SANTA CATARINA STATE UNIVERSITY – UDESC
COLLEGE OF TECHNOLOGICAL SCIENCE – CCT
GRADUATE PROGRAM IN APPLIED COMPUTING – PPGCAP

PAULO ROBERTO ALBUQUERQUE

MOBILITY-AWARE PROVISIONING OF SERVICE-COMPOSED APPLICATIONS ON CLOUD-EDGE CONTINUUM WITH QUALITY-OF-SERVICE REQUIREMENTS

JOINVILLE
2025

PAULO ROBERTO ALBUQUERQUE

MOBILITY-AWARE PROVISIONING OF SERVICE-COMPOSED APPLICATIONS ON CLOUD-EDGE CONTINUUM WITH QUALITY-OF-SERVICE REQUIREMENTS

Master's Degree Dissertation submitted to the Santa Catarina State University as a partial requirement for obtaining a Master's degree in Applied Computing.

Supervisor: Guilherme Piêgas Koslovski

JOINVILLE
2025

Ficha catalográfica elaborada pelo(a) autor(a), com
auxílio do programa de geração automática da
Biblioteca Setorial do CCT/UDESC

Albuquerque, Paulo Roberto
Mobility-aware Provisioning of Service-composed Applications on Cloud-Edge Continuum with Quality-of-Service Requirements / Paulo Roberto Albuquerque. - Joinville, 2025.
71 p. : il. ; 30 cm.

Supervisor: Guilherme Piêgas Koslovski.
Dissertação (Mestrado) - Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Mestrado em Computação Aplicada, Joinville, 2025.

1. Cloud-Edge. 2. Continuum. 3. Provisioning. 4. Mobility-Awareness. 5. Service Placement. I. Koslovski, Guilherme Piêgas . II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Mestrado em Computação Aplicada. III. Título.

PAULO ROBERTO ALBUQUERQUE

MOBILITY-AWARE PROVISIONING OF SERVICE-COMPOSED APPLICATIONS ON CLOUD-EDGE CONTINUUM WITH QUALITY-OF-SERVICE REQUIREMENTS

Master's Degree Dissertation submitted to the Santa Catarina State University as a partial requirement for obtaining a Master's degree in Applied Computing.

Supervisor: Guilherme Piêgas Koslovski

EXAMINATION COMMITTEE:

Guilherme Piêgas Koslovski, P.h.D.
Santa Catarina State University

Members:

Maurício Aronne Pillon, P.h.D.
Santa Catarina State University

Tiago Ferreto, P.h.D.
Pontifical Catholic University of Rio Grande do Sul

Joinville, February, 25th, 2025

ABSTRACT

The evolution of Cloud computing and the advent of Edge computing have paved the way to the concept of Cloud-Edge Continuum, where both of these domains are transparently available for Applications to use. However, provisioning an Application composed by many services in this Continuum can be challenging and complex due to the dynamic nature of the environment, especially considering new challenges regarding the users' mobility. This requires a careful analysis from the perspectives of users and service providers, regarding an application's requirements and Quality of Service constraints. This dissertation proposes a way to create provisioning policies based on user mobility and network requirements, while also analyzing the individual services that compose some Application, considering their needs and restrictions to better allocate them in the available infrastructure and deliver better Quality of Service. Through a series of evaluation scenarios using Cloud-Edge simulation tools, the proposal indicates that approaching Continuum scenarios with a mobility-aware perspective, by using novel service placement algorithms, can improve the Quality of Service of distributed applications. Ultimately, the present work aims to bring valid and innovative contributions to the Cloud and Edge computing field, with hybrid and continuous provisioning of distributed applications, discussing the challenges and opportunities of the Cloud-Edge Continuum, validating the proposal by using simulation tools, but also discussing how could these solutions be implemented in real-world scenarios. By using a Cloud-Edge simulation tools, in addition to a novel dataset composed of various sources, and a new Point of Interest mobility model, the present work was able to evaluate substantial improvements in overall delay metrics for a set of service-composed applications by using mobility-aware placement algorithms, when compared to traditional ones.
Keywords: Cloud-Edge. Continuum. Provisioning. Mobility-Awareness. Service Placement.

RESUMO

A evolução da computação em nuvem e o advento da computação de borda abriram caminho para o conceito de *Cloud-Edge Continuum*, em que ambos os domínios estão disponíveis de forma transparente para as aplicações utilizarem. No entanto, o provisionamento de uma aplicação composta por muitos serviços neste *Continuum* pode ser difícil e complexo devido à natureza dinâmica do ambiente, especialmente tendo em conta os novos desafios relacionados com a mobilidade dos usuários. Isto exige uma análise cuidadosa do ponto de vista dos usuários e dos provedores de serviços de computação, no que diz respeito aos requisitos de uma aplicação e às restrições da qualidade do serviço. Esta dissertação propõe uma forma de criar políticas de provisionamento baseadas na mobilidade dos usuários e nos requisitos da rede, enquanto analisa os serviços individuais que compõem uma aplicação, considerando as suas necessidades e restrições para melhor os alocar na infraestrutura disponível e proporcionar uma melhor Qualidade de Serviço. Por meio de uma série de cenários de avaliação usando ferramentas de simulação Cloud-Edge, a proposta indica que abordar cenários Continuum com uma perspectiva de mobilidade consciente, usando novos algoritmos de posicionamento de serviço, pode melhorar a Qualidade de Serviço de aplicativos distribuídos. Por fim, o presente trabalho visa trazer contribuições válidas e inovadoras para o campo de computação em Nuvem e Edge, com provisionamento híbrido e contínuo de aplicativos distribuídos, discutindo os desafios e oportunidades do Continuum Cloud-Edge, validando a proposta usando ferramentas de simulação, mas também discutindo como essas soluções podem ser implementadas em cenários do mundo real. Ao utilizar ferramentas de simulação Cloud-Edge, além de um novo conjunto de dados composto por várias fontes e um novo modelo de mobilidade de Ponto de Interesse, o presente trabalho foi capaz de avaliar melhorias substanciais nas métricas gerais de atraso para um conjunto de aplicativos compostos por serviços usando algoritmos de posicionamento com reconhecimento de mobilidade, quando comparados aos tradicionais ou de última geração.

Palavras-chave: *Cloud-Edge*. *Continuum*. Provisionamento. Consciência da Mobilidade. Posicionamento de Serviços.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Even though the beginning of Cloud Computing dates back to the last century, it truly started to take shape in the 2000s, when big companies started to invest heavily in building Data Center infrastructure to support the hosting of applications and renting this infrastructure to others in a plenitude of paradigms. As Cloud Computing gained momentum, more than just big companies started adopting it, and small to medium-sized enterprises began to realize what benefits could be gained by leveraging cloud services to suppress their computing needs. The scalability and flexibility offered by Cloud Computing, which was unmatched at the time, allowed these companies to access advanced technology and capabilities without the need for significant upfront investments in hardware and infrastructure.

Although the flexibility offered by Cloud Computing is a step-up from what came before it, it still has some flaws that come from the fundamental design of its concept. That is, hosting applications in centrally located Data Centers can introduce some hassles to users of said application that are not situated near that hosting site. Another aspect is that in Cloud Computing uprising era, applications were most commonly built as monoliths, meaning they had a single component executing all the functionalities that the application offered or supported. However, that is rarely the case in modern times, more often than not, applications are composed of big architectures of services and smaller components, each with a specific job/function. This is what we will be referring to as Service-Composed Applications in the following sections and chapters. It is a broader and more abstract concept of a microservice architecture, and each of the composing services has its own responsibility and also individual characteristics regarding all aspects of computing resources, such as processing power needs, memory profiles, networking needs, communication patterns, etc. This method of constructing applications is very beneficial to the Cloud Native way of deploying something, that is because it allows each component to scale individually according to its workload.

Most typical Data Center deployments often don't suffer from very high latencies and other problems that may arise from an end-user being very distant in relation to the computing source. Nevertheless, in certain scenarios, the real-timeliness of some components is of utmost importance, such as critical operational sensors accessing or communicating with an application through weak networks, or users accessing through mobile devices (MAHMOUDI; MOURLIN; BATTOU, 2018). The disparity between the desired conditions for applications, versus the conditions that Cloud Computing could deliver at the time, started to create a problem of sub-optimality. This phenomenon started to become known in the industry, through researchers and Cloud providers shining a light on it. The concept of Fog Computing started to emerge, proposing a solution to this exact situation, aiming to deliver a smaller, albeit still capable enough computational Cloud, closer to the Edge of the networks (BONOMI et al., 2012). This would have to be, however, a big investment from the Cloud Providers themselves, in order to build this substrate of Fog Computing infrastructure.

The cost and maintenance barriers throttled the adoption of the Fog architecture. Where it was adopted, it certainly helped bring the Quality of Service (QoS) expectations for those kinds of applications into reality, while also introducing its own sets of problems (DAS; INUWA, 2023). Fog computing was ultimately a first step into the future of distributed and serviced-composed applications. Nevertheless, those kinds of applications would still inevitably suffer from the restrictions of their substrates. Facilitated and influenced by Fog computing, for such applications, a new paradigm started to emerge. It had as one of its objectives to put such applications as near to their consumer/clients as possible. This new concept was called Edge Computing, because it is designed to operate precisely on the edge of the open internet network. At this stage, the industry started to mature and develop applications that would benefit from being hosted exclusively and specifically in the Edge. This created a new class of applications, Edge Applications, which could be designed in drastically different architectures than Cloud Native Applications.

Edge Computing had some serious advantages in these scenarios compared to Cloud Computing. It could offer a previously unachievable low latency for consumers. This was done by decreasing the number of network hops necessary to communicate with each other as low as possible. This usually meant that while the application and client were actually separate, they would still have a certain strong coupling in terms of their disposition in the network they were operating. The fact that both the Application and Client were in very close proximity also provided certain security brought by not depending as much on inter-network traffic and staying as much in LAN as possible. Of course, this isn't always possible, nevertheless, Edge Computing as a technology fundamentally enables and incentivizes computing to happen as near to the source of data, a client, or consumer, as possible (SHI et al., 2016).

Applications started to become dependent on this close proximity to their data source or target, while also rapidly expanding their computational necessities and needs. This created ever-growing limitations of solutions that could deliver the required QoS and Service Level Agreements (SLA). At the same time, some parts of applications could only be properly run in the Cloud, and other parts needed to be on the Edge in order to meet those SLAs. Needing some special hardware, or tools to execute some functions were some among the limitations that applications started to create upon themselves. This created an impossible scenario where provisioning the whole application in only one substrate would not be sufficient. Computation on the Edge follows this idea of closely placing Applications in relation to their clients. Although many of its benefits come exactly from this characteristic, it can also be a possible sensitive point of weakness. This is because, although the data sources can often be stationary sensors, home IoT devices, or other similar appliances, they can also be users with mobile devices, vehicle sensors, and moving field IoT appliances, among others.

This mobility of clients can suddenly create problems in Edge Computing scenarios, quickly increasing the distance in which data has to travel to reach a given Edge Server, thus rapidly extenuating the benefits of the architecture. In a quasi-static scenario, users simply select the best-fitting Edge Server to connect to, usually the nearest, but in a moving user scenario, this

is not enough, as the list of best-fitting servers sorted by such criteria is constantly changing (CAI; ZHU; ACKAH, 2024). It is, therefore, necessary to dynamically select to which Edge Server the user should be bound to, based on some information such as its geographical location.

Each application and by extension each of its individual composing services, has specific requirements and restrictions for many facets of computational resources. One service may need specific hardware, such as Graphics and Matrix Computing Units, or Tensor Processing Units, while others may need to meet specific latencies thresholds. These thresholds are the so-called Quality of Service requirements, often expressed through Service Level Agreements. They are vital to defining how well an application is running, and how the delivery of its functionality is meeting what was proposed.

All of these metrics and agreements are closely related to the substrate in which the application is being hosted. By means of provisioning the services into different hardware infrastructures, one can drastically alter the results of those metrics and achieved agreements. But one important question that arises from this hypothesis is how exactly would such provisioning of these services take place. The action of deciding which services are better off in which infrastructure/substrate is not a simple one. There are a few perspectives and aspects that need to be considered. The first is probably the Quality of Service model, or in other words, which of those metrics previously touched are considered, and what are their weights in the decision. In the context of the present work, QoS can be considered a proxy to user-perceived latency (OUYANG; ZHOU; CHEN, 2018), and can be very precisely determined by two principal metrics: the Computing Delay; and the Communication Delay.

Beyond the QoS model, there also needs to be a Service Placement model, whose actions are twofold: it is responsible primarily for the first placement of each service, and in conjunction with the QoS model, dynamically re-placing, or in other words, migrating them according to the movement patterns of users. This migration of course is not instantaneous, and also not free. Because of that, a Migration Cost Model might also be needed, in order to balance the frequency or magnitude of migrations more effectively (OUYANG; ZHOU; CHEN, 2018).

Of course, in the real world, all of this mixed and hybrid provisioning is made even more complicated by the fact that Internet/Infrastructure Service Providers are not unified entities, and one would have to integrate the private and public parts of the networks of many providers. Ranging from traditional Cloud Providers, such as Amazon Web Services, Google Cloud, and Microsoft Azure, to intercity Edge networks, to very locally installed Edge Servers and meshes, a provisioning policy would have to be able to connect, analyze, and deploy to all the layers of this Cloud-Edge Continuum. This task could prove itself to be very challenging because it would require some level of federation, or coordination between all the possible market competitors, providers.

1.1 OBJECTIVES

The present work aims to present the amplitude of possibilities that the Cloud Continuum paradigm can bring to some scenarios and applications, while also trying to provide a new perspective on user-mobility-aware provisioning policies. There is a prospect that many improvements can be achieved through the complete usage of Cloud-Edge Continuum for provisioning service-composed applications, where each service has different and individualized characteristic and requirement in terms of computation and network resources/necessities.

The hypothesis is that provisioning applications composed of varied and diversely defined services into a continuous substrate, organized by multiple layers of different kinds of computational infrastructure, could lead to better results in perceived quality of service by users in a geographical movement-heavy scenario, especially if the provisioning policy was to take into consideration this user mobility. Alongside this main hypothesis and primary research objective, the present work will also try to accomplish a few secondary objectives, which are described below:

- Define an efficient and realistic mobility model for simulating moving users in urban areas, and models to extract usage and performance data from said users.
- Elaborate policies and experiment with different provisioning algorithms for Cloud-Edge Continuum Applications.
- Measurably improve the user-perceived Quality of Service of Applications that fall into the Service-Composed class defined in this work, by applying such previously mentioned policies and algorithms.

1.2 METHODOLOGY

This academic text is a Dissertation for a Graduate Course in the STEM Computer Science field, specifically in the Applied Computing course in the Computing Systems line of Research. The text is classified as quantitative research in terms of the work's approach. With respect to the nature of research, it is classified as an applied research/topic, with exploratory objectives, and an experimental approach to research procedures.

The present work contains theoretical reviews of the main concepts approached in all discussions, as well as a systematic bibliography review. After a brief discussion on transitioning from theory to more practical scenarios, the implementation of the proposal is discussed in practical terms, in order to truthfully describe the challenges faced, and the solutions found. A simulation campaign is used to calibrate the parameters of the models and policies, and also to evaluate results of the proposed solutions,

1.3 CONTRIBUTIONS

A few key findings and developments are perceived to be achieved through the present work. Some of its main contributions could be listed as defining a Cloud-Edge Continuum model which considers user mobility and Quality of Service requirements, with a definition of a point of interest mobility model, as well as extensions to a state-of-the-art simulator to support said models. The results of the simulation campaign are in off itself contributions, as are the discussions, observations and findings that can be drawn from them. There is a clear indication of how the proposal could be implemented in a real scenario, and what would be necessary for it to be done so, and so this becomes a path for future work in the field.

1.4 TEXT ORGANIZATION

The remainder of the present work is divided into five chapters. Chapter 2 consists of a literature review, providing the necessary background information to the establishment of the proposal and understanding of the implementation of the experiment, including key theoretical concepts and examples relevant to this work of research, as well as a formal definition of the problems set to be solved In addition to this fundamental literature, a systematic review on the state-of-the-art literature is also conducted, regarding themes related to the present work. Chapter 3 goes on to describe the proposed contributions of this work, a deeper description of the methodology and architecture being used, how the decision policies work and are characterized, and also a discussion on how the proposal is expected to transition from theory to a more practical scenario in the simulation environment. After the formal formulation of the proposal, a discussion about the ways the proposal was implemented will be carried out in Chapter 4, including the challenges faced with implementing it, new and existing models used and developed, among other technical details relevant to the research. In Chapter 5, the methodology for the simulation campaign is described, including the simulation environment and setup, the metrics and objectives used in the evaluations, a calibration round to determine the parameters that better describe real scenarios, a comparison between proposed policies, baseline ones, and literature ones. Some time is also spent discussing the results of the simulation campaign, and the observations that can be drawn from them, as well as a discussion on how the results of the simulation campaign, its metrics and conclusions can be translated to the real world, and especially, what would be necessary for it to be implemented in a real scenario. Finally, in Chapter 6, we present the final considerations of the work, including a review of the proposal, the implementation, the simulation campaign, evaluations of the results, and some considerations about the future of the work, and the field of research in general.

2 LITERATURE REVIEW

In this chapter, a literature review will be conducted, going over some topics and concepts, deemed fundamental to understand the material that will be discussed and explained in the next chapters. In Section 2.1 discusses the concept of computing as a utility, Cloud, Fog, and Edge computing. Later, Section 2.2 elaborates on how some applications can be designed to take full advantage of new computing domains. Afterward, Section 2.3 touches upon the topic of mobile users in a geographical space, and how that mobility can affect applications. Subsequently, Section 2.4 examines the novel concept of a continuous substrate containing both Cloud and Edge computing domains. Finally, Section 2.5 presents a systematic literature review, discussing the current state-of-the-art in the field of Cloud-Edge Continuum, and the gaps in research that could be explored. To close the Chapter, we briefly discuss and reflect on its contents, and what will be the focus of the next chapters.

2.1 CLOUD AND EDGE COMPUTING

From the Mainframe Era in the 50s, through the arrival and popularization of personal computers in the 90s, up to the introduction and viability of virtualization in the 2000s, access to computing power had always been associated with big costs. Both in terms of upfront investments, regarding necessary hardware purchases, and in terms of very expensive maintenance, requiring highly skilled personnel and upgrades, also related to the inflexible nature of owning and managing finite computing hardware. This would be, however, in the brim of radical change, with many factors such as a widespread adoption and maturity of the internet, the popularization of Service-Oriented Architectures and web services, world leading companies starting to invest in providing computing services.

2.1.1 Cloud computing

Cloud computing has emerged as an innovative paradigm in the computer science field, and it fundamentally changed how organizations and individuals processed, stored and managed their data, by allowing on-demand access to a variety of different computing resources, like processing power and data storage, in a service-based model, eliminating the need for infrastructure and hardware management. The concept itself dates back to the 60s era, when the term was used to describe an ambition of having computing resources as a utility, likewise having electricity or clean water in homes (ELLISON, 1967). Nevertheless, it wasn't until the mid-2000s that this fantasy started to become a commercial reality, pioneered by industry pillars such as Amazon, Google, and Microsoft.

> "Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as *Software as a Service*

> *(SaaS)*. The datacenter hardware and software is what we will call a *Cloud*. When a Cloud is made available in a pay-as-you-go manner to the public, we call it a *Public Cloud*; the service being sold is *Utility Computing*. We use the term *Private Cloud* to refer to internal datacenters of a business or other organization, not made available to the public. Thus, Cloud Computing is the sum of SaaS and Utility Computing, but does not include Private Clouds. People can be users or providers of SaaS, or users or providers of Utility Computing." (ARMBRUST et al., 2009).

Providing computing power as a utility means offering a huge, flexible and cost-effective infrastructure in the form of ready to be used solutions and tools, for the most varied of computational needs. Cloud computing services have many benefits from previous paradigms, one of the main ones being its scalability, which allows for resources to be acquired based on the demand, easily adjusting the cost of operation according to its usage and return (ARMBRUST et al., 2009; MELL; GRANCE et al., 2011). Moreover, because of the sheer scale at which Public Cloud Providers operate, their services become extremely cost-effective for most scenarios, big or small, this combined with the pay-as-you-go model, it drastically reduces the entry-cost for newcomers to the industry. Beyond that, having managed and maintained computing diminishes the need for very specialized infrastructure engineers and similar personnel. It is intrinsic to the Cloud Provider to assume all the responsibilities in managing and controlling not only the infrastructure, but also the security, availability, and operability of applications, while the Cloud Consumers have limited access to administrative control over said applications (LIU et al., 2011).

Although Cloud Computing had many benefits, with the expansion of Internet of Things devices, the Industry 4.0 phenomena, and general availability of mobile internet connected devices in the early and mid-2010s, the need for better distributed computing started to arise (LOPEZ et al., 2015). Because of the way Cloud Computing is architected, it runs in big, sparsely spaced Data Centers, often afar from densely packed cities and locations. This means that, although providing excellent processing capabilities, Cloud Computing lacks the means to deliver or receive data very quickly, which is a need of many applications that started to appear in this new distributed era. With new mobile technologies such as 5G in the horizon, and ever-increasing restrictions of latency ((ITU), 2017), Computing and Network providers started to venture with new approaches to tackle these new scenarios.

2.1.2 Fog computing

Fog Computing was an attempt of bridging the technological gap in distributed systems that required abiding by these previously mentioned restrictions. It was an attempt at delivering relatively capable computing service closer to the edge of networks, aiming to minimize the liabilities of having distributed applications (BONOMI et al., 2012), and it worked by means of collaboration from Cloud Providers and Networking infrastructure organizations, such as Cisco.

Nevertheless, supplying Fog capabilities inferred a heavy investment from Providers in new and niche infrastructure, which did not closely abided by the principles of economy of scale as traditional Cloud Computing. Moreover, the concept of Fog Computing was fundamentally to be an extension of Cloud itself, and so, it carried over many core characteristics of Cloud Computing (DAS; INUWA, 2023). Ultimately, it wasn't received with a widespread adoption, and thus, the problem of distributed computing was still considered to be predominately unsettled.

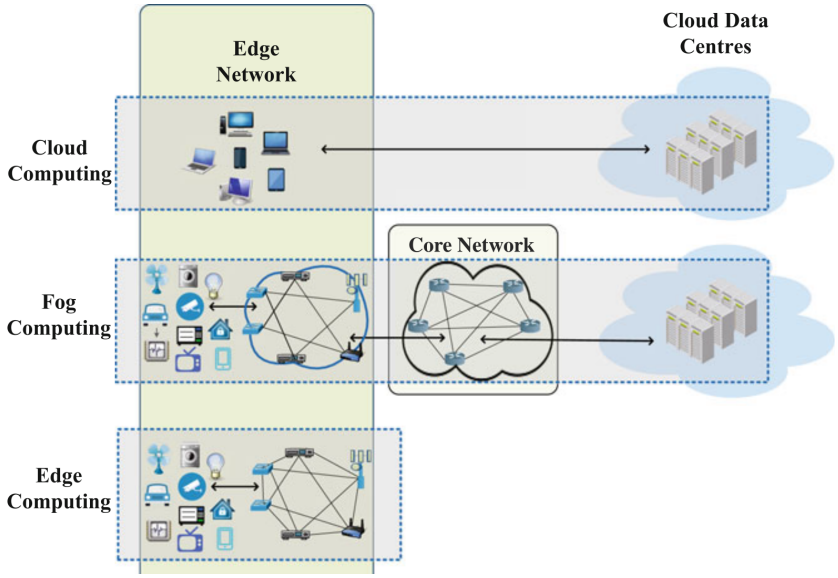
2.1.3 Edge computing

The need for real-timeness, lower latencies and round trip times of data processing were some forerunner motivators of the introduction of Edge Computing, which as the name implies, is the idea of bringing processing, storage and other computing resources to the complete edge of networks (LOPEZ et al., 2015). The network edge here means the closest point of contact that devices, such as industrial sensors, mobile phones, and autonomous appliances, have with the public internet, so having computational power at that point dramatically lowers response times and bandwidth usage for such devices connected to distributed applications, also known as Edge Devices. Because the edge of networks itself was always present, although previously limited to networking functions, the entry cost/investment for Edge Computing is much lower compared to it Fog Computing counterpart.

Figure 1 illustrates the different areas of domain Cloud, Fog, and Edge Computing. This figure can be a good illustration of the distance that data has to travel when being handled by each domain, for example, on the Cloud computing domain, the devices present at the edge network are far from the servers communicating with them at the Cloud Data Centers. On the Fog computing domain, the distance from devices to the Cloud is the same, but now there is a new element in the middle of them, which serves as an intermediary, however, as discussed previously, this new element is a whole new infrastructure that needs to be deployed by the interested parties. In the Edge computing domain, computing power is added to the already existing networking infrastructure, it is interesting to point out that in every domain illustrated in the image, the edge network was already present, simply because it is needed for any computing domain to be able to reach the devices of users.

Edge computing is not an entirely new concept, but rather an iteration on distributed computing with an evolution of older concepts such Content Delivery Networks (CDNs) (LOPEZ et al., 2015). Edge devices generate data that is initially processed by Edge servers that are responsible for data aggregation, filtering, and preliminary analysis. However, Edge only applications are very niche and specific, and many general purpose application could not be handled completely within Edge servers. As a result of the hardware, energy, and space limitations that Edge servers have to surmount, their general Computational power is dramatically lower than that of Cloud servers in robust Data Centers. Subsequently, they have a more narrow domain of operation, limiting the applications it can successfully deal with.

Deciding to use Edge Computing must be a very well-informed and intentional decision,



Source: Adapted from Mahmud, Kotagiri e Buyya (2018).

Figure 1 – Computation domain of Cloud, Fog, Edge, Mobile Cloud and Mobile Edge computing.

even though it can benefit applications in some aspects, others can rapidly become huge hurdles in terms of operation, maintainability, and management. Structuring a big application to be only partially hosted in Edge Computing can lead to better results than Cloud-only or Edge-only, however, it also immensely increases the complexity of operation as well.

Ultimately, Edge Computing can be a great tool when Cloud Computing leaves a lacking performance in some aspects. In the latter sections, other aspects that can encompass or are related to it are discussed, such as User Mobility in Section 2.3, and the abstraction of hybrid substrates called Cloud-Edge Continuum in Section 2.4. Later in Chapter 3, one of the more novel scenarios of using this abstraction is presented and discussed in depth.

2.2 SERVICE-COMPOSED APPLICATIONS

One of the main benefits of having on demand, and often automatically scalable resources through the use of Cloud Computing, is the ability to more granularity react to load/demand spikes in Applications (DRAGONI et al., 2017). However, traditionally designed applications often could not take advantage of this fact, on account of being architected in a monolithic approach, meaning every single component/module of the application is coupled together in a single, sizable artifact, and cannot operate independently. This software design approach was widely used before Cloud Computing, when a server usually used to host a single, exclusive application, and any adjustments regarding computational resources were usually done so via a technique called Vertical Scaling, where the hardware of that single server is upgraded, in order to accommodate increases in demand. The counterpart to Vertical Scaling would be Horizontal Scaling, where instead of giving the single instance of the application more resources, one would

increase the number of instances itself, however, needing another separate server for any new instances.

Nevertheless, as previously mentioned, one big differentiating characteristic of Cloud computing is that its servers are transparent to the application, meaning that the work necessary to obtain new servers is minimal. But a monolithic application is generally too big to have horizontal scaling be cost-effective, and so, a new paradigm started to become more viable, the Microservices architecture (DRAGONI et al., 2017). In this new concept, applications are composed by many smaller, independent processes, that have single, or reduced functionality and responsibilities individually, called Microservices.

When functionalities are very clearly separated, they can also react to loads separately, thus being individually scalable and elastic to demand. This flexibility can help to reduce costs and improve resource utilization, by scaling down when said resources are idle, a fact which can be crucial in a pay-per-use model such as Cloud computing. Being horizontal scalable also brings a side effect of better fault-tolerance and isolation, in view of one service failing not necessarily impacting others, which brings higher availability and resiliency to the application (DRAGONI et al., 2017).

For the application as a whole to operate, however, all microservices must cooperate, and for that to work, they need to communicate with each other, most often via the network. Network communication implies data transfer of varying sizes, restrictions on data freshness or staleness, higher analytics complexity, among other aspects that make a Microservice architecture more complicated to fully grasp. This can also lead to wider surface area for attacks against the application (PAHL et al., 2017), which must be counter-acted with much more meticulous data management.

Consider an application with a stream processing element to it, that has a few different responsibilities and constantly executing actions. The service responsible for ingesting data from the data source has a big real-timeness restriction to it, it must receive and acknowledge data as fast as possible, or it may even need to poll for that data, in order for it to not become stale in some way. On another part, there is a service responsible for filtering data according to some conditions, suppose this service could live with a constantly sized backlog. After that, there could be a service responsible for aggregating or conforming the data into a specific format. Finally, this application can have a service that sends this formatted data to an archival, and to save bandwidth, this service could do so in batches. Notably, each of these different services also has very different characteristics, in terms of resource usage, network restrictions, and also network footprint, and depending on the way it is deployed, and specially in which kind of computing provider, they could perform drastically different, above or below what could be deemed acceptable.

2.2.1 Network requirements

In the realm of distributed applications, especially in Microservices architectures, the leading factor that impacts performance is network communication (DRAGONI et al., 2017). This is only natural, seen as accessing information in-memory is orders of magnitude faster than accessing it via network calls. The degree of dependency on network calls will depend entirely on how the application is designed, the ratio of memory calls to the total number of calls is very related to components/modules sizes, and so a system with less/weaker coupling between components will experience less degradation due to network latency.

2.2.1.1 Latency, RTT, and Jitter

The problem with network latency, however, is that it can be very inconsistent and unpredictable (JAISWAL et al., 2002), due to a number of factors such as signal propagation time, transmission time, and processing delays at network nodes. As a consequence, any system that relies on a constant latency is more prone to failures, and applications that aim to be highly available and resilient must be prepared to experience latency spikes, and potentially even outages. For many applications, network communication needs to be bidirectional, and although rare, network latencies can vary depending on the direction of the flow, even in the same route, so another metric called Round-Trip-Time (RTT) can be useful to understand a bigger picture on an application's network performance (KUROSE; ROSS, 2012). It measures the time taken for a signal to travel from source to destination and back again, and is effective to measure processes that rely on acknowledgements, from lower levels such as TCP ACK, to higher levels such as a message consuming from a queue, and it can be a great indicative of network congestions.

For certain kinds of applications, such as live video streaming, and Voice-over-IP (VoIP), the variation in packets' queuing delays, also called Jitter, can be much more detrimental to performance. High Jitter can lead to problems such as packet loss, retransmissions, and unplanned buffering, which in turn degrade the quality of the service delivered by the application (TANENBAUM; WETHERALL, 2011). To counteract the effects of Jitter to some degree, applications can implement mechanisms of data buffering, and jitter correction algorithms, but to a higher degree, no method will completely eliminate its effects.

2.2.1.2 Bandwidth and Throughput

Alongside latency, one of the main aspects that can impact network intensive applications and networking in general is Bandwidth. It defines the maximum rate at which data can be transmitted over a network connection, and it is a fundamental parameter that dictates the volume of data that can be moved from point A to point B within a given timeframe.

The theoretical bandwidth of a certain network route is often very far off the practical throughput of applications using that route, not only in virtue of the route itself being shared with many other unrelated connections, but also because a huge number of factors can limit it, like

protocol inefficiencies and general network congestions (KUROSE; ROSS, 2012). Applications that are reliant in a specific minimum throughput rate are said to be bandwidth-sensitive, examples of these are current multimedia, although they may implement techniques which adaptively transcode data into smaller volumes, in order to fit the available throughput. Contrarily, Elastic applications are classified as such for their capacity to utilize of as much or as little throughput as may be vacant.

2.2.1.3 Network Congestion

All the previously mentioned metrics are affected by, or can be the culprits of, one common issue, which is network congestion. In simple terms, a network is congested when its demand exceeds the available capacity, be that of hardware, software or both (TANENBAUM; WETHERALL, 2011). Sudden spikes in networking load, bad designed topologies, or ill prepared hardware can also lead to congestions. They can occur in varying points of the network, from routers, switches, to end network interfaces, and the increase in packet queues, which in turn can lead to all the other problems discussed previously in this section, can also severely impact some application's performance.

Building loss-tolerant systems is a huge deterrent from suffering from network congestions and their effects, e.g., multimedia applications can withstand data loss to some degree, because it won't incur in a crucial inability to operate (KUROSE; ROSS, 2012). Not all applications, however, can be built this way, and so supporting such applications means the necessity to guarantee a reliable data transfer, which in network communication is often impossible to achieve.

Systems must be designed with some trade-offs in mind, notably, the CAP Theorem, which stands for Consistency; Availability and Partition-tolerance, states that in the case of a network separation, systems can either maintain 100% of data consistency, or 100% of service availability (GILBERT; LYNCH, 2012). In practice, a system can be implemented in such a way that it balances one with the other, but obtaining both completely is deemed to be unfeasible (NOORMOHAMMADPOUR; RAGHAVENDRA, 2018).

2.2.1.4 Flow Completion Time

Flow completion Time (FCT) is the duration required to complete a data transfer, from when the first packet of a flow is sent, up until the last is acknowledged (DUKKIPATI; MCKEOWN, 2006). It encompasses all the aforementioned metrics and factors, and can provide a comprehensive measurement of application performance, and successfully optimizing it can be critical, especially in distributed systems and applications. One can employ a number of different strategies to minimize FCT, including optimizing network paths, prioritizing critical data flows, and employing congestion control (TCP) techniques that are focused on optimizing FCT, such as Rate Control Protocol (DUKKIPATI; MCKEOWN, 2006). Essentially, FCT denotes a user-level

metric used to account the communication time required to transfer data among distributed microservices.

2.3 USER MOBILITY

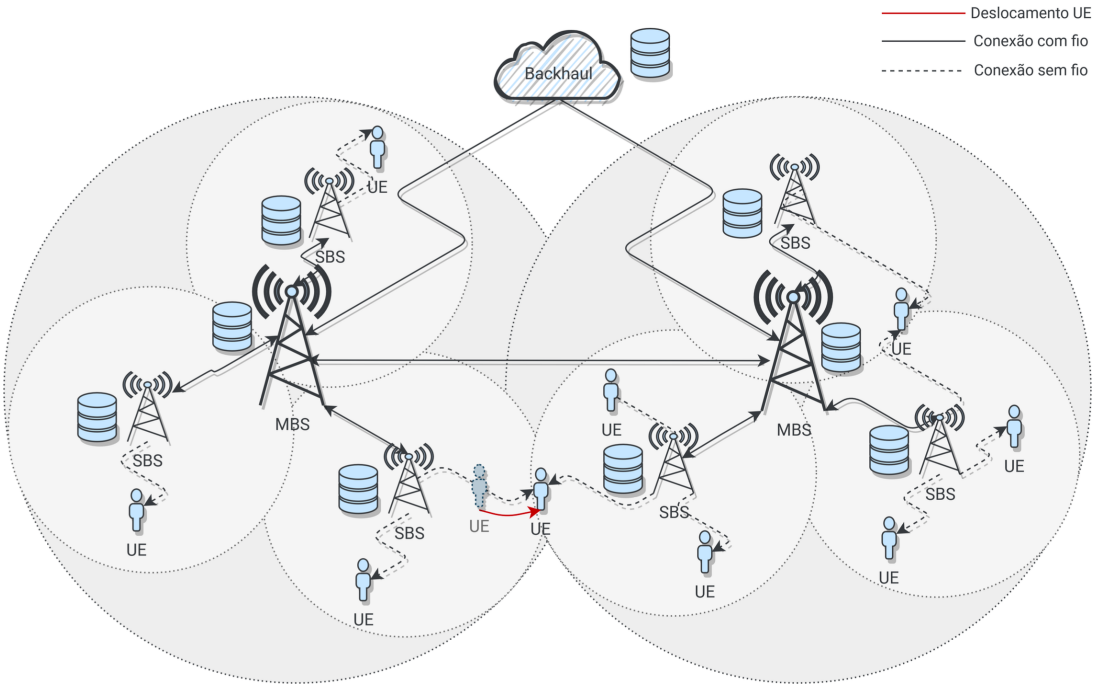
The rapid advancement of mobile and wireless technologies in the last decades drove up the demand for user mobility supported applications. In distributed computing environments, resources are scattered across multiple locations all connected through networks varying in quality, and users moving through these locations, often in unpredictable ways, can introduce many challenges (OUYANG; ZHOU; CHEN, 2018). If a certain application has some components statically hosted in some Edge servers, for example, any small change in the position/location of users/things connected to those components could dramatically influence the performance and perceived condition of the application as a whole (ALVES; KOSLOVSKI, 2022).

User mobility refers to the action of users accessing and interacting with applications while moving through a geographical space. Supporting this scenario requires the system to seamlessly deliver its services without significant disruptions, but there are different degrees of sensitivity to degrading of the service itself, depending on the kind and class of application (BONOMI et al., 2012).

Figure 2 illustrates a network architecture that has user mobility as a fundamental characteristic. The users can be connected to any of the base stations in the network, and when they move, they can be handed over to another base station, which can be in the same or in a different subnetwork, and the network must be able to handle this handover without significant disruptions to the user's service. There can be many aspects which can influence the handover complexity, such as the subnetworks having different providers, different technologies, or even different network architectures.

2.3.1 Mobility Models

In order to study and understand any scenario involving user mobility, it is necessary to be able to simulate in one way or another how users move through a geographical space in certain spans of time. The most accurate way of doing so is actually replicating real-world data, and there are many ways to do so, such as GPS data, mobile phone data, and even data from social networks. This method, however is not always practical or applicable, and so, many mobility models have been proposed in the literature to simulate user mobility in a more abstract way. These models vary in their characteristics, complexity and accuracy, but most can be classified into either pedestrian walk models; social network-based models; vehicular models; and models for disaster scenarios (SOLMAZ; TURGUT, 2019). The current work will focus more on vehicular based models, which tend to produce scenarios where network conditions can be more rapidly changing, and also sometimes more challenging to handle. Probably the simplest vehicular model which can be used is the Random Waypoint model, where users move randomly



Source: Alves e Koslovski (2022).

Figure 2 – Mobile network architecture.

through a space, and stop at random points for a random amount of time, and then move again to another random point (CAMP; BOLENG; DAVIES, 2002; BAI; HELMY, 2004), however, this kind of model can produce very unrealistic scenarios. One kind of model which is still very simple, while being more realistic, is the Work and Home Extracted REgions (WHERE) model, specifically the WHERE2 variation, which boils down to the idea that people tend to spend the vast majority of their time on two locations, either at home, or at work, which can be trivially simulated (SOLMAZ; TURGUT, 2019).

2.3.2 Quality of Service & Application Requirements

Applications classified as latency-sensitive are presumed to only become more common, with the ever-increasing capabilities that new wired and wireless transmission technologies start to offer ((ITU), 2017). Both developers and users will want to take full advantage of these capabilities, and naturally, applications start to become dependent on them. Some applications have no problem being in a “best performance under the circumstances” kind of existence, other however, require stricter performance guarantees (TANENBAUM; WETHERALL, 2011). One way to solve that would be setting fixed network metrics target minimums and maximums, however, that kind of rigidity not only can be actually detrimental to the application’s development, but also incredibly hard to establish.

A solution to guaranteeing a good service conditions is to deploy an infrastructure designed to sustain any kind of traffic and use case to it, which is called overprovisioning (TANENBAUM; WETHERALL, 2011). Any application in this infrastructure will be able to perform its

required affairs with not significant problems and losses, and its performance can't realistically get any better than this. This approach, however, can be absurdly expensive and wasteful.

A mid-point in terms of flexibility and harder guarantees would be to define more abstract Quality of Service (QoS) constraints for an application's execution, and create dynamic operators that could balance applications and their QoS restrictions (TANENBAUM; WETHERALL, 2011). Just as a simple example, Figure 3 relates some common application classes with their Application Requirements to maintain good QoS. Such requirements can be used to characterize an application's network behavior, and better handle its traffic. Some of them are more flexible than other, e.g., small amounts of packet loss are dealt with retransmissions, and the receiver buffering packets can smooth out slight jitter, but applications cannot be designed in such way to deal with not enough bandwidth or too much delay.

Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

Source: Tanenbaum e Wetherall (2011).

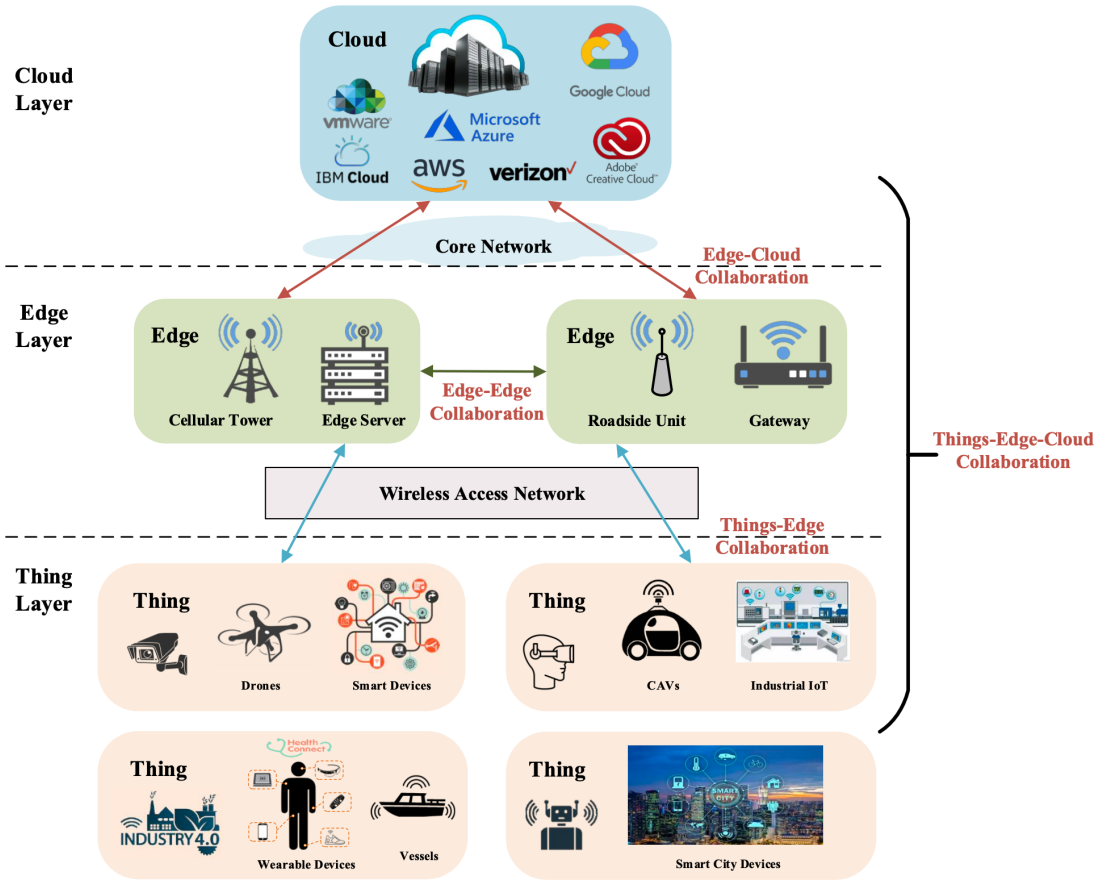
Figure 3 – Stringency of applications' quality-of-service requirements.

Of course, the data presented in Figure 3 is very abstract, used just to exemplify and materialize what these constraints can actually look like, and the actual requirements of an application can vary widely, and it is up to the application's architect to define them. The present work does not intend to delve into very specific of QoS requirements scenarios such as these, but rather to present a more general overview of the topic. Ultimately, adequately defining an application's requirements and its Quality of Service conditions can be very useful in distributed computing scenarios, where resources are limited, and knowing how some components can behave in certain conditions can make a huge difference in optimally provisioning then.

2.4 CLOUD-EDGE CONTINUUM

Computing resources can vary widely in terms of origin, scale, cost, management needs, among other factors, and correctly choosing a provider can deeply alter an application's overall performance and behavior. As previously discussed in Section 2.1, Cloud and Edge computing are two very distinct substrates, each with their benefits and drawbacks. Figure 1 illustrates the different domains in which computing resources could be hosted. Having to choose only one domain, however, can be very limiting, and lead to sub-optimality in distributed systems.

The Cloud-Edge Continuum is a heterogeneous substrate which consists of transparently covering both Cloud and Edge computing domains, in order to allow for applications to be architected in distributed systems with more flexibility in terms of latency, processing/storage power, distance to users/data sources, among other benefits (RODRIGUES et al., 2023). Figure 4 demonstrates the distance and how the overall architecture that a Cloud-Edge Continuum substrate would look like. Starting from the bottom Device layer, it consists of mobile smartphone users, IoT devices, sensors in cars, smart wearables, etc., and these devices have an extremely limited computing capacity. So any more complex action must partake execution in the second layer, the Edge layer, which consists of geographically spaced servers with increased computing capabilities, albeit still limited to a certain degree. If some action needs even more computing power, but most importantly, it can cope with looser network and application requirements as discussed in subsections 2.2.1 and 2.3.2, it may need to be hosted by some Cloud provider in a Data Center, potentially far away.



Source: Luo et al. (2021).

Figure 4 – The edge-cloud continuum architecture.

Statically choosing where each component is going to be hosted for the rest of its life is not that hard, one can just consider the aforementioned Application requirements, discussed in Subsection 2.3.2, and hand pick the computing domain of each component. Nevertheless, in reality applications and systems should dynamically adapt with their environment and their

demand, in order to better utilize its available resources, and also deliver the best quality of service possible. For this to be feasible, a complex provisioning of the available infrastructure must be made.

Optimally managing and distributing the resources in both computing domains in real time can be challenging, and a provisioning policy needs to consider and rely on many aspects. Especially considering that a Cloud-Edge Continuum would normally cross the boundaries to a single computing provider, demanding some level of federation of data management, responsibility, and ownership. It can be, however, one approach to improving applications even more, by bringing a higher resourcefulness and flexibility to designing complex distributed and even mobile systems.

2.5 RELATED WORK

The present section consists of a systematic literature review, the reasoning behind this choice is that seen as the concept of Cloud-Edge Continuum is very novel, there are not many literature publications, especially secondary works, with an emphasis on constructing an overview of this state-of-the-art theme. Beyond that, there is a substantial interest in identifying the current issues and gaps in research in this sphere, and which ones could be explored in future works. In the remaining of this section, all the relevant search characteristics and parameters are going to be explored, including the search motives, or research questions, the keywords considered, and how they are used in the complete search string, what fields were selected, what academic search engines were used and why, what was the search's publication period range, what kinds of publications were included, the objective and subjective inclusion and exclusion criteria, how the data was collected, what classifications were made from that data, and finally some analysis of the review itself, considering risks and potential conflicts of interests. With that in mind, there are a few very notable works in existing literature that have done just that. The present section aims to understand these related researches and compare how this work related to them.

2.5.1 Research Questions

There are some research questions that have been raised in order to conduct this review, based on the original hypothesis. The primary question is:

- Is it possible to formalize a provisioning heuristic for Applications with distributed services/components across different substrates in the Cloud-Edge Continuum?

There are also two secondary research questions, which are the following:

- Can a classification of different kinds of applications be made, in relation with the different behaviors of its composing services?

- Could hybrid/integrated provisioning in the Cloud-Edge Continuum improve the perceived performance for end-users?

2.5.2 Keywords

The search keywords are constructed in a certain way to try to encapsulate everything that the research wants to focus on. They are divided into four groups of synonyms: Firstly, we want to map the combinations of Cloud and Edge that are found in the literature, which look like any of the following: “Cloud-Edge"; “Edge-Cloud"; “Cloud Edge"; and “Edge Cloud". Second, this group simply consists of the term “Continuum". The third group consists of terms related to user mobility, which are any of the two “Mobility" and “Movement". Lastly, the fourth group consists of terms related to the provisioning aspect of the research, and can be any of the following: “Provisioning"; “Scheduling"; “Allocation; “Distributed"; “Deployment". Each keyword from each group is combined with an “OR" boolean operator, and for the groups themselves, they are combined with an “AND" boolean operator.

The combination of all the keywords presented before, with all the boolean operators, is as follows: *("Cloud-Edge" OR "Cloud Edge" OR "Edge-Cloud" OR "Edge Cloud") AND "Continuum" AND ("Mobility" OR "Movement") AND ("Scheduling" OR "Allocation" OR "Provisioning" OR "Distributed" OR "Deployment")*

2.5.3 Search Fields and Engines

With all the keywords defined above, the next step is defining where these keywords are going to be search, in terms of fields and metadata of publications. It must be addressed that some of the academic search engines are limited to "all metadata", and could not be further refined to only the chosen fields. The chosen fields are: Publication title; Abstract; and Publication keywords.

There were three different academic search engines used to conduct this systemic literature review. Each one has a reason behind the choice to use it, they are as follows:

1. Scopus: this engine was chosen because the author had already worked extensively with it, and had familiarity with its functionality, especially with the advanced search mode;
2. ACM DL: the reason this engine was chosen is because of the vast indexation of general and very specific Computer Science publications that it contains;
3. IEEE Xplore: for the great quantity and quality of the advanced search functionalities, this engine was selected;

2.5.4 Search Selection

After searching and matching the keywords in the academic search engines, there needed to be a way to select a subset of all the found publications. This subsection contains the details

of these choices, which impact what literature is going to be analyzed and reviewed.

2.5.4.1 Objective Criteria

Objective criteria are somewhat straightforward, and they represent many obvious choices in this review. The first criterion is the publication date range, which was chosen to be from 2015 to 2024. This choice has a sound reasoning behind it, 2015 was the year of publication of (LOPEZ et al., 2015), which is one of, if not the first proper and grounded literature on the topic of Cloud-Edge Continuum. So limiting from this date and beyond is thought to be a good choice in terms of sub-setting the emerging results of publications. The second criterion is the language of the work, which was chosen to be limited to English. That is because other than the mother language of the author, English is the only language that could be properly understood in an academic setting.

The third criterion is the limitation of primary works only. That is because secondary works were already touched upon at the beginning of Section 2.5, and tertiary works are outside the scope of this review. The fourth and final criterion is a limitation related to the type of publication, which was chosen to be only conference papers and journals.

2.5.4.2 Subjective Criteria

Contrary to the objective criteria, subjective criteria are more susceptible to interpretation, and so, are more open-ended. They are divided into inclusion criteria and exclusion criteria.

Firstly, in the inclusion criteria, publications must mention the origins of any used dataset; second, there must be an explanation for any algorithms present, or source code available; third the work's methodology must be reproducible.

As for the exclusion criteria, implicitly, if none of the inclusion criteria are present, the work is discarded. Beyond that, if the focus of the publication does not align with any of the research questions, the publication is also discarded.

2.5.5 Data

This subsection is responsible for discussing what kind of information and data the researchers aim to extract and analyze from the selected publication's subset. There are some data that are of great interest to the researchers, they are listed below, with a brief explanation of why they are important:

- Classification and methodology of user mobility: which is important for the reproducibility of the work;
- Dataset information: regarding it being simulated, real, or based on real data, but transformed in a way;

- Grouping and division of Applications: related to the services and components that compose the application;
- Provisioning aptitude: with respect to whether the scheduling is automatic or dependent in some kind of input from users/developers.
- Kind of resources considered: examples are processing power, storage amount, memory capacity, bandwidth, discrete computing, etc.

All the data extracted discussed in the previous subsection is going to be accumulated and classified in some kind of tabulated informative review. And one of the biggest focuses is classifying the different applications used in experiments.

2.5.6 Results and selection

As it can be seen in Table 1, for each of the chosen academic search engines, the following number of publications were found with the search string and chosen fields/metadata, and after executing the filtering, the following number of selected publications were kept. From the Scopus ASE, the number of selected publications, after applying the established objective and subjective criteria, was reduced from 22 to 6. For ACM DL, the number of returned publications was the biggest, but for unknown reasons, most of them were not really related to the focus of the review and therefore were discarded. From 32 results, only 1 was selected. In IEEE Xplore, there were only 9 initial results, but percentage-wise, it had the most relevant publications, from the 9 total, 7 were selected. The final number of selected publications came to 14 in total, although some publications appeared on different ASEs, this finding was common in publication present on Scopus and IEEE Xplore. With repeating publication ignored, the total number of articles came down to 11.

Table 1 – Number of selected publications for each ASE.

Academic Search Engine	**Number of returned publications**	**Number of selected publications**
Scopus	22	6
ACM DL	32	1
IEEE Xplore	9	7

Source: Developed by the author (2024).

2.5.7 Discussion

After a careful selection of the found publications, a review of their content was conducted, aiming to extract a few points of attention and classifying each of them into a few analysis points. All the debated points were subjective and qualitative in some manner, and could not

be statistically analyzed in any way. Table 2 shows all the selected publications, joined by five different aspects that were identified, classified and summarized, those being:

- The presence of user mobility in the research;
- Which dataset was used in the experiments, if any;
- What kind of general or specific application the research focus, or was used in the experiments;
- What was the prevailing focus of the work;
- What were the objectives of the developed or analyzed solution.

Firstly, concerning the presence of user mobility, some publications had no mention of it, while others did take it into consideration in varying degrees of importance, with only Kimovski et al. (2022) having it as a central focus point, moreover, Liwang e Wang (2022) mentioned user mobility as a potential influence point, but did not actually consider it in their experiments. With respect to datasets used in experimental analysis, some publications generated their own, while other used real world data in simulator, furthermore, some did practical experiments, and did not use any dataset. In regard to the application focus, many of the works used data intensive applications as examples, while some had very generic classes of applications, such as Filho et al. (2022) with a Web Application. The focus of each application is the most varying aspect, and also the most subjective, thus will not have any conclusions and generalizations drawn from it. Finally, the objectives for defined/proposed solutions varied a bit too, but a general trend towards network metrics and more broad aspects such as perceived application performance was identified, with more internal metrics such as CPU and Memory usage not being as predominant.

Table 2 – Aspects investigated in each of the selected references.

Ref	User Mobility	Dataset	Application	Focus	Objectives
Liwang e Wang (2022)	Present but not considered	Not discussed	Multi-user on Edge-Cloud	Overbooking and trading computing resources	Optimizing processing and storage resource usage
Filho et al. (2022)	Not present	N/A (Practical experiment)	Web App	Stateful and Scalable distributed Applications	Minimizing latency and response time
Tanaka et al. (2022)	Not present	N/A (Practical experiment)	Chameleon Cloud Testbed	Edge-to-Cloud workflow automation	Minimizing makespan
Masip-bruin et al. (2021)	Considered	N/A (Real deployments)	IoT and Industry 4.0	Platform for managing Cloud-Edge deployments	Maximizing usability and scenarios/device support
Raith et al. (2022)	Considered	Generated workloads	Serveless Functions	Pressure-based framework for systems placement	Optimizing novel Pressure metric
Balouek-Thomert, Rodero e Parashar (2021)	Not present	Hurricane Sandy	Data-driven applications	Proposal of a stack for adapting distributed analytics	Fitting user-defined constraints and resisting runtime events
Balouek-Thomert et al. (2021)	Not present	Generated based on IRIS's seismic data	Seismic data processing	Decentralization of data pipelines	Minimizing latency costs caused by data movement
Belcastro et al. (2023b)	Considered	EdgeCloudSim simulation	Taxi Fleets location	Urban mobility tasks through Compute Continuum	Efficiently perform data analysis tasks
Kimovski et al. (2022)	Considered	Global Ping Statistics in WonderNetwork	Health-Care applications	Mobility-aware Multi-objective application placement	Optimizing completion time, energy consumption and economic cost
Belcastro et al. (2023a)	Considered	EdgeCloudSim simulation	Geotagged data analysis	Cloud-Edge architecture for managing IoT devices	Minimizing processing time, network delay, failed tasks and resource usage
Aguzzi et al. (2020)	Considered	N/A (Practical experiment)	Mashup application	Web of Things applications migration policies	Optimizing network overhead, CPU fairness, Thing fairness and interaction latency

Source: Developed by the author (2024).

Furthermore, the findings and observations made in this review of state-of-the-art and similarly focused literature are going to be used to better inform the decisions in the implementation of a simulation campaign, which is going to be discussed in great details in Chapters 4 and 5, in addition to the analysis of a potential research gap which takes place in Section 3.2

2.6 PARTIAL CONSIDERATIONS

In this chapter, many crucial concepts were discussed, in order to better substantiate the next chapters. Broad and well known concepts such as Cloud and Edge computing were analyzed in Section 2.1. After that, some basic notions of Applications architecture were examined in Section 2.2, including how network requirements can impact and influence such applications. Later in Section 2.3, the concept and repercussions of mobile users were reviewed. Section 2.4 introduced and explained the novel concept of the Cloud-Edge Continuum, which is one of the fundamental aspects of the proposal of the present work, which is elaborated in Chapter 3. Finally, in Section 2.5, a systematic literature review was conducted, in order to better understand the state-of-the-art of the Cloud-Edge Continuum, and to identify gaps and potential future research directions.

3 PROPOSAL

In the current chapter of this work, the proposal is elaborated and discussed, touching topics such as what is the problem in question, how it materializes, and in what application scenarios it's relevant. A more thorough analysis is conducted about the proposal and which aspects are novel, and need some technical explanation. The potential contributions of the present work are also discussed, and how they could be beneficial to the state-of-the-art literature. Later, the decision policy is discussed, and what concepts are needed for it to be viable, and what are the potential challenges of it. Alongside that, a discussion is taken about considerations on implementing the proposal, and how the transition from the proposal to the implementation is going to be conducted. Finally, a section with some partial considerations is presented.

3.1 APPLICATION SCENARIO

A widespread architecture for modern Applications deployed on the Cloud is a decoupled microservices architecture, which has individual service horizontal scaling as one of its main benefits (FRITZSCH et al., 2019). This makes this architecture ideal for scenarios where not all components of the whole application have the same characteristics, enabling each service to have different workload patterns and different degrees of user-facing communication, if any at all, usually manifested by single or small-scoped responsibilities for each one.

In some cases, developers or application administrators specifically create architectures designed to handle the decoupling of such services better. Materialization of these designs comes through services being very independent of one another, often implementing asynchronous communication exchanges. This means that users can be strongly linked to one service, but have practically no relationship with other services, all of this being transparent for said users.

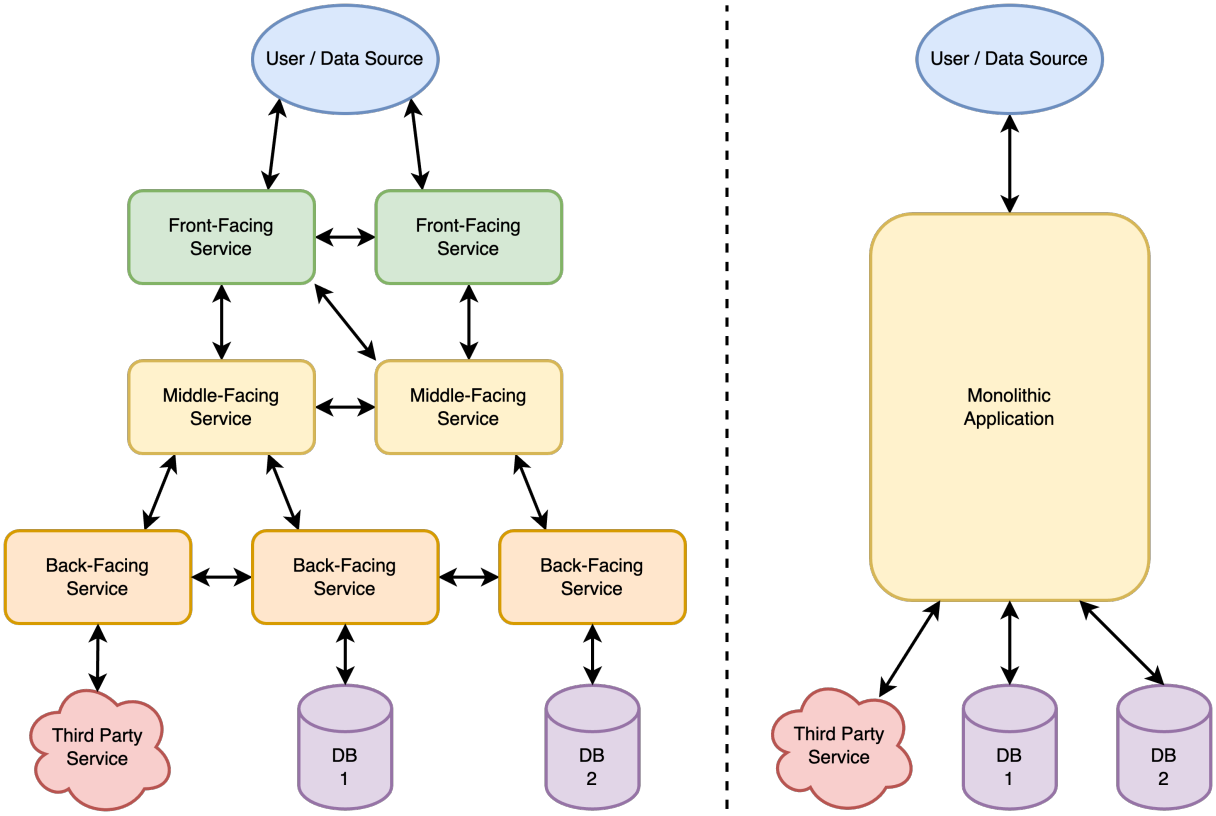
3.1.1 Monolith vs. Microservices

The decisions taken in order to achieve better reliability and availability are widespread, but at the same time, they could be explored further than they normally are. Unfortunately, such scenarios often don't take full advantage of the decoupling of those separate components, and more often than not, they are deployed in the exact same infrastructure, with the exact same capabilities. This pattern turns out very similar to a monolithic application, only with the added benefit of horizontal scaling.

A monolithic application is not necessarily a worse choice, but it has an enormous operational disadvantage, which is its inability to compartmentalize its functionality and responsibilities. This, in turn, makes it so that any functionality requiring some specific computational resource, applies a nonnegotiable requirement to all other parts of the application.

To illustrate and support the discussion, Figure 5 gives a simplified example of an abstract application designed in the two different architectures discussed previously: microservices and

monolith. In both architectures, there is the same number of external connections, including communication with users or any other data source, database channels, and even exchanges with a third-party service that provides some functionality to the application. Both approaches can accomplish the same features, although each has unambiguous trade-offs. In regard to our application scenario, the microservices approach stands out, it gives the necessary flexibility to characterize, classify, and parameterize the deployment of each service individually.



Source: Developed by the author (2024).

Figure 5 – Example of an application in a microservices architecture compared to a monolith architecture.

3.1.2 Network Requirements

Assume a hypothetical scenario where we have a stream processing application, consuming from many sensors at once, which in this case are data sources as shown in Figure 5, distributed throughout a large geographical area. The application has a constraint of needing to apply some pre-processing function to that data in a very immediate interval after it has been produced, otherwise, it is considered stale data, and it is no longer useful. Suppose the maximum time for data to be pre-processed before becoming stale is 50 milliseconds, the whole application would have to be deployed in the most central location considering all data sources, aiming to minimize their average distance. This would mean that those front-facing services (shown as the green rectangles in Figure 5) would have to be able to sustain connections (shown as the first two

edges from top to bottom) with all data sources in under 50 milliseconds consistently. This task has been proven to be very hard to achieve in public networks (JAISWAL et al., 2002), where jitter, traffic congestions, and general inconsistencies are extremely common.

As a supporting point for this fact, one could analyze the case with 5G networks, considering an Enhanced Mobile Broadband (eMBB) scenario. As per the International Telecommunication Union ((ITU), 2017), the requirements for such a scenario would be 20 milliseconds or fewer for the control plane latency, and an even lower 4 milliseconds for the user plane latency. Such values can not be consistently achieved in all the possible scenarios involving distributed and decentralized Data Sources, such as the hypothetical scenario used as an example previously.

By using only traditional Cloud Computing, even with multiple instances of those front-facing services, they will be very close to each other, so the margins for staying below that threshold get very slim. They are also statically allocated, meaning if the data sources need to move, they will instantly feel the connections with the application degrading.

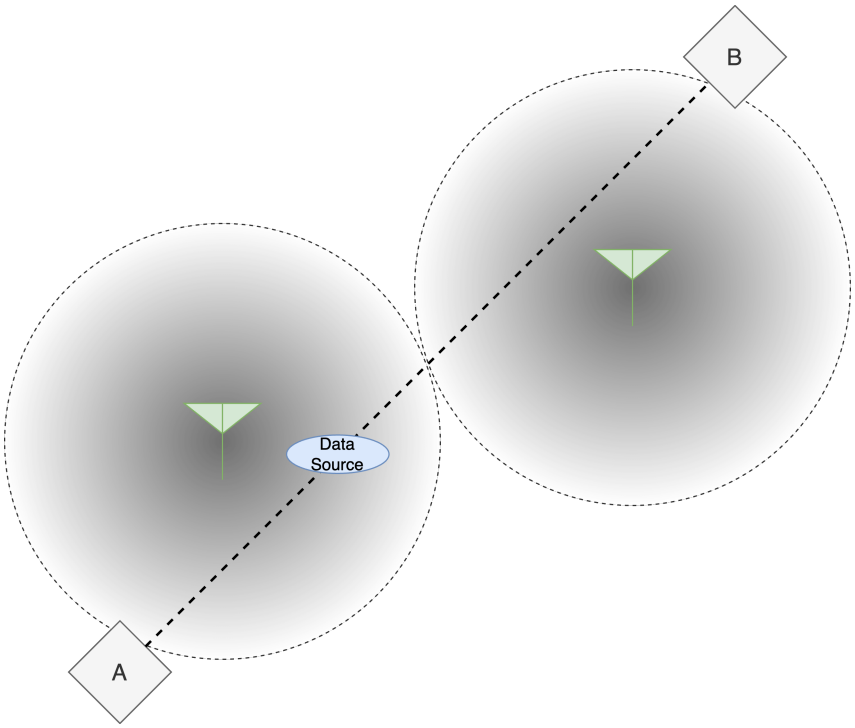
Nevertheless, trying to provision applications in such scenarios exacerbates the differences between building them in a Monolithic approach versus in a multiservice decomposed architecture such as microservices. Positioning such decomposed applications enables one to separate them into multiple Cloud or Edge Providers. In turn, this creates this new problem of optimal positioning which did not exist in a Monolithic Application scenario, where one could place the application with much less needed consideration, only needing to analyze the complete picture. Choosing which approach to take demands careful consideration of such dichotomy, among other aspects already discussed previously.

Following the same reasoning, a microservice architecture also creates the opportunity to make an analysis on top of the individual communication requirements of each component. Correctly reacting to each of those requirements can directly influence the performance of the final application. This possibility does not exist in the monolithic approach, and therefore is another advantage of going with a decomposed architecture for such classes and groups of applications.

Additionally, optimizing the first contact point of the User or Data Source with the Application is only but one aspect of the analysis. As already mentioned previously, optimally positioning the other components is really important to the macro-level behavior of the application. Just like the communication between the Data Source and the Front-facing services can have very tight network requirements, the internal communications from Front-facing to Middle-facing services and such can also have sensitive restrictions. Moving and disturbing one of them could create a necessity to adjust the other, which could in turn create a huge cascading effect inside the application. This means that repositioning one component could require the analysis of repositioning many others that are affected by that change in the architecture.

3.1.3 Mobility Awareness

Suppose a scenario similar to the one in Subsection 3.1.2, but in this instance, we have a mobile data source. Let's assume it is a sensor in some type of vehicle that needs to send its data to the application. Let's say that, for simplicity, this particular sensor that is being focused on is moving at a constant velocity from two points in a straight-line trajectory. From the application perspective, this data source is being serviced via the closest Edge Server located at some antenna, there are, however, other antennas capable of providing this service to the data source, but they are not currently being utilized because of cost-saving measures. Figure 6 shows this example scenario.



Source: Developed by the author (2024).

Figure 6 – Example of a Data Source moving from points A to B through the range of two Antennas.

When the data source moves far away enough from the first antenna, it will be outside its service range and therefore would not be able to communicate with the application. At this point in time, the application would need to reallocate and reposition one instance of its front-facing services, in order to not lose the data source's communication. In the case of the Figure, there is only one other antenna that could be used to reposition the service, and even then, there would be a period of transition between the connection from the first antenna to the connection to the second. There also isn't any guarantee that a second antenna would always be there, if the data source would, for some reason, travel in a different trajectory, it could leave the ranges of both antennas at the same time, rendering it unreachable from the service point of view.

With every data source and every user, careful consideration must be taken into account

when these present some degree of mobility. Unfortunately, many existing approaches only partially position components, as was discussed in Section 2.5. Partially positioning means they analyze the servers/antennas, and then they verify for network congestions, problematic traffic, and other problems. The present work aims to integrate this positioning and analysis into a single decision policy, which could allow for better start positions and also much more precise and resource-effective repositioning and re-allocations.

The principal idea in our application scenario is that every user or data source can have some basic degree of mobility within a specified area, and it will move periodically according to some subjective patterns that aim to mimic real-world behaviors. Such patterns could be simple point A to point B movements following some preconceived schedules, or any other algorithms. As discussed in Section 2.3.1, being able to simulate such patterns can be very beneficial for researching and developing any algorithm that acts upon user mobility.

Nevertheless, these mobility patterns are most often not random, and they can be generalized with modest confidence. The user/data source geographical distribution can be of huge help for the provisioning policies, benefiting from better placements of service instances. One aspect to consider is that mobility awareness in service provisioning is not necessarily only a reactive action. If one could clearly define the patterns in which the movement is made, one could feasibly predict the optimal service position, thus making a proactive decision.

Given the discussed scenario, many aspects of the present research have been discussed, including the fundamental literature and knowledge, presented in Chapter 2. Moreover, the application scenario was discussed in depth in this section. Section 3.2 tries to identify a research gap in the literature, which was reviewed in the previous Chapter's Section 2.5, and later the proposed solution, detailed in Section 3.3.

3.2 RESEARCH GAP AND POTENTIAL CONTRIBUTIONS

As discussed in Section 2.5, the literature has not approached all the elements cited above in the same proposal, especially the mobility-awareness aspect, which we consider very novel in this exact context. Further grounded by the findings demonstrated in Table 2, a clear gap exists in the state-of-the-art literature grasping all the cited aspects. Many of these aspects have been explored in exciting and promising ways individually, so one can only conclude that an amalgamation of all of them could potentially lead to great results. Exploring this literature gap is the main focus of the present work.

In Section 1.1, a broad initial hypothesis was constructed in some simplified terms, in Chapter 2 it was substantiated, and in the previous sections of the current chapter, the problems it aims to solve were further elaborated. Succinctly, the proposed hypothesis is that Service-composed Applications can noticeably improve their QoS metrics by being provisioned in the Cloud-Edge Continuum substrate and considering a novel prospect such as user geographical mobility into the decision policies. What is important from this summary, of the proposal is that

a decision policy that goes beyond simple placement of computational resources and congestion analysis can greatly outperform its counterparts.

This decision policy could potentially improve many aspects of the application scenario described in Section 3.1, such as the initial positioning of services, the most likely unavoidable reallocations/migrations of services, and the distribution of resources among the whole substrate, bringing a better overall processing/storage/network utilization. If some objectives of the present work could be achieved, it could bring valuable contributions to the community and the state-of-the-art literature.

3.3 DECISION POLICIES

In the previous sections, the problem's application scenario was discussed in detail, along with explanations as to why the problem exists, and how that scenario could be improved. By making the use of a decomposed into services arrangement for the application, which was called a service-composed design, and in practice is most often a microservice architecture, one could benefit from being able to configure and provision each individual component in a more granular way. This flexibility introduces by itself some potential perceived performance gains from the perspective of the application users, but it can be further explored if the possible substrates for component allocation are expanded to encompass the Cloud-Edge Continuum. Leveraging both Cloud and Edge computing at the same time can greatly improve the possibilities for best-fitting services, seen as every service can have very specific needs and restrictions, which would notably confine the application if it were to be allocated in only one of the cited substrates.

Furthermore, another aspect that can uplift the provisioning decisions is the awareness of user/data-source mobility, or in other words, the consideration of how, when, and to where the users move across some geographical location. Correctly anticipating and accommodating the application deployment and structure to this movement can lead to lower network traffic, better computing resource usage, and lower latencies and round trip times for user-service communications, consequently improving the overall end-users' performance indicators.

With the problem and its application scenario clearly stated, a thorough state-of-the-art literature review was conducted, together with an analysis of the aspects of each selected work. Although many aspects focused on the present work were approached, there was a clear missing conjunction of all of them, with each reviewed literature having a more narrow focus on a single or only a subset of the listed aspects. This fact was indicative of a suitable academic gap for research, it was sought to be explored in the present work.

The current section discusses and explains the many faces of a possible decision policy. Beyond that, some of the most important algorithms and processes for deciding the provisioning of applications are explored more abstractly, by using pseudocode, to better focus on the important aspects of the algorithms.

3.3.1 Onlineness of Provisioning

One of the more important facets of classical provisioning, scheduling, allocation, and similar problems is the onlineness of the whole procedure (FEITELSON; RUDOLPH, 1998). Being an online problem instead of an offline problem means that an iterative approach is necessary for taking the decisions in question. In other words, that means that the information about the whole scenario and execution is not all available at the start, needing to take action multiple times on different occasions. This iteration creates a sort of greedy approach to the problem, meaning that at any given moment, new information can reveal that very adversary actions must be taken to bring the scenario closer to the optimal solution.

Given the nature of the presented application scenario in Section 3.1, it is clear that the evolution of the application conditions (and similarly the users/data sources too) can constantly change. This change demands analysis and potential adjustments that could impact the position or distribution of resources for the multiple components of the application. In order to correctly decide upon these potential adjustments, some controlling policies have to be established, so that for each decision iteration, caused by the onlineness of the problem, an as close to optimal solution can be found.

3.3.2 Allocation and Reallocation

The necessity to iterate on the initial state of the application creates a conundrum of decisions based on the new information about the scenario. Initially, the decision on how and where to place the composing services of the application is taken by some policy algorithms, but after that initial state is defined, any new iteration is not as simple as repeating the same algorithms as in the beginning. This is because after initially allocating the services and distributing the resources to accommodate the application, any migration that could benefit the objective metrics, could also potentially cause disruptions or cost more resources than it would save (OUYANG; ZHOU; CHEN, 2018).

Any new migrations/reallocations should be analyzed not only on the merit of their potential benefit but also their inherent cost of execution, seen as constantly migrating a certain component could cause a pendulum (or ping-pong) effect. This is where some subpar performance calls for a migration, but after moving that component to a new state, the previous state then starts to perform poorly as well, so in the next iteration the component is moved back to its initial state, and this goes on indefinitely. As already discussed, the reallocation itself carries some performance cost, so this constant migration would drastically decrease the overall scenario performance according to the defined objective metrics.

3.3.2.1 Reallocation vs. Migration

It is important to discuss and elaborate on the distinction between a reallocation, as opposed to a migration. In the context of the present work, an allocation, and by extent a

reallocation, is the act of choosing and securing computational resources for a certain component of the system. The difference between the former and the latter is simply that one is the definition of the initial state, and the other is an iteration on top of that. In other words, a reallocation is the act of choosing which resources are going to be used for hosting the already accepted services.

Now for the migration, it is the actual technique used to take ownership of those chosen resources and typically means to change the physical/geographical location of the component that is being relocated. That is why it can be costly, because, in practice, a migration takes time, and necessitates many ephemeral/momentary actions to be taken, such as downloading the necessary components in the new location, or executing some extra work in the old location, so that new requests are redirected (RODRIGUES et al., 2019). This could even mean that, for some brief period of time, both the old and the new locations would cost their resource quotas at the same time. Specifically for the service-composed application's scenario, the migration process requires the download of container-related images for composing the new environment (SOUZA et al., 2022).

3.3.2.2 Allocation on Cloud-Edge

Another important perspective to consider in the decision policy is the underlying hardware substrate in which the components are allocated. As one of the focuses of the present work is the provisioning in the Cloud-Edge Continuum, it is valuable to assert the potential differences between those two distinct infrastructures. Per the previous discussion in Section 2.1, both kinds of infrastructure have their respective characteristics, such as the abundance of computing resources, and the speed at which those resources can be provisioned and obtained.

Just these two cited differences can already make an impact on the allocation of services in each substrate. For example, because the resources of each Edge Server are much more limited, allocating multiple services can cause a decrease in perceived performance due to resource competition or even starvation, so it is reasonable to try to keep the number of different services in the same server on the Edge to a minimum. In contrast, Cloud Servers are much more forgiving, as most of the time, the resources are all virtualized, and new resources can be transparently provisioned on the fly.

3.3.3 Mobility-aware Migration

Correctly considering, or even anticipating, the effect of users' movement throughout some geographical can lead to better overall resource usage and user-facing metrics, as already discussed in Section 3.1.3. A user/data source can have rather odd and irregular movement patterns at times, however, most of the time, their patterns can be successfully extracted into heuristics (OUYANG; ZHOU; CHEN, 2018). The data about future usage in a geographical dimension can lead to a more informed decision of reallocations and consequently migrations, although, integrating that into the general decision policy algorithm can provoke a huge increase

in algorithmic complexity due to the potential leap in search space.

There is also a potential rebound effect when constantly migrating services, as already discussed in Subsection 3.3.2, especially if based on user position. So beyond having to deal with the increased search space, one should also take into consideration a healthy interval in which a service should not be reallocated once again, or a service quality threshold, so that the rebound effect would be minimized (LIWANG; WANG, 2022).

3.4 TRANSITIONING FROM PROPOSAL TO IMPLEMENTATION

The presented proposal is a high-level view of the problem and the potential solution, in order to evaluate the feasibility of the proposed solution, especially doing so in a simulated manner as is the present work, some aspects of the proposal need to be translated into different abstractions of the originating concepts. This section aims to exemplify some of the more important aspects which need to be adapted in order to be simulated. Later on Section 4.1, other facets of this transition are discussed, for now, the main characteristics and concepts that need to be adjusted are as follows:

- Network requirements: Much has been said about network characteristics such as latency, jitter, and bandwidth, but these are not directly translatable to the simulation environment. In fact, simulating a fully distributed network hosting multiple applications with distinct requirements is a challenging problem (FLOYD; PAXSON, 2001). Starting with latency, the more appropriate metric to be used will be a generic delay, which could be drawn as a parallel to real world round-trip-time (two-way latency), but does not suffer from non-deterministic network degradation. Following this logic, jitter is therefore absent from the simulation, as it is defined as the variation in delay, which is not a concern in the simulation. Finally, bandwidth is considered as constant values for each link and server, and similarly, each service needs a fixed amount to be able to be allocated in a server, so in other words, there is no impact to and from bandwidth other that restricting which servers can host each service.

- Evaluation Metrics: The presented proposal intentionally does not go into much detail about how one could detect and evaluate the performance of the described scenarios, such as a service-composed applications with user mobility which could benefit from Cloud-Edge Continuum provisioning. This is because that is highly dependent on the specifics of the scenario and the available metrics to be measured. The same limitations to network characteristics when considered as requirements/restrictions, also apply when considering them as metrics which could be used to evaluate the performance, problems and bottlenecks of a scenario.

- User Mobility: The movement of users in theory is not something that should actually be simulated by the solution itself, seem as it is an external factor, which merely has

consequences on the scenario. However, being able to accurately portray and simulate user mobility can be very beneficial, even in a real-world scenario where this data is already available, seem as it can be used to predict future movements and further optimize the application provisioning. Furthermore, for a simulation environment, it is necessary to have some sort of model that can generate user movements, which can then be used to evaluate the performance of the proposed solution. As such, a user mobility model has to be implemented in order to simulate real world user mobility, the specifics of this model are discussed in a later section.

- Service Composed Applications: This concept, present in Section 3.1, is one of the main focuses of the present work. The services composing the application need to be modeled in a way that they can be allocated in the servers, while the application itself is only a collection of services, having no actual state of footprint. The services communicate with all the users of the application, but one key aspect which could not be expressed in the simulation environment, is communication between services. The specifics of this modeling are also discussed in later sections.

In the next sections in Chapter 4, more technical aspects of the simulation and its power simulator are discussed, as well as the necessary data sets for simulating scenarios discussed in this chapter. The decision policies discussed in this Chapter are expanded upon as well, with examples of algorithms used to model the decision-making process. Moreover, the chosen simulator was not actually capable of representing every details necessary, and so, some extensions and modifications were made, in order to accommodate for these missing features and characteristics.

3.5 PARTIAL CONSIDERATIONS

In this Chapter, we discussed the problem scenario in depth in Section 3.1, and demonstrated how all the different aspects approached in the present work fit together. Section 3.2 demonstrated that there is a potential research gap to be explored in the present work, and what are the potential contributions of doing so. Later in Section 3.3, we discussed more deeply how the present work aims to approach the problem, and how the different aspects can be dealt with. Finally, in Section 3.4, we discussed how the transition from the proposal to the implementation is going to be conducted, and what are the next steps to be taken.

There are a few points that need more discussing, for example, how the users' geographical data is accessed, measured and managed, and who owns that data. Or even the aspect of horizontal and/or vertical scaling of the services composing the applications, seen as that normally is internal information that stays within the application deployment, and the present work focus on the positioning of the application, which can be considered a separate dimension

to that of CPU or Memory usage. These aspects are not going to be discussed or considered in the present work, but they could potentially be explored in the future.

4 IMPLEMENTATION OF THE PROPOSAL

In order to better demonstrate and examine with the proposed concepts, discussed in Chapter 3, the present work aims to be able to conduct some experimental analysis, thoroughly substantiated by the discussions of core concepts in Chapter 2. We propose to utilize user mobility data through temporal and geographical axes to predict possible reallocations of services in Edge, while also leveraging concepts such as Infrastructure as Code to determine the Application's requirements, as discussed in Section 2.3.2. One of the main challenges is how to run experiments to a proper scale, taking into consideration the lack of access to a real substrate containing both enough Edge resources to demonstrate that proper provisioning can lead to gain in performance and other metrics, and also access to a Cloud infrastructure to properly leverage high performance computing. For this, the usage of an Edge Computing Simulator will be done, accompanied by the usage of real datasets of geographical distribution of Edge servers, base stations, antennas, and also, user mobility datasets and simulated models. We propose to pursue a classification of Applications such that the needs and restrictions for each of their services becomes clear and can be expressed declaratively, and also of an Edge substrate provisioning policy, to better accommodate with ever-changing user needs and demands.

4.1 SIMULATION

Distributed computing is a research domain that proves to be difficult to conduct experimental analysis, due to the massive extent of hardware, power, and general infrastructure needed to properly replicate certain scenarios. Beyond the economic reasons, there are other motivators to simulating experimental scenarios, such as the easier reproducibility and control over every major aspect and variable that could influence some experiment's results. This in turn gives more flexibility diversify scenarios and the ability to scale the simulation according to any desired equivalent real infrastructure.

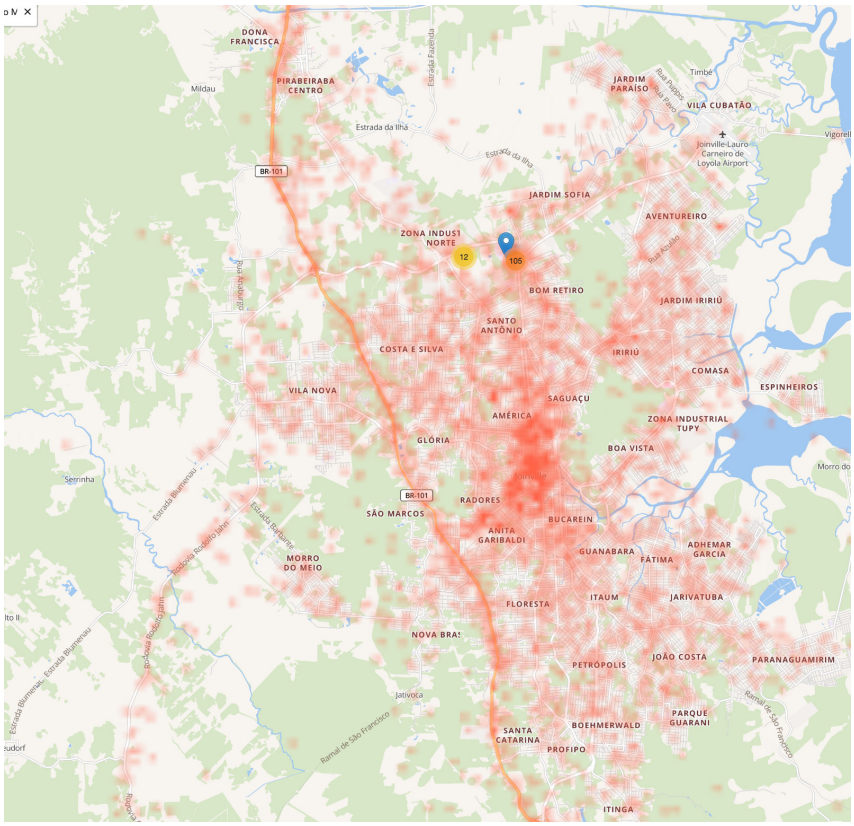
As formerly mentioned, the present work will apply the use of an Edge Computing Simulator to conduct the experimental analysis and search for empirical data to substantiate the proposed solutions and algorithms in Section 3.3. The simulator is called EdgeSimPy, and it consists of a "... framework for modeling and simulating resource management policies for Edge Computing environments developed in Python." (SOUZA; FERRETO; CALHEIROS, 2023). It is a great resource for experimental analysis because of its simplified prototyping model, which enables rapid testing and experimentation.

4.1.1 Antennas Dataset

The decision to create a new dataset based on many different sources was made due to the lack of a single dataset that could provide all the necessary information and characteristics expected for the present work. Combining data from various sources allows for a more compre-

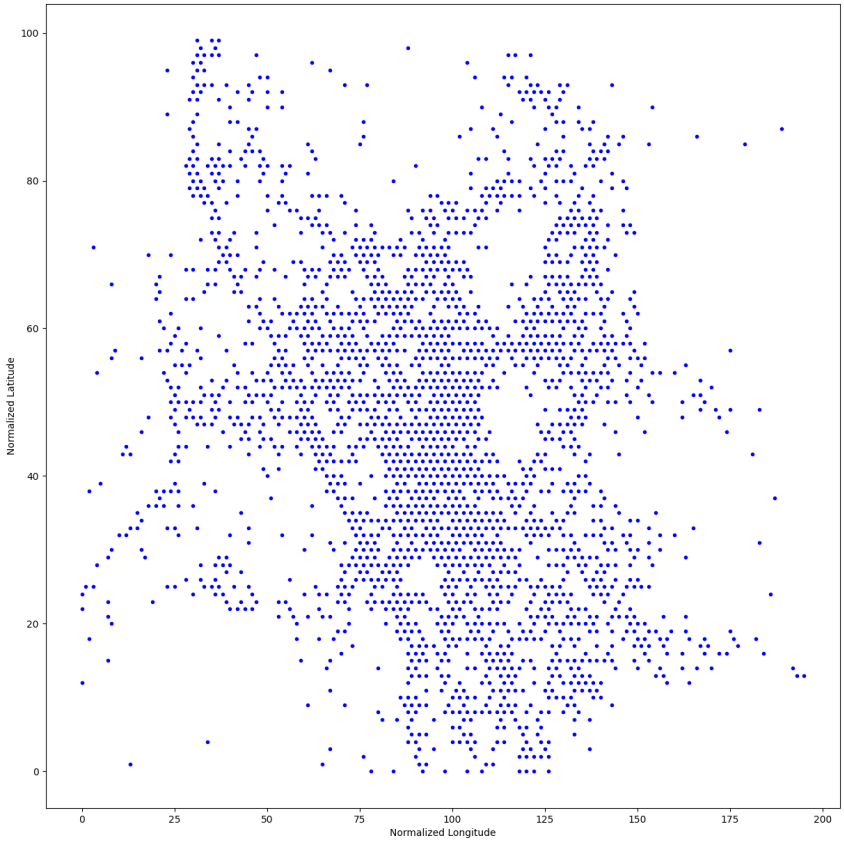
hensive and realistic simulation environment, which is crucial for the experimental analysis. The first component of the dataset is a base grid, which is constructed using an open cellular dataset, from OpenCellid (<https://opencellid.org>). This dataset is crucial to creating a simulation grid that closely replicates reality's network coverage patterns. It contains GPS data from mobile service antennas, or cell towers, their corresponding coverage information, Location Area Identity (LAI), and the technology of each one. We selected the city of Joinville, the third-largest municipality in the southern region of Brazil (total area 1,131 km^2), for composing the use case and for supporting the simulation campaign (detailed in Section 4.1).

Figures 7, 8, and 9 shows the transformation from the raw dataset to the grid used by the simulator, which was constructed by taking a cutoff of the coordinates of the city of Joinville, and downsampling the existing antennas into a smaller grid size of 100 by 100 points. The structure of the transformation is as follows: Figure 7 represents the cellular towers collected from OpenCellid (UNWIREDLABS, 2024) project. Assuming that edge servers are jointly placed with cellular towers, Figure 8 exemplifies a possible physical placement, in which blue dots represents edge servers. All possible coordinates for edge servers are then randomly sampled into a smaller sub-set, as shown in Figure 9. Figure 9 also shows 4 other blue dots in the far upper right corner, which represents Cloud servers, placed outside the normal grid, to represent the distance from traditional computing power originating from Data Centers. At the same time points of interest are generated into the resulting grid, these are further explained in Section 4.1.2.



Source: Developed by the author (2024).

Figure 7 – Cellular towers reported by OpenCellid project.



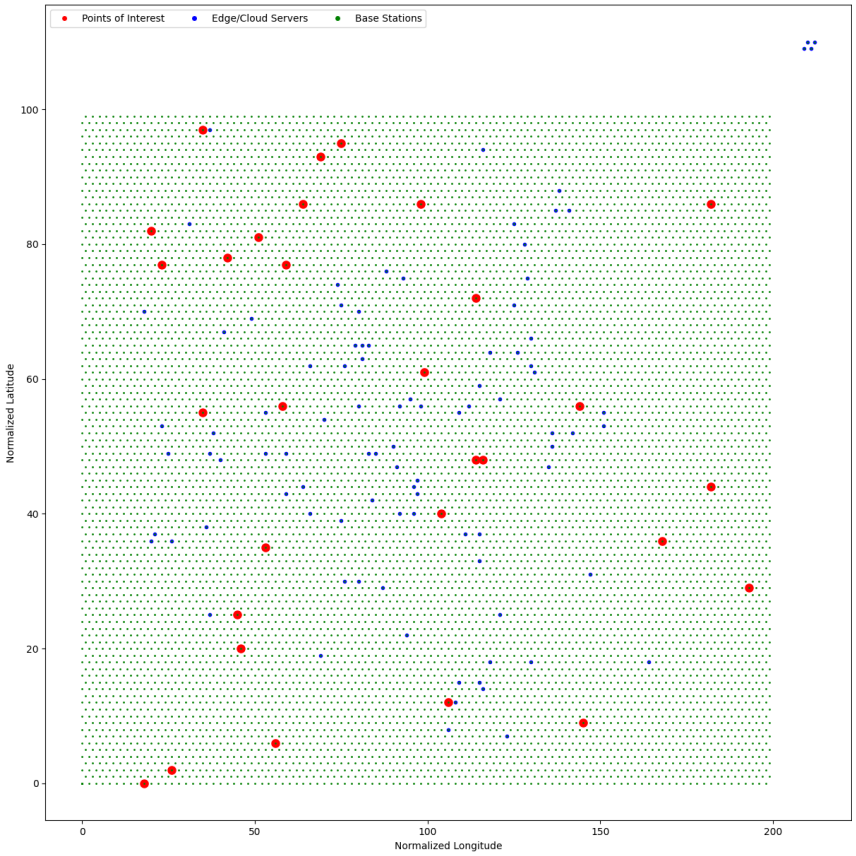
Source: Developed by the author (2024).

Figure 8 – Example of Edge servers' distribution into discrete grid.

All base stations, the green dots in Figure 9, are connected to each of their neighbors via links with 10 units of bandwidth, and 1 unit of delay, with the exception of the base stations where Cloud servers are placed, which have a bandwidth of 100 units, and a delay of 10 units.

4.1.2 User-Mobility Model

In order to take advantage of the base grid constructed, a User-Mobility model not previously present in the simulator was implemented. It is a Point-of-Interest (POI) based model, where the list of users can interact with a list of points of interest, created to allow for more modeling realistic movement profiles. Each POI represents a concentration point, in which users from the same city tend to group together at specific times of the day. Some users are more interested in universities and colleges, while others move more frequently to commercial and industrial locations. POIs are more attractive at certain times of the day: while industries and companies follow business hours (for example, starting work at 8:00 am), universities have night hours, that is, the model represents the behavior of a city. To represent this idea, we generate a number of points in the grid. These points of interests have a fixed position, and also have a schedule, that is composed by periods of time when it is considered in peak, meaning that it will potentially attract users. Each user starts at a random position in the grid, and as the simulation progresses through each step of it, the users has a chance of becoming engaged with any point of



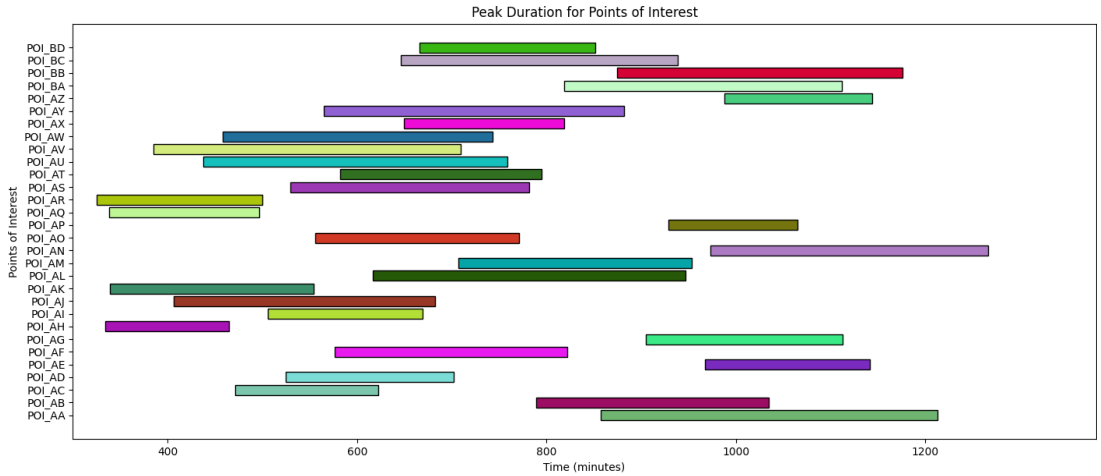
Source: Developed by the author (2024).

Figure 9 – Down sampled Edge servers and Cloud servers (blue) and generated Points of interests (red).

interest that is currently in its peak period. While engaged with a POI, the user will slowly move towards it, at a random fixed-range speed. After the peak period of the point of interest is over, the user disengages with it, and can now engage with another one at the next simulation step.

Figure 10 shows the generated points of interest, each with a different peak period, with different peak durations as well. Time is discretized in minutes, removing the initial and final windows (night period), and some POIs overlap in time intervals. The peak of each point of interest repeats day after day, and for better results, the generation had a minimum and a maximum duration for each, with a starting time and a cutout time as well, which is why the x axis starts at roughly 300 minutes and stops at around 1280. Each user is associated with a subset of POIs, and move to/from based on dynamically accounted interests.

This type of model has a resemblance to WHERE models (SOLMAZ; TURGUT, 2019), discussed in 2.3.1, with the difference that the users do not have predefined paths, and can interact with any point of interest at any time, creating slightly more chaotic/non-deterministic movement. This kind of model was deemed more appropriate for the present work, as it is representative of real-world user behavior, while being more challenging to placement algorithms. It is a good fit for representing big cities and metropolis, where users can interact with a multitude of different services and points of interest.



Source: Developed by the author (2024).

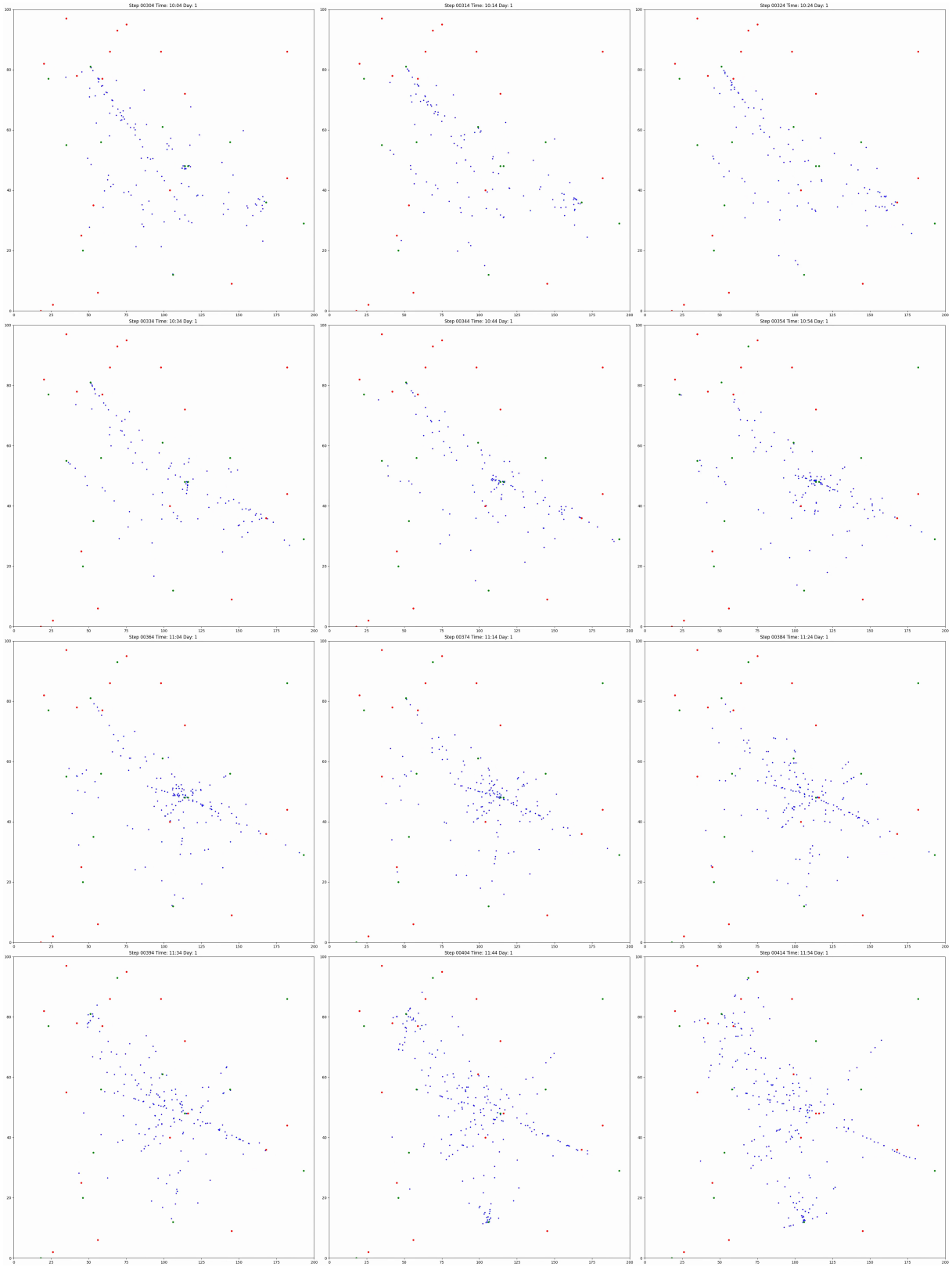
Figure 10 – Generated points of interest with their respective peak periods.

The number of users in the simulation is a variable parameter in the dataset generation, varying from 10 to 30 for each application. For the simulation campaign presented in Chapter 5, the dataset was only generated once, and reused for all different scenarios, in order to minimize non-deterministic behaviors caused by differences in the input data, in that case, the exact number of users was 288 in total. The speed of each user is also randomly defined in the dataset creation and can vary from 0.3 to 1.0 units of distance per time step. Figure 11 shows the users moving towards the grid, in the direction of points of interest, with the users represented by the blue dots, and the points of interest represented by either red dots (not in their peak period) or green dots (in their peak period). As the simulation progresses, the users can become interest in certain Points of Interest in the grid, and move towards that point. However, they can be interested only in a point currently in its peak period, and as this period ends, they disengage and can move towards another point. The figure represents twelve snapshots of this process, each separated by a time interval of 10 simulation steps, which is equivalent to 10 minutes. A full video of two entire days of simulation is available at <https://www.youtube.com/shorts/oXkZJhB2Dno>.

4.1.3 Simulator Extension

The chosen simulator, EdgeSimPy, is a Python-based simulation framework, but one of the reasons that it was chosen was its extensibility. It enables the creation of new models, algorithms, and policies, which can be easily integrated into the existing codebase. However, by actually working on an alternative version of its codebase, known as a fork, deeper changes to how the simulator operates are made possible.

As well as constructing the grid and mobility model for simulations, another aspect worked on is the fact that EdgeSimPy originally is an Edge Computing simulator, not Cloud or Cloud-Edge. This limitation prompts an extension and augmentation of the original implementation, thankfully, the implementation is rather extensible and straight forward to modify, thanks to the aforementioned fork of the simulator's codebase. The Cloud computing domain



Source: Developed by the author (2025).

Figure 11 – Users moving towards the grid to points of interest.

is represented via servers with significantly more computing power, but much worse latency capabilities. They are in fact, positioned outside the normal grid, which represents the distance

from traditional computing power originating from Data Centers. This can be seen as a simplification of the real world, but for the purposes of the present work, it is deemed a sufficiently realistic representation. The differences between Edge and Cloud servers are notable, but to keep a degree of standardization, the both follow templates of the same type, only varying in the number of resources available, and the latency to the users. The exact templates can be explored by accessing <https://github.com/paulora2405/espy-user-mobility>, which is the repository for the setup and execution of the simulation environment, it also contains a submodule for an alternative version of EdgeSimPy. To summarize, there were a total of 90 Edge servers, and 4 Cloud servers, all of which were generated once, and reused for all different scenarios in the simulation campaign as well.

The output model used by the simulator was also modified. The addition and modification of which metrics are outputted by the simulator is crucial to the experimental analysis, and was of great significance in order to correctly measure the performance and results of the proposed algorithms. In addition to the exported metrics, the output format was also modified, from the `msgpack` format, to Command Separated Values (CSV), which is more widely used and easier to manipulate in post-processing. This was done to better suit the needs of the various ingestion and processing scripts and flow created.

4.2 DECISION ALGORITHMS

In Chapter 3, we discussed the need for a decision policy that could take into account the user mobility data, and the application requirements, in order to make the best possible decision regarding the allocation of services in the Edge infrastructure. The decision policy is a crucial part of the present work, as it is the main mechanism that will enable the proposed solution to be viable. These policies are dictated, in practical terms, by decision algorithms, which are the actual implementation of the policy in the EdgeSimPy simulator.

There are a few algorithms that dictate the functioning of the present proposal, to some extent, they can be neatly classified into steps in the provisioning process. The first is the definition of the initial state, or in other words, the allocation policy for new applications, which at the beginning of the applications' life, is all of them. The second is the policy for existing applications, henceforth called reallocation, which must be reevaluated periodically, in order to maintain the defined SLAs. Both of these steps can be executed using a single algorithm, with the difference that the first step has no prior state or history of execution.

A pseudocode representation of both steps is demonstrated in Algorithms 1 and 2. Initially, the reallocation step (Algorithm 1) contains other internal steps itself, starting by creating a list of services from all applications, which fit into some restrictions. In this sense, instead of analyzing applications individually, the algorithms can make decisions regarding all services' requirements, at a given time instant. If a service is currently in the process of being migrated, it cannot be interacted with until that process finishes. Likewise, if a service was recently migrated, it must

not be migrated again until a certain period of time has passed, this period is defined via an input to the algorithm, called migration recency threshold. Moreover, if a service has experienced a decrease in some SLA metrics, defined via an input to the algorithm, but this decrease is not very significant, in other words, it does not go below some thresholds, also defined via an input, it should not be migrated. All of these three restrictions are made primarily to decrease the chance of the previously discussed pendulum effect occurring. After that filtering step, for each of the remaining services, we have to actually allocate them at some server in any of the available infrastructure. By looking at the potential improvement of each available server, and selecting one, the service is them relocated. Essentially, the pseudocode is externally parameterized by any Edge-Cloud Continuum orchestrator, which can inform the thresholds, SLA requirements, and objective function.

Algorithm 1: Application Reallocation algorithm $reallocateApps()$

Input: A set of service composed applications $\mathscr{A} = \{A_1, A_2, \ldots, A_n\}$
Input: A recency period threshold $\mathscr{P}$ expressed in some time unit
Input: A set of all available Servers $\mathscr{I}$
Input: SLA objectives $\mathscr{O}$
Input: A threshold of minimum values for the selected SLAs $\mathscr{T}$

```
servicesList ← joinAll(𝒜.services);
filteredList ← filter(s ∈ servicesList, s.isNotMigrating());
filteredList ← filter(s ∈ filteredList, s.notRecentlyMigrated(𝒫));
filteredList ← filter(s ∈ filteredList, s.decreasingSLA(𝒯));
for service ∈ filteredList do
    allocateService(service, ℐ, 𝒪);
end
```

The allocation of the services into actual servers is a subsequent step, which is once again represented as pseudocode in Algorithm 2. This algorithm is straight forward in its execution, but it uses another, more complex algorithm to select the best server candidate for the currently input service. Selecting a good server candidate it not at all trivial, and in the context of the present work, takes into consideration a lot of factors, including the users' geographical information.

Algorithm 2: Service Allocation algorithm $allocateService()$

Input: A Service $\mathscr{S}$
Input: A set of all available Servers $\mathscr{I}$
Input: SLA objectives $\mathscr{O}$

```
selectedServer ← selectServerCandidate(𝒮, ℐ, 𝒪);
selectedServer.allocate(𝒮);
```

The selection of the best candidate server for that service is represented as pseudocode in Algorithm 3. One of the complexities of the present work is being aware of users' mobility and position, but in order to do this in a reasonable amount of time, the search space must be somehow optimized. There are a number of different ways this could be achieved, one could use

a clustering algorithm to group users of a service, and select servers closest to the cluster's center. Another way would be to calculate a normalized score of the distance from all users of a service, to all available servers in the infrastructure, and select which would yield the lowest score. In this case, the second approach is deemed simpler and quicker to execute. In the next step, we try to find the best fitted server for the SLA objectives in the server group which is closest to the mass of users. After that, the best candidate server is used in the previously discussed Algorithm 2 to allocate the input service.

Algorithm 3: Server candidates selection algorithm $selectServerCandidate()$

Input: A Service $\mathscr{S}$
Input: A set of all available Servers $\mathscr{I}$
Input: SLA objectives $\mathscr{O}$
Output: A selected Server candidate $\mathscr{C}$ suitable for $\mathscr{S}$

1 $canditateServers \leftarrow sortByShortestDistance(\mathscr{S}.users, \mathscr{I})$;
2 $\mathscr{C} \leftarrow bestFitObjective(candidateServers.first, \mathscr{O})$;
3 $return\ \mathscr{C}$;

4.3 PARTIAL CONSIDERATIONS

In this chapter, we have delved into the implementation aspects of the proposed solution, focusing on the simulation environment and decision algorithms. We began by discussing the necessity of using an Edge Computing simulator, EdgeSimPy (SOUZA; FERRETO; CALHEIROS, 2023), to conduct experimental analysis due to the challenges of replicating real-world distributed computing scenarios. The simulator's extensibility and simplified prototyping model make it an ideal choice for our simulation campaigns.

We then explored the construction of a realistic simulation grid using an open cellular dataset from OpenCellid (UNWIREDLABS, 2024) as a base, specifically focusing on the city of Joinville. This dataset is crucial for creating a simulation environment that closely mirrors real-world network coverage patterns. The transformation process from the raw dataset to a usable grid was detailed, highlighting the importance of accurate geographical representation.

After that, the User-Mobility Model was introduced as a novel addition to the simulator, enabling more realistic movement profiles based on Points of Interest (POIs). This model captures the dynamic nature of user behavior in a big city, making the simulation more representative of real-world scenarios. The generated POIs and their peak periods were visualized, demonstrating the model's capability to simulate user interactions with various city locations.

We later discussed the extension of the EdgeSimPy simulator to include Cloud Computing capabilities, allowing for a more comprehensive analysis of Edge-Cloud Continuum scenarios. Modifications to the output model were made to better suit the needs of our experimental analysis, ensuring that the performance and results of the proposed algorithms could be accurately measured.

Finally, we presented the decision algorithms that form the core of our proposed solution. These algorithms are responsible for the initial allocation and periodic reallocation of services based on user mobility data and application requirements. The pseudocode representations provided a clear understanding of the steps involved in making optimal allocation decisions, emphasizing the importance of considering geographical information and SLA objectives.

In summary, this chapter has laid the groundwork for the experimental analysis by detailing the simulation environment, user mobility modeling, and decision algorithms. These components are essential for validating the proposed solution and demonstrating its effectiveness in real-world scenarios.

5 SIMULATION CAMPAIGN

On Chapter 4, we presented the implementation of the simulation environment, which is the basis for the present work's evaluation. This Chapter will present the simulation campaign, which aims to evaluate and test the proposed provisioning policies and the novel user-mobility model in a variety of scenarios. We discuss which metrics are relevant and used in the simulator context, the parameters and configurations initially tested in order to arrive at a baseline scenario, which is posteriorly compared with other proposals, including solutions from the reviewed literature. After discussing and collecting the relevant data in the simulation campaign, we present the key findings involving the implemented solutions, interpreting the results and discussing the implications of the findings. Beyond that, we take some considerations on what would be necessary to implement the proposed solutions in a real-world scenario. After that, we present some partial considerations on the simulation campaign, discussing the results and potential next steps to be taken.

5.1 SIMULATION SETUP

All simulation runs were executed in a dedicated research server, posing a `x86_64` dual socket Intel Xeon CPU E5-2620, each with 6 cores, 12 threads running at 2.00GHz base clock, with 160 GB of DDR3 Ram. In terms of software, the simulations were executed in a Linux environment, using Python as the main language for the simulator EdgeSimPy, running a fork of the 1.1 version, as well as new custom wrapper for executing, creating datasets, scenarios, generating results, and plotting graphs. The execution of every parameter combination, including calibration and comparison runs were executed in a span of several weeks, as some certain scenarios could surpass the 36 hour mark in terms of execution time. Every run result from all scenarios were collected and archived in comma-separated values (CSV) formatted logs, and subsequently aggregated, processed and summarized into plots using modular Python scripts.

5.2 METRICS

As discussed in Section 3.4, the metrics used to evaluate performance, potential problems, and eventual bottlenecks in any scenario are completely dependent on what can be extracted from the simulation environment. For this reason, we selected two metrics which are vital to the internal operation of the decision algorithms discussed in Section 4.2. The two selected metrics for composing and evaluating the simulation campaign are:

- Delay from users to applications: which consists of the sum of delays from each network link between a user and the services of an application they are interacting with.
- Normalized Distance from services to users: which is a value between 0.0 and 1.0 that represents the average distance of the mass of users of a particular service, to the server

where that service is currently allocated. It is dictated by the formula: $\frac{\sum_{i=1}^{N} D_i}{N \times D_{max}}$ where N is the number of users a service has, and D is the Euclidean distance between the user and the server where the service is allocated. This formula was created as a heuristic to approximate the distance from service to users in terms of network hoops, because the latter was to computationally expensive to calculate in the simulation environment.

These two metrics resonate with the main objective of the present work, which is to improve the overall performance of the Service Composed Application by optimizing the placement of its composing services. A smaller delay implies a better and more responsive experience for the users of some application. The delay itself is correlated with the distance from the service to its users, and so in short, a lower distance in general means a lesser delay as well. Both of them are intrinsically connected to the characteristics of distributed applications, such as service-composed ones, as discussed at length in Chapter 3. Constructing mobility awareness is done so via mainly these metrics, as a way to capture aspects of the users' positions, relative to the services they are connected to. Being able to detect a bigger delay in that communication, and being able to pinpoint that to a larger distance, or just random network noise is crucial to a correct user mobility consideration. This correctness is a fundamental aspect of the present work, and through the usage of the Delay from users to applications, and Normalized Distance from services to users, we can achieve algorithms that can perform decisions with bigger confidence figures.

5.3 PARAMETERS AND CONFIGURATIONS

Correctly calibrating a simulation environment is vital to the quality of the obtained results. In order to be able to achieve a certain degree of precision and overall quality in our simulation campaign, a calibration process takes place before any real proposal evaluation. The parameters used to vary the scenarios of the simulation campaign are significant individually, and Table 3 discusses both of them in detail, including a description about them, which values were initially chosen and tested, and the rationale behind their selection.

5.3.1 Calibration

The first scenario in the simulation campaign aims at identifying the appropriated parameters for configuring the environment. In this sense, we measured the total distance from services to users when orchestrating a baseline scenario, varying the internal thresholds. The implementation of the placement heuristic itself is fairly straightforward, and consists of a simplified version of Algorithm 3, with the only difference being that the reallocation is done in a greedy worst fit fashion, selecting the first server that yields a better distance metric, while fitting the requirements, with the list of available servers being sorted by available resources, in decreasing order. In Figures 12 and 13, we present the preliminary results from the simulation

Table 3 – Parameters used for calibrating and evaluation the baseline scenario.

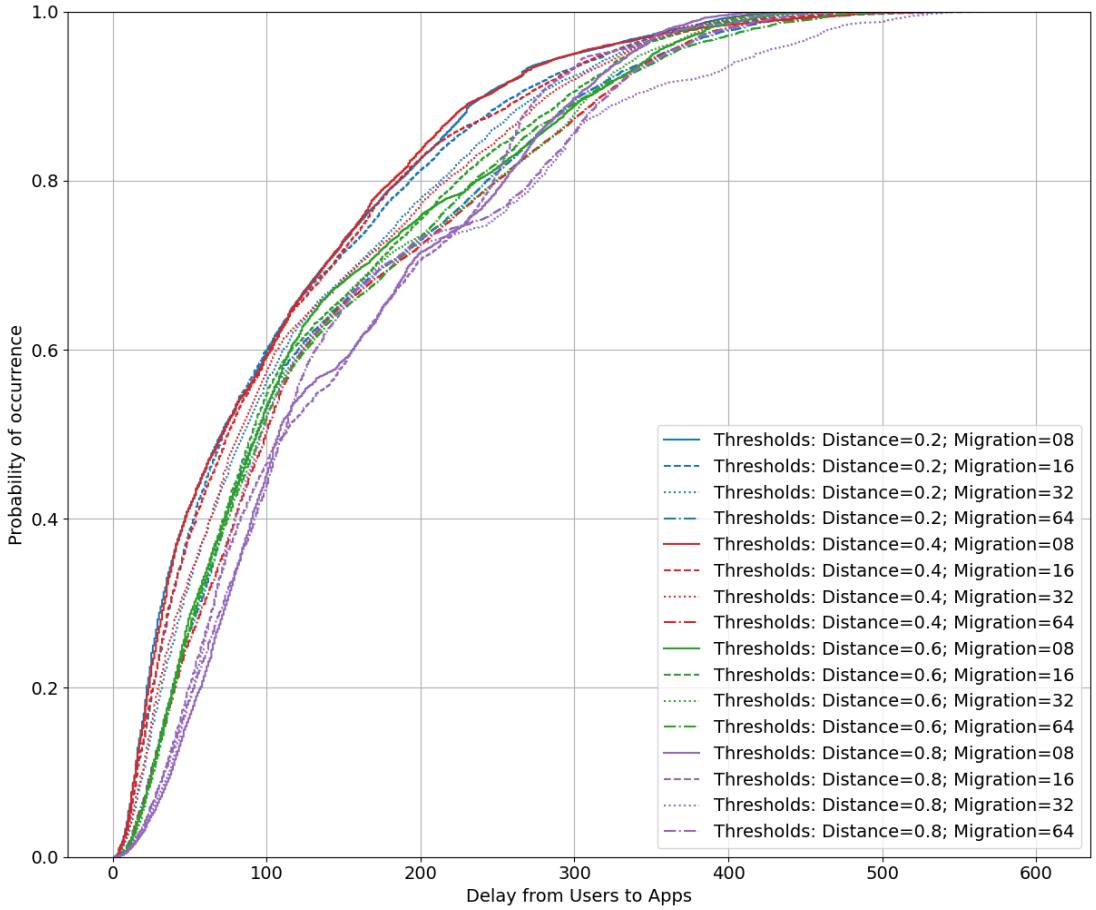
Parameter Name	Migration Recency Threshold	Normalized Distance Threshold
Description	The minimum time before a recently migrated service can consider migrating again. The unit used to measure it is simulation steps, which are equivalent to minutes in a ratio of 1 : 1.	The minimum normalized grid distance between a service and its users before the service is considered for migrating. The distance from all users of a service is averaged, and normalized to a value between 0.0 and 1.0.
Values	8, 16, 32, 64	0.2, 0.4, 0.6, 0.8
Rationality	Allowing for services that have just been migrated to be migrated again instantly can lead to over migration, or a ping-pong effect, as discussed in Section 3.3.2.	A minimum degradation of service from the perspective of the users must exist, in order to avoid trying to achieve a perfect placement, which in almost all scenarios is impossible, seem as a service has multiple users connected to it, many of which are in vary distinct positions in the grid.

Source: Developed by the author (2025).

campaign, which are used for calibrating the environment's parameters. They are presented in a Cumulative Distribution Function (CDF) graph.

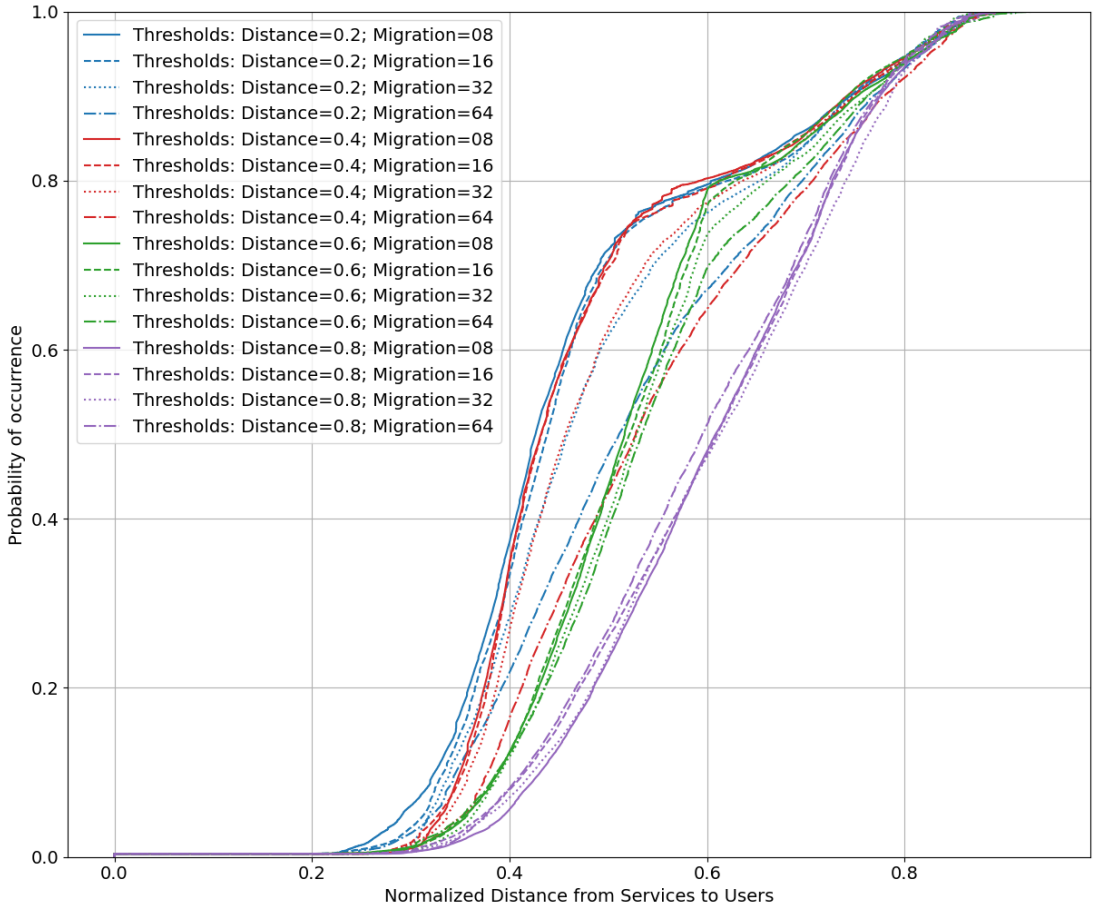
By analyzing these preliminary results, we can identify that a few parameter combinations produce better overall metrics, such as lower distance threshold and lower migration recency threshold. Regarding the migration recency threshold, the lowest values of 8 time steps was the best for the large majority of combinations, so this is the only value which will be used in the comparison campaign. Regarding the normalized distance threshold, a value of 0.8 seems to be too tolerant to high distances, so we will only use values of 0.6 and bellow in the comparison campaign. With this preliminary testing, we can decrease the number of combinations from 16 to only 3, which will make the comparison campaign more straightforward.

In terms of the users' perspective, a higher distance from the services they are connected directly implies in a lower quality and consistency of the provided service. So a lower distance is always better, but not all users connected to a certain service are going to be in the same region of the city, and so, the service has to optimize to providing the lowest average delay, which means sometimes having a higher distance to a few users, while maintaining a lower one to the majority. This is precisely why we have constructed this Normalized Distance metric. As users move throughout the city, they naturally form clusters, especially if many of them are engaged with the same points of interest, or to multiple that are in proximity, and so, readjusting the service placement ensures that the distance stays as low as possible to the big mass of users.



Source: Developed by the author (2024).

Figure 12 – Delay from users to applications. Part of results from simulation campaign used for calibrating environment parameters.



Source: Developed by the author (2024).

Figure 13 – Normalized distance from services to users. Part of results from simulation campaign used for calibrating environment parameters.

5.4 COMPARISON WITH LITERATURE'S PROPOSALS

After the initial calibration campaign, we introduced two new scenarios for comparing our proposal with proposals found in the reviewed literature, in addition to the one used in Section 5.3.

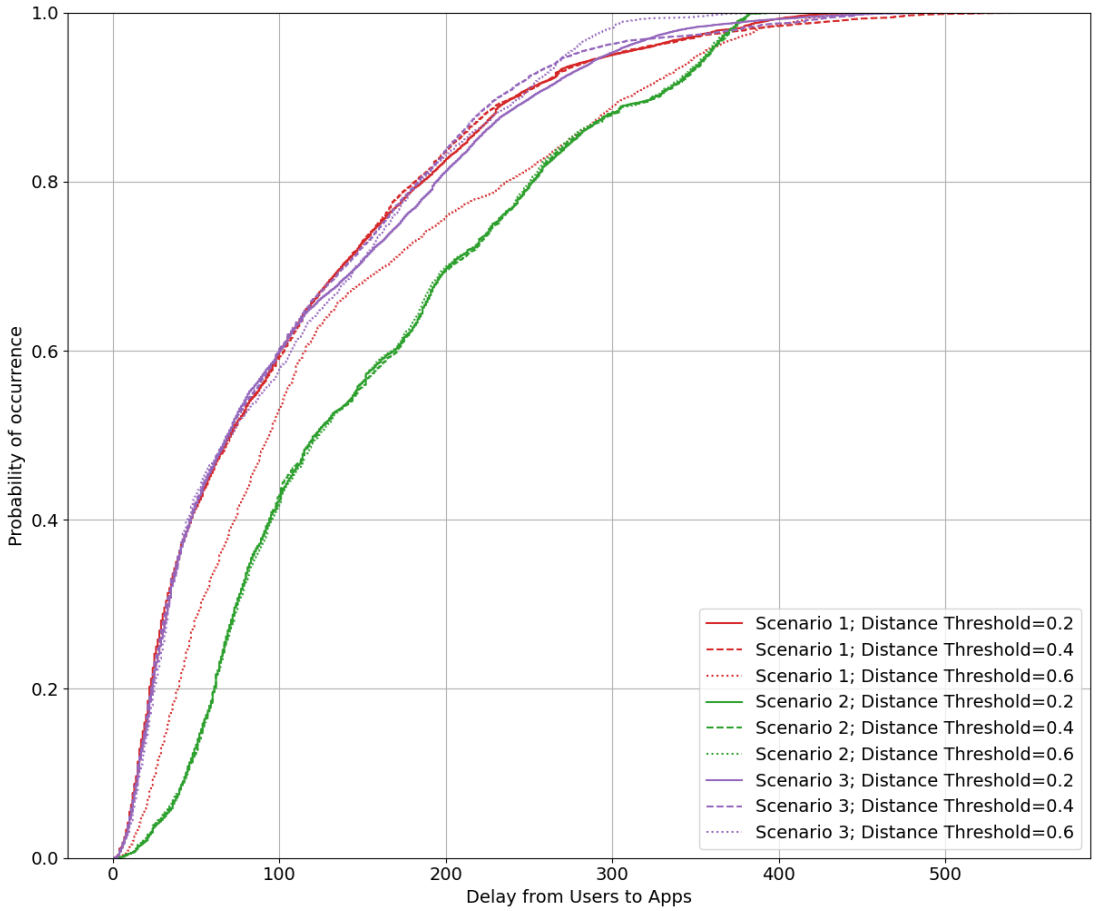
- Scenario 1: is identical the baseline used in the calibration campaign, and it consists of a worst fit algorithm which finds the first server that can accommodate the service, and has a lower normalized distance.
- Scenario 2: is a no-reallocation approach, which does not consider the normalized distance at all for reallocations, which in turn means that the mobility of users is ignored by the service placement algorithm. This type of placement algorithm is similar to solutions used in works such as Liwang e Wang (2022) and Balouek-Thomert, Rodero e Parashar (2021), where user mobility is not considered in decision making.
- Scenario 3: is similar to scenario 1, but with an improved server filtering, which is described in Algorithm 3, where servers are sorted by the lowest normalized distance from the mass of users to a service.

The heuristic used in Scenario 2 may seem rather naive, but for certain systems trying to optimize other metrics, such as energy consumption, it could be a valid approach. For example, in Masip-bruin et al. (2021), dynamic infrastructure provisioning for scaling services is proposed, but not necessarily considering the users' mobility itself.

In Figures 14 and 15, we present the results from the simulation campaign, represented once again as CDF graphs. All three scenarios were run with the same combinations of parameters, 8 steps of Migration Recency Threshold, combined with 0.2, 0.4, and 0.6 of Normalized Distance Threshold.

There are a few possible points to be inferred by the results presented in Figures 14 and 15. Firstly, the naive approach (Scenario 2) proved to deliver higher delays from applications to users, which is expected, as it does not take into consideration the users' mobility, so as soon as the users move, the services are not reallocated to accommodate them. In this case, the variation of the normalized distance threshold does not seem to have a significant impact on the results. Scenarios 1 and 3 were similar in terms of delay from applications to users, having a slight advantage for Scenario 3. One notable thing is that for both of these, the 0.6 distance threshold seems to be a little too tolerant still, and inversely, the 0.2 threshold seems to be too strict, with practically no percentage of distances lower than 0.2, in other words, it is virtually unachievable.

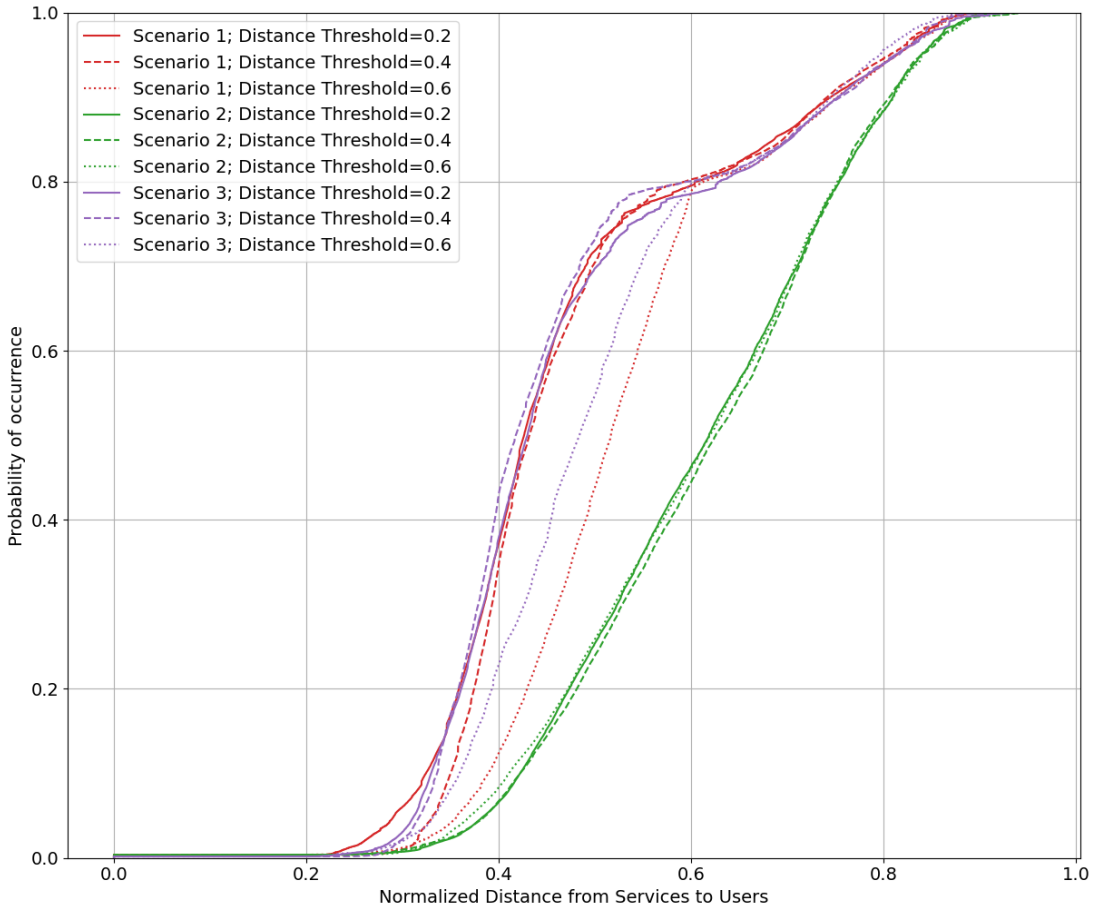
Another aspect which was not yet touched upon, is the simulation complexity, which was measured by comparing the execution time of each variation of the simulation campaign. The more complex Scenario 3 took about twice as much time to execute the same step limit as the



Source: Developed by the author (2024).

Figure 14 – Delay from users to applications. Part of results from simulation campaign with 3 different scenarios.

more intermediary Scenario 1. This fact, combined with the very close results shows that the algorithmic efficiency is very relevant in this context, and thus, can lead to the inference that a good enough result can be obtained with a not so complex heuristics.



Source: Developed by the author (2024).

Figure 15 – Normalized distance from services to users. Part of results from simulation campaign used for calibrating environment parameters.

5.5 REFLECTING ON REMAINING METRICS

In Sections 5.3 and 5.4, we discussed the metrics used in the simulation campaign, and how they were used to evaluate the performance of the proposed heuristics. However, the chosen metrics were not the only ones available in the simulation environment. In order to further demonstrate the relevance of the chosen metrics, this section presents a reflection on the remaining metrics, and the reasons why they were not included and used as evaluation criteria.

Some discussion about the outputs of the simulator took place in Section 4.1, yet, the specific data that is output was not approached. The simulator exposes metrics from many entities from the environment, such as the applications, services, users, servers (both edge and cloud ones), network switches, network links, network flows, and points of interest. Some of these entities, although relevant in a conceptual level, simply did not have a direct impact on the simulation, thus not producing any relevant data, such as network links and switches. Other entities were redundant, seem as the relevant data they produced was already captured by another entity, such as the network flows, which its important data could be traced back to the services and users. Entities such as Point of Interest, although relevant in the context of the simulation

execution itself, did not produce data important to the results of any scenarios, because its data was not dependent, nor impactful to these scenarios.

For the entities that were chosen to be evaluated, such as the Servers, had many available data points, such as CPU, RAM, disk, resources score, ongoing migrations, hosted services, and power consumption. The resource score was a custom metric created to represent the server's overall capacity, being influenced by its available CPU, RAM and Disk. This metric was initially considered to be used in the comparisons and evaluations, however, it was later discarded, as it did not have a link with relevant findings, and remained almost constant independently of the scenario. For the User and Service entities, their data points were used to extract and/or generate the final metrics used, as show in Section 5.3.

5.6 KEY FINDINGS

If we recall what the simulation scenario represents in reality, as show in Figure 7, what happens when a service falls below our selected SLA objectives, is that the service is reallocated to a server that is closer to the mass of users, any blue point in Figure 9 is potentially an eligible server. Similarly, the service has fallen below the SLA objectives, because the users have moved to a different location, any of the users connected to that service could be travelling to any point of interest, which are the red points in Figure 9. With this context in mind, our results show that the proposed heuristics tested in Scenarios 1 and 3 are able to place the services closer to their users, consequently decreasing the delay from users to applications, which translates directly to a better QoS for the users.

When comparing the best results from Scenario 1 and 3, 60% of users experienced a delay lower than 100 units, while in Scenario 2, only 40% of users experienced the same delay. In the same sense, the normalized distance from services to users was also better in Scenarios 1 and 3, with all parameter combinations staying bellow the 0.6 threshold, while in Scenario 2, the 0.6 threshold was achieved by less than 50% of the services.

Imagine a user which is connected to an application, travelling from work or to class in an urban city, constantly sending and receiving data to and from said application. It can be quite disruptive of the application's usage if the quality of the connection suddenly starts degrading as they move away from their starting point, and begin to approach their destination. Simply put, a better placement algorithm for service composed applications like shown in Scenarios 1 and 3 can minimize, if not completely eradicate, this undesired behavior of QoS degradation. At the same time, the concept of service composed applications itself allows for such optimizations for individual services, by characterizing the needs, requirements and behaviors of each one separately.

5.7 TRANSLATING TO REAL WORLD

Implementing the solutions proposed in this work in a real-world scenario requires careful consideration of several factors to ensure the feasibility and effectiveness of the proposed provisioning policies and user-mobility models. These factors are crucial for translating the theoretical framework into practical applications.

One of the primary challenges is the integration of Cloud and Edge computing services, which are often provided by different vendors. This integration necessitates a federated approach, where multiple service providers collaborate to offer a seamless computing continuum. Such federation requires the usage of standardized protocols and interfaces to ensure interoperability across different platforms. Additionally, Edge servers may also be managed by various providers, each with different capabilities, policies, and pricing models.

These facts are important considerations individually, but also, all of them together raise the question of where and how would an implementation of the proposed solutions be placed. A possibility would be to create a framework/platform that integrates many computing services providers, and allows for developers/engineers to configure and setup applications on top of these available resources. The implementation must be deployed in a manner that is accessible and usable by application engineers. This could involve integrating the tool into existing DevOps tools and frameworks, allowing for automated and continuous optimization of service placement. The deployment environment should support real-time data processing and decision-making to respond promptly to changes in user mobility and network conditions. The parameters and configurations of the placement algorithm may need to be tailored to specific applications or use cases. Different applications may have varying QoS requirements, resource constraints, and user behavior patterns. Therefore, the platform should be flexible enough to accommodate these differences, either through global configurations or application-specific settings.

Beyond that, the applications often experience fluctuations in their user base, with new users joining and existing users leaving. The placement algorithm must be capable of dynamically adjusting to these changes, ensuring that the services remain optimally placed to serve the current user population. This requires continuous monitoring and adaptive reallocation of resources as user patterns evolve. Applications themselves, or the services composing them, may have redundancy and availability requirements to ensure high reliability and fault tolerance. The solution should be flexible to consider these requirements when allocating resources, ensuring that critical services are replicated and distributed across multiple servers to prevent single points of failure. This approach enhances the overall resilience of the application and maintains consistent QoS levels.

Another very important factor to consider is restrictions and concerns about users' data collection. Collecting network metrics, such as latency, bandwidth, and jitter, is essential for optimizing service placement. However, this data collection must be balanced with privacy concerns. Users' network data can be sensitive, and its collection and usage must comply with

privacy regulations and user consent. To circumvent users' geo data collection, one approach could involve using anonymized metrics or indirect measurements, such as ping or Round-Trip Time (RTT) from service servers to users' devices, to infer network conditions without compromising privacy. Accurately determining users' geographical locations is critical for mobility-aware service placement. However, obtaining this information requires explicit user consent due to privacy regulations. In scenarios where direct location data is unavailable, the placement algorithm may need to rely on alternative methods, such as network-based location estimation or user-provided location data, to approximate users' positions.

In summary, translating the proposed solutions to real-world scenarios involves addressing several practical considerations, including the federation of services, privacy concerns, dynamic user behavior, and application-specific needs. By carefully managing these factors, the proposed provisioning policies and user-mobility models can be effectively implemented to enhance the performance of service-composed applications in the Cloud-Edge Continuum.

5.8 PARTIAL CONSIDERATIONS

Provisioning an application composed of many services in Cloud-Edge Continuum can be challenging and complex due to the dynamic nature of the environment, especially considering new challenges regarding the users' geographical mobility. This requires a careful analysis from the perspectives of service providers, regarding an application's requirements and users' Quality of Service constraints. We propose, implement, and analyze a mobility-aware orchestration solution for placing service-composed applications atop Cloud-Edge Continuum substrates. By more accurately tracking and managing user's mobility inside an urban environment, the proposal can improve the overall performance of the Service-Composed Application (SCA). The preliminary results comparing with placement heuristics proposed in the reviewed literature shows that a more mobility-aware algorithm can deliver better results in respect to some QoS objectives for users and applications. Beyond that, a parameterized orchestrator proved to be a good approach for the problem, as it can be easily adapted to different scenarios and requirements. Overall, more tightly integrating Edge and Cloud Computing, proved to be a dependable way to improve the performance of the service compose applications.

6 FINAL CONSIDERATIONS

The present work embarked on a comprehensive exploration of the Cloud-Edge Continuum, a novel paradigm that integrates the capabilities of Cloud and Edge computing to enhance the performance and flexibility of distributed applications. In Chapter 1, the present work's objectives were discussed, implementing a new Point of Interest mobility model, which included creating a novel dataset with realistic user-mobility data, defining service provisioning/placement policies for Cloud-Edge Computing, and measurably improving user-perceived QoS metrics for service-composed applications.

In Chapter 2, we conducted an extensive literature review to identify the fundamental aspects and the current state-of-the-art in Cloud-Edge Continuum research. This review revealed a gap in the literature, particularly in the area of mobility-aware provisioning policies. One of the focuses of present work is the novel idea of a Cloud-Edge Continuum, that emerged through the development of Cloud computing and the introduction of Edge computing. This concept, discussed in Section 2.4, consists of enabling applications and distributed systems transparent access to both of these domains, and benefiting from all their capabilities combined. We also explored the critical aspect of user mobility, by understanding how users move through geographical spaces and interact with applications, we can develop more effective provisioning policies that dynamically adapt to changing user patterns. This led us to connecting user-mobility with the concept of the Cloud-Edge Continuum, where applications can seamlessly leverage both Cloud and Edge resources to meet their diverse requirements and doing so while maintaining Quality of Service for these users.

However, deploying an application made up of various services within this continuum can be difficult and intricate due to the environment's dynamic nature, especially with the added challenge of user mobility. Chapter 3 presented our detailed proposal, outlining the application scenario, research gap, and potential contributions. Addressing this issue requires thorough analysis from both user and service provider perspectives, focusing on the application's requirements and Quality of Service (QoS) constraints. We introduced the concept of mobility-aware provisioning policies and discussed the challenges associated with implementing such policies in a real-world environment. Our proposal emphasized the importance of considering user mobility and network demands when allocating services in the Cloud-Edge Continuum. It also examines the individual services within an application, taking into account their specific needs and limitations, to better allocate them within the available infrastructure and enhance the Quality of Service.

In Chapter 4, we transitioned from the theoretical proposal to practical implementation, detailing the design and development of our simulation environment. We developed a simulation environment using the EdgeSimPy simulator, which we extended to support Cloud computing capabilities. We also created a new dataset based on real GPS data of mobile antennas and implemented a novel user-mobility model. This simulation environment allowed us to test and

evaluate our proposed provisioning policies in a controlled setting.

Chapter 5 detailed our simulation campaign, where we calibrated and evaluated our proposed heuristics. We compared our mobility-aware provisioning policies with state-of-the-art solutions, demonstrating that our approach can significantly improve the performance of service-composed applications, when compared to traditional state-of-the-art placement policies and algorithms. Our results showed that considering user mobility leads to lower delays and better QoS for users, validating the effectiveness of our proposed solution. We reflected on the key findings of our research and discussed the practical implications of implementing our proposed solutions in real-world scenarios, highlighting the challenges associated with integrating Cloud and Edge services from different providers and emphasized the importance of privacy considerations when collecting user data. We also outlined potential future research directions, including the implementation of our solutions in real-world environments and the exploration of additional metrics and optimization techniques.

In conclusion, this dissertation has made significant contributions to the field of Cloud-Edge Continuum research by proposing and validating mobility-aware provisioning policies. Our work has demonstrated the potential of these policies to enhance the performance and scalability of distributed applications, paving the way for future advancements in this exciting and rapidly evolving domain.

6.1 CONTRIBUTIONS

In order to substantiate the proposal presented in Chapter 3, we implemented a variety of test scenarios using Cloud-Edge simulation tools, as presented in Chapters 4 and 5. A new Cloud-Edge dataset with user mobility aspects was created, based on real GPS data of mobile antennas, as well as a novel user-mobility model, and an expansion on the simulator to support the Cloud computing domain. Ultimately, the present work aimed to bring valid and innovative contributions to the Cloud and Edge computing field, with hybrid and continuous provisioning of distributed applications. This was achieved by creating and comparing different provisioning policies, as well as evaluating the impact of user mobility on the application's Quality of Service.

6.2 FUTURE WORK

The present work has opened up several possibilities for future research. One of the biggest potentials lies in implementing the proposed solutions on real-world scenarios, as discussed in Section 5.7. This would lead to a whole new set of hard challenges to be solved, but at the same time, it could provide many benefits to service-composed applications, such as the ones evaluated throughout this dissertation.

6.3 PUBLICATIONS

In regard to the content presented in this work, an article was submitted to the *25th IEEE International Symposium on Cluster, Cloud and Internet Computing* (CCGrid 2025), titled *Mobility-aware placement of service-composed applications on Cloud-Edge Continuum*. At the time of writing this dissertation, however, the article was still under review.

BIBLIOGRAPHY

AGUZZI, Cristiano et al. From cloud to edge: Seamless software migration at the era of the web of things. **IEEE Access**, v. 8, p. 228118–228135, 2020. ISSN 2169-3536. Cited on page 31.

ALVES, Marisangila; KOSLOVSKI, Guilherme Piĉgas. Joint cache placement and request routing optimization in heterogeneous cellular networks. In: IEEE. **2022 IEEE Symposium on Computers and Communications (ISCC)**. [S.l.], 2022. p. 1–6. Cited 2 times on pages 22 and 23.

ARMBRUST, Michael et al. Above the clouds: A berkeley view of cloud computing. **Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS**, v. 28, n. 13, p. 2009, 2009. Cited on page 16.

BAI, Fan; HELMY, Ahmed. A survey of mobility models in wireless adhoc networks. **Wireless ad hoc and sensor networks**, Kluwer Academic Publishers Dordrecht, p. 1–30, 2004. Cited on page 23.

BALOUEK-THOMERT, Daniel; RODERO, Ivan; PARASHAR, Manish. Evaluating policy-driven adaptation on the edge-to-cloud continuum. In: . [S.l.: s.n.], 2021. p. 11 – 20. Cited by: 2. Cited 2 times on pages 31 and 59.

BALOUEK-THOMERT, Daniel et al. Mdsc: Modelling distributed stream processing across the edge-to-cloud continuum. In: . [S.l.: s.n.], 2021. Cited by: 1. Cited on page 31.

BELCASTRO, Loris et al. Edge-cloud continuum solutions for urban mobility prediction and planning. **IEEE Access**, v. 11, p. 38864–38874, 2023. ISSN 2169-3536. Cited on page 31.

BELCASTRO, Loris et al. Using the compute continuum for data analysis: Edge-cloud integration for urban mobility. In: **2023 31st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)**. [S.l.: s.n.], 2023. p. 338–344. ISSN 2377-5750. Cited on page 31.

BONOMI, Flavio et al. Fog computing and its role in the internet of things. In: **Proceedings of the first edition of the MCC workshop on Mobile cloud computing**. [S.l.: s.n.], 2012. p. 13–16. Cited 3 times on pages 10, 16, and 22.

CAI, Jianhong; ZHU, Xiaorong; ACKAH, Amuah Ebenezer. Mobility-aware task offloading scheme for 6g networks with temporal graph and graph matching. **IEEE Internet of Things Journal**, IEEE, 2024. Cited on page 12.

CAMP, Tracy; BOLENG, Jeff; DAVIES, Vanessa. A survey of mobility models for ad hoc network research. **Wireless communications and mobile computing**, Wiley Online Library, v. 2, n. 5, p. 483–502, 2002. Cited on page 23.

DAS, Resul; INUWA, Muhammad Muhammad. A review on fog computing: Issues, characteristics, challenges, and potential applications. **Telematics and Informatics Reports**, Elsevier, v. 10, p. 100049, 2023. Cited 2 times on pages 11 and 17.

DRAGONI, Nicola et al. Microservices: yesterday, today, and tomorrow. **Present and ulterior software engineering**, Springer, p. 195–216, 2017. Cited 3 times on pages 18, 19, and 20.

DUKKIPATI, Nandita; MCKEOWN, Nick. Why flow-completion time is the right metric for congestion control. **ACM SIGCOMM Computer Communication Review**, ACM New York, NY, USA, v. 36, n. 1, p. 59–62, 2006. Cited on page 21.

ELLISON, Frank. The challenge of the computer utility. **Journal of the Operational Research Society**, Taylor & Francis, v. 18, n. 3, p. 324–325, 1967. Cited on page 15.

FEITELSON, Dror G; RUDOLPH, Larry. Metrics and benchmarking for parallel job scheduling. In: SPRINGER. **Job Scheduling Strategies for Parallel Processing: IPPS/SPDP'98 Workshop Orlando, Florida, USA, March 30, 1998 Proceedings 4**. [S.l.], 1998. p. 1–24. Cited on page 39.

FILHO, Roberto Rodrigues et al. Exploiting the potential of the edge-cloud continuum with self-distributing systems. In: . [S.l.: s.n.], 2022. p. 255 – 260. Cited by: 1; All Open Access, Green Open Access. Cited 2 times on pages 30 and 31.

FLOYD, Sally; PAXSON, Vern. Difficulties in simulating the internet. **IEEE/ACm Transactions on Networking**, IEEE, v. 9, n. 4, p. 392–403, 2001. Cited on page 41.

FRITZSCH, Jonas et al. From monolith to microservices: A classification of refactoring approaches. In: SPRINGER. **Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers 1**. [S.l.], 2019. p. 128–141. Cited on page 33.

FU, Kaihua et al. Adaptive resource efficient microservice deployment in cloud-edge continuum. **IEEE Transactions on Parallel and Distributed Systems**, IEEE, v. 33, n. 8, p. 1825–1840, 2021. No citation on text.

GILBERT, Seth; LYNCH, Nancy. Perspectives on the cap theorem. **Computer**, IEEE, v. 45, n. 2, p. 30–36, 2012. Cited on page 21.

GKONIS, Panagiotis et al. A survey on iot-edge-cloud continuum systems: Status, challenges, use cases, and open issues. **Future Internet**, MDPI, v. 15, n. 12, p. 383, 2023. No citation on text.

GRIGOROPOULOS, Nasos; LALIS, Spyros. Fractus: Orchestration of distributed applications in the drone-edge-cloud continuum. In: . [S.l.: s.n.], 2022. p. 838 – 848. Cited by: 0. No citation on text.

GU, Ji; LIU, Yushi; XU, Xiaolong. Sharpedge: A qos-driven task scheduling scheme with blockchain in mobile edge computing. **Concurrency and Computation: Practice and Experience**, n/a, n. n/a, p. e8161. No citation on text.

(ITU), International Telecommunication Union. **Minimum Requirements Related to Technical Performance for IMT-2020 Radio Interface(s)**. [S.l.], 2017. v. 2410, 2410–2017 p. Disponível em: <https://www.itu.int/pub/R-REP-M.2410>. Cited 3 times on pages 16, 23, and 35.

JAISWAL, Sharad et al. Measurement and classification of out-of-sequence packets in a tier-1 ip backbone. In: **Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment**. [S.l.: s.n.], 2002. p. 113–114. Cited 2 times on pages 20 and 35.

KIMOVSKI, Dragi et al. Mobility-aware iot application placement in the cloud – edge continuum. **IEEE Transactions on Services Computing**, v. 15, n. 6, p. 3358–3371, Nov 2022. ISSN 1939-1374. Cited 2 times on pages 30 and 31.

KUROSE, James; ROSS, Keith. **Computer networks: A top down approach**. 6. ed. [S.l.]: Pearson Addison Wesley, 2012. Cited 2 times on pages 20 and 21.

LIU, Fang et al. Nist cloud computing reference architecture. **NIST special publication**, v. 500, n. 2011, p. 1–28, 2011. Cited on page 16.

LIWANG, Minghui; WANG, Xianbin. Overbooking-empowered computing resource provisioning in cloud-aided mobile edge networks. **IEEE/ACM Trans. Netw.**, IEEE Press, v. 30, n. 5, p. 2289–2303, apr 2022. ISSN 1063-6692. Cited 4 times on pages 30, 31, 41, and 59.

LOPEZ, Pedro Garcia et al. **Edge-centric computing: Vision and challenges**. [S.l.]: ACM New York, NY, USA, 2015. 37–42 p. Cited 3 times on pages 16, 17, and 28.

LUO, Quyuan et al. Resource scheduling in edge computing: A survey. **IEEE Communications Surveys & Tutorials**, IEEE, v. 23, n. 4, p. 2131–2165, 2021. Cited on page 25.

MAHMOUDI, Charif; MOURLIN, Fabrice; BATTOU, Abdella. Formal definition of edge computing: An emphasis on mobile cloud and iot composition. In: IEEE. **2018 Third international conference on fog and mobile edge computing (FMEC)**. [S.l.], 2018. p. 34–42. Cited on page 10.

MAHMUD, Redowan; KOTAGIRI, Ramamohanarao; BUYYA, Rajkumar. Fog computing: A taxonomy, survey and future directions. **Internet of everything: algorithms, methodologies, technologies and perspectives**, Springer, p. 103–130, 2018. Cited on page 18.

MASIP-BRUIN, Xavi et al. Managing the cloud continuum: Lessons learnt from a real fog-to-cloud deployment. **Sensors**, v. 21, n. 9, 2021. Cited by: 12; All Open Access, Gold Open Access, Green Open Access. Cited 2 times on pages 31 and 59.

MELL, Peter; GRANCE, Tim et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National . . . , 2011. Cited on page 16.

NOORMOHAMMADPOUR, Mohammad; RAGHAVENDRA, Cauligi S. Datacenter traffic control: Understanding techniques and tradeoffs. **IEEE Communications Surveys and Tutorials**, v. 20, n. 2, p. 1492–1525, 2018. Cited on page 21.

OUYANG, Tao; ZHOU, Zhi; CHEN, Xu. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. **IEEE Journal on Selected Areas in Communications**, IEEE, v. 36, n. 10, p. 2333–2345, 2018. Cited 4 times on pages 12, 22, 39, and 40.

PAHL, Claus et al. Cloud container technologies: a state-of-the-art review. **IEEE Transactions on Cloud Computing**, IEEE, v. 7, n. 3, p. 677–692, 2017. Cited on page 19.

RAITH, Philipp et al. Mobility-aware serverless function adaptations across the edge-cloud continuum. In: . [S.l.: s.n.], 2022. p. 123 – 132. Cited by: 4. Cited on page 31.

RODRIGUES, Diego O. et al. Service provisioning in edge-cloud continuum: Emerging applications for mobile devices. **Journal of Internet Services and Applications**, v. 14, n. 1, p. 47 – 83, 2023. Cited by: 0; All Open Access, Gold Open Access, Green Open Access. Cited on page 25.

RODRIGUES, Leonardo R. et al. Cloud broker proposal based on multicriteria decision-making and virtual infrastructure migration. **Software: Practice and Experience**, v. 49, n. 9, p. 1331–1351, 2019. Cited on page 40.

ROSENDO, Daniel et al. Distributed intelligence on the edge-to-cloud continuum: A systematic literature review. **Journal of Parallel and Distributed Computing**, Elsevier, v. 166, p. 71–94, 2022. No citation on text.

SHI, Weisong et al. Edge computing: Vision and challenges. **IEEE internet of things journal**, Ieee, v. 3, n. 5, p. 637–646, 2016. Cited on page 11.

SINGH, Ramesh; SUKAPURAM, Radhika; CHAKRABORTY, Suchetana. A survey of mobility-aware multi-access edge computing: Challenges, use cases and future directions. **Ad Hoc Networks**, Elsevier, v. 140, p. 103044, 2023. No citation on text.

SOLMAZ, Gürkan; TURGUT, Damla. A survey of human mobility models. **IEEE Access**, v. 7, p. 125711–125731, 2019. Cited 3 times on pages 22, 23, and 47.

SOUMPLIS, Polyzois et al. Resource allocation challenges in the cloud and edge continuum. **Lecture Notes in Networks and Systems**, v. 289, p. 443 – 464, 2022. Cited by: 3. No citation on text.

SOUZA, Paulo S.; FERRETO, Tiago; CALHEIROS, Rodrigo N. Edgesimpy: Python-based modeling and simulation of edge computing resource management policies. **Future Generation Computer Systems**, Elsevier, v. 148, p. 446–459, 2023. ISSN 0167-739X. Cited 2 times on pages 44 and 52.

SOUZA, Paulo S. et al. Location-aware maintenance strategies for edge computing infrastructures. **IEEE Communications Letters**, v. 26, n. 4, p. 848–852, 2022. Cited on page 40.

TANAKA, Ryan et al. Automating edge-to-cloud workflows for science: Traversing the edge-to-cloud continuum with pegasus. In: **2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)**. [S.l.: s.n.], 2022. p. 826–833. Cited on page 31.

TANENBAUM, Andrew S; WETHERALL, David J. **Computer Networks**. 4. ed. [S.l.]: Pearson Education, Inc., 2011. Cited 4 times on pages 20, 21, 23, and 24.

UNWIREDLABS. **The world's largest Open Database of Cell Towers**. 2024. On-line. Https://www.opencellid.org/. Cited 2 times on pages 45 and 52.