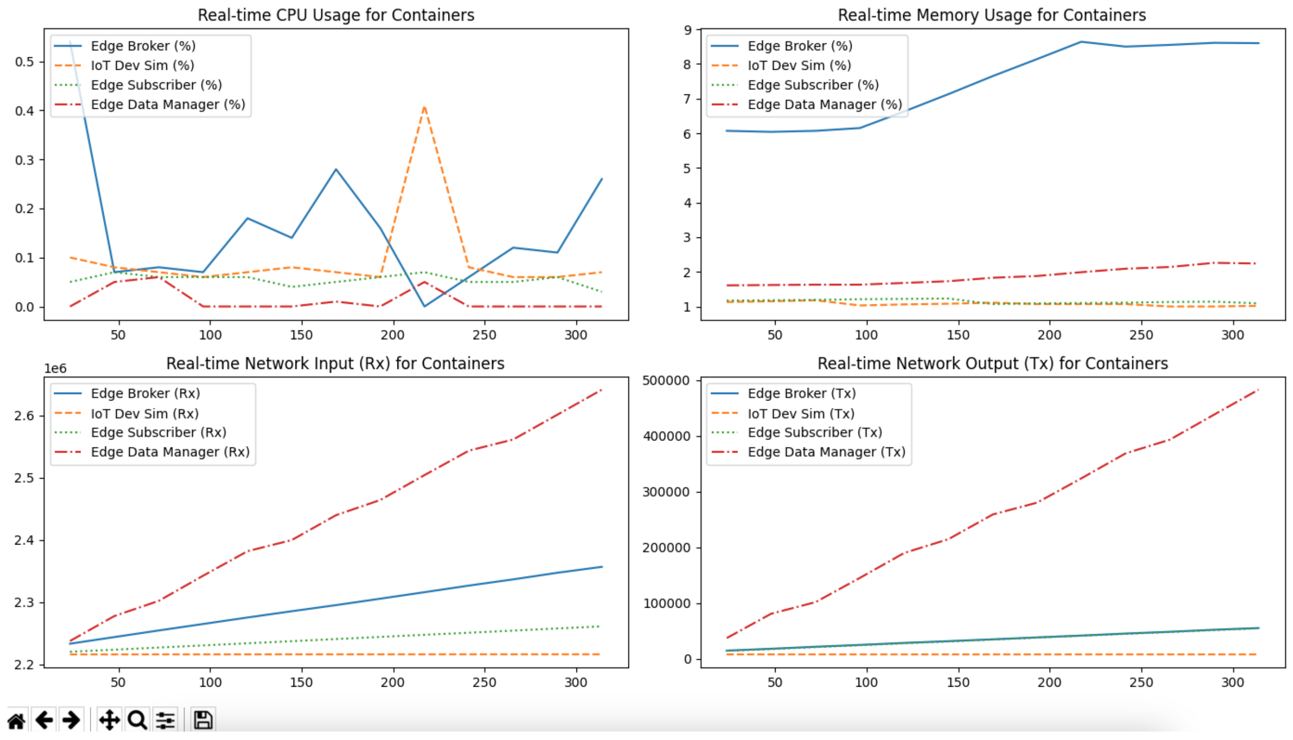

Figure 5.5 – Prototype real-time charts for container resource consumption for 300 seconds (5 minutes). Charts represent the usage of CPU, Memory, Network Rx, and Network Tx considering four different containers.

We believe that this monitoring layer adds value to the emulator since it makes it easier to identify behavior patterns. However, plotting real-time charts generates computational overhead, which might overshadow experiment results. Therefore, in our implementation, there are other types of visualization, such as the feature presented in Figure 5.6, which generates a minor overhead by monitoring general values for CPU, Memory, Network I/O, and Disk. Also, it presents a minor summary of the local machine resources, such as the Disk, Memory, and Processor model to serve as a reference for the user. Furthermore, it was added another monitoring feature that only generates log files.

The architecture monitoring layer might be implemented in different ways. The features we showed above are focused on capturing the resource consumption of each Docker container. There are established monitoring tools that may also be used for that purpose. One example is Grafana[2], an open-source analytics and monitoring platform designed to visualize and analyze data from various sources in real time. It acts as a central dashboard where users can integrate data from multiple systems, databases, and applications. It also supports alerting mechanisms, allowing users to set up notifications based on specific conditions. What proved to be interesting about using this tool in our implementation was the focus on monitoring the data that passed between the containers and not on its resource

[2]https://grafana.com/

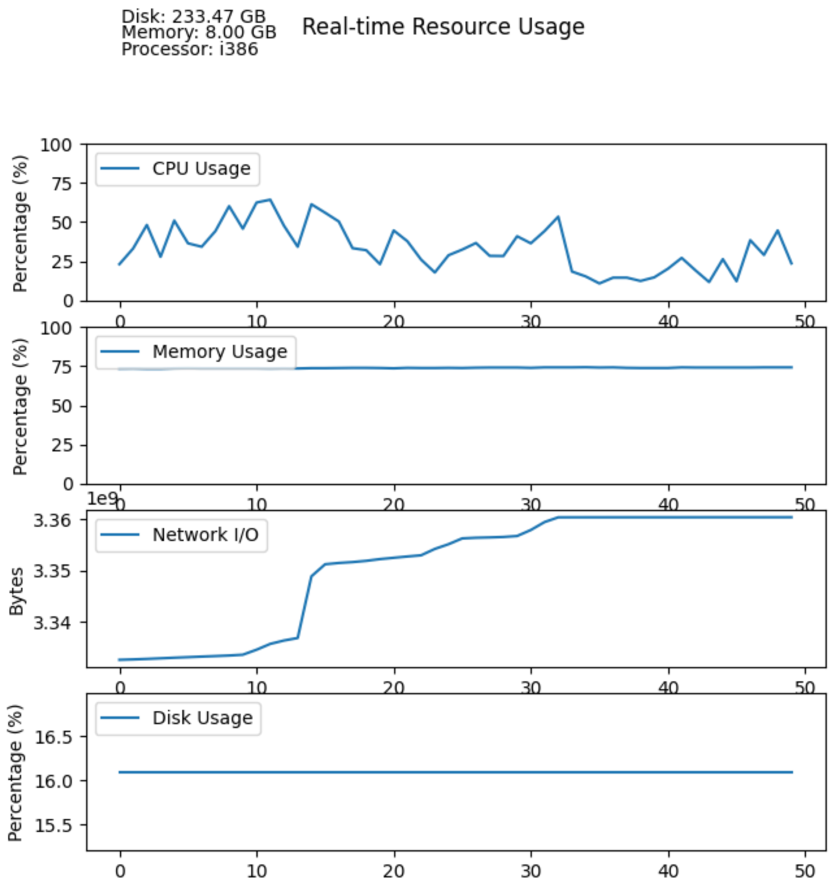

Figure 5.6 – Prototype real-time charts for CPU, Memory, Network I/O, and Disk resource consumption.

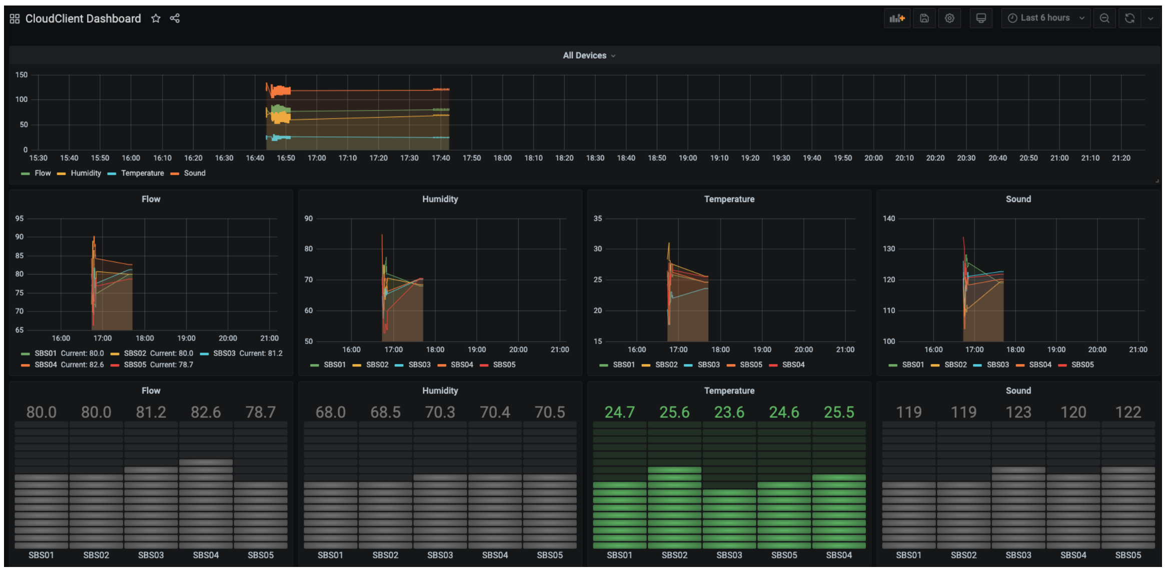

Figure 5.7 – Readings collected from Grafana. The monitoring layer in this scenario is built over the data stored in the Cloud Database. The readings refer to Flow, Humidity, Temperature, and Sound considering five different sensors SBS01, SBS02, SBS03, SBS04, and SBS05.

consumption. While our previous charts focused on getting data per container, in this case, the focus was on the dataflow itself. Each message sent in our prototype has tags of where it was sent for and what sensor reading it represents. These values are stored at the edge and cloud, which makes it possible for Grafana to monitor data. Figure 5.7 presents Grafana representation from readings stored in the Cloud DB. As mentioned, these refer to readings of Flow, Humidity, Temperature, and Sound considering five different sensors SBS01, SBS02, SBS03, SBS04, and SBS05.

6. PROTOTYPE VALIDATION AND USE CASES

This chapter presents the results obtained from the experiments conducted with the implementation described in Chapter 5.4, referred to in this chapter as the edge emulation tool. First, we present our resource constraints experiments. To validate the effectiveness of our tool, we present a second experiment on a use case in the field of smart agriculture, where the emulation tool successfully reproduces a vineyard use case and evaluates the scalability of the system. Thereafter, we present a scalability experiment based on the same smart agriculture use case.

6.1 Resource Constraints Experiments

This section navigates through our resource constraints experiments. The explored dimensions are focused on the restriction of general resources such as container CPU sharing and memory, along with Traffic Control. In combination with our real-time plots, we explore how limitations in these foundational elements impact resource utilization and communication. This section dives into the validation of how our tool might be used in experimenting, design, and execution. In this experiment, in order to observe the resource consumption behavior and the real-time plots, ten containers were allocated to represent the sensors containers' multiple deployments running on a testbed consisting of a MacBook Pro, 2,3 GHz Dual-Core Intel Core i5, 8 GB 2133 MHz LPDDR3. The experiment protocol comprised three distinct stages. In the initial phase, the round was conducted without imposing resource limitations per container or network. Subsequently, during the second stage, the restrictions were introduced, and in the third stage, an intentional overrun of these predefined values was implemented.

The primary objective of the experiment was to observe the processing metrics of the Edge Broker meticulously. Concerning network parameters, the focus was on monitoring general network metrics, consideration the presence of specific limiters. This meticulous de-

Table 6.1 – Network Configurations at Different Stages of the Resource Constraint Experiments

Stage	Network	Driver	Ingress Bandwidth	Ingress Burst	Ingress Latency	Egress Bandwidth	Egress Burst	Egress Latency
1	Edge	bridge	-	-	-	-	-	-
	Cloud	bridge	-	-	-	-	-	-
2	Edge	bridge	100kbit	10kbit	70ms	100kbit	100kbit	70ms
	Cloud	bridge	100kbit	10kbit	70ms	100kbit	100kbit	70ms
3	Edge	bridge	500kbit	100kbit	500ms	500kbit	100kbit	500ms
	Cloud	bridge	500kbit	100kbit	500ms	500kbit	100kbit	500ms

Table 6.2 – Service Configurations at Different Stages

Stage	Service	cloudBroker	cloudClient	cloudServer	edgeBroker	edgeDataManager	edgeIoTSmartDev
1	CPU Shares	-	-	-	-	-	-
	Memory Limit	-	-	-	-	-	-
2	CPU Shares	10 (1%)	10 (1%)	10 (1%)	102 (10%)	10 (1%)	10 (1%)
	Memory Limit	-	-	-	1gb	-	1gb
3	CPU Shares	10 (1%)	10 (1%)	10 (1%)	102 (10%)	10 (1%)	10 (1%)
	Memory Limit	-	-	-	1gb	-	1gb

sign allowed for a nuanced exploration of the interplay between processing capabilities and network performance, providing a comprehensive understanding of the system's behavior under varying conditions. The strategic manipulation of processing and network variables at different stages contributes to the experimental robustness, allowing for a detailed analysis of the Edge Broker's processing dynamics and the broader implications on network metrics. Table 6.1 presents the Docker network configuration restrictions used throughout the three stages of the experiment. Table 6.2 presents the CPU Shares and Memory limits values. The "cpu_shares" value in Docker represents a proportional weight, not an absolute percentage. Therefore, the weight is relative to the CPU shares of other containers on the system as presented in Equation 6.1:

$$\mathit{SharePercentage} = \left(\frac{\mathit{ContainerShares}}{\mathit{TotalSharesOnSystem}}\right) x100 \tag{6.1}$$

Where SharePercentage denotes the proportion of CPU time allocated to a container relative to the total CPU time. ContainerShares denotes the weight assigned to a container for determining its share of CPU resources. TotalSharesOnSystem denotes the total sum of CPU shares assigned to all containers.

A consistent pattern is discernible across various execution stages, as illustrated in Figures 6.1, 6.2, and 6.3. Whether examining CPU, Memory, Network Rx, or Network Tx consumption, each container exhibits a similar behavior when compared to itself throughout each stage. However, it is noteworthy that variations in the depicted volume urge a closer observation of the dynamic changes in the container interactions. The primary bottleneck in the system is the Edge Broker container. This container becomes the focal point for communication, thereby leading to increased demand for computational resources compared to other components. Notably, in resource restriction experiments, the consumption values of this container demonstrate heightened sensitivity and variation in response to the parameter changes outlined in Tables 6.1 and 6.2.

Our main concern in this experiment was to demonstrate how constraints effectively alter the behavior and dynamics of containers. As elucidated in the preceding chapters, the ability to generate diverse scenarios relies on the capacity to impose constraints on the regular operational state. In Stage 1, as depicted in Figure 6.1, normal operating conditions are portrayed without any restrictions, presenting minor variation in the CPU and Network in behavior compared to Stage 2 represented in Figure 6.2. However, it is possible to observe

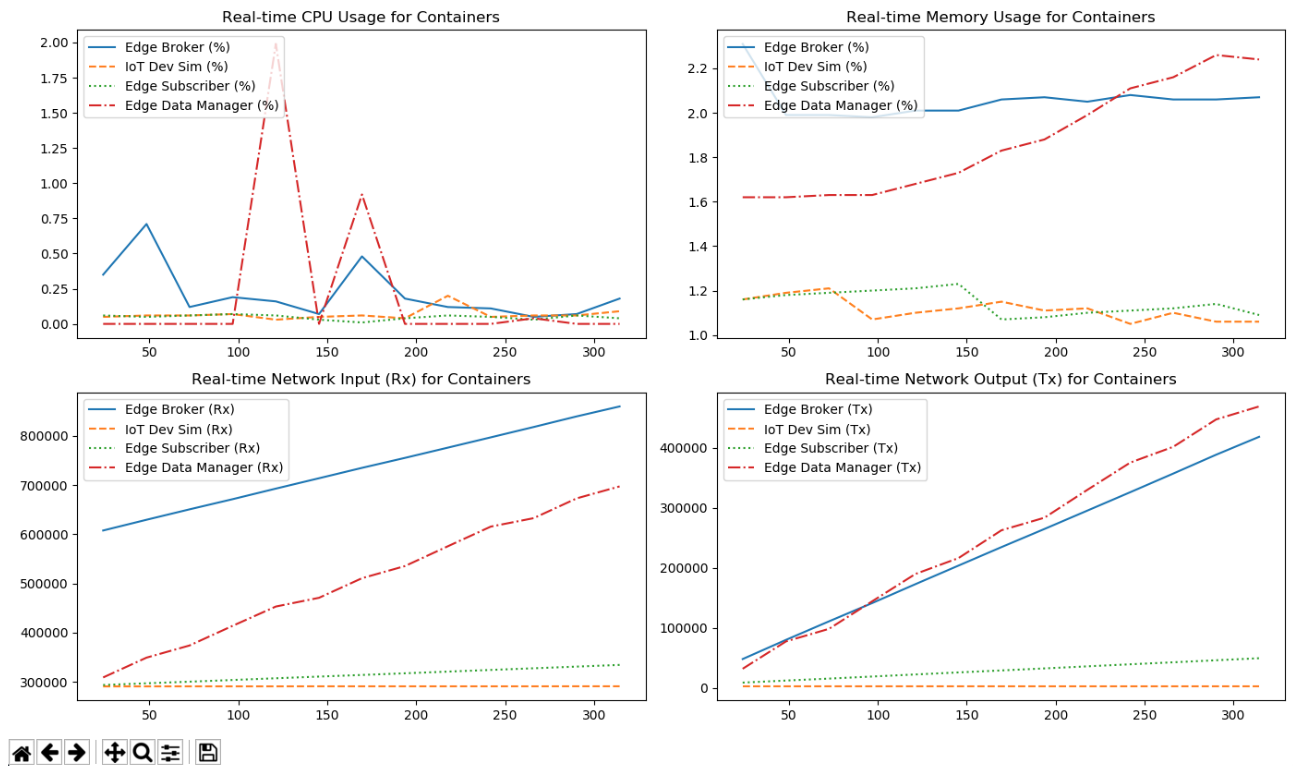

Figure 6.1 – Stage 1 of resource constraints readings for 300 seconds (5 minutes). Charts represent the usage of CPU, Memory, Network Rx, and Network Tx considering four different containers.

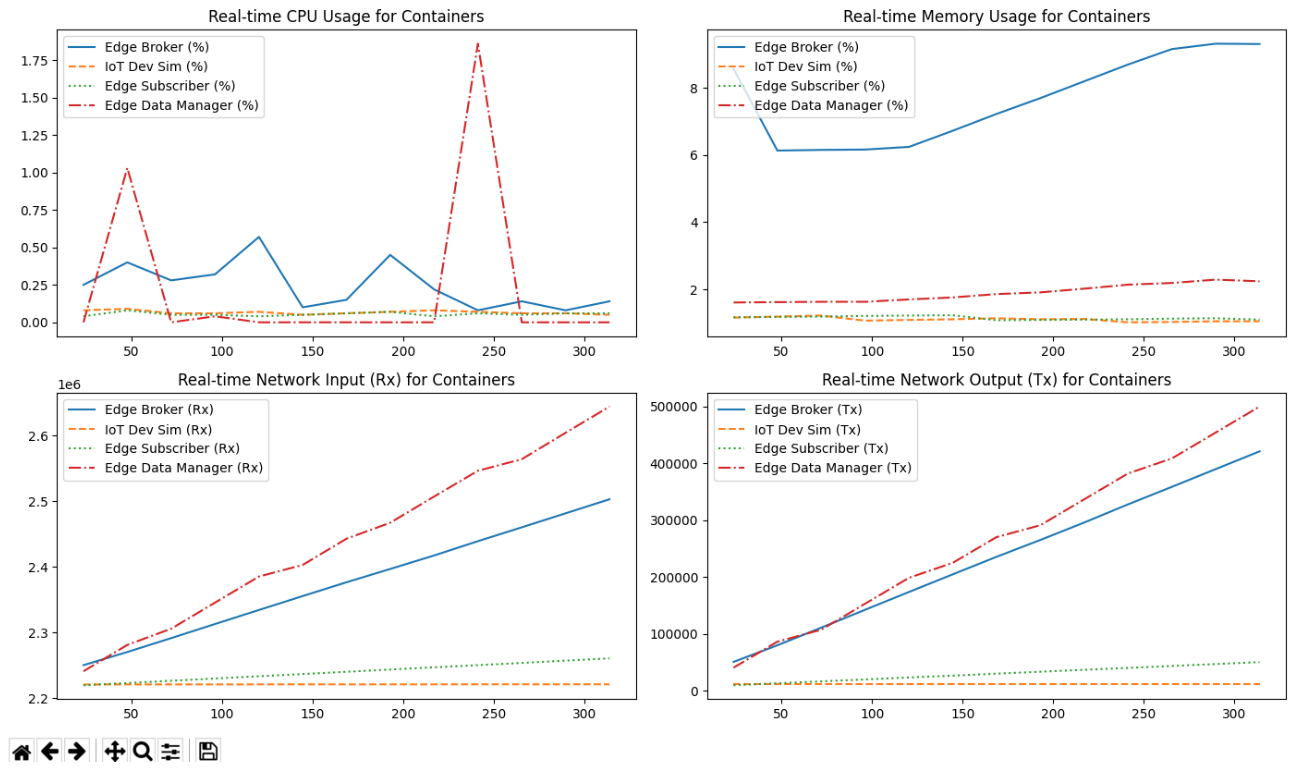

Figure 6.2 – Stage 2 of resource constraints readings for 300 seconds (5 minutes). Charts represent the usage of CPU, Memory, Network Rx, and Network Tx considering four different containers.

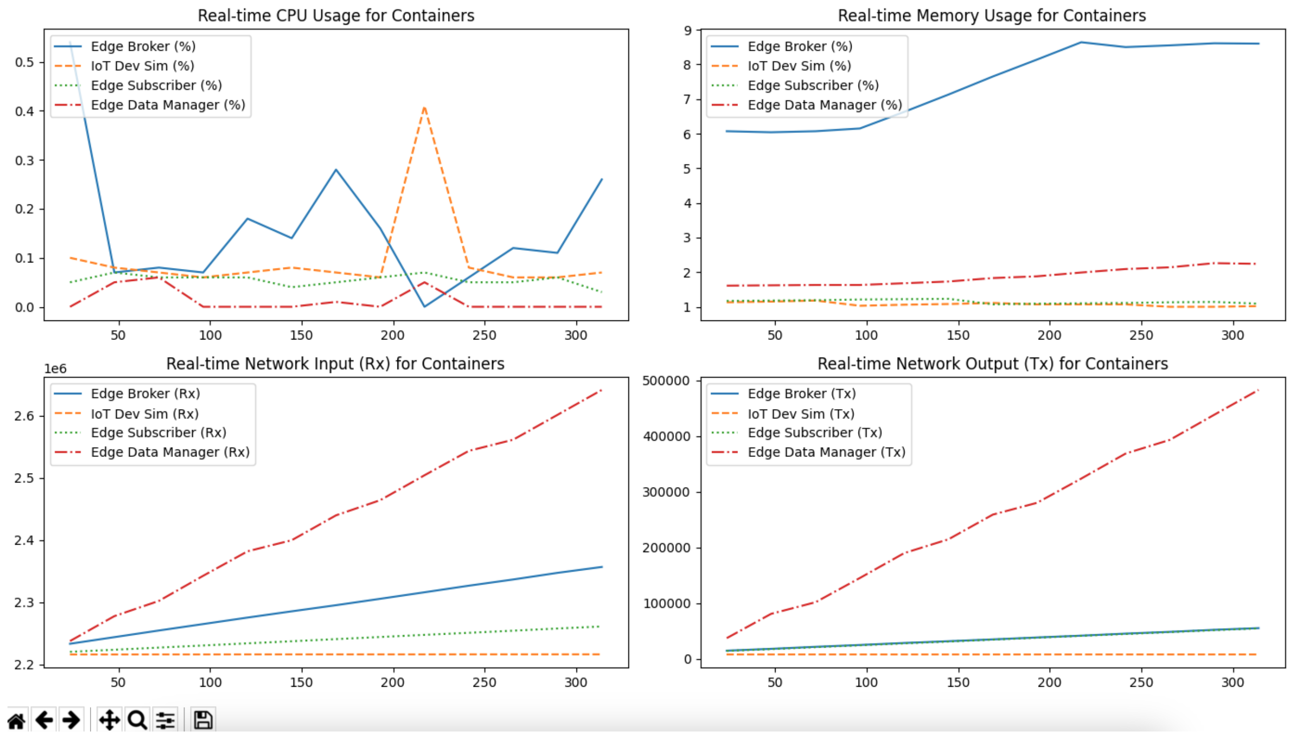

Figure 6.3 – Stage 3 of resource constraints readings for 300 seconds (5 minutes). Charts represent the usage of CPU, Memory, Network Rx, and Network Tx considering four different containers.

that the values presented in the stage 1 graph regarding memory usage show changes for the Edge Broker and Edge Data Manager containers. It is noticeable a difference in the total consumption values compared to the other two stages where restrictions were applied. It is interesting to note that this restriction was only applied to the Edge Broker. However, due to its central role in communication, it also affected the Edge Data Manager container. We believe that this cross-alteration occurs because the latter performs computation on the data received from the Edge Broker. Thus, when the Edge Broker operates with limitations, its consumption is also affected.

A Higher difference appears comparing stage 3 to the previous two, as illustrated in Figure 6.3, where a noticeable shift in behavior is evident, which is even more evident in the Network output. The CPU consumption exhibits a typical pattern, punctuated by occasional peaks that underscore the imposed limitations. To effectively illustrate this, it is imperative to employ a use case characterized by even more intensive computations on the edge broker. In this scenario, merely redirecting MQTT protocol messages proved insufficient to sufficiently highlight its impact on the graph.

The experiment elucidates the impact of resource limitations on the performance of a distributed system, focusing on memory restrictions, traffic control, networking, and CPU. The strategic allocation of our containers in different stages provided a structured approach to assess the system under varying conditions. By systematically adjusting network configurations, including bandwidth and latency parameters, the experiment simulated scenarios with and without processing limitations. This allowed for a comprehensive exploration of the interplay between hardware resources and network performance, providing valuable insights into potential bottlenecks and optimizing strategies. Furthermore, the experimentation extended to service configurations, where CPU shares and memory limits were manipulated across stages. Notably, the allocation of computing resources to the edge broker played a critical role in influencing overall system performance. The inclusion of memory limits for specific services, such as the edge IoT device simulator, introduced an additional layer of complexity, showcasing the experiment's adaptability to diverse real-world scenarios.

6.2 Experimentation Over a Smart Agriculture Use Case

In this Section, we present a vineyard use case chosen to validate the developed edge emulator tool due to the complexity of edge infrastructure that is involved and the required scalability needed to cover an entire vineyard, making an in-site PoC too complex and time-consuming. The results of this experiment were also part of our publication "A versatile emulator tool to evaluate the scalability of Edge architectures in IoT scenarios" at the 9th IEEE World Forum on Internet of Things (WF-IoT 2023).

In smart vineyards, temperature, humidity, and light sensors are distributed along the whole territory. These sensors collect data from each grapevine or grapevine group, depending on the vineyard size and the financial costs, and transmit it to a smartphone app, where wine producers would be able to monitor in real-time or near real-time the quality of their product. The collected data from the vineyard are usually transmitted through LPWAN protocols, Wi-Fi, or a combination of both, using the MQTT protocol to publish and subscribe information to/from an MQTT broker.

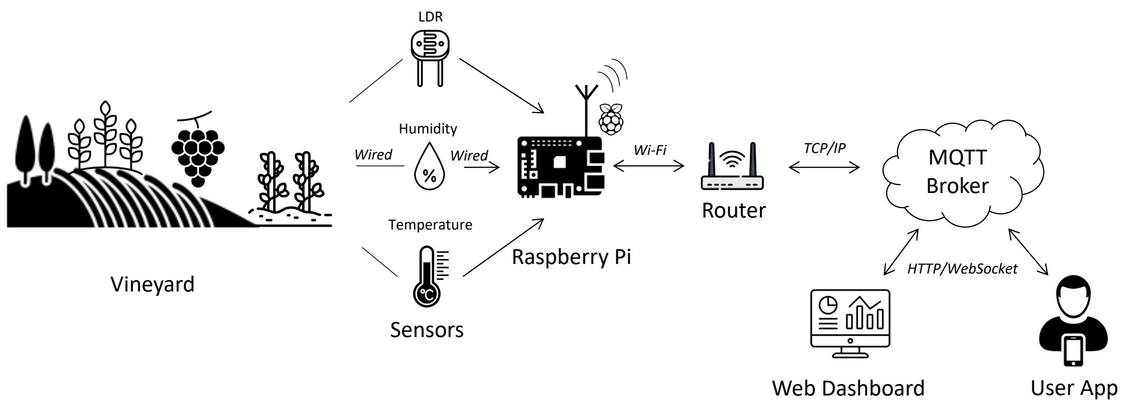

Figure 6.4 – Vineyard use case scheme.

The sensors responsible for monitoring each grapevine or a group of grapevines were connected to a Raspberry Pi with a Wi-Fi module on board (edge device). The first problem that comes to mind is the signal collisions between the Wi-Fi modules and the high data rate handled by the router. The big challenge is ensuring that all data read from the grapevine sensors will be successfully received by the MQTT broker, even in a large-scale application. For a smart vineyard solution, a simulation architecture would require multiple Raspberry Pi boards exchanging data with an MQTT broker. The data loss estimation could be considered in future collision-tolerant algorithm development. This extra validation layer would increase the chances of detecting bugs that would be identified only in the PoC, saving money and engineering time. Figure 6.4 presents the considered smart vineyard diagram and Figure 6.5 its state machine.

State machine:

1. Read Sensor: reads the sensors (humidity, temperature, luminosity) and goes to the Publish Data state.

2. Publish Data: publishes sensor data via MQTT to the broker and goes into low-power mode.

3. Sleep: low consumption mode. It remains in this state for 15 minutes. Wakes up and returns to the Read Sensor state.

4. Subscribe Handle: works like an external interrupt. It is only activated when the device detects the receipt of a subscription. This subscription can be a command asking to read a specific sensor. Then it handles the received command and goes to Read Sensor, asking only for the requested sensor data.

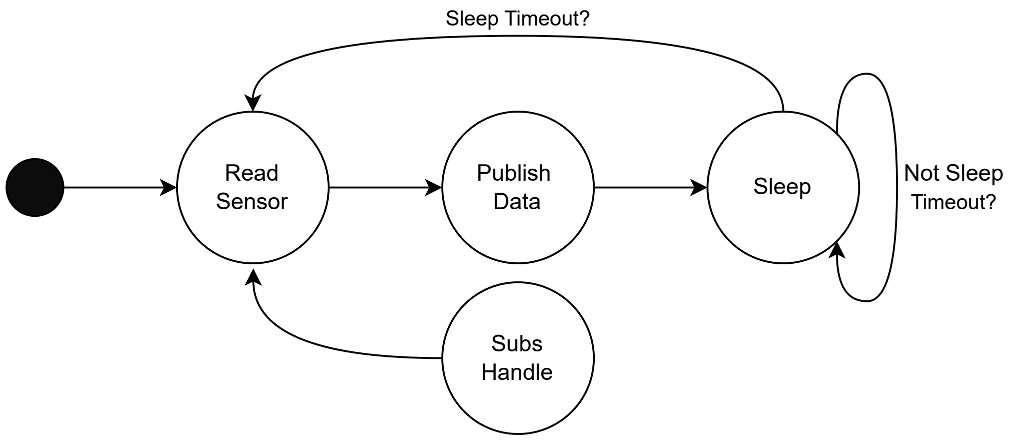

Figure 6.5 – Vineyard state machine.

The vineyard hardware configuration consists of a Raspberry Pi 3B+: 1GB RAM; Quadcore 1.2 GHz; SD of 64 GB. Temperature sensor: analog NTC thermistor part number B57164K0103. Light Sensor (LDR): Part number 02-LDR1 50-100K ohms. Humidity Sensor: Part number DHT22, range 0-100%. For our emulation tests, the computational testbed

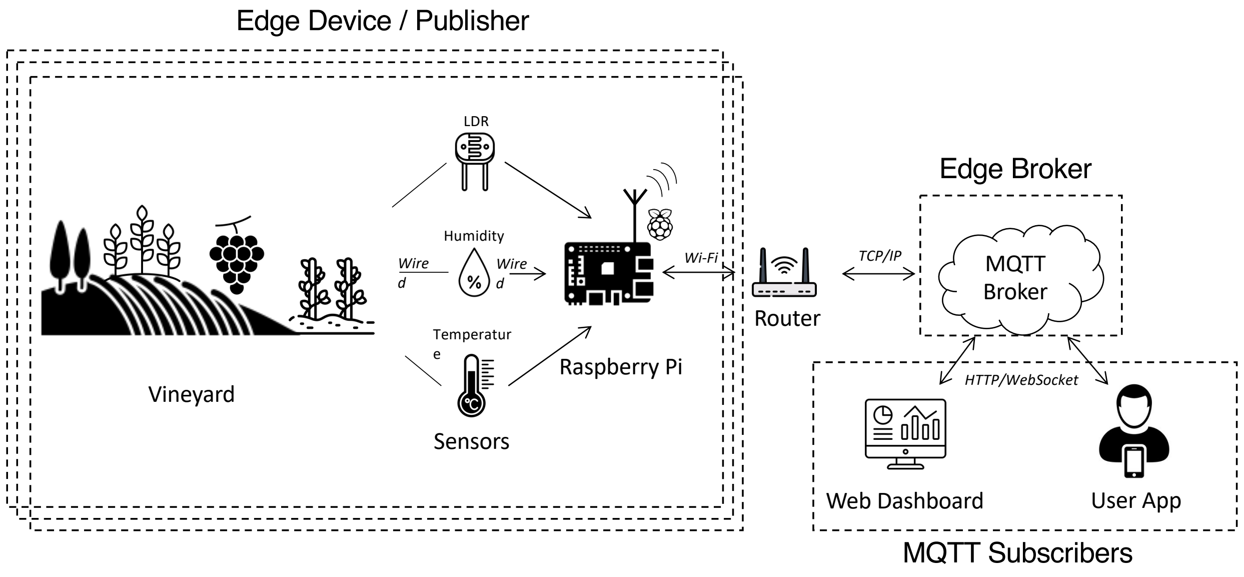

Figure 6.6 – Vineyard use case scheme considering the distribution of the containers.

consists of a three-node cluster in an ethernet network. Each node is a Dell EMC PowerEdge R740 server with two sockets, each one containing a 5118 Xeon Gold processor of 2.30 GHz, 12 Cores/24 Threads 12 MB L2 cache. The total number of cores is 24/48, with 16.5 MB of L3 cache shared by the two sockets and 322 GB of main memory. Since an IoT network presents limitations regarding the delay between nodes and rate traffic inherent to its nature, we take into consideration those concerns and control via software the number of messages sent by the emulated sensors.

In the experiment data, the observed behavior is that, as the number of edge device containers increases from 1 to 50, the resource consumption also does. For the chosen use case, the edge devices containers presented in the Architecture of Figure 5.2, emulate the Raspberry Pi behavior from the use case scheme of Figure 6.4, along with the sensors' data. Figure 6.6 shows the use case elements presented in the Architecture, throughout the emulator containers presented in the use case. The message exchange that happens in the edge is concentrated in the edge broker node due to the MQTT protocol behavior. Therefore, the flow starts in the edge devices containers and passes through the edge broker, which propagates to the subscribers. One of the subscribers is the data manager container, which will create another flow between edge and cloud. Figure 6.7 represents the network consumption of three specific containers considering the increasing replication of the Edge Devices containers as mentioned above. The other containers represented are the Edge Broker, Data Manager, and, MongoDB (chosen storage for this case).

We might observe that the input and output regarding the Edge Broker node tend to be higher than the other containers. This might be caused due to the inherent behavior of the MQTT protocol mentioned above, concentrating most of the message exchange at

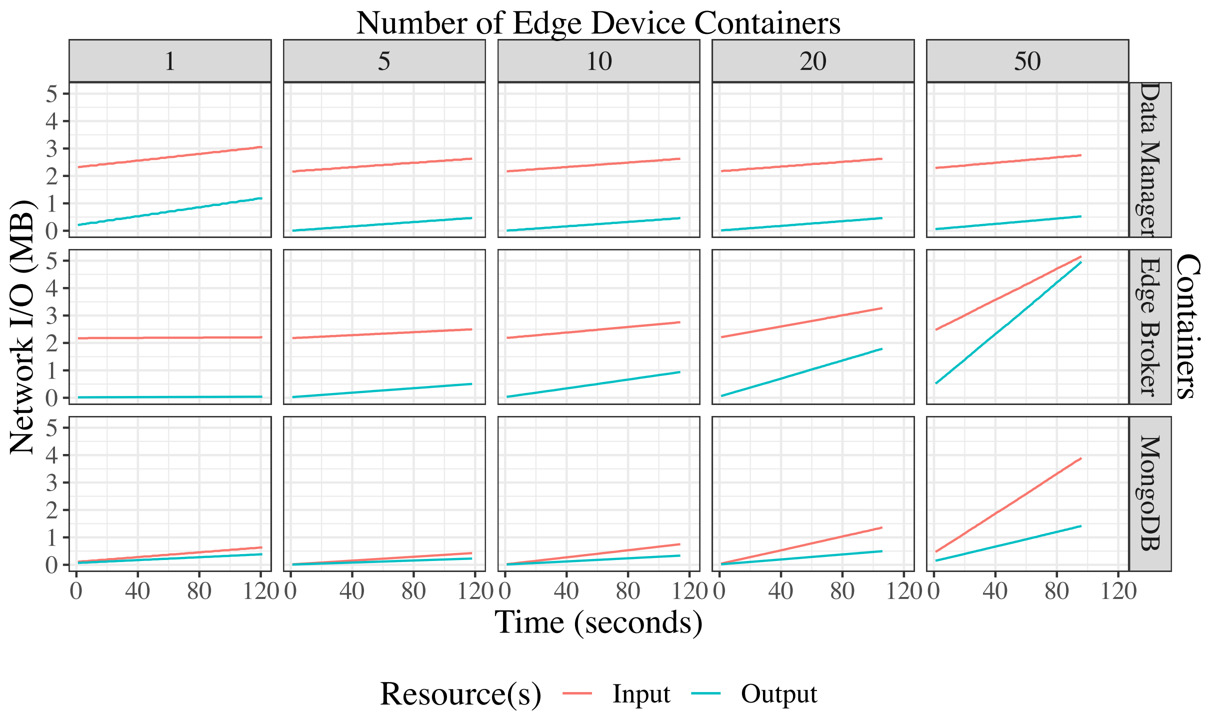

Figure 6.7 – Network input and output of the data manager, edge broker, and MongoDB containers (rows), considering the increase from 1 to 50 edge devices containers (columns).

the broker. For the Data Manager and MongoDB containers, the input network consumption turned out to be higher than the output. In that case, as shown in Figure 5.2, the database, represented here as the MongoDB instance, does not propagate the data from the edge. For the data manager instance, the input represents the traffic that comes from all the Edge Devices passing through the Edge Broker. An output bandwidth lower than the input demonstrates how the data volume that is sent to the cloud container is considerably lower than the real traffic at the edge, which is one of the main goals of an edge architecture. These results point out how the application would perform and may identify possible bottlenecks in resource consumption. For the observed use case, the Edge Broker and MongoDB are shown as the main bottlenecks, and the behavior is dictated by the volume created by the number of Edge Devices instances. Therefore, the results achieved with fifty Edge Devices indicate that the application is performing until now, but from these results of this experiment only we cannot determine if the application would scale if the vineyard had a more expressive number of edge devices.

Regarding processing utilization, as shown in Figure 6.8, the observed behavior is similar to what was discussed in the network analysis. With the increase in the number of edge device instances, there is, as expected, also an increase in CPU usage. However, this behavior is restricted to the containers that exercise some computation (not significant), being the mean utilization never higher than 15%. Therefore, we might observe a tendency

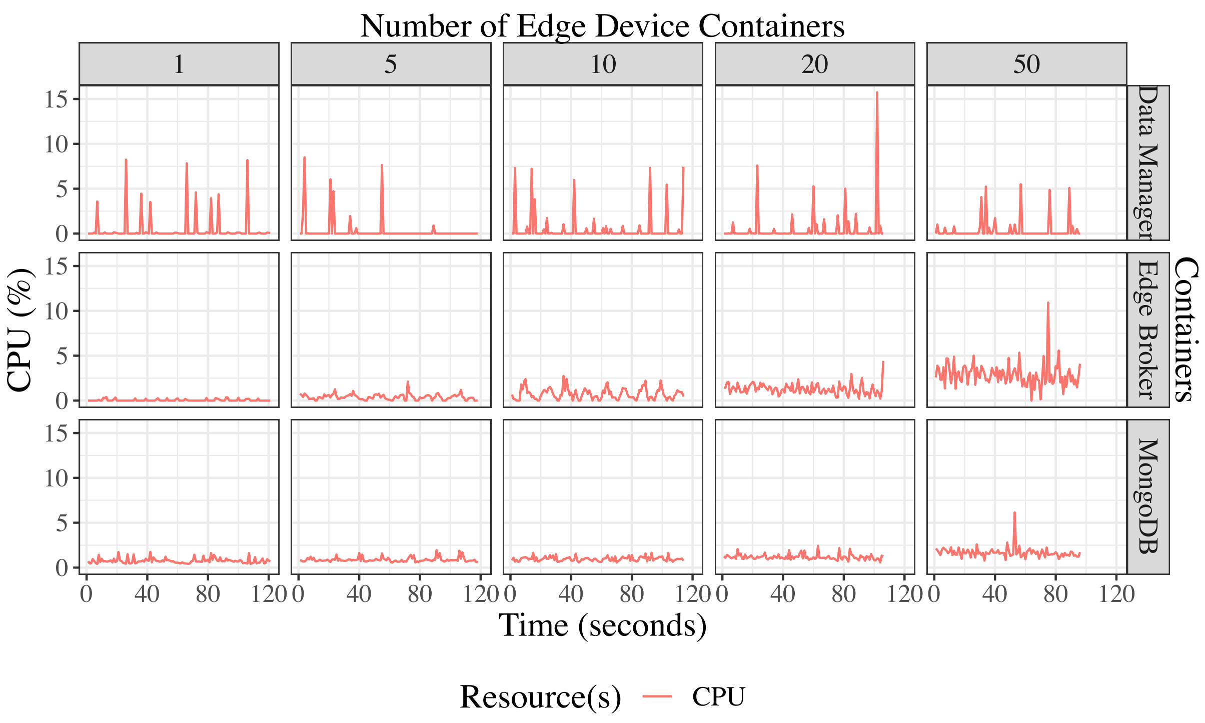

Figure 6.8 – Percentage of the CPU consumption for the data manager, edge broker, and MongoDB containers (rows), considering the increase from 1 to 50 edge devices containers (columns).

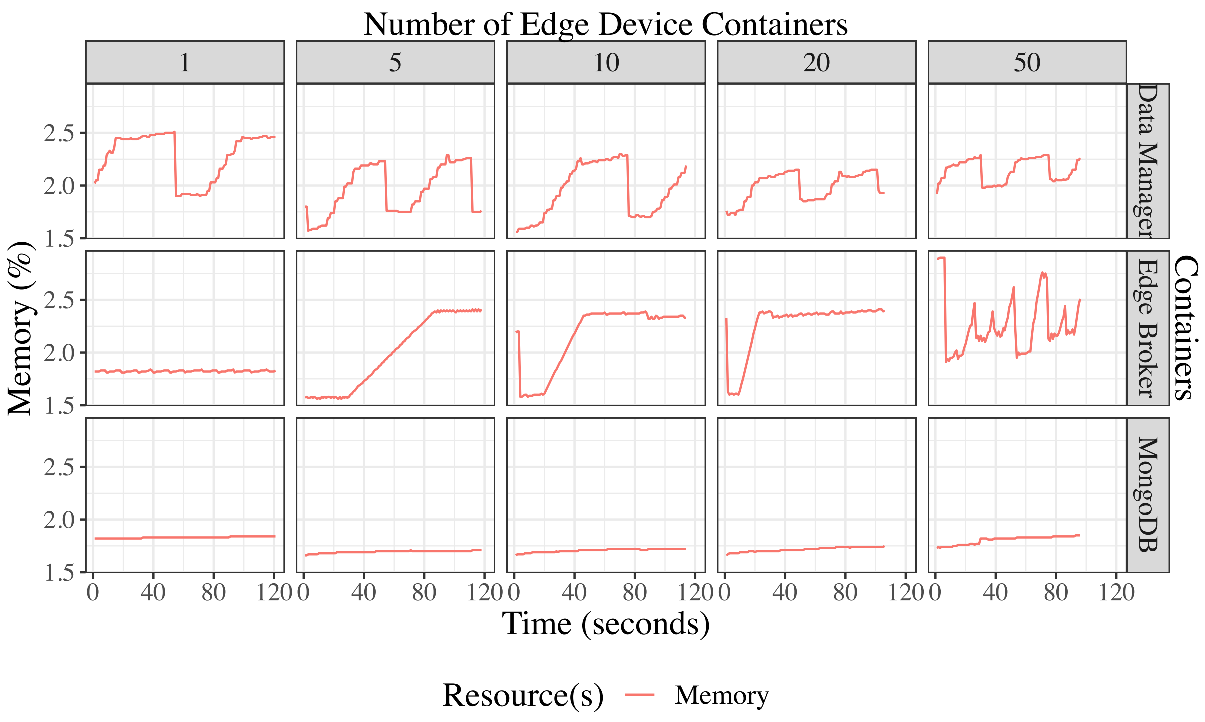

Figure 6.9 – Percentage of the memory consumption according to the max of 3.44GB pre-defined for the data manager, edge broker, and MongoDB containers (rows) considering the increase from 1 to 50 edge devices containers (columns).

to increase the resource consumption of the edge broker container that does not happen to the data manager in the same way. Since the former routes all messages and the latter keeps executing the same function. Despite the higher data volume, it does not result in much more CPU consumption. The same happens to the MongoDB instance. Regarding the memory consumption presented in Figure 6.9, there is also an increased consumption when more edge devices are used, not representing a significant value since it never goes over 3%. Therefore, based on this scenario, we can assume that memory requirements for a real application can be lower than the parameters we predefined. This result can be used to save money when buying edge devices for the real system.

6.2.1 Scalability Experimentation: Benchmark of the Smart Agriculture Use Case

The experiment presented in the previous section demonstrated how our proposal can be used to reproduce real-world scenarios in digitalized environments that resemble specific conditions, thereby becoming a strong ally during the development and testing phases of a system. Furthermore, due to the modularity of our architecture, validated in the prototype implementation, new nodes can be added to the emulator, allowing our proposal to become specialized for specific requirements. In addition to the modularity, scalability was also a considered aspect of this work. Therefore, in this session, we present a new experiment, reproducing the smart agriculture use case and running the prototype implementation for five minutes over the same testbed, containing modifications that affect the scale aspects of the previous experiment. The main variation is the number of Edge Devices, with iterations changing to 100, 200, 300, 400, 500. Since the emulator prototype creates an environment where the instances communicate with each other respecting a publishing pattern, an increase of that scale in the number of Edge Devices instances represents a much higher number of message exchanges which pushes the system behavior to its limits.

The goal of this experiment was to push the containers to their limits, understanding the behavior of nodes that had previously shown themselves as bottlenecks in a more extreme situation regarding resource consumption. With the variation increasing from 100 to 500 instances, Figure 6.10 represents the network consumption of three bottleneck containers alike the previous experiment, followed by Figure 6.11 representing the CPU consumption of the Edge Broker and Data Manager instances. The CPU consumption of the MongoDB instances is represented apart from the others in Figure 6.12. Figure 6.13 presents the memory consumption.

Considering the components of the edge block presented in the Architecture of Figure 5.2, what the experiment determines is how many edge devices are supported by the emulated environment without outages. Regarding the Edge Broker instances, the resource consumption of the observed behavior demonstrates that the maximum availability

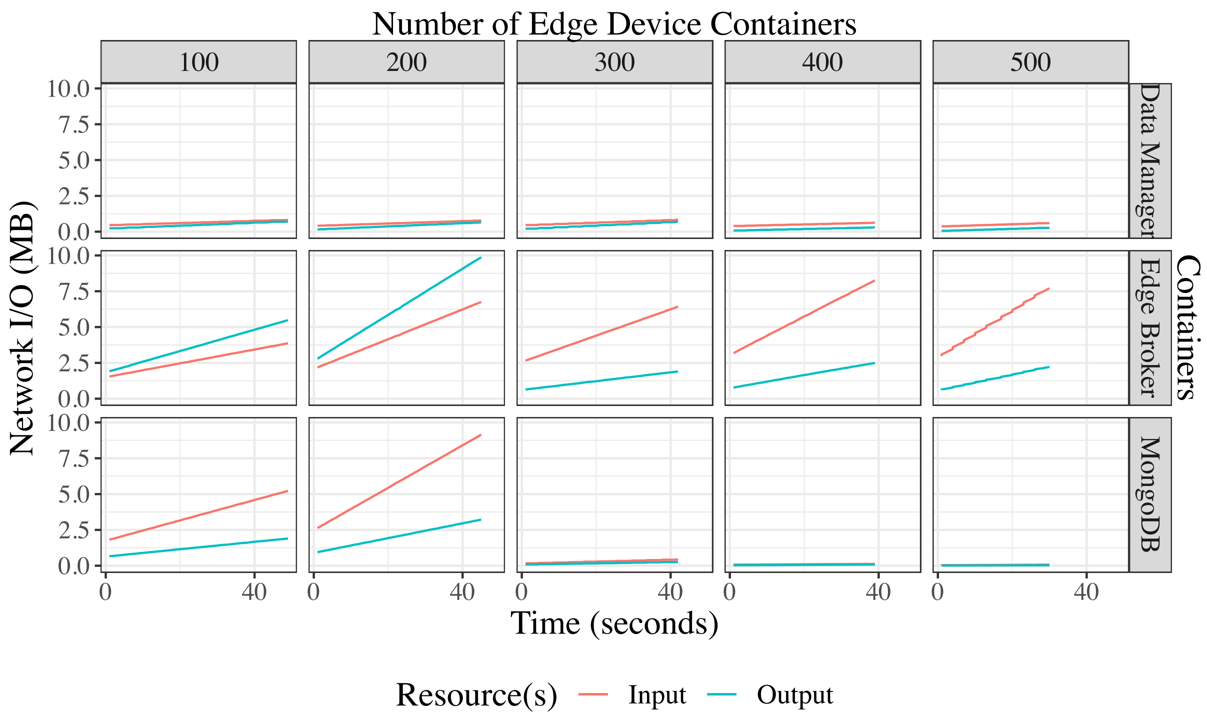

Figure 6.10 – Network input and output of the data manager, edge broker, and MongoDB containers (rows), considering the increase from 100 to 500 edge devices containers (columns).

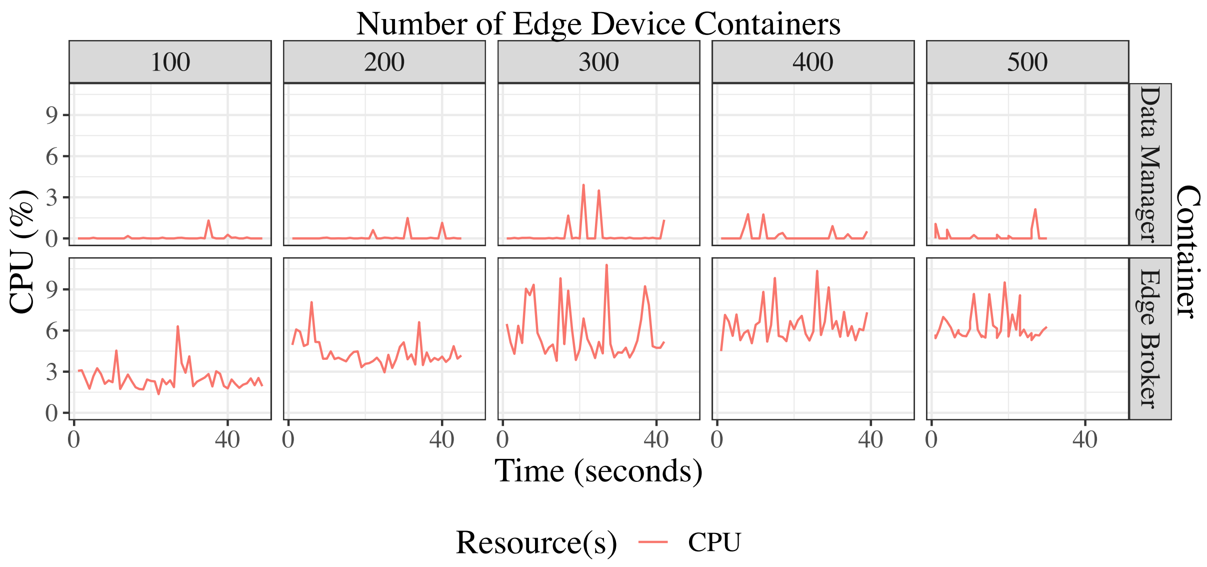

Figure 6.11 – Percentage of the CPU consumption for the data manager and edge broker containers (rows), considering the increase from 100 to 500 edge devices containers (columns).

for orchestrating messages between publishers and subscribers for the configuration is determined by the number of publishers. In a cross-analysis of the data presented in the

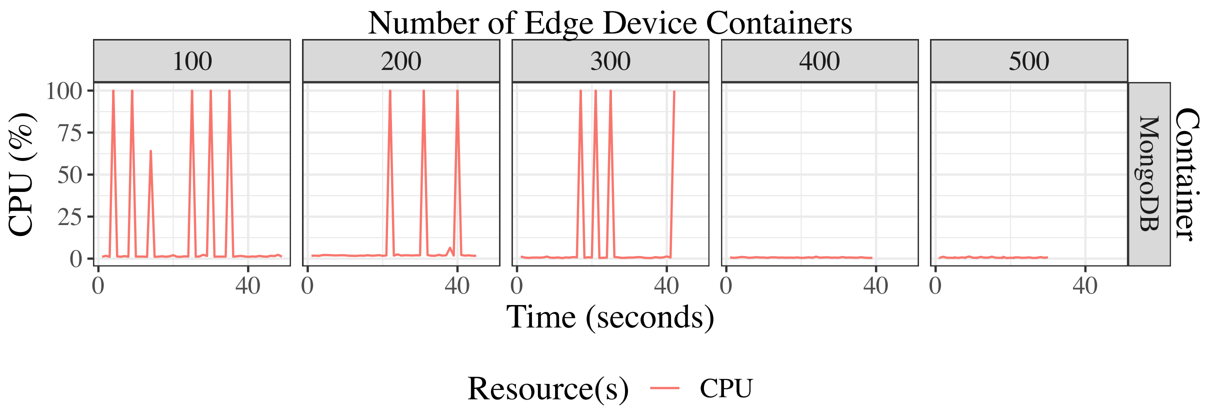

Figure 6.12 – Percentage of the CPU consumption for the MongoDB container (rows), considering the increase from 100 to 500 edge devices containers (columns).

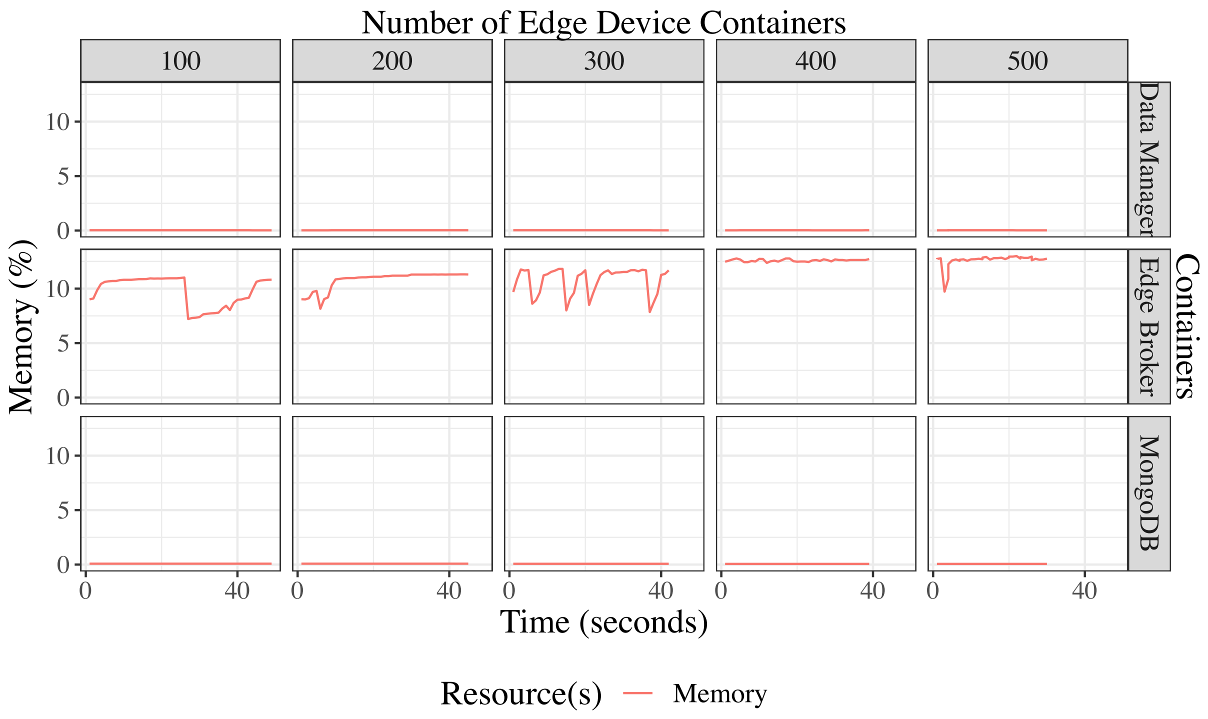

Figure 6.13 – Percentage of the memory consumption according to the max of 3.44GB predefined for the data manager, edge broker, and MongoDB containers (rows) considering the increase from 100 to 500 edge devices containers (columns).

images, as well as a log observation during the experimentation, a variation in the behavior pattern can be spotted between the phases of 200 to 300 Edge Device instances.

From the data we collected, the 300 stage is the point where we observe that the MongoDB instance stopped responding to requests. The CPU consumption of this container, as shown in Figure 6.12, presents spikes of 100%, unlike the other instances that have a considerably low average, as presented in Figure 6.11. Those spikes appeared since the first stage of the experiment. However, the buffering problem appears only in the mentioned threshold, when the volume of messages floods the container interactions. Therefore, the Edge Broker instance while trying to communicate with the database starts to reach a buffering timed out after 10000ms. It is also possible to observe that network usage is inverted compared to the two previous testing stages. In this case, for the Edge Broker container, the volume of incoming data becomes greater than the output, indicating that the container loses the ability to orchestrate message delivery to subscribers, as well as to persist data in the database after this point. Regarding the network metrics for the MongoDB instance, it is possible to observe that the input data is higher than the output in the 100 and 200 phases, and close to zero from 300 to 500. A representation of this threshold is presented in Table 6.3.

Table 6.3 – Edge broker, data manager, and MongoDB containers behavior considering the increase from 100 to 500 edge devices containers

	Edge Broker		**Data Manager**		**MongoDB**
Edge Devices	Availability	Orchestration	Availability	Summarization	Availability
100	✓	✓	✓	✓	✓
200	✓	✓	✓	✓	✓
300	✓	x	✓	x	x
400	✓	x	✓	x	x
500	✓	x	✓	x	x

The experiment results determine 200 as the observed maximum number of Edge Devices that the combination of a single Edge Broker and a single MongoDB instance can support on the applied conditions. The obtained results points the MongoDB instance to be the cause of the outages, which leads to the conclusion that the Edge Broker would be able to handle a higher number of devices if its operations were not as dependent on the database. These findings can be aligned with the use case we addressed. Considering that within a vineyard, the sensors will be scattered across the entire territory relying on many edge brokers geographically close to the sensors and the overall resource consumption of the testbed was considerably low, it can be assumed that if we take 200 as the limit, to reach numbers like 1000 Edge Devices, it would be sufficient to instantiate only 5 instances of Edge Brokers and MongoDB, distributing the Edge Devices among them. This would effectively increase the scale of the created environment.

Therefore, we consider this experiment, to successfully benchmark the scalability of our prototype implementation and proposed architecture. This scalability experiment alludes to the architecture objectives mentioned in Chapter 5.2, showing that emulating edge computing applications in clustered environments allows the replication of specific conditions found in real-world scenarios and that analyzing the performance dynamics and resource utilization patterns of a digitized edge emulator environment also leads to the understanding of real-world application behavior.

7. CONCLUSION

In the course of our work, we presented a systematic mapping of 219 papers on edge computing in the context of the IoT and industry 4.0 which provides a comprehensive overview of the current state-of-the-art. Our research endorses the prospective utilization of digital environments, specifically emphasizing the merits of edge application development. Emphasizing the preference for emulation over simulation, our study underscores the distinct advantages that emulation brings to digital edge environments, aligning effectively with the unique requirements of IoT applications and enhancing realism and efficiency in testing and development processes. The proposed architecture for edge emulation is designed to create realistic edge applications within clustered environments. This architecture offers flexibility, enabling the evaluation of scalability in IoT scenarios by replicating complex real-world conditions. Through this approach, a comprehensive understanding of system behaviors in edge environments is fostered. The prototype implementation of this architecture demonstrates its modular and easily expandable nature, facilitating the design, experimentation, and testing of edge environments. Notably, the integration of pre-programmed popular IoT protocols and standards, such as MQTT, enhances the tool's applicability and adaptability.

In conclusion, we present an emulation architecture and its prototype, a versatile emulator tool focused on the scalability evaluation of edge environments. Furthermore, we present a performance analysis that evidences the potential usage of the proposed tool. Our validation efforts included a case study in the domain of smart agriculture, where the developed emulation tool was applied to assess system scalability. In these analyses, we conduct two experiments, where in the first we emulate up to fifty Edge Devices showing that the vineyard use case was scalable through the experiments, and the second experiment points out that the current state supports in a single Edge Broker and MongoDB instance the maximum number of 200 Edge Devices. Other configurations with more IoT devices might be easily tested by changing the emulator parameters if the host machine has sufficient resources for all deployed containers.

The results yielded accurate metrics for resource utilization, traffic control, and other general resources. This empirical validation demonstrates the practical utility and effectiveness of our emulation tool in real-world scenarios. The proposed architecture and its prototype implementation offer a valuable tool for researchers and practitioners, facilitating the design and evaluation of scalable edge environments. The successful validation in the context of smart agriculture further establishes the applicability and reliability of our approach, paving the way for advancements in the development and testing of edge applications for IoT and industry 4.0. In the course of our work, the results benchmark the scalability aspects of our proposal by experimenting with a real-world use case, answering the objectives observed in the early stages of research. This work demonstrates that emulating edge computing applications in clustered environments allows the replication of specific conditions and may serve to analyze performance dynamics to achieve a better understanding of real-world application behavior. In summary, our research contributes to the field of edge computing by advocating for the adoption of edge emulation in digitalized environments for IoT applications. We strongly believe it might become an ally in the evaluation of IoT research, serving as an aid in the development and testing of the application response on different scenarios.

7.1 Publications

During the Ph. D. course, the following research results were published:

- MEYER, V.; KIRCHOFF, D.; DA SILVA, M.; DE ROSE, C. A. F.; An Interference-Aware Application Classifier Based on Machine Learning to Improve Scheduling in Clouds. In: 28th Euromicro International Conference on Parallel, Distributed and NetworkBased Processing (PDP 2020), Västerås, Suécia, p. 80. (QUALIS - A3).
- DA SILVA, M.; MEYER, V.; KIRCHOFF, D.; NETO, J.; VIEIRA, R.; DE ROSE, C. A. F.; Evaluating the performance and improving the usability of parallel and distributed Word Embeddings tools. In: 28th Euromicro International Conference on Parallel, Distributed and NetworkBased Processing (PDP 2020), Västerås, Suécia, p. 201. (QUALIS - A3).
- MEYER, V.; KIRCHOFF, D.; DA SILVA, M.; DE ROSE, C. A. F.; ML-driven Classification Scheme for Dynamic Interference-aware Resource Scheduling in Cloud Infrastructures. Journal of Systems Architecture (JSA), 2021, Elsevier. (QUALIS - A2).
- MEYER, V.; DA SILVA, M.; KIRCHOFF, D.; DE ROSE, C. A. F.; IADA: A dynamic interference-aware cloud scheduling architecture for latency-sensitive workloads, Journal of Systems and Software, 2022, Elsevier. (QUALIS - A1).

Additionally, we have the following research papers accepted:

- DA SILVA, M.; MEYER, V.; KIRCHOFF, D.; DE ROSE, C. A. F.; Edge Computing and its Boundaries to IoT and Industry 4.0: A Systematic Mapping Study, International Journal of Grid and Utility Computing (IJGUC), 2021, Elsevier. (QUALIS - A4).
- DA SILVA, M.; GONÇALVES, H.; MEYER, V.; HESSEL, F.; DE ROSE C. A. F.; A versatile emulator tool to evaluate the scalability of Edge architectures in IoT scenarios. In: 9th IEEE World Forum on Internet of Things (WF-IoT 2023), Aveiro, Portugal. (QUALIS A2).

7.2 Future Directions

Even though this thesis marks the end of a cycle of our research, the work presented here opens a new path in our research group, and for that reason, we believe that this work can serve as a foundation for others. We see the potential for exploring topics that we were unable to cover in our study, by adding different communication protocols, such as LoRa, SigFox, and NB-IoT, and also exploring the digital twins approach, to allow the connection of real edge devices, fiscally deployed and integrating with the proposed tool, which already has ongoing works considering the vineyard use case.

As presented in this work, determining which part of a system should be processed at the edge has the potential to reduce costs. Therefore, the edge emulation applied to that cause is a field that can be roughly explored. We believe that the billing aspect is a topic that could receive more attention by generating metrics and analytics considering the resource consumption results provided by our emulation tool. Furthermore, we believe our work could be applied to other areas despite smart agriculture. One of them is federated learning, which focuses on training machine learning models at the edge, also dealing with problems in the development phase due to the necessity of having many clients connected at the edge to generate individual models.

REFERENCES

[1] J. Wu and W. Zhao, “Design and realization of winternet: From net of things to internet of things,” *ACM Transactions on Cyber-Physical Systems*, vol. 1, no. 1, pp. 1–12, 2016.

[2] D. Linthicum, “Responsive data architecture for the internet of things,” *Computer*, vol. 49, no. 10, pp. 72–75, 2016.

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[4] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, “A survey on the edge computing for the internet of things,” *IEEE access*, vol. 6, pp. 6900–6919, 2017.

[5] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab, “Edge computing enabling the internet of things,” in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pp. 603–608, IEEE, 2015.

[6] V. Meyer, D. F. Kirchoff, M. L. Da Silva, and C. A. F. De Rose, “An interference-aware application classifier based on machine learning to improve scheduling in clouds,” in *28th Euromicro Int. Conf. on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 80–87, 2020.

[7] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[8] I. Halachmi, M. Guarino, J. Bewley, and M. Pastell, “Smart animal agriculture: application of real-time sensors to improve animal well-being and production,” *Annual review of animal biosciences*, vol. 7, pp. 403–425, 2019.

[9] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, and X. Wang, “Internet of things for the future of smart agriculture: A comprehensive survey of emerging technologies,” *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 4, pp. 718–752, 2021.

[10] R. K. Singh, R. Berkvens, and M. Weyn, “Agrifusion: An architecture for iot and emerging technologies based on a precision agriculture survey,” *IEEE Access*, vol. 9, pp. 136253–136283, 2021.

[11] R. N. Calheiros, R. Ranjan, C. A. De Rose, and R. Buyya, “Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services,” *arXiv preprint arXiv:0903.2525*, 2009.

[12] P. Mell, T. Grance, *et al.*, “The nist definition of cloud computing,” 2011.

[13] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[14] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pp. 27–33, 2010.

[15] T. Diaby and B. B. Rad, "Cloud computing: a review of the concepts and deployment models," *International Journal of Information Technology and Computer Science*, vol. 9, no. 6, pp. 50–58, 2017.

[16] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson, "Cost-benefit analysis of cloud computing versus desktop grids," in *2009 IEEE International Symposium on Parallel Distributed Processing*, pp. 1–12, 2009.

[17] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2011.

[18] X. R. Luo, W. Zhang, R. Bose, H. Li, and Q. B. Chung, "Producing competitive advantage from an infrastructure technology: The case of cloud computing," *Information Systems Management*, vol. 35, no. 2, pp. 147–160, 2018.

[19] K. Xu, Y. Wang, L. Yang, Y. Wang, B. Qiao, S. Qin, Y. Xu, H. Zhang, and H. Qu, "Clouddet: Interactive visual analysis of anomalous performances in cloud computing systems," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1107–1117, 2020.

[20] C. Lu, K. Ye, G. Xu, C. Z. Xu, and T. Bai, "Imbalance in the cloud: An analysis on alibaba cluster trace," *Proc. - 2017 IEEE Int. Conf. Big Data, Big Data 2017*, vol. 2018-Janua, pp. 2884–2892, 2017.

[21] V. Nguyen and T. Dang, "Cloudtraceviz: A visualization tool for tracing dynamic usage of cloud computing resources," *Proc. DAAC 2019 3rd Ind. Jt. Int. Work. Data-Center Autom. Anal. Control - Held conjunction with SC 2019 Int. Conf. High Perform. Comput. Networking, Storage Analy*, pp. 1–6, 2019.

[22] M. A. A. da Cruz, J. J. P. C. Rodrigues, J. Al-Muhtadi, V. V. Korotaev, and V. H. C. de Albuquerque, "A reference model for internet of things middleware," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 871–883, 2018.

[23] S. Li, L. Da Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[24] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, “Internet of things (iot): Research, simulators, and testbeds,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1637–1647, 2018.

[25] Y. Lu, “Industry 4.0: A survey on technologies, applications and open research issues,” *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, 2017.

[26] K. Zhou, T. Liu, and L. Zhou, “Industry 4.0: Towards future industrial opportunities and challenges,” in *2015 12th International conference on fuzzy systems and knowledge discovery (FSKD)*, pp. 2147–2152, IEEE, 2015.

[27] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, “Edge computing: A survey,” *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[28] M. Satyanarayanan, “The emergence of edge computing,” *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[29] B. M. Campbell, P. Thornton, R. Zougmoré, P. Van Asten, and L. Lipper, “Sustainable intensification: What is its role in climate smart agriculture?,” *Current Opinion in Environmental Sustainability*, vol. 8, pp. 39–43, 2014.

[30] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, “A survey on the role of iot in agriculture for the implementation of smart farming,” *Ieee Access*, vol. 7, pp. 156237–156271, 2019.

[31] B. B. Sinha and R. Dhanalakshmi, “Recent advancements and challenges of internet of things in smart agriculture: A survey,” *Future Generation Computer Systems*, vol. 126, pp. 169–184, 2022.

[32] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software engineering,” in *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, pp. 1–10, 2008.

[33] M. O. Ozcan, F. Odaci, and I. Ari, “Remote debugging for containerized applications in edge computing environments,” in *2019 IEEE International Conference on Edge Computing (EDGE)*, pp. 30–32, 2019.

[34] S. K. Datta and C. Bonnet, “Mec and iot based automatic agent reconfiguration in industry 4.0,” in *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1–5, 2018.

[35] G. Signoretti, M. Silva, A. Dias, I. Silva, D. Silva, and P. Ferrari, “Performance evaluation of an edge obd-ii device for industry 4.0,” in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT)*, pp. 432–437, 2019.

[36] N. Dao, Y. Lee, S. Cho, E. Kim, K. Chung, and C. Keum, “Multi-tier multi-access edge computing: The role for the fourth industrial revolution,” in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1280–1282, 2017.

[37] M. Ramadan, “Industry 4.0: Development of smart sunroof ambient light manufacturing system for automotive industry,” in *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, pp. 1–5, 2019.

[38] C. Shi, Z. Ren, K. Yang, C. Chen, H. Zhang, Y. Xiao, and X. Hou, “Ultra-low latency cloud-fog computing for industrial internet of things,” in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2018.

[39] P. Ferrari, S. Rinaldi, E. Sisinni, F. Colombo, F. Ghelfi, D. Maffei, and M. Malara, “Performance evaluation of full-cloud and edge-cloud architectures for industrial iot anomaly detection based on deep learning,” in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT)*, pp. 420–425, 2019.

[40] S. Trinks and C. Felden, “Edge computing architecture to support real time analytic applications : A state-of-the-art within the application area of smart factory and industry 4.0,” in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 2930–2939, 2018.

[41] S. Kulkarni and A. Guha, “Industrial internet of things (iot) computational architectures and novel neural network for smart computing,” in *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 925–930, 2019.

[42] R. Hill, J. Devitt, A. Anjum, and M. Ali, “Towards in-transit analytics for industry 4.0,” in *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 810–817, 2017.

[43] A. Buzachis, A. Galletta, A. Celesti, L. Carnevale, and M. Villari, “Towards osmotic computing: a blue-green strategy for the fast re-deployment of microservices,” in *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2019.

[44] E. Gyamfi, J. A. Ansere, and L. Xu, “Ecc based lightweight cybersecurity solution for iot networks utilising multi-access mobile edge computing,” in *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 149–154, 2019.

[45] A. Zomaya, “Keynote 2: Resource management in edge computing: Opportunities and open issues,” in *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–1, 2019.

[46] H. Rahimi, A. Zibaeenejad, and A. A. Safavi, “A novel iot architecture based on 5g-iot and next generation technologies,” in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 81–88, 2018.

[47] A. R. M. Forkan, F. Montori, D. Georgakopoulos, P. P. Jayaraman, A. Yavari, and A. Morshed, “An industrial iot solution for evaluating workers’ performance via activity recognition,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1393–1403, 2019.

[48] I. Bedhief, L. Foschini, P. Bellavista, M. Kassar, and T. Aguili, “Toward self-adaptive software defined fog networking architecture for iiot and industry 4.0,” in *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–5, 2019.

[49] R. Hamzeh, R. Zhong, X. W. Xu, E. Kajáti, and I. Zolotova, “A technology selection framework for manufacturing companies in the context of industry 4.0,” in *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, pp. 267–276, 2018.

[50] C. Um, J. Lee, and J. Jeong, “Industrial device monitoring and control system based on onem2m for edge computing,” in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1528–1533, 2018.

[51] A. Damljanovic and J. M. Lanza-Gutierrez, “An embedded cascade svm approach for face detection in the iot edge layer,” in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, pp. 2809–2814, 2018.

[52] G. Bloom, B. Alsulami, E. Nwafor, and I. C. Bertolotti, “Design patterns for the industrial internet of things,” in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, pp. 1–10, 2018.

[53] I. D. S. Batista, I. M. Sardina, and R. R. Dantas, “Monitoring restaurants in real-time,” in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT)*, pp. 202–206, 2019.

[54] A. Badar, D. Z. Lou, U. Graf, C. Barth, and C. Stich, “Intelligent edge control with deterministic-ip based industrial communication in process automation,” in *2019 15th International Conference on Network and Service Management (CNSM)*, pp. 1–7, 2019.

[55] M. Al Yami and D. Schaefer, “Fog computing as a complementary approach to cloud computing,” in *2019 International Conference on Computer and Information Sciences (ICCIS)*, pp. 1–5, 2019.

[56] M. Ghazi Vakili, C. Demartini, M. Guerrera, and B. Montrucchio, "Open source fog architecture for industrial iot automation based on industrial protocols," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 570–578, 2019.

[57] H. Derhamy, M. Andersson, J. Eliasson, and J. Delsing, "Workflow management for edge driven manufacturing systems," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pp. 774–779, 2018.

[58] T. P. Raptis, A. Passarella, and M. Conti, "Maximizing industrial iot network lifetime under latency constraints through edge data distribution," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pp. 708–713, 2018.

[59] T. Kumar, A. Braeken, V. Ramani, I. Ahmad, E. Harjula, and M. Ylianttila, "Sec-blockedge: Security threats in blockchain-edge based industrial iot networks," in *2019 11th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pp. 1–7, 2019.

[60] D. W. McKee, S. J. Clement, J. Almutairi, and J. Xu, "Massive-scale automation in cyber-physical systems: Vision challenges," in *2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS)*, pp. 5–11, 2017.

[61] S. Alharthi, P. Johnson, M. Alharthi, and C. Jose, "Iot/cps ecosystem for efficient electricity consumption : Invited paper," in *2019 Tenth International Green and Sustainable Computing Conference (IGSC)*, pp. 1–7, 2019.

[62] M. V. Souza Costa, V. B. Souza, and S. S. Araújo Júnior, "Dynamic control-as-a-service provisioning in fog computing," in *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 1–6, 2019.

[63] G. Lee, W. Saad, and M. Bennis, "Online optimization for uav-assisted distributed fog computing in smart factories of industry 4.0," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2018.

[64] C. Yang, S. Lan, W. Shen, G. Q. Huang, and L. Wang, "Software-defined cloud manufacturing in the context of industry 4.0," in *2019 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, pp. 184–190, 2019.

[65] L. P. I. Ledwaba, G. P. Hancke, S. J. Isaac, and H. S. Venter, "Developing a secure, smart microgrid energy market using distributed ledger technologies," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, pp. 1725–1728, 2019.

[66] N. Sharma and C. Tham, “Predictive maintenance using gpu-accelerated partially observable markov decision process,” in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 769–773, 2019.

[67] M. W. Condry and C. B. Nelson, “Using smart edge iot devices for safer, rapid response with industry iot control operations,” *Proceedings of the IEEE*, vol. 104, no. 5, pp. 938–946, 2016.

[68] X. Li, D. Li, J. Wan, C. Liu, and M. Imran, “Adaptive transmission optimization in sdn-based industrial internet of things with edge computing,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1351–1360, 2018.

[69] P. Pace, G. Aloi, R. Gravina, G. Caliciuri, G. Fortino, and A. Liotta, “An edge-based architecture to support efficient applications for healthcare industry 4.0,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 481–489, 2019.

[70] C. Lin and J. Yang, “Cost-efficient deployment of fog computing systems at logistics centers in industry 4.0,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4603–4611, 2018.

[71] C. Alcaraz, G. Bernieri, F. Pascucci, J. Lopez, and R. Setola, “Covert channels-based stealth attacks in industry 4.0,” *IEEE Systems Journal*, vol. 13, no. 4, pp. 3980–3988, 2019.

[72] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, “Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study,” *IEEE Access*, vol. 5, pp. 9882–9910, 2017.

[73] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, “Industrial iot data scheduling based on hierarchical fog computing: A key for enabling smart factory,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4590–4602, 2018.

[74] C. Paniagua, J. Eliasson, and J. Delsing, “Efficient device-to-device service invocation using arrowhead orchestration,” *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 429–439, 2020.

[75] P. K. Illa and N. Padhi, “Practical guide to smart factory transition using iot, big data and edge analytics,” *IEEE Access*, vol. 6, pp. 55162–55170, 2018.

[76] R. W. L. Coutinho and A. Boukerche, “Modeling and analysis of a shared edge caching system for connected cars and industrial iot-based applications,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2003–2012, 2020.

[77] H. Tang, D. Li, J. Wan, M. Imran, and M. Shoaib, “A reconfigurable method for intelligent manufacturing based on industrial cloud and edge intelligence,” *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4248–4259, 2020.

[78] W. Wang, L. Fan, P. Huang, and H. Li, “A new data processing architecture for multi-scenario applications in aviation manufacturing,” *IEEE Access*, vol. 7, pp. 83637–83650, 2019.

[79] N. G. Nayak, F. Dürr, and K. Rothermel, “Incremental flow scheduling and routing in time-sensitive software-defined networks,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2066–2075, 2018.

[80] R. Mahmud, A. N. Toosi, K. Rao, and R. Buyya, “Context-aware placement of industry 4.0 applications in fog computing environments,” *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[81] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, “Multi-agent deep reinforcement learning for joint multi-channel access and task offloading of mobile edge computing in industry 4.0,” *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[82] C. Jiang, J. Wan, and H. Abbas, “An edge computing node deployment method based on improved k-means clustering algorithm for smart manufacturing,” *IEEE Systems Journal*, pp. 1–11, 2020.

[83] Y. Zhan, J. Zhang, P. Li, and Y. Xia, “Crowdtraining: Architecture and incentive mechanism for deep learning training in the internet of things,” *IEEE Network*, vol. 33, no. 5, pp. 89–95, 2019.

[84] B. Chen, J. Wan, Y. Lan, M. Imran, D. Li, and N. Guizani, “Improving cognitive ability of edge intelligent iiot through machine learning,” *IEEE Network*, vol. 33, no. 5, pp. 61–67, 2019.

[85] A. P. Fournaris, C. Alexakos, C. Anagnostopoulos, C. Koulamas, and A. Kalogeras, “Introducing hardware-based intelligence and reconfigurability on industrial iot edge nodes,” *IEEE Design Test*, vol. 36, no. 4, pp. 15–23, 2019.

[86] R. Munoz, J. Mangues-Bafalluy, R. Vilalta, C. Verikoukis, J. Alonso-Zarate, N. Bartzoudis, A. Georgiadis, M. Payaro, A. Perez-Neira, R. Casellas, R. Martinez, J. Nunez-Martinez, M. Requena Esteso, D. Pubill, O. Font-Bach, P. Henarejos, J. Serra, and F. Vazquez-Gallego, “The cttc 5g end-to-end experimental platform : Integrating heterogeneous wireless/optical networks, distributed cloud, and iot devices,” *IEEE Vehicular Technology Magazine*, vol. 11, no. 1, pp. 50–63, 2016.

[87] Y. Xia, “Resource scheduling for piano teaching system of internet of things based on mobile edge computing,” *Computer Communications*, vol. 158, pp. 73 – 84, 2020.

[88] L. Beňo, R. Pribiš, and R. Leskovský, “Processing data from opc ua server by using edge and cloud computing,” *IFAC-PapersOnLine*, vol. 52, no. 27, pp. 240 – 245,

2019. 16th IFAC Conference on Programmable Devices and Embedded Systems PDES 2019.

[89] A. Celesti, M. Fazio, A. Galletta, L. Carnevale, J. Wan, and M. Villari, "An approach for the secure management of hybrid cloud–edge environments," *Future Generation Computer Systems*, vol. 90, pp. 1 – 19, 2019.

[90] L. Hu, Y. Miao, G. Wu, M. M. Hassan, and I. Humar, "irobot-factory: An intelligent robot factory based on cognitive manufacturing and edge computing," *Future Generation Computer Systems*, vol. 90, pp. 569 – 577, 2019.

[91] C. Catal and B. Tekinerdogan, "Aligning education for the life sciences domain to support digitalization and industry 4.0," *Procedia Computer Science*, vol. 158, pp. 99 – 106, 2019.

[92] G. Jo, S.-H. Jang, and J. Jeong, "Design and implementation of cpps and edge computing architecture based on opc ua server," *Procedia Computer Science*, vol. 155, pp. 97 – 104, 2019. The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2019),The 14th International Conference on Future Networks and Communications (FNC-2019),The 9th International Conference on Sustainable Energy Information Technology.

[93] Y.-J. Lin, C.-F. Tan, and C.-Y. Huang, "Integration of logic controller with iot to form a manufacturing edge computing environment: A premise," *Procedia Manufacturing*, vol. 39, pp. 398 – 405, 2019. 25th International Conference on Production Research Manufacturing Innovation: Cyber Physical Manufacturing August 9-14, 2019 | Chicago, Illinois (USA).

[94] D. Fernández-Cerero, A. Fernández-Montes, F. J. Ortega], A. Jakóbik, and A. Widlak, "Sphere: Simulator of edge infrastructures for the optimization of performance and resources energy consumption," *Simulation Modelling Practice and Theory*, vol. 101, p. 101966, 2020. Modeling and Simulation of Fog Computing.

[95] B. Mayer, D. Tantscher, and C. Bischof, "From digital shop floor to real-time reporting: an iiot based educational use case," *Procedia Manufacturing*, vol. 45, pp. 473 – 478, 2020. Learning Factories across the value chain – from innovation to service – The 10th Conference on Learning Factories 2020.

[96] B. Liu, Y. Zhang, G. Zhang, and P. Zheng, "Edge-cloud orchestration driven industrial smart product-service systems solution design based on cps and iiot," *Advanced Engineering Informatics*, vol. 42, p. 100984, 2019.

[97] J. Tang, S. Liu, J. Cao, D. Sun, B. Ding, J.-L. Gaudiot, and W. Shi, "pi-hub: Large-scale video learning, storage, and retrieval on heterogeneous hardware platforms," *Future Generation Computer Systems*, vol. 102, pp. 514 – 523, 2020.

[98] A. Buzachis, A. Celesti, A. Galletta, M. Fazio, G. Fortino, and M. Villari, “A multi-agent autonomous intersection management (ma-aim) system for smart cities leveraging edge-of-things and blockchain,” *Information Sciences*, vol. 522, pp. 148 – 163, 2020.

[99] A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, R. M. Parizi, and K.-K. R. Choo, “Fog data analytics: A taxonomy and process model,” *Journal of Network and Computer Applications*, vol. 128, pp. 90 – 104, 2019.

[100] R. S. Alonso, I. Sittón-Candanedo, Óscar García, J. Prieto, and S. Rodríguez-González, “An intelligent edge-iot platform for monitoring livestock and crops in a dairy farming scenario,” *Ad Hoc Networks*, vol. 98, p. 102047, 2020.

[101] Y.-J. Lin, Y.-H. Lin, and C.-Y. Huang, “Using cyber plc to link physical operations with cyber control decisions,” *Procedia Manufacturing*, vol. 39, pp. 294 – 299, 2019. 25th International Conference on Production Research Manufacturing Innovation: Cyber Physical Manufacturing August 9-14, 2019 | Chicago, Illinois (USA).

[102] M. Gushev, “Dew computing architecture for cyber-physical systems and iot,” *Internet of Things*, vol. 11, p. 100186, 2020.

[103] C. Zhang, G. Zhou, J. He, Z. Li, and W. Cheng, “A data- and knowledge-driven framework for digital twin manufacturing cell,” *Procedia CIRP*, vol. 83, pp. 345 – 350, 2019. 11th CIRP Conference on Industrial Product-Service Systems.

[104] T. C. Xavier, I. L. Santos, F. C. Delicato, P. F. Pires, M. P. Alves, T. S. Calmon, A. C. Oliveira, and C. L. Amorim, “Collaborative resource allocation for cloud of things systems,” *Journal of Network and Computer Applications*, vol. 159, p. 102592, 2020.

[105] M. Isaja and J. K. Soldatos, “Distributed ledger architecture for automation, analytics and simulation in industrial environments,” *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 370 – 375, 2018. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[106] J. Zietsch, L. Büth, M. Juraschek, N. Weinert, S. Thiede, and C. Herrmann, “Identifying the potential of edge computing in factories through mixed reality,” *Procedia CIRP*, vol. 81, pp. 1095 – 1100, 2019. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

[107] H. Zörrer, R. Steringer, S. Zambal, and C. Eitzinger, “Using business analytics for decision support in zero defect manufacturing of composite parts in the aerospace industry,” *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1461 – 1466, 2019. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.

[108] L. Valerio, M. Conti, and A. Passarella, "Energy efficient distributed analytics at the edge of the network for iot environments," *Pervasive and Mobile Computing*, vol. 51, pp. 27 – 42, 2018.

[109] N. Verba, K.-M. Chao, A. James, D. Goldsmith, X. Fei, and S.-D. Stan, "Platform as a service gateway for the fog of things," *Advanced Engineering Informatics*, vol. 33, pp. 243 – 257, 2017.

[110] S. Sahmim, H. Gharsellaoui, and S. Bouamama, "Edge computing: Smart identity wallet based architecture and user centric," *Procedia Computer Science*, vol. 159, pp. 1246 – 1257, 2019. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019.

[111] M. Tortonesi, M. Govoni, A. Morelli, G. Riberto, C. Stefanelli, and N. Suri, "Taming the iot data deluge: An innovative information-centric service model for fog computing applications," *Future Generation Computer Systems*, vol. 93, pp. 888 – 902, 2019.

[112] I. Sittón-Candanedo, R. S. Alonso, J. M. Corchado, S. Rodríguez-González, and R. Casado-Vara, "A review of edge computing reference architectures and a new global edge proposal," *Future Generation Computer Systems*, vol. 99, pp. 278 – 294, 2019.

[113] L. Verderame, I. Merelli, L. Morganti, E. Corni, D. Cesini, D. D'Agostino, and A. Merlo, "A secure cloud-edges computing architecture for metagenomics analysis," *Future Generation Computer Systems*, 2019.

[114] P. Petrali, M. Isaja, and J. K. Soldatos, "Edge computing and distributed ledger technologies for flexible production lines: A white-appliances industry case," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 388 – 392, 2018. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[115] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, and P. D. Drobintsev, "Trust management in a blockchain based fog computing platform with trustless smart oracles," *Future Generation Computer Systems*, vol. 101, pp. 747 – 759, 2019.

[116] I. L. Santos, L. Pirmez, F. C. Delicato, G. M. Oliveira, C. M. Farias, S. U. Khan, and A. Y. Zomaya, "Zeus: A resource allocation algorithm for the cloud of sensors," *Future Generation Computer Systems*, vol. 92, pp. 564 – 581, 2019.

[117] Z. Zhao, P. Lin, L. Shen, M. Zhang, and G. Q. Huang, "Iot edge computing-enabled collaborative tracking system for manufacturing resources in industrial park," *Advanced Engineering Informatics*, vol. 43, p. 101044, 2020.

[118] P. Zheng, Z. Wang, and C.-H. Chen, "Industrial smart product-service systems solution design via hybrid concerns," *Procedia CIRP*, vol. 83, pp. 187 – 192, 2019. 11th CIRP Conference on Industrial Product-Service Systems.

[119] E. Kajati, P. Papcun, C. Liu, R. Y. Zhong, J. Koziorek, and I. Zolotova, "Cloud based cyber-physical systems: Network evaluation study," *Advanced Engineering Informatics*, vol. 42, p. 100988, 2019.

[120] X. Li and J. Wan, "Proactive caching for edge computing-enabled industrial mobile wireless networks," *Future Generation Computer Systems*, vol. 89, pp. 89 – 97, 2018.

[121] A. Carvalho, N. O. Mahony, L. Krpalkova, S. Campbell, J. Walsh, and P. Doody, "At the edge of industry 4.0," *Procedia Computer Science*, vol. 155, pp. 276 – 281, 2019. The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2019),The 14th International Conference on Future Networks and Communications (FNC-2019),The 9th International Conference on Sustainable Energy Information Technology.

[122] D. J. Ahn and J. Jeong, "A pmipv6-based user mobility pattern scheme for sdn-defined smart factory networking," *Procedia Computer Science*, vol. 134, pp. 235 – 242, 2018. The 15th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2018) / The 13th International Conference on Future Networks and Communications (FNC-2018) / Affiliated Workshops.

[123] M. Bauer, E. Kovacs, A. Schülke, N. Ito, C. Criminisi, L.-W. Goix, and M. Valla, "The context api in the oma next generation service interface," in *2010 14th International Conference on Intelligence in Next Generation Networks*, pp. 1–5, IEEE, 2010.

[124] Q. Qi and F. Tao, "A smart manufacturing service system based on edge computing, fog computing, and cloud computing," *IEEE Access*, vol. 7, pp. 86769–86777, 2019.

[125] A. Gilchrist, *Industry 4.0: the industrial internet of things*. Springer, 2016.

[126] S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert, "Industrial internet of things and cyber manufacturing systems," in *Industrial internet of things*, pp. 3–19, Springer, 2017.

[127] R. Mayer, L. Graser, H. Gupta, E. Saurez, and U. Ramachandran, "Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures," in *2017 IEEE Fog World Congress (FWC)*, pp. 1–6, IEEE, 2017.

[128] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pp. 1–6, 2010.

[129] P. Wette, M. Dräxler, A. Schwabe, F. Wallaschek, M. H. Zahraee, and H. Karl, "Maxinet: Distributed emulation of software-defined networks," in *2014 IFIP Networking Conference*, pp. 1–9, IEEE, 2014.

[130] Y. Zeng, M. Chao, and R. Stoleru, "Emuedge: A hybrid emulator for reproducible and realistic edge computing experiments," in *2019 IEEE International Conference on Fog Computing (ICFC)*, pp. 153–164, IEEE, 2019.

[131] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srslte: An open-source platform for lte evolution and experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, pp. 25–32, 2016.

[132] C. Andrés Ramiro, C. Fiandrino, A. Blanco Pizarro, P. Jimenez Mateo, N. Ludant, and J. Widmer, "openleon: An end-to-end emulator from the edge data center to the mobile users," in *Proceedings of the 12th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, pp. 19–27, 2018.

[133] A. Coutinho, F. Greve, C. Prazeres, and J. Cardoso, "Fogbed: A rapid-prototyping emulation environment for fog computing," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2018.

[134] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[135] D. Fernández-Cerero, A. Fernández-Montes, F. J. Ortega, A. Jakóbik, and A. Widlak, "Sphere: Simulator of edge infrastructures for the optimization of performance and resources energy consumption," *Simulation Modelling Practice and Theory*, vol. 101, p. 101966, 2020.

[136] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018.

[137] T. Qayyum, A. W. Malik, M. A. K. Khattak, O. Khalid, and S. U. Khan, "Fognetsim++: A toolkit for modeling and simulation of distributed fog environment," *IEEE Access*, vol. 6, pp. 63570–63583, 2018.

[138] J. Hasenburg, S. Werner, and D. Bermbach, "Fogexplorer," in *Proceedings of the 19th International Middleware Conference (Posters)*, pp. 1–2, 2018.

[139] I. Lera, C. Guerrero, and C. Juiz, "Yafs: A simulator for iot scenarios in fog computing," *IEEE Access*, vol. 7, pp. 91745–91758, 2019.

[140] J. Byrne, S. Svorobej, A. Gourinovitch, D. M. Elango, P. Liston, P. J. Byrne, and T. Lynn, "Recap simulator: Simulation of cloud/edge/fog computing scenarios," in *2017 Winter Simulation Conference (WSC)*, pp. 4568–4569, IEEE, 2017.

[141] V. De Maio and I. Brandic, "First hop mobile offloading of dag computations," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CC-GRID)*, pp. 83–92, IEEE, 2018.

[142] C. Huttenhower, M. Schroeder, M. D. Chikina, and O. G. Troyanskaya, "The sleipnir library for computational functional genomics," *Bioinformatics*, vol. 24, no. 13, pp. 1559–1561, 2008.

[143] B. Heller, *Reproducible network research with high-fidelity emulation*. Stanford University, 2013.

[144] S. P. Ong, S. Cholia, A. Jain, M. Brafman, D. Gunter, G. Ceder, and K. A. Persson, "The materials application programming interface (api): A simple, flexible and efficient api for materials data based on representational state transfer (rest) principles," *Computational Materials Science*, vol. 97, pp. 209–215, 2015.

[145] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[146] T. Pfandzelter and D. Bermbach, "Celestial: Virtual software system testbeds for the leo edge," in *Proceedings of the 23rd ACM/IFIP International Middleware Conference*, pp. 69–81, 2022.

MARISTA PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Pesquisa e Pós-Graduação
Av. Ipiranga, 6681 – Prédio 1 – Térreo
Porto Alegre – RS – Brasil
Fone: (51) 3320-3513
E-mail: propesq@pucrs.br
Site: www.pucrs.br