DOCTORAL PROGRAM IN
INFORMATION AND COMMUNICATION TECHNOLOGY

Doctoral Candidate

Ayub Shah

Cycle	**35th (January 2020 - June 2024)**
Thesis	**Resource Optimization Strategies and Optimal Architectural Design for Ultra-Reliable Low-Latency Applications in Multi-Access Edge Computing**
Advisor	**Roberto Passerone, Associate Professor, DISI, University of Trento**
Co-advisor	**N/A**

1. PhD. Abstract:

The evolution and deployment of fifth-generation (5G) and beyond (B5G) infrastructure will require a tremendous effort to specify the design, standards, and manufacturing. 5G is vital to modern technological evolution, including industry 4.0, automotive, entertainment, and health care. The ambitious and challenging 5G project is classified into three categories, which provide an essential supporting platform for applications associated with:

- Enhanced mobile broadband (eMBB)
- Ultra-reliable low-latency communication (URLLC)
- Massive machine-type communication (mMTC)

The demand for URLLC grows, particularly for applications like autonomous guided vehicles (AGVs), unmanned aerial vehicles (UAVs), and factory automation, and has a strict requirement of low latency of 1 ms and high reliability of 99.999%.

To meet the needs of communication-sensitive and computation-intensive applications with different quality-of-service (QoS) requirements, this evolution focuses on ultra-dense edge networks with multi-access edge computing (MEC) facilities. MEC emerges as a solution, placing resourceful servers closer to users. However, the dynamic nature of processing and interaction patterns necessitates effective network control, which is challenging due to stringent requirements on both communication and computation.

In this context, we introduce a novel approach to optimally manage task offloading, considering the intricacies of heterogeneous computing and communication services. Unlike existing methods, our methodology incorporates the number of admitted service migrations and QoS upper and lower bounds as binding constraints. The comprehensive model encompasses agent positions, MEC servers, QoS requirements, edge network communication, and server computing capabilities. Formulated as a mixed-integer linear program (MILP), it provides an optimal schedule for service migrations and bandwidth allocation, addressing the challenges posed by computation-intensive and communication-sensitive applications.

Moreover, tailoring to an indoor robotics environment, we explore optimization-based approaches seeking an optimal system-level architecture while considering QoS guarantees. Optimization tools, e.g., ARCHEX, prove their ability to capture cyber-physical systems (CPS) requirements and generate correct-by-construction architectural solutions. We propose an extension in ARCHEX by incorporating dynamic properties, i.e., robot trajectories, time dimension, application-specific QoS constraints, and finally, integrating the optimization tool with a discrete-event network simulator (OMNeT++).

This extension automates the generation of configuration files and facilitates result analysis, ensuring a comprehensive evaluation. This part of the work focuses on the dynamism of robots, server-to-service mapping, and the integration of automated simulation. The proposed extension is validated by optimizing and analyzing various indoor robotics scenarios, emphasizing critical performance parameters such as overall throughput and end-to-end delay (E2E). This integrated approach addresses the complex interplay of computation and communication resources, providing a solution for dynamic mobility, traffic, and application patterns in edge server environments.

2. List of publications

1. **Resource Optimization in MEC-based B5G Networks for Indoor Robotics Environment,** Proceedings of the International Conference on Applications in Electronics Pervading Industry, Environment and Society, Amsterdam, Netherlands: Elsevier, 2021. Proceedings of: International Conference on Applications in Electronics Pervading Industry, Environment and Society, Pisa, Italy, 21-22 September 2021.

2. **Migration-Aware Optimized Resource Allocation in B5G Edge Networks,** Proceedings of the IEEE Consumer Communications & Networking Conference, Piscataway, NJ, USA: IEEE, 2022. Proceedings of: IEEE Consumer Communications & Networking Conference, Las Vegas, NV, 8-11 January 2022.
3. **Architectural Exploration and Design for Ultra-Reliable Low-Latency Indoor Robotics Systems**, IEEE Consumer Communications & Networking Conference, CCNC, USA: IEEE, 2024. Proceedings of: IEEE CCNC, Las Vegas, Nevada, USA, 6-9, January, 2024.
4. **Toward Simulation-Assisted Architecture Design Space Exploration of Indoor Robotics Networks,** IEEE 20th International Conference on Factory Communication Systems (WFCS), WFCS 2024, (Toulouse, France), April 17–19, 2024.

3. Research/study activities

Courses taken in Ph.D.

1. **Academic Writing for the Sciences and Engineerings, University of Trento**
2. **Research Methodology, University of Trento**
3. **Computing in Communication Networks, University of Trento**
4. **Industrial IoT Markets and Security, University of Colorado, online**

Summer and Winter Schools

1. **5G International PhD School, Winter School, Consorzio nazionale interuniversitario per le telecomunicazioni (CNIT), Rome, Italy, 2020.**
2. **SIE PhD School on Electronics and IOT, Summer School, Associazione Società Italiana di Elettronica (SIE), Trieste, Italy, 2021.**

Seminars

1. **Future Internet: a bridge from Artificial Intelligence to Quantum Intelligence, DISI, Antonio Manzalini Affiliation: TIM, 2020**
2. **Some recent advancements in machine learning for embedded systems, DISI, Claudio Turchetti, Universitá Politecnica delle Marche, 2021.**
3. **Dos and don'ts of being a researcher: some lessons I've learned on the way, DISI, Marco Baroni (ICREA, Pompeu Fabra University), 2022.**
4. **Breaking the Hype: Are 5G Networks Secure? Security Research Labs, DISI, University of Trento, 2022.**
5. **AWS Research Roadshow, AWS Research Roadshow, DISI, University of Trento, 2023.**
6. **Towards a Green Smart City: Harvesting RF Energy from V2X Communications, Federico Librino, DISI, University of Trento, 2024.**

ICT Days- Poster Presentation

1. **Poster Presentation, DISI, University of Trento, 2022.**

SELF-DECLARATION AND AUTHORIZATION

I hereby declare that the above-mentioned information is true to the best of my knowledge and belief.

Ayub Shah

Trento, Italia

12/June/2024

UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Resource Optimization Strategies and Optimal Architectural Design for Ultra-Reliable Low-Latency Applications in Multi-Access Edge Computing

Doctoral Dissertation

Candidate: Ayub Shah, University of Trento, Italy

Advisor: Professor Roberto Passerone, University of Trento, Italy

Committee: Professor Mihai Teodor Lazarescu, Politecnico di Torino, Italy
Professor Alessandra Rizzardi, University of Insubria, Italy
Professor Philippe Velha, University of Trento, Italy

June 2024

- **Honesty** is the first chapter in the book of **Wisdom**. [Thomas Jefferson]
- **Wisdom** is not a product of **schooling** but of the lifelong attempt to acquire it. [Albert Einstein]
- If we knew what it was we were doing, it would not be called research, would it? [Albert Einstein]
- Try not to become a man of **success**, but rather try to become a man of **value**. [Albert Einstein]

Acknowledgement

I want to start by saying **THANK YOU** to many people, but above all, I want to thank the **University of Trento** for providing me with this platform where I spent the last four years of my life. I find no words to express my gratitude, respect, and compliments to my kind, humble, and generous supervisor, **Professor Roberto Passerone**, who was and is always there for me whenever I needed him, who always listened to me without any restrictions, who always encouraged me, put aside my weakness, and tried to extract the best out of me. Aside from all his positive habits, I wish I could have as much patience and tolerance as he has. I appreciate our **Department's (IECS)** administration staff, secretary office, mission office, and even the canteen staff, who always provided their services whenever I asked.

Not to forget about **Trento city**, which has given me a splendid natural and cultural experience. The people, mountains, lakes, and especially the summer are exceptional. Joining a PhD program was not an easy choice; one must sacrifice, especially as an international student. I can proudly say that these four years taught me a lot; they helped me improve my personal and professional skills and ability to represent myself in society. When this journey is about to end, I find no words to express my feelings from the deepest of my heart and pay regard to my parents, siblings, and, above all, my wife.

There is no shame in admitting that this journey was impossible to accomplish alone **without my supervisor and wife**. I also want to say **Sorry** to my son, with whom I couldn't play much after going back home from university, but I will always admire his positive attitude towards me

because he never left me alone in any moment and never missed a single chance to jump over me without looking at my mood.

Last but not least, **a huge thanks** to all my friends and office colleagues, especially to Dr. Abhishek, Dr. Junaid, Dr. Tadeus, Dr. Arsalan, Mr. Rawlings, Mr. Enrico, Dr. Haroon, and Dr. Praveen who listened to me and helped me in the completion of this journey.

Ayub Shah
June, 2024

Abstract

The evolution and deployment of fifth-generation (5G) and beyond (B5G) infrastructure will require a tremendous effort to specify the design, standards, and manufacturing. 5G is vital to modern technological evolution, including industry 4.0, automotive, entertainment, and health care. The ambitious and challenging 5G project is classified into three categories, which provide an essential supporting platform for applications associated with:

- Enhanced mobile broadband (eMBB)
- Ultra-reliable low-latency communication (URLLC)
- Massive machine-type communication (mMTC)

The demand for URLLC grows, particularly for applications like autonomous guided vehicles (AGVs), unmanned aerial vehicles (UAVs), and factory automation, and has a strict requirement of low latency of 1 ms and high reliability of 99.999%. To meet the needs of communication-sensitive and computation-intensive applications with different quality-of-service (QoS) requirements, this evolution focuses on ultra-dense edge networks with multi-access edge computing (MEC) facilities. MEC emerges as a solution, placing resourceful servers closer to users. However, the dynamic nature of processing and interaction patterns necessitates effective network control, which is challenging due to stringent requirements on both communication and computation.

In this context, we introduce a novel approach to optimally manage task offloading, considering the intricacies of heterogeneous computing and communication services. Unlike existing methods, our methodology

incorporates the number of admitted service migrations and QoS upper and lower bounds as binding constraints. The comprehensive model encompasses agent positions, MEC servers, QoS requirements, edge network communication, and server computing capabilities. Formulated as a mixed-integer linear program (MILP), it provides an optimal schedule for service migrations and bandwidth allocation, addressing the challenges posed by computation-intensive and communication-sensitive applications.

Moreover, tailoring to an indoor robotics environment, we explore optimization-based approaches seeking an optimal system-level architecture while considering QoS guarantees. Optimization tools, e.g., ARCHEX, prove their ability to capture cyber-physical systems (CPS) requirements and generate correct-by-construction architectural solutions. We propose an extension in ARCHEX by incorporating dynamic properties, i.e., robot trajectories, time dimension, application-specific QoS constraints, and finally, integrating the optimization tool with a discrete-event network simulator (OMNeT++).

This extension automates the generation of configuration files and facilitates result analysis, ensuring a comprehensive evaluation. This part of the work focuses on the dynamism of robots, server-to-service mapping, and the integration of automated simulation. The proposed extension is validated by optimizing and analyzing various indoor robotics scenarios, emphasizing critical performance parameters such as overall throughput and end-to-end delay (E2E). This integrated approach addresses the complex interplay of computation and communication resources, providing a solution for dynamic mobility, traffic, and application patterns in edge server environments.

Keywords

[Mixed-Integer Linear Program, Design Space Exploration, Ultra-Reliable Low-Latency Communications, 5G and Multi-Access Edge Computing, Automated-Simulation]

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Fifth-generation (5G) communication is evolving into three main categories, i.e., enhanced mobile broadband (eMBB), ultra-reliable and low-latency communications (URLLC), and massive machine type communications (mMTC) [1]. The International Telecommunication Union (ITU) has formulated these three categories shown in Figure 1.1. The ongoing development and implementation of 5G wireless communication aim

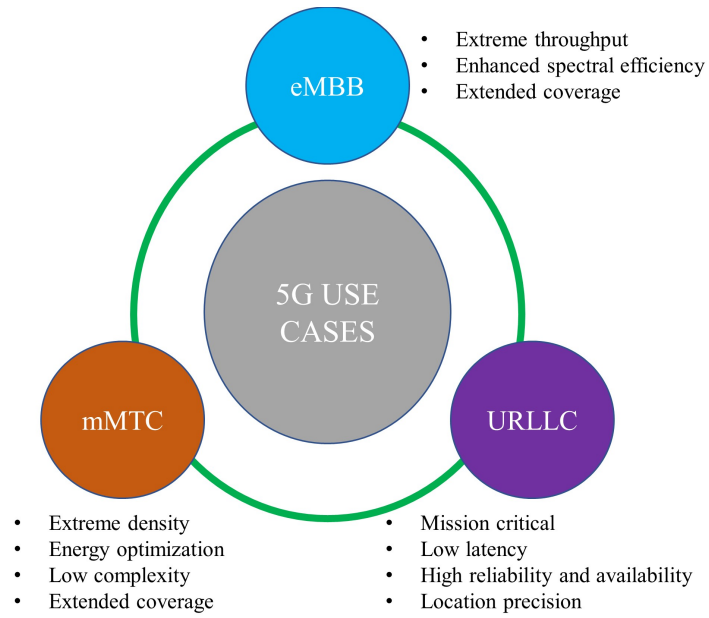


Figure 1.1: Use cases and features of future 5G network [2]

to make the ubiquitous connectivity of many Internet-of-Things (IoT) devices and applications possible. The user devices and applications vary from smartphones and computers to autonomous guided vehicles (AGVs), unmanned aerial vehicles (UAVs), industry automation, remote surgery, augmented reality (AR), and virtual reality (VR). These emerging applications are computation-intensive and delay-sensitive, imposing strict quality-of-service (QoS) requirements on the network. For example, applications in eMBB have a stringent requirement for throughput, but in the case of URLLC and mMTC, the focus is on reliability (99.999%) and latency ($\leq$ 1 ms) [3].

The ITU formulated the intended features and significance of all these three categories, which are [2, 4]:

- URLLC-based applications depend on high reliability and low latency. This category's applications do not focus on high data rates but require high responsiveness from the network. Important applications of URLLC include industry 4.0 automation, autonomous vehicle communication, mission-critical applications, remote surgery, and medical assistance.

- In eMBB, the applications focus mainly on the availability of high data rates and uninterrupted internet service. Potential applications and services include augmented and virtual reality, online gaming, and ultra-high-definition videos.

- In mMTC applications, the main concern is the connectivity of many Internet-of-Things (IoT) devices. Long-range communication with asynchronous access and energy efficiency is possible, making it suitable for many connected devices with low power.

It is challenging to satisfy the diverse requirements of all three categories simultaneously, as it will require a high-level modification in the architecture

of the present telecommunication infrastructure. Although the primary demand of smart device users in eMBB applications is high bandwidth and uninterrupted service, in the case of URLLC and mMTC, latency and reliability have a significant impact on real-time applications and mission-critical networks [1]. Table 1.1 shows the QoS requirements for different URLLC applications [5].

Table 1.1: Expected QoS requirements for URLLC applications

Industry	**Latency [ms]**	**Reliability**
Internet of Things	1	10^{-5}
Industry Automation	1	10^{-5}–10^{-9}
Autonomous Guided Vehicles	5–10	$\geq 10^{-3}$
Augmented and Virtual reality	5–10	10^{-3}–10^{-5}

As URLLC will play a key role in industry and vehicle automation, we aim to plan and deploy a network with optimal design, cross-optimization of computation, and optimal architectural design and resource optimization strategies for URLLC-based applications in the context of multi-access edge computing (MEC). The importance and implementation of URLLC in mission-critical applications are highlighted in Table 1.2 [5, 6].

With 5G, the user is expected to receive high computation and transmission speed, coverage, QoS, and responsiveness from wireless networks. Table 1.3 highlights the performance parameters of 5G with their expected ranges [7]. Recent research activities in academia and industry have highlighted the following significant requirements: [8, 9]

- Data Rates: The user is expected to get 1–10 Gbps of data rate, which is ten times higher than the traditional Long-Term Evolution (LTE) network theoretical peak data rate of 150 Mbps.

- Latency: The round-trip latency time is expected to be 1 ms in 5G, which is ten times less than 4G's 10 ms latency time.

Table 1.2: Importance of reliability and low latency as per industrial/user requirements

Industry	**Application**	**Importance of URLLC**
Industrial Automation	Various applications in industrial automation include building automation, floor automation, electric power automation, supervisory control systems, automation of factory floor and assembly lines with robots, machine-generated status reports, process automation, and Power management in the smart grid.	The Importance of URLLC services and applications is well known in the case of factory 4.0 automation. Robots are capable of replacing the entire factory management system, from development to processing, packing, and delivery. Reliability and latency are the two significant requirements for robots to perform a smooth operation.
Transportation systems	Intelligent Transportation Systems (ITS), including autonomous driving, vehicle-to-everything communication, UAVs, and traffic management systems.	The future of the transportation system is fully autonomous. Cars, drones, and traffic management will be operated without human intervention. Reliability and low-latency communication will be a priority for communication between cars, drones, and infrastructure.
Entertainment and media services	Ultra-high-definition (UHD) videos, online gaming, AR, and VR. Live reporting of an event, coverage, and telecasting of an event.	Online businesses, from entertainment to media coverage, are getting much attention with the advancement of technology. The required availability of network bandwidth and uninterrupted services provides the platform for online gaming, media coverage, and businesses to be done online.
Medical facility and health care services	Remote surgery and patient treatment.	With reliable and low-latency networks, performing remote surgery and providing medical assistance are now possible. Instead of humans, robots can operate in long-distance areas and countries. The desirable latency and reliability in the network are critical for the smooth performance of robots.

- High bandwidth: The connection of mobile devices with higher bandwidth and better QoS for long-duration among different users is expected.

- Connection of mobile devices: With data-sensitive and interactive ap-

Table 1.3: The performance parameters of 5G

Key parameters	**Ranges and Values**
Data transmission rate	100 Mbps – 1 Gbps
Peak data transfer rate	20 Gbps
Efficiency in spectrum transmission	Three-times higher than 4G
Latency rate	1 ms
Mobility	500 km/h

plications in the system, the connection to many devices must be maintained. Availability: In the vision of 5G, the network would always be available to users.

- Coverage anytime, anywhere: 5G holds up the coverage responsibility for users everywhere, anywhere, and anytime.
- Minimizing energy usage: With the development of green technology, energy consumption is expected to be reduced to 90%; however, with higher data rates and massively connected devices in 5G, it would be crucial.

High-performance computation driven by 5G networks has the potential to enable fundamentally new applications, industries, and business models and dramatically improve the quality of life around the world. The unprecedented use cases and applications require high data-rate instantaneous communications, low latency, and massive connectivity for mobile, e-health, autonomous vehicles, smart cities, smart homes, and the IoTs [10]. In this situation, to cope with resource hungry applications with minimum latency and higher reliability, MEC as an infrastructure is introduced that provides the computation and storage of data at the edge of a network close to user devices to minimize any delay between the network nodes and remote servers, which is typically a scenario in cloud computing architecture.

Specifically, the services run on physical computers called MEC hosts that can be placed even at the radio units of base stations (BS) [11], in which case the services are connected to their users directly through the shortest single-hop communication path instead of indirectly through a multi-hop communication path. Furthermore, to maintain a service's target latency as its mobile user moves to another BS, the network intelligently migrates the service to different MEC hosts. A service migration, however, entails a service downtime that increases the service's latency, which, if unacceptable, can be reduced if not eliminated by intelligently replicating the service on different MEC hosts.

To validate the resource management schemes and assess the performance of the 5G network, planning a reliable 5G network is of paramount importance. However, the evolution of novel technologies, varying QoS requirements, and complex scenarios with dynamic network architecture make it quite a challenging task [12].

Considering the complex network design challenges, network providers can benefit from design-space exploration (DSE) tools such as ARCHEX [13]. These tools allow network designers to optimally satisfy a set of requirements (e.g., the various QoS levels required by different services) by choosing a set of components (e.g., access points (APs), servers, and others) from a network component library [14]. For example, in designing a 5G network for a large factory that hosts specific services, a network designer first chooses several network components and places them on the factory model (e.g., a floor plan).

1.1 Challenges in the implementation of ultra-reliable low-latency communication

Realizing the importance of URLLC applications in various environments, this use case became a trending research area for academia and industry. Moreover, the ultra-reliability and ultra-low latency requirements make it challenging for network operators to design an optimal 5G network that supports and satisfies the URLLC applications and QoS requirements. Below are some challenges and issues that can be raised in implementing 5G use cases with the URLLC applications.

- **Quality-of-service (QoS)**: The end-to-end (E2E) delivery of data in URLLC applications strictly relies on high reliability, security, and low latency. To satisfy the stringent requirements of URLLC applications, the Third Generation Partnership Project (3GPP) has determined the latency and reliability parameters to 1 ms and 99.999% respectively [15].

- **Co-existence with eMBB:** 5G network will face the challenge of providing and satisfying diverse QoS requirements for various applications from URLLC and eMBB. However, the heterogeneity of the involved application scenarios in the same physical layer will pose a challenge in achieving the optimization solutions [16].

- **Energy consumption concern for the user devices:** To save the battery life of user devices, a sleep mode operation is employed in most wireless devices. The device immediately acts upon a packet's arrival to minimize network delay. However, this energy-saving technique is unsuitable for the URLLC scenarios because the device must continuously check incoming packets, which causes the battery of user equipment (UE) to drain drastically [17].

1.2 Multi-access edge computing and Resource optimization strategies

To enhance the QoS and quality of experience (QoE), as well as to make sure that users' sensitive and varied applications have enough resources, it is ideal to have a resourceful infrastructure that can offer high bandwidth, low latency, reliability, and storage at the network's edge. The European Telecommunications Standards Institute (ETSI) introduced MEC in 2014 [18], which offers the 4 C's (Computing, Caching, Communication, or Control) at the edge of the network to reduce any delays between network users and remote servers that are typically present in cloud computing architecture.

The deployment of MEC with 5G is gaining much attention from researchers and industrialists in wireless communication as it can provide elastic resources for the computation and latency-sensitive applications with ultra-low latency and high bandwidth. It facilitates mobile operators and service providers to provide storage and computation resources at the network's edges, i.e., BSs and APs [19].

ETSI published a white paper that characterizes MEC in various aspects mentioned below [20].

- The deployment of MEC computing facilities closer to user premises and in dense geographical locations brings cloud computing services to the network's edge. The facility of location-based mobility services, accuracy in big data analytics, and the ability to monitor real-time environments like sensor networks are possible. While accessing local resources, MEC can perform tasks in isolation from the rest of the network. This feature of MEC is considered an essential aspect of the machine-to-machine (M2M) scenario, making it less vulnerable.

- The deployment of a MEC server close to the user at the network's edge has the advantages of low latency, fast computation, reliability, and high bandwidth for sensitive applications like AR and VR.

- Edge-distributed users can use the attribute of location awareness, which is made possible in edge computing, to access the services of edge servers near its location. The devices share information using low-level signals. Implementing MEC in the business model allows the applications to benefit from real-time network data information and services. The applications based on this information can provide better delivery services to customers by estimating the network bandwidth and radio cell congestion.

Resource allocation and optimization of MEC-based infrastructure is a trending research topic for dynamic environments where different tools and techniques are adopted to handle user mobility and utilize MEC servers efficiently. Recent research shows how to handle user mobility and service migration and improve QoS for different system settings.

Afrin et al. [21] developed a use case of fire emergency management service in a smart factory environment and proposed a novel idea, such as edge cloud (EC), to overcome the remote cloud limitations. The proposed idea is to develop an evolutionary algorithm based on multi-robots in a smart factory environment to overcome the limitations of a remote cloud, such as executing latency-sensitive tasks. In their work, the authors extended the Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) and modeled the resource allocation for the robotic workflow as a constrained multi-objective optimization problem, i.e., to minimize the time to complete all tasks, the energy consumption, and the cost of task execution.

1.2.1 State-of-the-art related technologies of Multi-access edge computing

MEC is not the only technology providing computation resources to mobile users; some alternative technologies also exist in the relevant literature. The rest of the terms are mobile cloud computing (MCC), Cloudlet, and fog computing (FC) [22]. MCC collectively provides the advantages of mobile internet, mobile, and cloud computing (CC). CC focuses on the isolated virtualization computing, communication, and storage resources leveraged by end-mobile users [23]. Amazon EC2, Microsoft Azure, Google, and Aneka are examples of CC infrastructure. The demand for resources in a mobile environment, which includes computing, network servers, applications, and storage, is provided by the deployment of MCC Resource management, which could be easily managed and focused on MCC.

Local Cloud (LC) is a computational resource provider in a local area network (LAN) that coordinates with the remote cloud server. LC is designed for an organization or institution controlled by internal or external sources to provide strict data privacy [24]. The LC is integrated with the remote cloud by installing software on the local server. However, LC provides a better communication cost because of its close installation to the network, but it also has the problem of limited resources and less computation capacity [25]. A cloudlet is another small or box computation resource provider deployed immediately after the mobile devices and before the remote cloud.

Public spaces like shopping malls, hospitals, universities, and office buildings can install cloudlets to enable easy access to the remote cloud [26]. Cloudlet is formed by combining single or multiple units of multi-core computers, which are then connected to remote cloud computing servers. Cloudlet supports resource-hungry, data- and latency-sensitive applica-

tions by bringing the remote cloud servers closer to end-mobile users [27]. Cloudlet services depend on reliable and uninterrupted internet facilities as they utilize Wi-Fi technology near the edge of the internet.

Moreover, there are privacy and security issues related to cloudlets, which creates a more serious concern for the service providers; e.g., cloud-assisted healthcare big data computing becomes critical to meet users' ever-growing demands on health consultation. The body data collected by wearable devices is transmitted to the nearby cloudlet. Those data are further delivered to the remote cloud, where doctors can access them for disease diagnosis. With the advances in cloud computing, a large amount of data can be stored in various clouds, including cloudlets. However, cloud-based data sharing entails the following fundamental problems: [28, 29]

- How to protect the security of the user's data during its delivery to cloudlet?
- How to make sure the data sharing in cloudlet will not cause privacy problems.
- How to effectively protect the whole system from malicious attacks?

Another promising technology in CISCO's edge computing paradigm is FC. It enables global device connectivity and brings cloud services to the edge of an enterprise network. The processing in FC is mainly performed in a LAN, fog server, or IoT gateway. Although FC has much less latency than CC, a major disadvantage of fog computing technology is that it depends heavily on a wireless connection. Also, the technological terms MEC and fog computing are used and vice versa, providing almost the same features to network users. MEC is well-supported by mobile network operators (MNO) as it is implemented close to or at the BS, whereas fog computing receives support from internet service providers and processes data at the fog node or IoT gateway [30].

1.3 The Context

The emerging use cases of 5G presents a break with 4G by making the network intelligent, not to mention the tremendous increase in network density and heterogeneity [31,32]. Several applications like AR, VR, video-conferencing, especially URLLC-based applications, such as autonomous vehicles, industrial automation, and remote surgery, will impose latency-critical requirements, and a slight change in network configuration may drastically impact the QoS. Eventually, network providers will face a complex network design task, specifically, when they want to minimize their costs while guaranteeing a certain QoS levels, i.e., by ensuring sufficient resources between the service and its user [31, 32].

URLLC is characterized by a convergence of different areas, including control, robotics, signal processing, artificial intelligence, and data analysis, which provide fertile ground for novel and innovative applications. To jointly support a large number of heterogeneous services, the optimal management of computing and communication resources available at the network edge for task offloading is fundamental for successfully satisfying the strict QoS requirements of emerging applications.

Considering the approaching era of technology and industry requirements, as well as to narrow down the scope of this research, we consider a URLLC-based dynamic indoor robotics scenario in a 5G and beyond edge network where multiple robots are deployed in an ultra-dense cell environment and are continuously migrating from one cell to another to perform their tasks. While moving across the network attachment points, robots should be served by different servers at the edge whose positions and capabilities can satisfy the expected service requirement.

The contribution of this research provides a novel methodology for optimally and flexibly managing task offloading in the context of real-time

applications requiring heterogeneous computing and communication services. The network entities are connected to an edge network hosting MEC servers with computing capabilities. Assuming the evolution of the agent position in time, we describe the evolution of the QoS in terms of the position of the agents, the expected service requirements, the position of the VMs, the communication capabilities of the links, and the computing capabilities of the MEC servers.

Moreover, the evolution of novel technologies imposes a varying QoS requirement on the network, making it quite a challenging and computationally demanding procedure [33]. Due to the complex scenarios and dynamic network architecture, it is necessary to assess the network parameters before the actual deployment. The associated parameters, such as the increase in network capacity, latency, reliability, and mobility environment, make it difficult to find a particular analytical solution, as required by the optimization tool [34]. To handle the complexity of network design, network providers can utilize DSE tools such as ARCHEX. These tools help optimize network designs by allowing designers to select the best combination of components, like APs and servers, from a library of network elements [35].

Using ARCHEX[1], a system designer first expresses a system architecture as a directed graph, a set of nodes and their interconnecting links, where the nodes and links have specific attributes and are chosen from a pre-defined component library. The system designer can then use a predefined pattern language to specify a set of system properties and an objective function before running ARCHEX to obtain an optimal system architecture.

[1] `https://bitbucket.org/regkirov/archex.git`

1.4 The Problem

General Framework: The 5G communication systems are evolving to support hundreds of users with heterogeneous applications and diverse QoS requirements. The low-latency and ultra-reliability QoS requirements of applications such as AGVs, UAVs, and industrial automation will pave the way to MEC. Depending on the specific use case, mobile networks are called to support peak data rates of around 1–10 Gbps, real-time communications (e.g., telesurgery, extended reality, and robotics) asking for latency of a few milliseconds, higher mobility scenarios with users moving at potentially high speeds, ultra-dense scenarios with hundreds of devices concentrated in small areas, and high service reliability [36].

At the same time, processing algorithms based on artificial intelligence, data analysis, mission planning, and automated reasoning, compounded with both the autonomous nature of new devices and the necessity for a federated and cooperative infrastructure, require extreme computational resources that are hard to provide in isolated, resource-constrained systems. These strict requirements inevitably introduced the need to formulate novel and sophisticated methodologies to optimize network and computation functionalities and resources distributed across radio interface, edge network, and core network.

However, the high dynamicity of the involved processing and interaction patterns requires planning and deployment of architectures with optimal design and cross-optimization of computation and communication resources. The emergence of data-sensitive and human-machine interactive (HMI) applications has made it difficult for low-power, low-computability mobile devices to cope with the resource demands.

In this context, to address this complexity and deal with the constrained resources typically available on end devices, computational offloading has

been proposed as a way to migrate computation-intensive tasks from smart devices to external sources like clouds, edge servers, or cloudlets for better performance [37]. Service offloading is therefore taken as the solution for devices to improve their battery power and performance by transferring intensive computation tasks to resourceful, rich servers located at a remote location [38]. Therefore, the primary system design goal is to provide an optimal architecture, quality, and cost constraints that can efficiently schedule, migrate, and execute tasks in dynamic environments.

Proposed Methodology: In this research, our goal is to design and develop an optimal network where we will merge scheduling techniques for flexible task management, guaranteeing consistent service levels. Alongside, we will employ simulations and explore diverse strategies to enhance network performance. Our objective is to construct a system capable of dealing with migrations, mobility, and evolving application requirements within the MEC domain.

Moreover, we also aim to develop a design methodology and supporting tool infrastructure that combines and enriches ideas from architectural exploration, network optimization, and simulation. To do so, we start from an abstract, high-level specification of the system and encode the problem in a formalism that optimization algorithms can handle. We will employ an iterative optimization methodology to make the approach effective, where optimization and simulation are interleaved to contribute to the final result.

Relevant Tools/Systems: There are a number of tools that can be used for optimization. Our method involves formalizing the problem using generic patterns that make it easier to represent the system. These patterns are then turned into linear constraints that use both continuous and discrete variables. This results in a MILP optimization problem, which can be solved using state-of-the-art solvers like CPLEX and Gurobi. This class of problems is characterized by high computational complexity, which

can be addressed using iterative techniques, where simplified optimization problems are formulated and verified using analytical or simulation methods.

Given the complexity of the networks, we plan to resort to simulation to verify if the generated architecture satisfies the given requirements. Several simulators can be employed, but their capabilities and application spaces differ. We are particularly interested in network simulators that can handle complex interactions and computation nodes and can be extended to cover our use cases better. Among these, the following are potential candidates for our implementation:

- OMNeT++ is an extensible, modular, component-based C++ simulation library and framework primarily for building network simulators that include wired and wireless communication networks, on-chip networks, queueing networks, etc. Several extensions are available to model existing networks and protocols.

- NS-3 is a discrete-event network simulator targeted primarily for research and educational use. The goal of the ns-3 project is to develop a preferred, open simulation environment for networking research.

- LENA and 5G-LENA are GPLv2 NR network simulators designed as plug-able modules to NS-3. Their development is open to the community to foster early adoption, contributions by industrial and academic partners, collaborative development, and reproducibility of results.

- 5G-Air-Simulator is an open-source tool modeling the 5G air interface.

- Simu5G is the evolution of the popular SimuLTE 4G network simulator, incorporating 5G New Radio access.

The objectives of the thesis therefore include:

- A methodology for designing and optimizing complex dynamic architectures involving computational and communication resources, latency, mobile agents with wireless access to the network, and dynamic mapping of services.
- A set of pre-defined specification patterns that express the design intention in a compact yet semantically reachable formalism and that can be tailored to the desired application domain.
- A tool that supports the methodology by analyzing the patterns and by generating a mixed-integer optimization problem to be handed over to a MILP solver.
- A case study of industrial automation involving mobile robots to validate the approach.
- Developing an automated simulation environment by integrating an optimization tool with a simulation framework to perform the optimization and simulation in a loop to analyze and verify the generated candidate architecture for various network KPIs, e.g., E2E latency and overall throughput.

1.5 Thesis Contributions

Our major scientific contribution is divided into three Chapters.

Chapter 3 [39]: In this contribution, we formulated a novel MILP-based mathematical model. The formulation consists of multiple linear constraints, decision variables, and objective functions. As an example, we consider a small factory floor environment with multiple connected cells with one robot and its associated multiple services with different QoS requirements. To solve the MILP optimization model, we leveraged state-of-the art solvers like GLPK to achieve optimal solutions. Our solution

clearly shows how the solver optimally chose the service to be migrated from one cell to another in order to be processed successfully with the desired resource requirements.

Chapter 4 [40]: We developed and evaluated MILP formulation for a complex and scalable environment where we have hundreds of cells, agents, and servers. Our formulation's scalability in handling complex systems is presented in detail. We evaluate the effectiveness and scalability of our MILP formulation in a realistic ultra-dense network where many robots and services are deployed in an indoor environment (i.e., an industrial scenario). The impact of various quality functions on both star and mesh networks demonstrates our formulation's effectiveness.

Chapter 5 [41,42]: The contribution in [41] is based on the optimization tool called ARCHEX. The tool makes an architectural design process more efficient by automatically producing an optimal system architecture with respect to an objective function while guaranteeing that the resulting system remains correct with respect to the set of system properties. By leveraging the optimization tool, we are interested in the earlier phase of architectural design: the placement of the infrastructure, the deployment of the servers, their sizing, and the distribution of communication and services. We aim to design and develop a dynamic robotic case study where we implement various QoS constraints and time dimensions into the tool so that connections between entities can change dynamically, and the mapping of services to architecture components can be performed to satisfy the QoS requirements and achieve the desired performance.

In [42], the contribution consists of extending ARCHEX core structure to support the design of systems on which 5G's promise of URLLC has a significant impact (e.g., on the design of smart factories) and has excellent potential when coupled with open-source simulators, e.g., a discrete event

simulator, i.e., OMNeT++[2]. The idea has two important points: first, to use the optimization tool to find feasible candidate architectures, and second, to integrate ARCHEX framework with the simulation to validate the dynamic parameters of the network.

1.6 Structure of the Thesis

This thesis consists of six chapters, which are summarized as follows:

- Chapter 1 presents a thorough study and provides an in-depth analysis of 5G and B5G mobile wireless networks, the detailed background of the 5G use cases, specifically URLLC, the problem statement, and the contributions of our work. We also discussed state-of-the-art MEC technology and provided an in-depth analysis of its various versions using recent research work.

- Chapter 2 presents the detailed state-of-the-art literature and their analysis related to our area of interest. Specifically, the chapter includes study about resource allocation and optimization in 5G and beyond (B5G) edge networks and architectural exploration and design for indoor systems. Moreover, a detailed background of the optimization tool, i.e., ARCHEX is also presented.

- Chapter 3 represents a novel case study for an indoor robotics environment for which we developed a MILP-based mathematical formulation. Initially, we consider a smaller scenario for a factory floor, which consists of cells, robots, and MEC servers. The model, being deterministic, assumes that the robot is present in the first cell as a starting point. Then, at every discrete time-step, it changes its position from one cell to another and finally completes a round trip and

[2] https://omnetpp.org/

returns to its original position.

- In Chapter 4, we extended one of our previous contributions and formulated a MILP-based optimization problem where we account for a complex architecture with ten's of robots, services, servers, and cells. The optimal solutions achieved for different network and scenarios shows the effectiveness and scalability of our formulated MILP model. For clarity of illustration and without loss of generality, we use different edge networks, different MILP-solving strategies, and other MILP parameters while considering that in an individual scenario, the agents and the cells have the same types, the same type of links statically interconnect the cells, the MEC servers are of the same kind with the same software stack, every server and every link have the same capacities, respectively; every VM migration has the same cost, and every VM has the same bounds on the computation and communication bandwidth and latency and the same quality function.

- In Chapter 5, we efficiently explored and utilized an architectural exploration tool, ARCHEX 2.0, an extensible framework for CPS architecture exploration, and implemented a dynamic robotic scenario and various constraints to satisfy our problem requirements. We discuss in detail the methodology flow, outlining the main steps and providing the formulations for various network requirements (e.g., LQ, routing, and QoS). To demonstrate the design effectiveness of ARCHEX, particularly to highlight its potential benefits in our case study, we adapt the methodology of WSN from ARCHEX as an example and extend it further because WSN is not only the example most relevant to indoor robotics scenarios but also an underpinning communication technology in both cyber-physical and internet-of-things systems.

  However, the current version of the tool is limited to supporting only

static scenarios and is therefore not suited for a dynamic environment. Considering this limitation as a starting point, we plan to introduce the time component also, to further support the dynamicity, one proposition is to overcome ARCHEX limitations by integrating a network simulator in a loop to validate the dynamic properties of the generated candidate architectures.

- Chapter 6 summarizes the thesis with a detailed revision and the contributions of the research work. Among the conclusions, we also highlighted promising research directions for future work.

Chapter 2

Related work

This chapter is organized to present a detailed analysis of the state-of-the-art literature concerning resource allocation and optimization in 5G and beyond (B5G) edge networks, as well as the architectural exploration and design of indoor systems. The chapter provides a comprehensive review of existing methodologies and approaches in these areas, highlighting their contributions and limitations. Additionally, the chapter discusses the background and functionalities of the optimization tool, ARCHEX, which plays a crucial role in the analysis and model development of the case study in this research work. This foundational overview provides the necessary context to understand the advancements and challenges addressed in this research.

2.1 Resource Allocation and Optimization in 5G and beyond (B5G) Edge Networks

The research on the next generation of mobile networks is chasing the ambitious objective of jointly supporting, within a flexible and powerful communication infrastructure, a vast number of heterogeneous services belonging to mobile broadband reliable low latency communications (MBRLLC), massive ultra-low latency communications (mURLLC), and human-centric

services (HCS) categories [43]. In this context, effective management of task offloading is crucial to delivering high-quality services [44], hence the need for the optimal management of computing and communication resources at the network edge. It is easy to understand that mobile network optimization is one of the major multidisciplinary research topics addressed in the latest decade.

Recent contributions published in the scientific literature proposed several interesting approaches to optimize the use of computing and communication resources exposed at the edge of the 5G system for task offloading. Many scientific papers [45–47] focus on static scenarios, where user mobility is not explicitly taken into account, while other papers [48–51], explicitly account for user mobility. The literature mentioned above proposes optimization algorithms or iterative procedures to optimize the allocation of servers to tasks while respecting energy, latency, and communication delay constraints. Task offloading in mobile scenarios is a complex problem, and the overall management is extremely dynamic. While moving across network attachment points, mobile users should be served by different servers at the edge whose position and capabilities can satisfy the expected service level. In the event that the offered service is implemented through dedicated virtual machines (VMs) or containers, such a dynamic scenario requires frequent migration operations [49, 52–54].

Wang et al. [55] focused on user mobility and dynamic service migration in the aspect of MEC. Considering the challenges of making optimal migration decisions and resource allocation for the users in an uncertain environment, the authors formulated the problem as a Markov decision process (MDP). The proposed method provides a general cost and mathematical framework to design the optimal policies for service migration. The results of the MDP model provide an exact optimal solution and close approximation for 1-D and 2-D users' mobility, respectively.

Plachy et al. [56] jointly investigated the computation and communication resource allocation to handle the mobility and meet the user QoS demands. The authors proposed two approaches in the user mobility environment. First, they designed a path selection algorithm with handover (PswH) to select a suitable path for flexible communication between the user and the VM. Second, to predict the user movement using an MDP model to place the VM at the most suitable BS for the user to offload their tasks.

The concept of MEC is also known as cloudlet as in [57]; moreover, Zhang et al. [58] investigated the user mobility scenario and proposed an MDP framework to exploit the user's task offloading in a mobile cloudlet system. The authors proposed a dynamic decision-making algorithm for executing user tasks locally or on Cloudlets. The authors consider the intermittent connectivity problem in mobile cloudlet systems and propose an MDP model that is solved using a value iteration algorithm. The proposed model obtained an optimal offloading policy and minimized communication and computation costs.

Liu et al. [33], to meet the user's QoS requirements, proposed a two-stage multi-objective MILP-based optimization strategy in a multi-user and multi-cloudlet environment. The first and second stages consider the optimal cloudlet selection and resource allocation model based on MILP by optimizing the latency and mean resource usage.

MEC-based networks are not limited to mobile networks but also AGVs and UAVs. In the case of vehicle-everything (V2X) communication, MEC has been adopted as a primary solution for many scenarios, e.g., Balid et al. [59] proposed MEC technology for their work on the real-time traffic surveillance and counting of vehicles. A low-cost sensor network was developed and installed on the roadside for traffic monitoring. Emara et al. [60] proposed to deploy MEC technology for V2X communication,

assisted by cellular networks. The study focuses on the safe interaction between vulnerable road users (VRU) and vehicles through cooperative awareness messages (CAM) and aims to reduce the E2E latency between VRU and vehicles.

The authors validated their work through extensive simulations and claimed that their system could achieve up to an 80 percent reduction in latency. Moubayed et al. [61] suggested a system considering MEC parameters, including latency, redundancy, and resource availability. The authors considered a hybrid cloud and edge environment and proposed a solution by developing a greedy-based algorithm. They evaluated the performance of their algorithm for V2X applications in terms of efficient service placement on servers, latency, and resource utilization.

The desired migration model that is intended for our research work is similar to a VM copying technique from a source to a destination that is called pre-copy memory data migration, which is usually used for a VM live migration in a data center but is also adapted for a service migration in MEC [62,63]. Our copying technique shares similarities with the process of copying a VM while it's still running on the source. Unlike traditional methods, our technique ensures that any changes made to VM data on the source during migration are not re-copied to the destination until the source VM is terminated and the destination VM is initialized. While our intended copying technique may not work for all kinds of services, it works for those that are either easily made if not already stateless (e.g., video streaming services) or tolerant of some data losses and aging (e.g., robotic control services [64]).

Wang et al. [62] show different techniques for migrating an amount of data depending on whether the destination already hosts the needed operating system and application code as an expected service migration technique, and concerning a pipelined migration involving change logging, data

compression, and other kinds of pipelined processes as another expected service migration technique, and aside from that, with respect to deciding whether to migrate based on the remaining duration of a service (e.g., it may not be worthwhile to migrate a video streaming service when only ten seconds are left).

Our migration model, however, can model any amount of data and can also model a pipeline migration as long as the pipeline's computation bandwidth is accounted for in the executing service's computation bandwidth. Our migration model should not take a service's remaining duration into account because it assumes that mobile agents need their services indefinitely. Similar to [62, 63], our migration model does not allow migration cancellation and enables multiple services to migrate from a source to a destination concurrently.

However, Yang et al. [63] consider the optimal concurrent migration of a given number of services from a given source to a given destination and only consider the available end-to-end communication bandwidth. Our model would consider whether, for every possible source and destination, it is optimal to migrate several services, and our model would also consider the computation bandwidth available in both endpoints. Once a migration is started, our migration model will no longer consider any migrated data changed on the source. Moreover, our measure of QoS is not the size of data that a service may change every time unit on the source, but the computation and communication bandwidth that a service can use to serve its user, its distance to the user, and in case of migration, the time duration needed to complete the migration.

Unfortunately, state-of-the-art solutions have two main problems. First, many B5G use cases (e.g., indoor robotic applications) will be enabled through ultra-dense cell deployments, where numerous base stations cover a limited area. The frequent handovers triggered by user mobility would

produce erratic task offloading management, resulting in excessive migrations of VMs or containers deployed at the network edge. Unfortunately, to our knowledge, no solution in the literature considers the number of migrations as a system variable to be optimized (e.g., minimized).

Second, in real-time systems, a service's QoS can be linked to its computation bandwidth [65]. But in edge computing, the degrees of freedom that affect QoS are more important and include the computation and communication bandwidth as well as the physical movement of the VM that provides the service. Therefore, the choice of these parameters allows for adjusting the QoS level dynamically within a range spanning from a minimum to a maximum value. Despite the evident benefits of an adaptive QoS, the flexible provisioning of advanced services in dynamic network conditions is quite ignored in the literature.

To close the gap, we propose a novel methodology to manage B5G task offloading optimally and flexibly in the context of real-time applications, represented well in the robotic domain: AGVs tour a large logistic facility, using the edge facilities for computation-intensive tasks [66]. Network attachment points offer wireless connectivity to agents (e.g., AGVs) that require heterogeneous services. The attachment points are connected to an edge network with computing capabilities provided by MEC servers. The agents then connect to one of the available MEC servers through edge links with fixed communication capabilities.

In the robotic domain, it is reasonable to assume that agent mobility is predictable [67]. Because of this, the QoS dynamics are modeled based on the positions of the agents and VMs, the service needs, the communication capabilities of the link, the computing capabilities of the MEC server, and other factors. The model is then translated into a MILP to get optimal VM positions and their optimal communication and computation bandwidth. Unlike the current state of the art, QoS is a function of computing

and communication requirements, end-to-end communication latency, and migration cost. In the current state of the art, computing and communication capabilities are seen as constraints. However, the proposed new approach takes into account the number of allowed VM migrations and the lower and upper QoS bounds that the agents expect.

2.2 Architectural Exploration and Design for Indoor Systems

CPSs are the backbone for diverse applications such as healthcare, smart grids, transportation, and smart homes [68]. These systems comprise numerous interconnected digital, analog, physical, and human components meticulously designed to execute specific functions by seamlessly integrating physics and logic. The complexity of CPS designs becomes particularly evident in multidisciplinary analyses, such as optimizing an aircraft's operational cost and range. Notably, conceptual designs contribute to approximately 80% of the development cost in aircraft manufacturing [69].

Therefore, a priority is to devise search techniques that minimize the need for expensive CPS simulations and efficiently navigate the broad design search space [70]. The process of CPS design involves determining both the system architecture and the values of the appropriate components, aiming to achieve a final design that aligns with predefined system specifications. Traditional approaches to CPS design conventionally involve human input for selecting architectures [71]. Complexity arises also from the heterogeneous nature of the models involved, which often requires the use of meta-modeling and hybrid techniques [72–75].

This entails simulating the architecture across various component values to identify the combination that satisfies the system requirements. However, this process imposes limitations on the search space, and including

expensive CPS simulations extend the overall development timeline. To enhance DSE efficiency, it is crucial to integrate innovative automated search techniques while retaining the adaptability to incorporate existing knowledge derived from architectures designed by humans. Formulating CPS design as a multi-objective optimization (MOO) problem offers a solution to address diverse system design objectives, considering constraints imposed by available components for synthesis. For example, in the design of a drone, constraints arise from the maximum torque the motor can generate, with the primary objective of maximizing payload capacity and the drone's travel distance [76].

Research institutions and industry have developed several approaches to cope with the exponential growth in systems complexity. Of particular interest are *layered design* and *component-based design*, supported for instance, by the AUTOSAR standard[1] in the automotive sector and the ARINC standard[2] in the avionic domain. In this context, *model-based development* is supported by several frameworks and tools, such as Modelica [77] and Matlab-Simulink [78] for system modeling, virtual integration (Ptolemy [79] and Metropolis [80]), and platform-based design [80–83]. There are two basic principles followed by these methods: abstraction/refinement and composition/decomposition.

Abstraction and refinement are processes that relate to the flow of design between different layers of abstraction (vertical process) [84], while composition and decomposition operate at the same level of abstraction (horizontal process) [85]. Layered design and model-based development focus on the vertical process, while component-based design deals principally with the horizontal process. The platform-based design combines the two aspects in a unified framework. Contracts or interface models

[1] http://www.autosar.org/

[2] http://www.aeec-amc-fsemc.com/standards/index.html

can be used to enrich the specification with environmental assumptions in addition to guarantees [86–90]. These methods have been applied to both behavior-based verification and non-functional requirements [91]. We currently do not support contracts, although our methodology can be extended in that direction. Behavioral models that rely on tag-systems are particularly interesting in this regard, since they support orchestrating potentially heterogeneous domains [92–94].

Optimization-based approaches are applied to find an optimal high-level architecture and feasible solution to reduce the cost value. The scope of CPS and the experiments performed on ARCHEX prove its usability, expressiveness, scalability, and extensibility characteristics. The proposed methodology of the tool can select and interconnect physical and electronic components of a CPS at a conceptual level of design and satisfy the system-level requirements, paving the way for many potential applications in multiple domains. However, many DSE techniques and tools have appeared over time in different domains, but the tool adopted for this research, i.e., ARCHEX is complementary and can be combined with simulation-based methods [95] in the future to verify the architecture's dynamic properties. To date, no standard technique is commonly adopted to solve optimization-based problems. Still, for corresponding problems, MILP is used with state-of-the-art MILP solvers, such as CPLEX [96], Gurobi [97], and MOSEK [98].

Existing approaches show how DSE problems are formulated as an optimization problem, where the MILP-based formulation is increasingly adopted to solve the optimization problems [99]. An alternate technique is MILP combined with simulation and convex problems optimization, as in [95, 100]. Mansoori et al. [101] proposed a general framework for joint optimization based on multi-objective Bayesian Optimization (BO). The proposed techniques use high-level synthesis (HLS) and combine a two-

stage DSE for FPGA-based hardware performance. The two DSE stages consider the training of model parameters to find the best ML and configuration for hardware performance, respectively. Recently, formal specifications have been proposed for handling robustness using a formalism based on satisfiability modulo theory (SMT) [102, 103].

Prerit et al. [76] addressed the CPS design problem as a MOO problem and proposed a two-stage design methodology known as "DISPATCH." The first stage of the method is to use a genetic algorithm (GA) for a predetermined architecture. GA serves the purpose of exploring a discrete design space and simplifying the encoding of architectures. Notably, it explores either both architectures and component values or exclusively focuses on component values. To mitigate the sample inefficiency of GA and its tendency to require a substantial number of iterations to meet design requirements, the algorithm is terminated before complete requirement fulfillment, resulting in a preliminary coarse design. The second stage involves refining the coarse design by searching for component values within a continuous search space. In this phase, a neural network (NN) is leveraged as a surrogate function to model system response. The NN is then transformed into a MILP to encompass constraints imposed by inputs (i.e., component values), outputs (i.e., desired response), and the NN itself. This formulation facilitates the derivation of an inverse design for the system, identifying component values that adhere to a predefined set of system constraints. A feasible solution from the MILP provides the component values used to simulate the system.

In the literature, many scientific articles highlighted an interesting technique of co-simulation for simulation-based exploration, e.g., Finn et al. [95] using MODELICA, combines the discrete optimization routine (MILP) that generates architecture candidates of an aircraft environmental control system over a discrete space with continuous sizing and optimization,

whereas the MODELICA simulator runs the optimization and performs the analysis over a continuous space. Moin et al. [100] performed a synthesis of topologies for body area networks (BANs) and adopted an iterative optimization technique where a MILP-based problem generates candidate network architectures under energy constraints, which are then analyzed in a loop under reliability constraints with a discrete-event network simulator using the Castalia toolbox [104]. An alternative approach consists in integrating simulation and verification, using monitors and transaction detectors derived from formal specifications [105–107]. We do not pursue this avenue in this work.

Moreover, our proposed tool in this research work, i.e., ARCHEX, complements the above-mentioned co-simulation-based techniques. It can fully capture the system requirements and generate correct-by-construction candidates' architecture solutions. The latter can be used for analysis and synthesis using simulation models in a loop; e.g., in [95, 100] the authors adopted a problem-specific MILP formulation; instead, our proposed tool and methodology prove its characteristics to be re-usable, expressible, and extensible, which paves the way for a large number of potential applications in multiple CPS domains. The scope of CPS and the experiments performed into ARCHEX prove its usability, expressiveness, and scalability characteristics. The proposed tool provides the ability to select and interconnect physical and electronic components of a CPS at a conceptual level of design and satisfy the system-level requirements that pave the way for many potential applications in multiple domains. The motivation for extensibility in the proposed architecture exploration methodology came from successfully implementing multiple design examples into the framework.

Additionally, integrating the framework with a simulator is another considerable achievement of this research work. The main challenge in exploit-

ing the integration of any simulator with the optimization tool is to extract meaningful constraints that can guide the optimization tool in converging towards an optimal and feasible solution. The method starts by parsing the simulation output and then tracking the components involved in constraint violation through a process of abstraction to return information that can be used to exclude the solution in the succeeding optimizations. The algorithm then terminates either with a feasible solution or the optimization problem will be declared inconsistent.

Also, simulators are considered by many researchers in almost every domain because of their lower risk, low cost, and effective nature, e.g., Bouras et al. [108] highlight the importance of selecting a suitable simulator depending on the speed, cost, accuracy, modularity, multi-protocol support, and convenience of use. Several simulators can be employed in wireless sensor networks [109,110]. Both academia and industry have developed numerous effective network simulators, particularly for 5G systems, to perform testing and simulation of new algorithms and network behavior, respectively. These simulators have proven to be useful tools for evaluating realistic scenarios. These simulators include: Simu5G [111], Open-Air-Interface [112], Vienna5G [113], 5G-K simulator [114], 5G-LENA [115], WiSE [116], and 5G air- interface [117].

While simulators are effective tools for realizing the complex scenarios of a real-time system, their capabilities are still limited because they are time-consuming, provide a limited number of configuration evaluations, and do not generate the best possible strategies.

2.3 Background: Optimization Tool

ARCHEX is an optimization-based system-level architectural exploration framework that allows the network designer to produce an optimized can-

didate architecture of a system (w.r.t. an objective function) for evaluation. The tool supports an efficient design, produces high-quality analytical solutions, and guarantees correctness. ARCHEX represents a system architecture in the form of a directed graph and generates a network of interconnected nodes in which the components are chosen from a pre-defined system-specific library having specific attributes [118, 119].

The tool can produce an optimal system architectural design by automatically generating and solving multiple constraints by expressing the requirements as a pattern-based formal language. Moreover, the software structure of the ARCHEX framework is modular and amenable to extensibility and design reuse. The algorithms and solvers in ARCHEX provide expressiveness and usability in the tool for a domain-specific network designer to express a set of requirements and exploration problems to be efficiently formulated and solved as MILP optimization problems.

The internal structure of ARCHEX is designed such that it can work on a pattern of multiple class-connected files as shown in Figure 2.1. To generate an optimal system architectural design, the network designer has to provide two text files as input: one is the problem description file that contains various constraints in the form of system specifications, and the second is the library file, which includes the list of components and their associated values and parameters.

2.3.1 Architecture, Template, Topology configuration

Architecture: The architecture generated by ARCHEX is a directed graph model ($\mathbb{G} = \mathbb{V}, \mathbb{E}$), which represents a network of interconnected components, where $\mathbb{V}$ is a set of components (nodes), i.e., $V_1, \ldots, V_{|V|}$, where $\texttt{|V|} \in \mathbb{V}$ and $\mathbb{E}$ is a set of edges, i.e., $e_{ij} \in \mathbb{E}$, where e_{ij} presents the link (edge) between the source i and destination j node in the network. Nodes and edges in the graph are labeled with different attributes, e.g.,

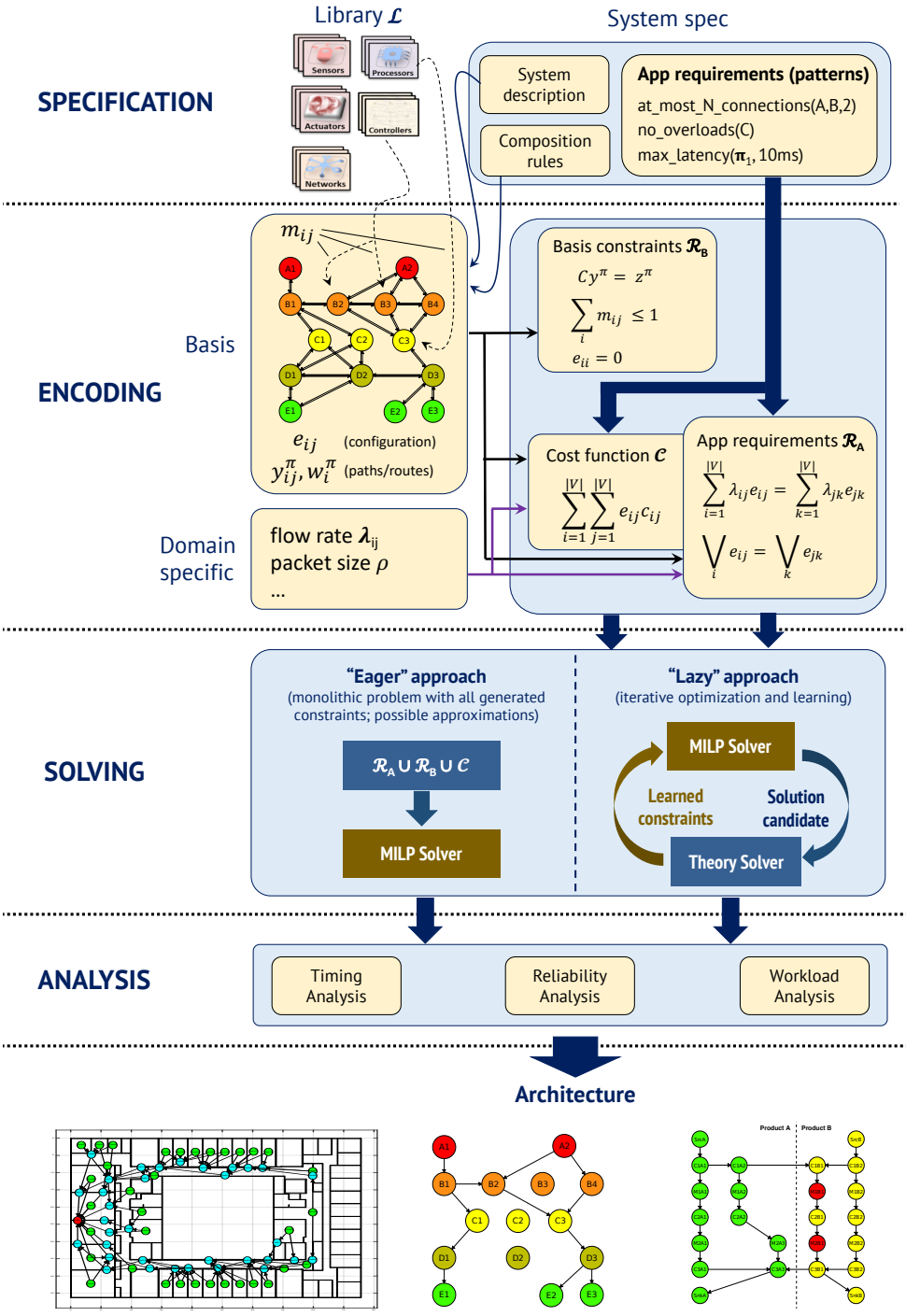


Figure 2.1: Flow of the proposed architecture exploration methodology [14]

"Node_labeling" and "Edge_labeling," such as name, type, cost, sub-type, delay, etc. corresponding to those from the components library $\mathbb{L}$.

The labeled nodes in the interconnected network comply with the composition rules; the rules define a functional flow $\mathbb{F}$ which is an ordered sequence of component types $(t_1 \dots t_n)$ needed to implement a link between a source and destination node. The edges e_{ij} of $\mathbb{G}$ are binary variables presenting a connection between v_i and v_j, where $e_{ij} = 1$ or 0 indicates the presence or absence of connection between the source and destination nodes. Similarly, a node v_i is represented by a binary variable δ_i, which evaluates to 1 if at least one incoming edge e_{ji} or outgoing edge $e_{ij} = 1$.

Template: A template $\mathcal{T}$ is a re-configurable architecture and one of the inputs of the exploration problem. It represents a graph with a fixed set of nodes $\mathbb{V}$ and a variable set of edges $\mathbb{E}$, with the possibility of altering the graph using edge variables e_{ij}. The template contains information about the "what" type and "how" many components can be used in the problem, as shown in Figure 2.2. Initially, the template size is big as it contains the maximum number of nodes and all possible edges, where the size is determined by the number of nodes in the graph, i.e., $|\mathcal{T}| = |\mathbb{V}|$.

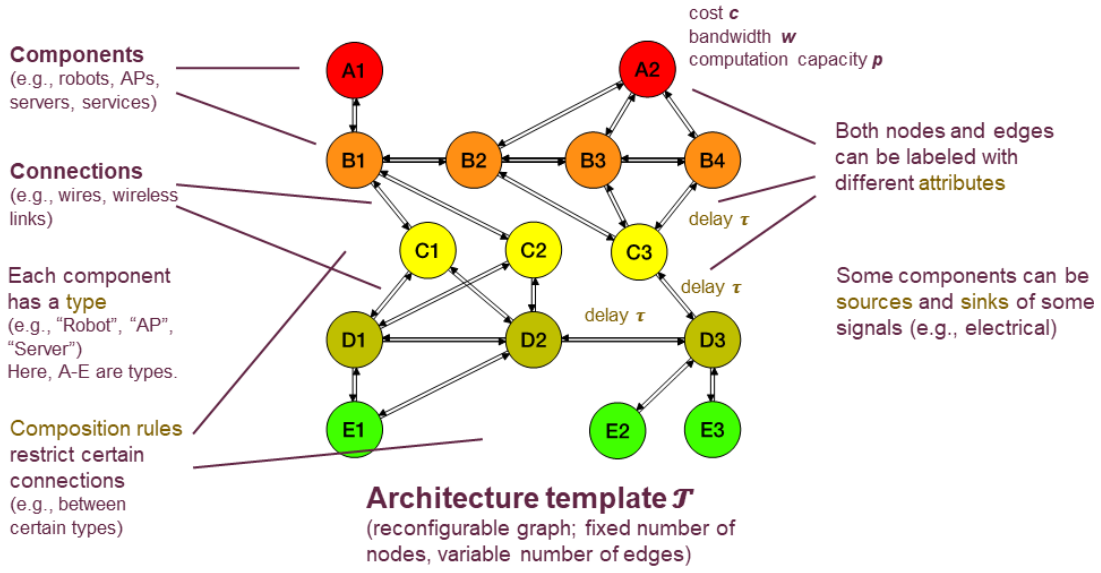


Figure 2.2: A template $\mathcal{T}$ of components, edges, and associated attributes [14]

Mapping: A mapping $\mathbb{M}$ is a function denoted with $\mathbb{M} : \mathbb{V} \rightarrow \mathbb{L}$. The map associates each component of the template (a "virtual component") $v \in \mathbb{V}$ with one or many components in the library (a "real component") $l \in \mathbb{L}$, as shown in Figure 2.3. The mapping $\mathbb{M}$ is defined as a binary variable $m_{ij} \in \mathbb{M}$, which is equal to 1 if a virtual node $v_i \in \mathbb{V}$ is mapped to a real node $l_j \in \mathbb{L}$; otherwise, it is equal to 0.

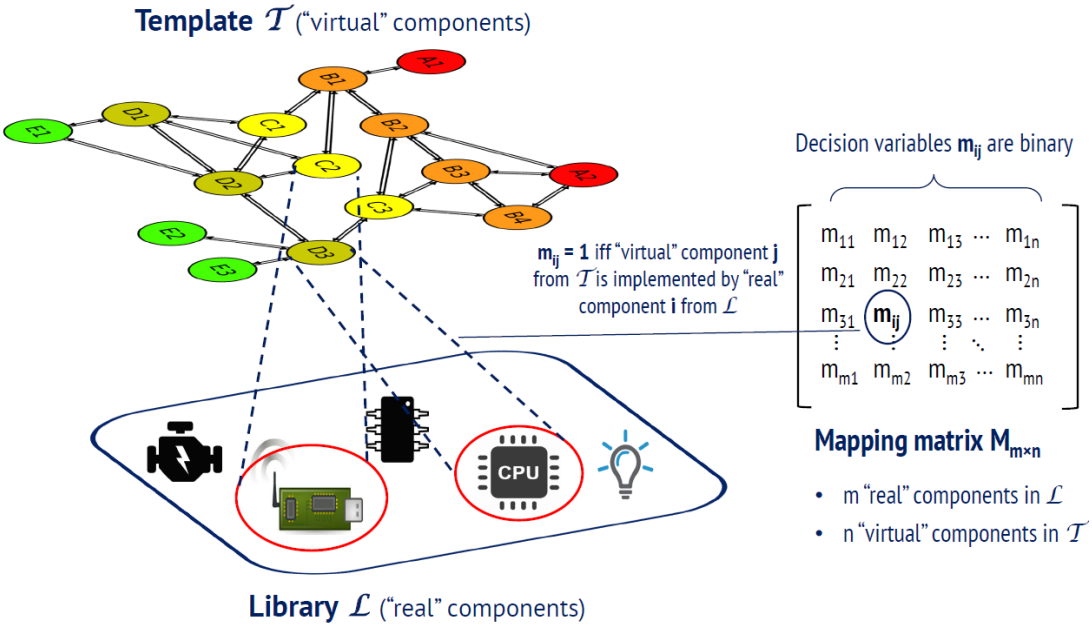


Figure 2.3: Mapping of $\mathcal{T}$ component with a $\mathbb{L}$ component [14]

Topology configuration: Given a template $\mathcal{T} = (\mathbb{V}, \mathbb{E})$ and a library of components $\mathbb{L}$, a topology configuration is a design decision where the optimization find a topology $\mathbb{E}^*$ and a map $\mathbb{M}^*$ to satisfy a set of constraints requirements (e.g., interconnection, routing, link-quality, and QoS) with an objective function to minimize the overall cost. The topology configuration defines which components are used and how they are connected in the final architecture; also, topology configuration can be seen as an instance of $\mathcal{T}$ because the final architecture is much smaller as compared to the $\mathcal{T}$ as many nodes and edges are not used as they can be unnecessary or unused and thus are pruned away during the optimization to minimize the cost.

Figure 2.4*a* shows a $\mathcal{T}$ with all the possible nodes and edges and an adjacency matrix containing binary decision variable e_{ij}, while Figure 2.4*b* shows the topology configuration, which shows the final architecture of the network. In the final topology configuration, only those nodes are used that are connected by edges, while the rest of the nodes without an edge are unused.

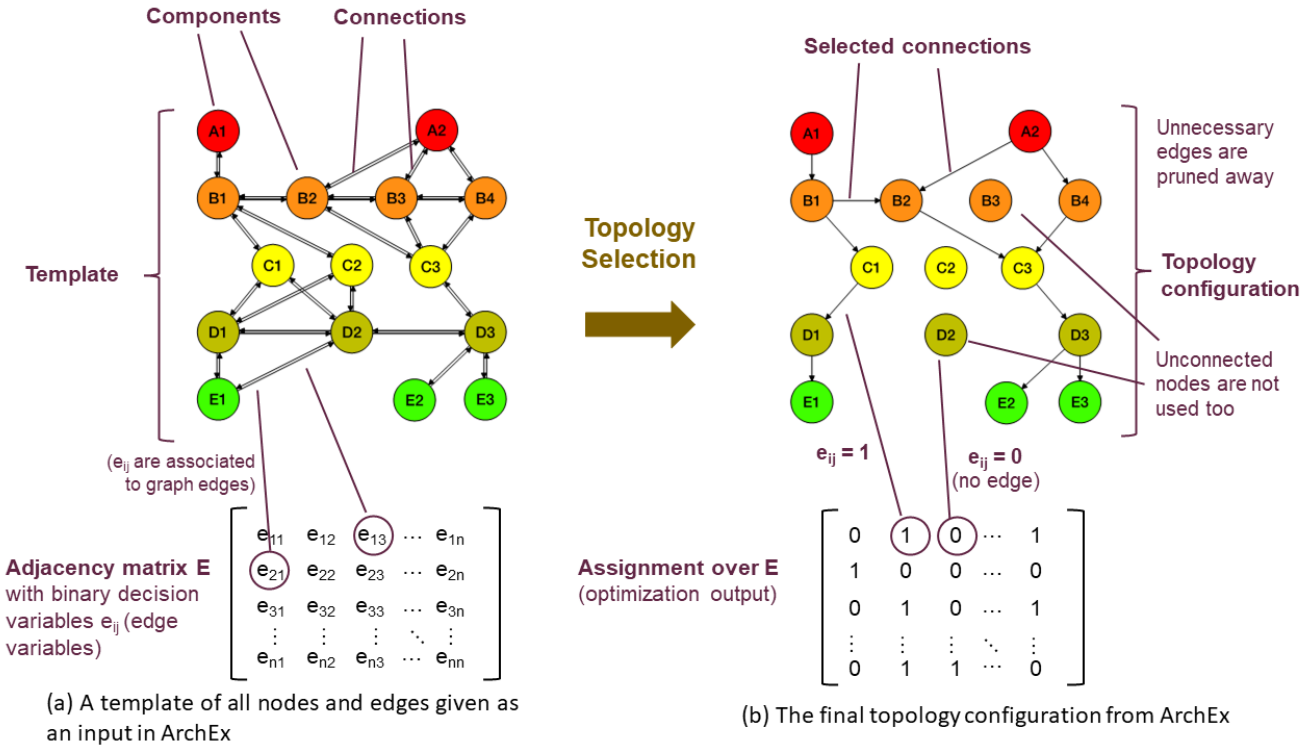


Figure 2.4: Example of template $\mathcal{T}$ and final topology configuration [14]

The next step in the exploration tool to be considered by a system designer is defining application-specific constraints and patterns, e.g., as mentioned in Chapter 5. In the next section, we highlight how ARCHEX supports the system requirements and application constraints to be defined as a set of natural language patterns that automatically generate the MILP constraints for the solver.

2.3.2 Natural language-based patterns in ARCHEX

ARCHEX introduces a concept of requirement patterns, serving as an intermediary layer between natural language and the underlying MILP formulation. Illustrated in Figure 2.5, this approach allows users, particularly designers, to work within a formal language based on patterns. These patterns simplify the formulation of complex problems by automatically handling translations and definitions within the ARCHEX tool.

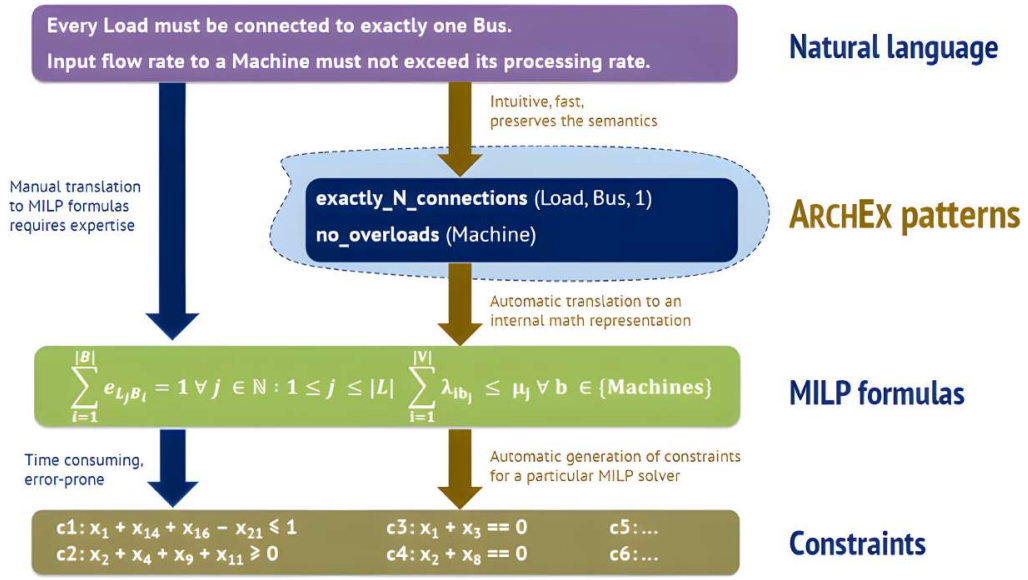


Figure 2.5: Pattern-based formal language as an intermediate between the requirements expressed in natural language and the underlying MILP formulation [14]

In ARCHEX, each requirement pattern has a name that mirrors the associated requirement and a set of arguments specifying its application, such as component types or relevant paths. The tool then automatically generates MILP constraints based on these patterns, operating on corresponding subsets of decision variables. Users interact with the abstract level of these patterns, and the internal workings of problem variables remain transparent. This abstraction facilitates problem formulation, as designers can leverage higher-level primitives to encode problems with-

out manually generating optimization constraints, a typically tedious and error-prone task. The use of patterns contributes to error reduction and minimizes debugging efforts due to their abstract nature.

Table 2.1 presents a list of patterns supported by ARCHEX, categorized based on classes of system requirements (e.g., path, QoS, timing, etc). Each pattern intuitively represents a requirement, maintaining formal semantics while resembling natural-language expressions. For instance, the pattern at_least_n_connections (A, B, 1) represents the requirement that "there must be at least one connection between components of type A and components of type B." Another example is in_conn_implies_out_conn, which encodes balance constraints on component connections. It states that a corresponding type of outgoing edge must be present if there is a specific type of incoming edge. Optional arguments marked with a prime (e.g., S') can be omitted.

The complexity gap between specification and generated MILP formulation significantly diminishes the usability challenges, especially as the problem size increases. This pattern-based language substantially enhances the framework's user-friendliness. The patterns presented in Table 2.1 cover various constraint categories discussed in Chapter 5, demonstrating their versatility across different application domains and their efficacy in capturing the core aspects of the addressed problems.

Table 2.1: List of requirement patterns supported by ARCHEX

Pattern_name	**Description**
Path and Routing	
$\mathcal{P}$ = has_path(src_node, dst_node)	A connection between src_node and dst_node is assigned, denoted as $\mathcal{P}$, which can be utilized in other specifications to reference the current path.
disjoint_edges $(\mathcal{P}_1, \mathcal{P}_2)$	Two Paths denoted as $\mathcal{P}_1, \mathcal{P}_2$ must have disjoint edges.
disjoint_nodes $(\mathcal{P}_1, \mathcal{P}_2)$	Two Paths denoted as $\mathcal{P}_1, \mathcal{P}_2$ must have disjoint nodes, except for the src_node & dst_node.

Reliability	
min_redundant_components (T, S', N)	There must be at least N redundant components of type T and sub-type S (minimal degree of redundancy).
max_fail-prob_of_connection $(T_1, S'_1, T_2, S'_2, value)$	The failure probability of all functional links between components of type T_1 and T_2 must not exceed value.
Timing	
max_latency_of_path $(p, value, units)$	The latency (delay) of the path p must not exceed *value* units (e.g., seconds, minutes).
max_latency_of_component $(T, S', value, units)$	The latency (delay) of components of type T and sub-type S must not exceed *value* units (e.g., seconds, minutes).
Mapping	
at_most_one_mapping (T)	Components of type T can be mapped to at most one element in the platform library.
if_conn_then_has_mapping_strict (T)	If a component of type T is connected, then it must have a mapping to the library.
if_conn_then_has_mapping_soft (T)	a component of type T is not connected, it must not (may) be mapped.
Link-Quality (WSN)	
min_received_sig_strength $(p, value)$	The RSS of every link along the path p must be at least *value* dBm (p can also be a single link).
min_signal_to_noise $(p, value)$	The SNR of every link along the path p must be at least *value* dB (p can also be a single link).
max_bit_error_rate $(p, value)$	The bit error rate (BER) of every link along the path p must be at most *value* % (p can also be a single link).
Localization (WSN)	
min_reachable_devices $(loc, N, metric, value)$	Every location in the set *loc* must be reachable by at least N devices, such that the LQ metric (e.g., "RSS" or "ETX") of corresponding links does not go beyond (or below) *value*.
QoS (WSN)	
atleast_beta $(p, value)$	Every path p between src_node & dst_node must have at least *value* communication bandwidth (p can also be a single link).
atleast_alpha $(p, value)$	Every path p between src_node & dst_node must have at least *value* computation bandwidth (p can also be a single link).
atleast_lambda $(p, value)$	Every path p between src_node & dst_node must have a minimum *value* latency (p can also be a single link).

no_server_overload (p, T_1, T_2)	Every service (T_2) connected to a server (T_1) in a path p (src_node & dst_node) must not exceed server_capacity (*server_capacity*) can be a fixed value).

Moreover, ARCHEX also has the limitation of not providing system-level design optimization in a dynamic environment, e.g., for a network containing movements of robots and its associated service mappings to the resourceful servers, and that is one of the objectives of this research we achieved in (Chapter 5), to overcome the tool deficiency by developing a case study of an industrial automation scenario where multiple robots are moving around the factory floor, acquiring heterogeneous services from MEC servers to perform different tasks simultaneously and by integrating the tool with a simulator.

Although in previous work, ARCHEX framework is extended such that it can support the WSN scenario as shown in Figure 2.6, the scenario is implemented in an indoor and static environment where the components are always placed at a fixed position, and the data has to be transmitted from the sensors through relays to sink node.

Our case study considers robots' trajectories and efficiently utilizing server resources and service mappings. So, to achieve the model objectives and satisfy the system QoS requirements, a comprehensive modification in the previous version of the WSN model and the current version of ARCHEX is required. It will require the development of a new library of components, problem description file, and other files to make possible the implementation of our scenario in ARCHEX. We must generalize the core of ARCHEX by modifying its internal structure with additional QoS constraints for this specific application scenario to make the optimization and mapping process effective while satisfying system requirements and accounting for the distinctive architectural aspects related to our domain.

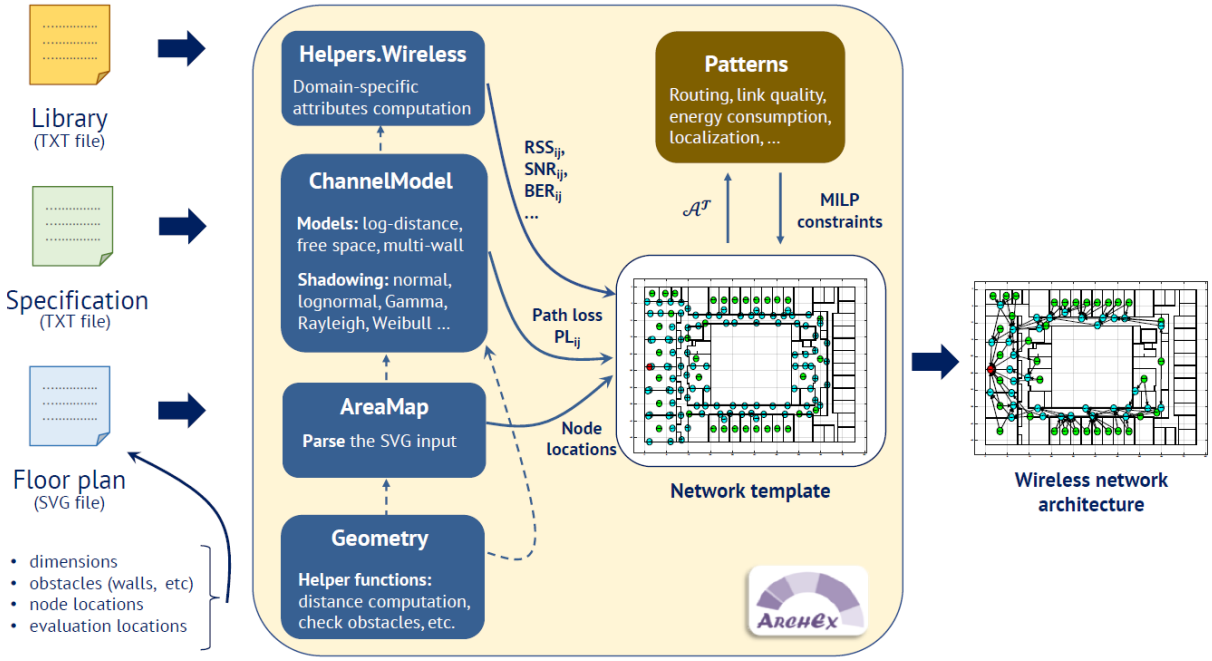


Figure 2.6: The Wireless Sensor Network extension in ARCHEX [14]

We aim to target the tool's static nature and overcome this limitation by designing a dynamic scenario where robots follow a trajectory and continuously change their positions at each discrete time step. We aim to exploit the scalable, expressible, and adaptable capabilities of ARCHEX and then demonstrate various interesting ways a designer can explore the design space of a URLLC-based network that hosts multiple services and maintains their QoS levels.

Chapter 3

Resource Optimization in MEC-based B5G Networks for Indoor Robotics Environment

This work considers a dynamic indoor robotic scenario for B5G edge networks where agents continuously need MEC services and migrate from one cell to another to perform their tasks in an ultra-dense cell environment. Assuming that every MEC service is a virtual machine (VM) to execute in one of the cells with the possibility of migrating the VM to another cell by paying some cost, we formalize the joint problem of (1) placing/migrating the VMs to respect their end-to-end communication latency requirements and (2) allocating their computation and communication bandwidth as a mixed-integer linear program (MILP). A MILP solver is then used to find optimal VM placements/migrations and bandwidth allocations over a time horizon.

3.1 Introduction

The 5G (fifth-generation) and B5G (beyond 5G) network infrastructures will play a vital role in modern technological evolution, including industry 4.0 and autonomous vehicles. In particular, B5G services will require

diverse quality-of-service (QoS) in terms of processing, latency, reliability, and bandwidth [36] and place heavy communication and computation loads on the network, as exemplified by ultra-reliable low-latency communication (URLLC) services whose domain is the convergence of different areas, including control, robotics, signal processing, artificial intelligence, and data analysis. As different areas converge on URLLC services, the services are inherently heterogeneous, increasing the complexity of managing the computing and communication resources available in MEC (multi-access edge computing) servers at the network edge. Therefore, the success of deploying heterogeneous services on B5G networks crucially depends on the optimal management of MEC servers.

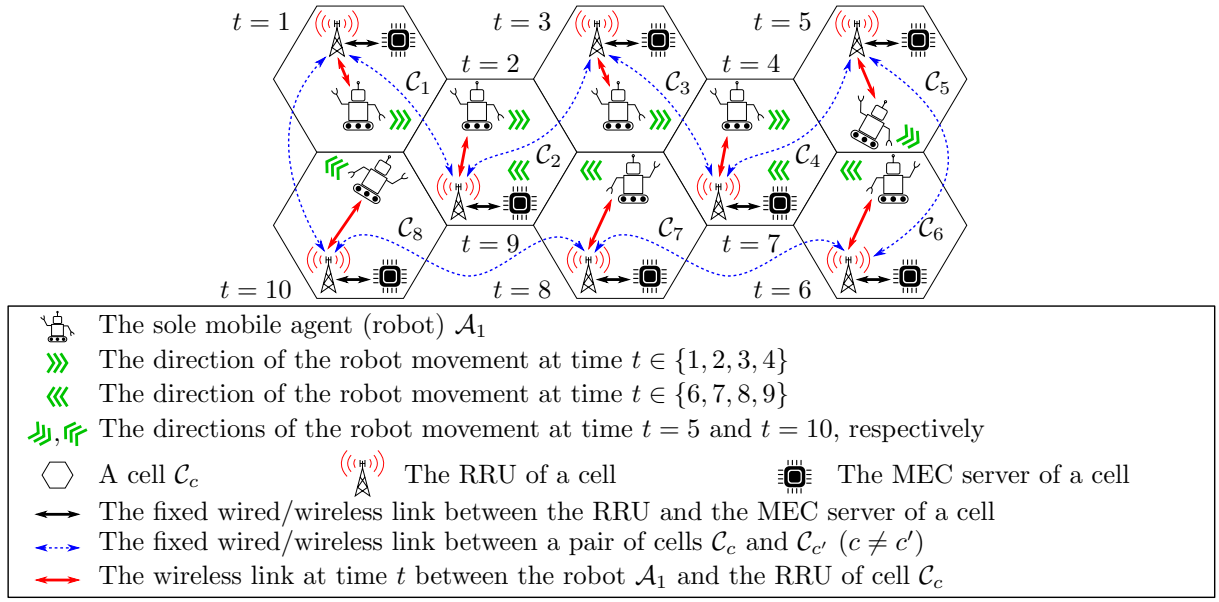


Figure 3.1: A robot serviced by MEC-equipped cells finishes a circuit every ten-time units.

Many researchers have investigated MEC for offloading computation and optimizing resource allocations in static and dynamic scenarios. Markov decision process has been proposed to solve the problem of placing/migrating services and allocating their resources in an uncertain environment [55, 58]. However, the proposals do not jointly optimize the compu-

tation and communication resources at the network edge, which is crucial as the tasks offloaded by emerging applications become more considerable and heterogeneous in terms of their QoS requirements.

Hence, [45, 120, 121] propose solutions for a static user environment and closer to our research objective, [48–50, 122] propose optimization algorithms and iterative procedures that take into account the optimal resource allocations for offloaded tasks subject to latency and energy consumption constraints when the users are mobile. However, none holistically considers all characteristics we consider on an ultra-dense cell environment populated by agents with predictable mobility that continuously offload heterogeneous tasks to MEC servers (Figure 3.1) with diverse QoS requirements.

Section 3.2 presents a novel methodology for flexibly and optimally managing task offloading in the context of RT (real-time) applications that require heterogeneous computing and communication services. Section 3.3 shows an application of the methodology on the network shown in Figure 3.1. Finally, Section 3.4 presents discussions and future work.

3.2 Methodology

Figure 3.2 shows our proposed methodology to flexibly and optimally manage task offloading in RT applications requiring heterogeneous computing and communication services. The flexibility is achieved by first decomposing an RT application into several services, each implemented by a single VM, that impose different upper bounds on their end-to-end communication latency and various ranges of computation and communication bandwidth. Then, each service quantifies its QoS as a function of the experienced end-to-end latency and allocated computation and communication bandwidth. By using quality functions to quantify the QoS of the different

services, the optimality of managing task offloading can then be measured against the greatest possible total QoS achievable in the lifetime of the RT application. To illustrate the steps of the proposed methodology, the scenario depicted in Figure 3.1 is used as a running example in the rest of this paper.

The design in Step 1 assumes a set of mobile agents $\mathbb{A}$ that roam a cellular network $\mathbb{G}$ where each cell $\mathcal{C}_c \in \mathbb{G}$ is equipped with a MEC server. Hence, if multiple RRUs (remote radio units/heads) are served by one MEC server, the RRUs are taken as one cell altogether. For example,

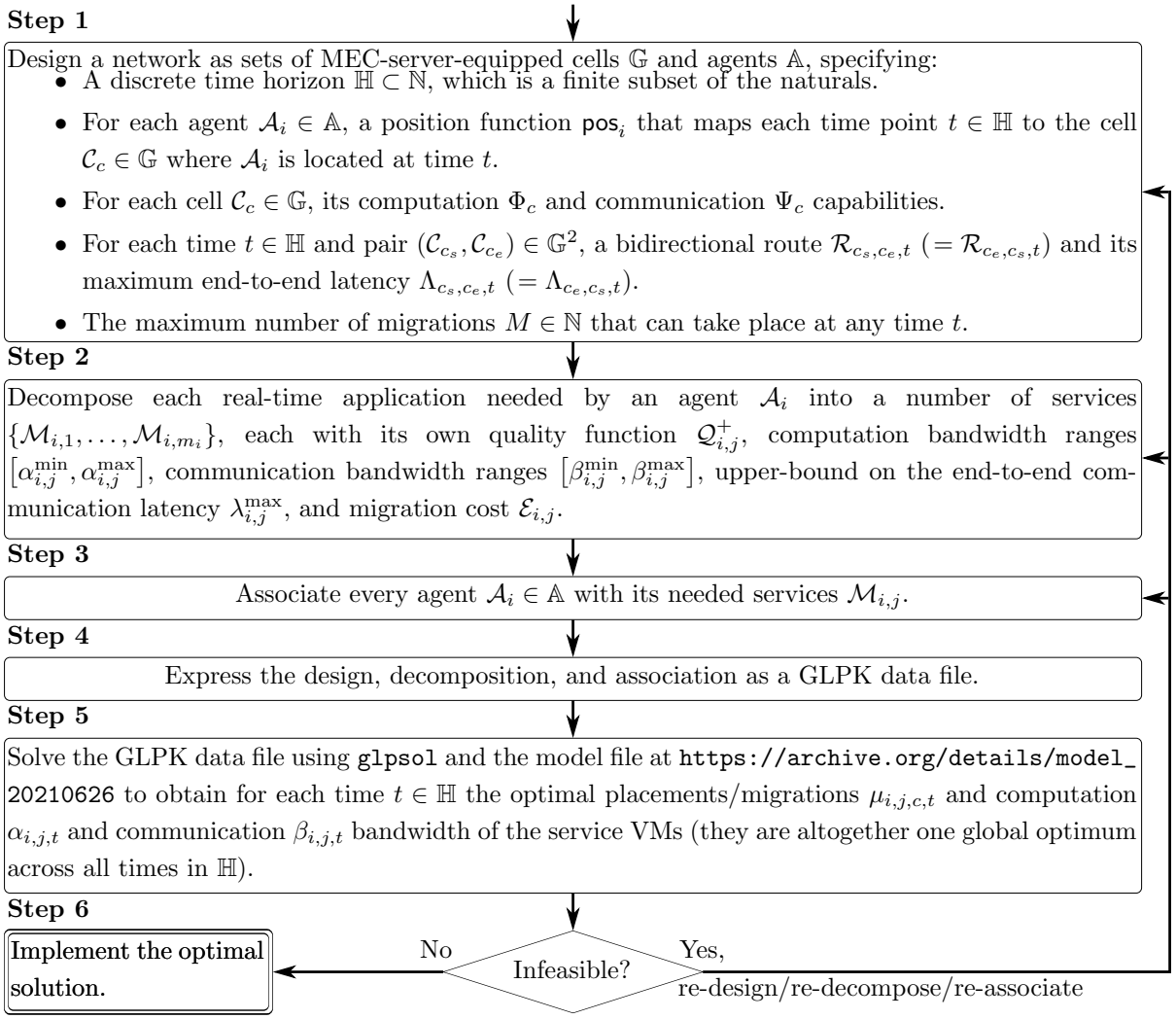


Figure 3.2: Our methodology to flexibly and optimally manage network edge resources.

the network shown in Figure 3.1 has $\mathbb{G} = \{\mathcal{C}_1, \ldots, \mathcal{C}_8\}$. The design then considers a finite discrete time horizon $\mathbb{H}$. For example, for the case shown in Figure 3.1, using $\mathbb{H} = \{1, \ldots, 11\}$ is sufficient to obtain a complete task-offloading plan as the sole agent repeats its complete tour every ten-time units.

The design then specifies a position function $\mathsf{pos}_i : \mathbb{H} \to \mathbb{G}$ for each agent $\mathcal{A}_i \in \mathbb{A}$. For example, the sole agent $\mathcal{A}_1$ in Figure 3.1 has its position function pos_1 defined as $\mathsf{pos}_1(t) = \mathcal{C}_t$ if $1 \leq t \leq 6$, $\mathsf{pos}_1(t) = \mathcal{C}_{11-t}$ if $t \in \{7, 9\}$, and $\mathsf{pos}_1(t) = \mathcal{C}_{7+\frac{t-8}{2}}$ if $t \in \{8, 10\}$. Denoting the latest time in $\mathbb{H}$ as t^* (i.e., $t^* = \max_{t \in \mathbb{H}} t$), the design reserves enough computation and communication bandwidth to allow at most M services to migrate at any

Table 3.1: One possible set of end-to-end routes for the network in Figure 3.1 for all $t \in \mathbb{H}$.

$\boldsymbol{\mathcal{R}_{c_s,c_e,t}}$	$\boldsymbol{c_e = 1}$	$\boldsymbol{c_e = 2}$	$\boldsymbol{c_e = 3}$	$\boldsymbol{c_e = 4}$	$\boldsymbol{c_e = 5}$	$\boldsymbol{c_e = 6}$	$\boldsymbol{c_e = 7}$	$\boldsymbol{c_e = 8}$
$\boldsymbol{c_s = 1}$	$\{\mathcal{C}_1\}$	$\{\mathcal{C}_1, \mathcal{C}_2\}$	$\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$	$\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$	$\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5\}$	$\{\mathcal{C}_1, \mathcal{C}_8, \mathcal{C}_7, \mathcal{C}_6\}$	$\{\mathcal{C}_1, \mathcal{C}_8, \mathcal{C}_7\}$	$\{\mathcal{C}_1, \mathcal{C}_8\}$
$\boldsymbol{c_s = 2}$	$\mathcal{R}_{1,2,t}$	$\{\mathcal{C}_2\}$	$\{\mathcal{C}_2, \mathcal{C}_3\}$	$\{\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$	$\{\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5\}$	$\{\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6\}$	$\{\mathcal{C}_2, \mathcal{C}_1, \mathcal{C}_8, \mathcal{C}_7\}$	$\{\mathcal{C}_2, \mathcal{C}_1, \mathcal{C}_8\}$
$\boldsymbol{c_s = 3}$	$\mathcal{R}_{1,3,t}$	$\mathcal{R}_{2,3,t}$	$\{\mathcal{C}_3\}$	$\{\mathcal{C}_3, \mathcal{C}_4\}$	$\{\mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5\}$	$\{\mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6\}$	$\{\mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6, \mathcal{C}_7\}$	$\{\mathcal{C}_3, \mathcal{C}_2, \mathcal{C}_1, \mathcal{C}_8\}$
$\boldsymbol{c_s = 4}$	$\mathcal{R}_{1,4,t}$	$\mathcal{R}_{2,4,t}$	$\mathcal{R}_{3,4,t}$	$\{\mathcal{C}_4\}$	$\{\mathcal{C}_4, \mathcal{C}_5\}$	$\{\mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6\}$	$\{\mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6, \mathcal{C}_7\}$	$\{\mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6, \mathcal{C}_7, \mathcal{C}_8\}$
$\boldsymbol{c_s = 5}$	$\mathcal{R}_{1,5,t}$	$\mathcal{R}_{2,5,t}$	$\mathcal{R}_{3,5,t}$	$\mathcal{R}_{4,5,t}$	$\{\mathcal{C}_5\}$	$\{\mathcal{C}_5, \mathcal{C}_6\}$	$\{\mathcal{C}_5, \mathcal{C}_6, \mathcal{C}_7\}$	$\{\mathcal{C}_5, \mathcal{C}_6, \mathcal{C}_7, \mathcal{C}_8\}$
$\boldsymbol{c_s = 6}$	$\mathcal{R}_{1,6,t}$	$\mathcal{R}_{2,6,t}$	$\mathcal{R}_{3,6,t}$	$\mathcal{R}_{4,6,t}$	$\mathcal{R}_{5,6,t}$	$\{\mathcal{C}_6\}$	$\{\mathcal{C}_6, \mathcal{C}_7\}$	$\{\mathcal{C}_6, \mathcal{C}_7, \mathcal{C}_8\}$
$\boldsymbol{c_s = 7}$	$\mathcal{R}_{1,7,t}$	$\mathcal{R}_{2,7,t}$	$\mathcal{R}_{3,7,t}$	$\mathcal{R}_{4,7,t}$	$\mathcal{R}_{5,7,t}$	$\mathcal{R}_{6,7,t}$	$\{\mathcal{C}_7\}$	$\{\mathcal{C}_7, \mathcal{C}_8\}$
$\boldsymbol{c_s = 8}$	$\mathcal{R}_{1,8,t}$	$\mathcal{R}_{2,8,t}$	$\mathcal{R}_{3,8,t}$	$\mathcal{R}_{4,8,t}$	$\mathcal{R}_{5,8,t}$	$\mathcal{R}_{6,8,t}$	$\mathcal{R}_{7,8,t}$	$\{\mathcal{C}_8\}$

time $t'' \in \mathbb{H}$ with $t'' < t^*$. In reserving the computation and communication bandwidth, the design makes it so that M migrations taking place at any time t'' complete within one time unit Δ (i.e., $t'' + \Delta = t'' + 1 \in \mathbb{H}$), which in reality can be some seconds or minutes depending on the cell sizes, agent speeds, and migration technique. Hence, when the design specifies for each cell $\mathcal{C}_c$ its computation capability Φ_c and its communication capability Ψ_c, the specified capabilities exclude the bandwidth reserved for migrations. For example, if the network shown in Figure 3.1 reserves Δ_1 computation bandwidth and Δ_2 communication bandwidth to allow at most one migration at any time t'' (i.e., $M = 1$), and every cell $\mathcal{C}_c$ in the network has the same capabilities with $\Phi_c = 100$ GIPS (gigainstructions/second) and $\Psi_c = 1000$ Mbps (megabits/second), then every cell is indeed capable of computing at $(100 + \Delta_1)$ GIPS and communicating at $(1000 + \Delta_2)$ Mbps.

Afterwards, the design designates every end-to-end route $\mathcal{R}_{c_s,c_e,t}$ as a nonempty subset of $\mathbb{G}$. Specifically, $\mathcal{R}_{c,c,t} = \{\mathcal{C}_c\}$ refers to both the wireless link between an agent $\mathcal{A}_i$ and the RRU of cell $\mathcal{C}_c$ and the wired/wireless link between the RRU and the MEC server of $\mathcal{C}_c$. On the other hand, $\mathcal{R}_{c_s,c_e,t} \supseteq \{\mathcal{C}_{c_s}, \mathcal{C}_{c_e}\}$ refers to the wired/wireless links between either $\mathcal{A}_i$ in $\mathcal{C}_{c_s}$ and the MEC server at $\mathcal{C}_{c_e}$ or $\mathcal{A}_i$ in $\mathcal{C}_{c_e}$ and the MEC server at $\mathcal{C}_{c_s}$. For example, one possible design of the routes for the mesh network in Figure 3.1 is shown in Table 3.1. Finally, the design specifies for each route $\mathcal{R}_{c_s,c_e,t}$ its end-to-end latency $\Lambda_{c_s,c_e,t}$, which always includes the latency in the agent-RRU wireless link and, additionally if $c_s \neq c_e$, the total latency of every wired/wireless links between $\mathcal{C}_{c_s}$ and $\mathcal{C}_{c_e}$. For example, the end-to-end latency of each route in Table 3.1 can be stated in terms of hop count so that $\Lambda_{c_s,c_e,t} = |\mathcal{R}_{c_s,c_e,t}|$ (e.g., $\Lambda_{1,1,t} = 1$ hop).

The decomposition in Step 2 assumes that each service communicates with no other service but an agent. Furthermore, since each service $\mathcal{M}_{i,j}$ at any time $t \in \mathbb{H}$ needs some computation bandwidth $\alpha_{i,j,t}$ and some

communication bandwidth $\beta_{i,j,t}$ and expects to experience some end-to-end latency $\lambda_{i,j,t}$ in the communication between $\mathcal{M}_{i,j}$ and $\mathcal{A}_i$, each service specifies their bounds $\alpha_{i,j}^{\min}$, $\alpha_{i,j}^{\max}$, $\beta_{i,j}^{\min}$, $\beta_{i,j}^{\max}$, and $\lambda_{i,j}^{\max}$ such that the correct functioning of the service necessitates $\alpha_{i,j}^{\min} \leq \alpha_{i,j,t} \leq \alpha_{i,j}^{\max}$, $\beta_{i,j}^{\min} \leq \beta_{i,j,t} \leq \beta_{i,j}^{\max}$, and $\lambda_{i,j,t} \leq \lambda_{i,j}^{\max}$. Each service $\mathcal{M}_{i,j}$ then quantifies its QoS at any time $t \in \mathbb{H}$ based on $\alpha_{i,j,t}$, $\beta_{i,j,t}$, and $\lambda_{i,j,t}$ using a quality function $\mathcal{Q}_{i,j}^{+}$.

Every quality function should map to the same range (e.g., $[0, 1]$) so that they are comparable and every migration cost $\mathcal{E}_{i,j}$ is easily expressible in the range. For example, suppose the agent $\mathcal{A}_1$ shown in Figure 3.1 is served by an RT application that is decomposed into three services $\mathcal{M}_{1,1}$, $\mathcal{M}_{1,2}$, and $\mathcal{M}_{1,3}$ with different QoS characteristics shown in Table 3.2. Since $\mathcal{Q}_{1,j}^{+}$ shares the same range $[0, 3]$ due to $25\,\text{GIPS} \leq \alpha_{1,j,t} \leq 75\,\text{GIPS}$, $250\,\text{Mbps} \leq \beta_{1,j,t} \leq 750\,\text{Mbps}$, and $1 \leq \lambda_{1,j,t} \leq \lambda_{1,j}^{\max}$, the migration costs are easily derived to be $\mathcal{E}_{1,1} = 0.6$, $\mathcal{E}_{1,2} = 1.5$, and $\mathcal{E}_{1,3} = 2.4$, which are 20%, 50%, and 80% of 3, respectively.

On the other hand, the association in Step 3 assumes that each service is used exclusively by one agent, and hence, if multiple agents need the same service, then the VM implementing the service is duplicated for each agent (e.g., if $\mathcal{A}_1$ and $\mathcal{A}_2$ need the same service, then the VM of the service is duplicated as $\mathcal{M}_{1,j}$ for $\mathcal{A}_1$ and as $\mathcal{M}_{2,j'}$ for $\mathcal{A}_2$ for some j and j').

The expression of the design, decomposition, and association in Step 4 can be manual, as in Figure 3.3 or automated by some sophisticated network design software. We use GLPK [1] due to its flexibility and extensibility. To save time, CPLEX[2] can be substituted for GLPK in Step 5. The model file in Step 5 expresses our formulation of a MILP problem whose objective is to maximize $\sum_{\mathcal{M}_{i,j} \in \mathbb{M}, t \in \mathbb{H}} (\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t}) - \mathcal{E}_{i,j}\, h_{i,j,t})$ where $\mathbb{M}$ is the set of all services, $h_{i,j,t_0} = 0$ for $t_0 = \min_{t \in \mathbb{H}} t$ and, for $t' > t_0$,

[1] `https://gnu.org/software/glpk`

[2] `https://ibm.com/analytics/cplex-optimizer`

Table 3.2: An example of three heterogeneous services used by the agent in Figure 3.1.

$\boldsymbol{\mathcal{M}_{1,j}}$		$\boldsymbol{j=1}$	$\boldsymbol{j=2}$	$\boldsymbol{j=3}$
Computation Bandwidth	**Min** ($\boldsymbol{\alpha_{1,j}^{\min}}$)	25 GIPS		
	Max ($\boldsymbol{\alpha_{1,j}^{\max}}$)	75 GIPS		
Communication Bandwidth	**Min** ($\boldsymbol{\beta_{1,j}^{\min}}$)	250 Mbps		
	Max ($\boldsymbol{\beta_{1,j}^{\max}}$)	750 Mbps		
Latency Upper-Bound	($\boldsymbol{\lambda_{1,j}^{\max}}$)	2 ms	4 ms	6 ms
Normalized Migration Cost		20%	50%	80%
Quality Function	($\boldsymbol{\mathcal{Q}_{1,j}^{+}(\alpha_{1,j,t}, \beta_{1,j,t}, \lambda_{1,j,t})}$)	$U_{1,j}^{(1)}\left(m_{1,j}^{(1),1}\left(\alpha_{1,j,t}-L_{1,j}^{(1),0}\right)\right)+C_{1,j}^{(1)}+$ $U_{1,j}^{(2)}\left(m_{1,j}^{(2),1}\left(\beta_{1,j,t}-L_{1,j}^{(2),0}\right)\right)+C_{1,j}^{(2)}+$ $U_{1,j}^{(3)}\left(m_{1,j}^{(3),1}\left(\lambda_{1,j,t}-L_{1,j}^{(3),0}\right)\right)+C_{1,j}^{(3)}$		
Weight and Offset	$\boldsymbol{\alpha_{1,j,t}}$, **i.e.,** $\boldsymbol{k=1}$ $\left(\boldsymbol{U_{1,j}^{(k)}, C_{1,j}^{(k)}}\right)$	$(1,0)$		
	$\boldsymbol{\beta_{1,j,t}}$, **i.e.,** $\boldsymbol{k=2}$ $\left(\boldsymbol{U_{1,j}^{(k)}, C_{1,j}^{(k)}}\right)$	$(1,0)$		
	$\boldsymbol{\lambda_{1,j,t}}$, **i.e.,** $\boldsymbol{k=3}$ $\left(\boldsymbol{U_{1,j}^{(k)}, C_{1,j}^{(k)}}\right)$	$(1,1)$		
Interval and Gradient with $\boldsymbol{l=1}$	$\boldsymbol{\alpha_{1,j,t}}\left(\left[\boldsymbol{L_{1,j}^{(k),l-1}, L_{1,j}^{(k),l}}\right], \boldsymbol{m_{1,j}^{(k),l}}\right)$	$\left([25,75], \frac{1}{50}\right)$		
	$\boldsymbol{\beta_{1,j,t}}\left(\left[\boldsymbol{L_{1,j}^{(k),l-1}, L_{1,j}^{(k),l}}\right], \boldsymbol{m_{1,j}^{(k),l}}\right)$	$\left([250,750], \frac{1}{500}\right)$		
	$\boldsymbol{\lambda_{1,j,t}}\left(\left[\boldsymbol{L_{1,j}^{(k),l-1}, L_{1,j}^{(k),l}}\right], \boldsymbol{m_{1,j}^{(k),l}}\right)$	$([1,2],-1)$	$\left([1,4], -\frac{1}{3}\right)$	$\left([1,6], -\frac{1}{5}\right)$

$h_{i,j,t'} = 1$ if $\mathcal{M}_{i,j}$ was on the MEC server of another cell at time $t'-1$ (i.e., $\mathcal{M}_{i,j}$ migrated at time $t'-1$ and completed by time t') but $h_{i,j,t'} = 0$ otherwise. The decision variables $\alpha_{i,j,t}$, $\beta_{i,j,t}$, $\lambda_{i,j,t}$, and $h_{i,j,t}$ depend on the decision variable $\mu_{i,j,c,t}$ (not shown) that specifies which cell $\mathcal{C}_c$ has $\mathcal{M}_{i,j}$ running on its MEC server at time t because $\mu_{i,j,c,t}$ determines $\alpha_{i,j,t}$ and $\beta_{i,j,t}$ due to the limited computation Φ_c and communication Ψ_c capabilities, $\lambda_{i,j,t}$ due to the mobility of $\mathcal{A}_i$ and latency $\Lambda_{c_s,c_e,t}$, and $h_{i,j,t}$ due to $h_{i,j,t'} = 1$ whenever $\mu_{i,j,c,t'-1} \neq \mu_{i,j,c,t'}$.

3.3 Application

To demonstrate the proposed methodology, we apply the methodology in Figure 3.2 on the network in Figure 3.1 with the resulting data file in Step 4 given in Figure 3.3, and henceforth, every line number refers to Figure 3.3. The 11 time points and the eight cells $\mathcal{C}_{c_1}, \ldots, \mathcal{C}_{|\mathbb{G}|}$ are specified in line 1 (`Hbb` $t_0 \ldots t^*$ and `Gbb` $c_1 \ldots c_{|\mathbb{G}|}$) with their equal capabilities specified in line 2 (`Phi default` Φ_c, `Psi default` Ψ_c, and `M` M). The routes shown in Table 3.1 are specified in lines 3–12 (`Rcal[`c_s`,`c_e`,1]` $\mathcal{R}_{c_s,c_e,1}$) with

```
1  set Hbb 1 2 3 4 5 6 7 8 9 10 11; set Gbb 1 2 3 4 5 6 7 8;
2  param Phi default 100; param Psi default 1000; param M 1;
3  # The model file states that unspecified Rcal[s,e,1] defaults to {s,e} while
4  # Rcal[s,e,t'] for every t'>1 defaults to Rcal[s,e,t'-1].
5  set Rcal[1,3,1] 1 2 3;    set Rcal[2,4,1] 2 3 4;   set Rcal[3,5,1] 3 4 5;
6  set Rcal[1,4,1] 1 2 3 4;  set Rcal[2,5,1] 2 3 4 5; set Rcal[3,6,1] 3 4 5 6;
7  set Rcal[1,5,1] 1 2 3 4 5;set Rcal[2,6,1] 2 3 4 5 6;
8  set Rcal[1,6,1] 1 8 7 6;  set Rcal[2,7,1] 2 1 8 7; set Rcal[3,7,1] 3 4 5 6 7;
9  set Rcal[1,7,1] 1 8 7;    set Rcal[2,8,1] 2 1 8;   set Rcal[3,8,1] 3 2 1 8;
10 set Rcal[4,6,1] 4 5 6;    set Rcal[5,7,1] 5 6 7;
11 set Rcal[4,7,1] 4 5 6 7;  set Rcal[5,8,1] 5 6 7 8;
12 set Rcal[4,8,1] 4 5 6 7 8;set Rcal[6,8,1] 6 7 8;
13 # Unspecified RcalLambda[s,*,t'] for t'>1 defaults to RcalLambda[s,*,t'-1].
14 param RcalLambda [1,*,1] 1 1, 2 2, 3 3, 4 4, 5 5, 6 4, 7 3, 8 2
15                   [2,*,1] 2 1, 3 2, 4 3, 5 4, 6 5, 7 4, 8 3
16                   [3,*,1] 3 1, 4 2, 5 3, 6 4, 7 5, 8 4
17                   [4,*,1] 4 1, 5 2, 6 3, 7 4, 8 5
18                   [5,*,1] 5 1, 6 2, 7 3, 8 4
19                   [6,*,1] 6 1, 7 2, 8 3
20                   [7,*,1] 7 1, 8 2
21                   [8,*,1] 8 1;
22 param pos [1,*] 1 1, 2 2, 3 3, 4 4, 5 5, 6 6, 7 4, 8 7, 9 2, 10 8, 11 1;
23 set Mbb (1,*) 1 2 3; param Ecal [1,*] 1 0.6, 2 1.5, 3 2.4;
24 param alpha_min default 25; param alpha_max default 75;
25 param beta_min default 250; param beta_max default 750;
26 param lambda_max [1,*] 1 2, 2 4, 3 6;
27 param U [1,1,*] 1 1, 2 1, 3 1 [2,1,*] 1 1, 2 1, 3 1 [3,1,*] 1 1, 2 1, 3 1;
28 param n [1,1,*] 1 1, 2 1, 3 1 [2,1,*] 1 1, 2 1, 3 1 [3,1,*] 1 1, 2 1, 3 1;
29 param L [1,0,1,*] 1  25, 2  25, 3  25 [1,1,1,*] 1  75, 2  75, 3  75
30         [2,0,1,*] 1 250, 2 250, 3 250 [2,1,1,*] 1 750, 2 750, 3 750
31         [3,0,1,*] 1   1, 2   1, 3   1 [3,1,1,*] 1   2, 2   4, 3   6;
32 param C [1,1,*] 1 0, 2 0, 3 0 [2,1,*] 1 0, 2 0, 3 0 [3,1,*] 1 1, 2 1, 3 1;
33 param m [1,1,1,*] 1 0.02,  2 0.02, 3 0.02 [2,1,1,*] 1 0.002, 2 0.002, 3 0.002
34         [3,1,1,*] 1    -1, 2 -.333333, 3 -.2;                          end;
```

Figure 3.3: The data file `data.glp` in Step 4 of our methodology application on Figure 3.1.

their latency in lines 13–21 (`RcalLambda [`c_h`,*,1]` c_h $\Lambda_{c_h,c_h,1}$, (c_h+1) $\Lambda_{c_h,c_h+1,1}, \ldots,$ $|\mathbb{G}|$ $\Lambda_{c_h,|\mathbb{G}|,1}$). The mobility of the agent $\mathcal{A}_1$ is then specified in line 22 (`pos [`i`,*]` t_0 $\mathsf{pos}_i(t_0), \ldots,$ t^* $\mathsf{pos}_i(t^*)$). The services in Table 3.2 are specified in line 23 (`Mbb (`i`,*)` $j_{i,1} \ldots j_{i,m_i}$) with their migration costs in line 23 (`Ecal [`i`,*]` $j_{i,1}\,\mathcal{E}_{i,j_{i,1}}, \ldots, j_{i,m_i}\,\mathcal{E}_{i,j_{i,m_i}}$), computation bounds in line 24 (`alpha_min default` $\alpha_{i,j}^{\min}$ and `alpha_max default` $\alpha_{i,j}^{\max}$), communication bounds in line 25 (`beta_min default` $\beta_{i,j}^{\min}$ and `beta_max default` $\beta_{i,j}^{\max}$), latency bounds in line 26 (`lambda_max [`i`,*]` $j_{i,1}\,\lambda_{i,j_{i,1}}^{\max}, \ldots, j_{i,m_i}\,\lambda_{i,j_{i,m_i}}^{\max}$), and quality functions in lines 27–34 ($U[k,i,*]\,j_{i,1}\,U_{i,j_i,1}^{(k)}, \ldots, j_{i,mi}\,U_{i,j_i,mi}^{(k)}$, $n[k,i,*]\,j_{i,1}\,l, \ldots, j_{i,mi}\,l$, $L[k,l',i,*]\,j_{i,1}\,L_{i,j_i,1}^{(k)}, \ldots, j_{i,mi}\,L_{i,j_i,mi}^{(k),l'}$, $C[k,i,*]\,j_{i,1}\,C_{i,j_i,1}^{(k)}, \ldots,$ $j_{i,mi}C_{i,j_i,mi}^{(k)}$, and $m[k,l',i,*]\,j_{i,1}\,m_{i,j_i,1}^{(k)}, \ldots, j_{i,mi}\,m_{i,j_i,mi}^{(k),l'}$).

Finally, to get a complete task-offloading plan that is repeatable every ten-time units, the following constraint is added to the model file in Step 5 so that each service is on the same server both at time 1 and 11: `rep{(i,j) in Mbb,c in Gbb}: mu[i,j,c,1] = mu[i,j,c,11]`. In Step 5, the optimal solution can be obtained by running GLPK as `glpsol -m model.glp -d data.glp -o out.sol`. To substitute CPLEX for GLPK, the input file `in.lp` can be obtained by running GLPK as `glpsol --check --wlp in.lp -m model.glp -d data.glp`.

Using CPLEX CC8ATML (i.e., 20.1.0 for GNU/Linux) by running `cplex -c "read in.lp" mipopt "write out.sol"` on Ubuntu 16.04.7 on a Lenovo E40-80 laptop with a 4-core Intel Core i3-5010U (2×64-bit 2.1-GHz cores, 2 threads/core), the optimal solution (`out.sol`) was obtained in 78.96 s (13677.56 ticks) without memory swapping to disk. Table 3.3 shows the result of post-processing `out.sol` to start Step 6 which is as follows: $\alpha_{1,j,t} = 75$ GIPS, the average $\lambda_{1,1,t}$, $\lambda_{1,2,t}$, and $\lambda_{1,3,t}$ are 1.4, 2.3, and 2.8 hops, respectively. The numbers in parentheses are $\beta_{1,j,t}$ in tens of Mbps, and the colors are alternated across a row whenever the value changes to double-check that $M = 1$ is respected.

Table 3.3: Communication bandwidth (β) in tens of Mbps, and the alternated colors across a row to double-check service migration (M) is respected

t	1	2	3	4	5	6	7	8	9	10
$\mathcal{A}_1$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$	$\mathcal{C}_4$	$\mathcal{C}_7$	$\mathcal{C}_2$	$\mathcal{C}_8$
$\mathcal{M}_{1,1}$	$\mathcal{C}_1(50)$	$\mathcal{C}_3(25)$	$\mathcal{C}_3(25)$	$\mathcal{C}_4(25)$	$\mathcal{C}_6(25)$	$\mathcal{C}_6(25)$	$\mathcal{C}_4(25)$	$\mathcal{C}_7(50)$	$\mathcal{C}_1(50)$	$\mathcal{C}_1(50)$
$\mathcal{M}_{1,2}$	$\mathcal{C}_8(25)$	$\mathcal{C}_8(25)$	$\mathcal{C}_5(25)$	$\mathcal{C}_5(25)$	$\mathcal{C}_5(25)$	$\mathcal{C}_5(50)$	$\mathcal{C}_5(50)$	$\mathcal{C}_5(25)$	$\mathcal{C}_5(25)$	$\mathcal{C}_8(25)$
$\mathcal{M}_{1,3}$	$\mathcal{C}_2(25)$	$\mathcal{C}_2(50)$	$\mathcal{C}_2(50)$	$\mathcal{C}_2(50)$	$\mathcal{C}_2(50)$	$\mathcal{C}_2(25)$	$\mathcal{C}_2(25)$	$\mathcal{C}_2(25)$	$\mathcal{C}_2(25)$	$\mathcal{C}_2(25)$

3.4 Discussions

For a dynamic indoor B5G network roamed by agents that continuously offload their heterogeneous RT tasks in an ultra-dense cell environment, we have proposed a methodology for flexibly and optimally managing the computing and communication resources at the network edge, demonstrating its application on a mesh network roamed by one agent that needs three services with distinct end-to-end latency upper bounds. The demonstration-scale is small for clarity, but in our ongoing investigation on the scalability of the methodology, a 10-by-10 grid (100 cells) with a mesh topology roamed by 40 agents that each needs three services can be solved within 5% of the optimum in a few minutes by CPLEX 20.1.0.

Chapter 4

Migration-Aware Optimized Resource Allocation in B5G Edge Networks

This work focuses on an ultra-dense edge network with multi-access edge computing (MEC) facilities, serving agents that execute their tasks by touring the cells. Specifically, we propose a novel methodology for optimally and flexibly managing task offloading in the context of heterogeneous computing and communication services required by real-time robotic applications. Differing from many related works, the proposed approach takes the number of admitted service migrations and the QoS upper and lower bounds as binding constraints. We model the QoS evolution based on the agent positions, the MEC servers serving the agents, the QoS requirements, the communication capabilities in the edge network, and the computing capabilities of the servers. The model is formalized as a mixed-integer linear program (MILP) to obtain an optimal schedule for the service migrations and communication and computation bandwidth allocation.

4.1 Introduction

The research on the next generation of mobile networks is chasing the ambitious objective of jointly supporting, within a flexible and robust com-

munication infrastructure, a vast number of heterogeneous services belonging to mobile broadband reliable low latency communications (MBRLLC), massive ultra-low latency communications (mURLLC), and human-centric services (HCS) categories [36, 43]. The target values of key performance indicators (KPIs) are tremendously challenging compared to those of current 5th-generation (5G) deployments. Depending on the specific use case, effective management of task offloading is crucial to deliver high-quality services [44]; hence, there is a need for the optimal management of computing and communication resources at the network edge.

It is easy to understand that mobile network optimization is one of the significant multidisciplinary research topics addressed in the latest decade. Recent contributions published in the scientific literature, for example, proposed several interesting approaches to optimize the use of computing and communication resources exposed at the edge of the 5G system for task offloading. Many papers [45–47, 120, 121, 123–132] focus on static scenarios where user mobility is not explicitly taken into account. Other papers, e.g., [48–54, 56, 122, 133–137], explicitly account for user mobility.

They propose optimization algorithms or iterative procedures to optimize the allocation of servers to the tasks while respecting energy, latency, and communication delay constraints. Task offloading in mobile scenarios is a complex problem. While moving across the network attachment points, mobile users should be served by different servers at the edge whose position and capabilities can satisfy the expected service level. As the overall management is extremely dynamic, the offered service is implemented through dedicated VMs (virtual machines) or containers; such a dynamic scenario requires frequent migration operations [49, 50, 52–54, 56].

Unfortunately, state-of-the-art solutions have two main problems. First, many B5G use cases (e.g., indoor robotic applications) will be enabled through ultra-dense cell deployments, where a limited number of base sta-

tions cover the area. The frequent handovers triggered by user mobility would produce erratic management of task offloading, resulting in excessive migrations of VMs or containers deployed at the network edge. Unfortunately, to our knowledge, no solution in the literature considers the number of migrations as a system variable to be optimized (e.g., minimized).

Second, while in real-time systems, the QoS of a service can be related to its computation bandwidth [65], in edge computing, the degrees of freedom that affect the QoS are more significant and include at least the computation and communication bandwidth and the physical displacement of the VM delivering the service. Hence, the choice of these parameters can make for an adaptive QoS level between some minimum and maximum. Despite the evident benefits of an adaptive QoS, the flexible provisioning of advanced services in dynamic network conditions is quite ignored in the literature.

To close the gap, we propose a novel methodology to manage B5G task offloading optimally and flexibly in the context of real-time applications, which are represented well in the robotic domain: AGVs (automated guided vehicles) touring a large logistic facility, using the edge facilities for computation-intensive tasks. Network attachment points offer wireless connectivity to agents (e.g., AGVs) that require heterogeneous services. The attachment points are connected to an edge network with computing capabilities provided by MEC servers. The agents then connect to one of the available MEC servers through edge links with fixed communication capabilities. In the robotic domain, it is reasonable to assume that agent mobility is predictable. Hence, the QoS dynamics are modeled in terms of the agent and VM positions, the service requirements, the link communication capabilities, the MEC server computing capabilities, and other parameters.

The model is then translated into a MILP to get optimal VM positions

and their optimal communication and computation bandwidth. Unlike the current state of the art, QoS is a function of computing and communication requirements, end-to-end communication latency, and migration cost. Furthermore, while the computing and communication capabilities are constraints in state of the art, the proposed novel approach is characterized by considering the number of admitted VM migrations and the QoS lower and upper bounds expected by the agents. The three main contributions in this work include the model of the QoS dynamic, translating the model into a MILP, and evaluating the MILP's effectiveness and scalability in various scenarios.

4.2 System Model

B5G network dynamics are viewed at discrete times t with the network edge being required by every mobile agent $\mathcal{A}_i \in \mathbb{A}$ to deliver several real-time services by running their VMs $\mathbb{M}_i = \{\mathcal{M}_{i,1}, \ldots, \mathcal{M}_{i,m_i}\}$. We denote the set of all VMs with $\mathbb{M} = \bigcup_{\mathcal{A}_i \in \mathbb{A}} \mathbb{M}_i$. While $\mathcal{A}_i$ uses several VMs, $\mathcal{M}_{i,j}$ serves exactly one $\mathcal{A}_i$ and communicates with no other VM in $\mathbb{M}$. Every cell $\mathcal{C}_c$ in the network $\mathbb{G}$ hosts a MEC server that has a fixed computation capacity Φ_c. Similarly, the links that connect every cell with the edge network has a fixed communication capacity Ψ_c, as well.

Thanks to the edge network, a VM can migrate from the MEC server in some cell $\mathcal{C}_c$ to the MEC server in another cell $\mathcal{C}_{c'}$ to maximize its QoS. We use $\mu_{i,j,c,t}$ (resp. $\rho_{i,c,t}$) to denote the location of $\mathcal{M}_{i,j}$ (resp. $\mathcal{A}_i$) in terms of the network cells $\mathbb{G}$ such that $\mu_{i,j,c,t} = 1$ (resp. $\rho_{i,c,t} = 1$) if the MEC server that hosts $\mathcal{M}_{i,j}$ is in cell $\mathcal{C}_c$ (resp. $\mathcal{A}_i$ connects to the network by attaching itself wirelessly to cell $\mathcal{C}_c$) and $\mu_{i,j,c,t} = 0$ (resp. $\rho_{i,c,t} = 0$) otherwise. While the position of the agents $\rho_{i,c,t}$ is assumed to be known a priori as given by the function pos_i such that $\rho_{i,c,t} = 1$ if and only if $\mathsf{pos}_i(t) = \mathcal{C}_c$, the location

of the VMs $\mu_{i,j,c,t}$ is to be decided optimally.

Initially at time $t = 0$, every $\mathcal{M}_{i,j}$ runs on exactly one MEC server in some cell $\mathcal{C}_c$, and every $\mathcal{A}_i$ is hosted by exactly one cell $\mathcal{C}_{c'}$, possibly $c = c'$. At any later time $t' > 0$, while migration may have occurred between $t' - 1$ and t', every $\mathcal{M}_{i,j}$ still runs on exactly one possibly-different MEC server and every $\mathcal{A}_i$ is still hosted by one possibly-different cell. It follows that (4.1) holds for every $\mathcal{M}_{i,j}$ and every $\mathcal{A}_i$ at any time t.

$$\begin{aligned} \sum_{\mathcal{C}_c \in \mathbb{G}} \mu_{i,j,c,t} &= 1 \\ \sum_{\mathcal{C}_c \in \mathbb{G}} \rho_{i,c,t} &= 1 \end{aligned} \tag{4.1}$$

Migrating $\mathcal{M}_{i,j}$ is assumed to consume negligible bandwidth and take one time unit with cost $\mathcal{E}_{i,j}$ and with the total migrations that can take place at any time t' being limited by some constant M.[1] We use $h_{i,j,t'}$ to denote the migration of $\mathcal{M}_{i,j}$ at time t' such that $h_{i,j,t'} = 1$ if $\mathcal{M}_{i,j}$ was at time $t' - 1$ not hosted by the MEC server in $\mathcal{C}_c$ but at time t' is hosted by the server in $\mathcal{C}_c$, else $h_{i,j,t'} = 0$. Hence, (4.2) and (4.3) hold.

$$h_{i,j,t} = \begin{cases} 0 & \text{, if } t = 0 \\ 1 - \sum_{\mathcal{C}_c \in \mathbb{G}} \mu_{i,j,c,t-1} \cdot \mu_{i,j,c,t} & \text{, otherwise} \end{cases} \tag{4.2}$$

$$\sum_{\mathcal{M}_{i,j} \in \mathbb{M}} h_{i,j,t} \leq M \tag{4.3}$$

Beside $\mu_{i,j,c,t}$, the computation bandwidth, denoted $\alpha_{i,j,t}$, and the communication bandwidth, denoted $\beta_{i,j,t}$, given to $\mathcal{M}_{i,j}$ at any time t are to be decided optimally as well subject to (4.4) and (4.5) with $\alpha_{i,j}^{\min}$ and $\beta_{i,j}^{\min}$ being the least bandwidth below which the service cannot be provided and

[1] This is realistic by choosing a suitable time unit and by reserving some computation and communication bandwidth, if not dedicating processor cores and network links to support M concurrent migrations.

$\alpha_{i,j}^{\max}$ and $\beta_{i,j}^{\max}$ being the greatest bandwidth above which further allocations yield no benefits on the QoS.

$$\alpha_{i,j}^{\min} \leq \alpha_{i,j,t} \leq \alpha_{i,j}^{\max} \tag{4.4}$$

$$\beta_{i,j}^{\min} \leq \beta_{i,j,t} \leq \beta_{i,j}^{\max} \tag{4.5}$$

On the other hand, the total computation and communication bandwidth allocated to the VMs hosted by a MEC server cannot exceed the maximum computation and link capacities of the server. It follows that the computation bandwidth $\phi_{i,j,c,t}$ allocated to $\mathcal{M}_{i,j}$ by the MEC server in a cell $\mathcal{C}_c$ at time t has to satisfy (4.6) and (4.7). Note that if the MEC server in $\mathcal{C}_c$ executes the VM $\mathcal{M}_{i,j}$ at time t, then $\phi_{i,j,c,t} > 0$ but $\phi_{i,j,c',t} = 0$ for the MEC servers in all other cells $\mathcal{C}_{c'}$. Furthermore, the communication bandwidth $\psi_{i,j,c,t}$ of the link that at time t connects an agent $\mathcal{A}_i$ to its VM $\mathcal{M}_{i,j}$ running on the MEC server in $\mathcal{C}_c$ (i.e., $\mu_{i,j,c,t} = 1$) depends on whether $\rho_{i,c,t} = 1$. If it is, only the $\mathcal{A}_i \leftrightarrow \mathcal{M}_{i,j}$ link in $\mathcal{C}_c$ allocates some communication bandwidth.

Else, the communication bandwidth is also allocated by all other links involved in routing the $\mathcal{A}_i \leftrightarrow \mathcal{M}_{i,j}$ communication. With respect to the routing, we assume that at any time t there exists exactly one logical loop-free cycle-free bidirectional route $\mathcal{R}_{c_s,c_e,t} \in \wp(\mathbb{G})$ from $\mathcal{C}_{c_s}$ to $\mathcal{C}_{c_e}$ where $\wp(\mathbb{G})$ is the power set of $\mathbb{G}$. Clearly, $\{\mathcal{C}_{c_s}, \mathcal{C}_{c_e}\} \subseteq \mathcal{R}_{c_s,c_e,t}$ (i.e., the start and end cells are on the route), $\mathcal{R}_{c_s,c_e,t} = \mathcal{R}_{c_e,c_s,t}$ (i.e., the route is bidirectional), and $\mathcal{R}_{c,c,t}$ refers to the agent-connecting wireless link. The binary value $\eta_{c,c_s,i,t}$ is used to denote whether at time t, cell $\mathcal{C}_c$ is on the route $\mathcal{R}_{c_s,c_e,t}$ between some cell $\mathcal{C}_{c_s}$ and the cell $\mathcal{C}_{c_e}$ where the agent $\mathcal{A}_i$ is. Specifically, if $\rho_{i,c_e,t} = 1$ and $\mathcal{C}_c \in \mathcal{R}_{c_s,c_e,t}$, then $\eta_{c,c_s,i,t} = 1$ (i.e., $\mathcal{C}_c$ is in $\mathcal{R}_{c_s,c_e,t}$ to route any bidirectional communication between some VM $\mathcal{M}_{i,j}$ running on the MEC server in $\mathcal{C}_{c_s}$ and its agent $\mathcal{A}_i$ in $\mathcal{C}_{c_e}$), else $\eta_{c,c_s,i,t} = 0$.

The communication bandwidth allocated to each VM $\psi_{i,j,c,t}$ then has to

satisfy (4.8) and (4.9) (i.e., the total allocated bandwidth cannot exceed the available one). Clearly, (4.7)/(4.9) fails for some VM $\mathcal{M}_{i,j}$ whenever at time t the MEC server in some cell $\mathcal{C}_c$ hosts too many VMs such that the sum of $\alpha^{\min}_{i,j}/\beta^{\min}_{i,j}$ of all the VMs $\mathcal{M}_{i,j}$ hosted by the server in $\mathcal{C}_c$ exceeds the respective capacity Φ_c/Ψ_c of the server in order to satisfy (4.4)/(4.5). In case of failing (4.7), migrating $\mathcal{M}_{i,j}$ to another server in $\mathcal{C}_{c'}$ may respect both (4.7) and (4.4) at time t. However, in the case of failing (4.9), the migration cannot respect both (4.9) and (4.5) in $\mathcal{C}_{c'}$ as $\mathcal{M}_{i,j}$ still needs at least $\beta^{\min}_{i,j}$ of the link capacity at $\mathcal{C}_c$ to communicate with $\mathcal{A}_i$, and hence, it follows that $\sum_{\mathcal{A}_i \in \mathbb{A}} \rho_{i,c,t} \left(\sum_{\mathcal{M}_{i,j} \in \mathbb{M}_i} \beta^{\min}_{i,j} \right) \leq \Psi_c$ for every cell $\mathcal{C}_c$ that hosts some agent at time t.

$$\phi_{i,j,c,t} = \mu_{i,j,c,t} \cdot \alpha_{i,j,t} \tag{4.6}$$

$$\sum_{\mathcal{M}_{i,j} \in \mathbb{M}} \phi_{i,j,c,t} \leq \Phi_c \tag{4.7}$$

$$\psi_{i,j,c,t} = \sum_{\mathcal{C}_{c_s} \in \mathbb{G}} \eta_{c,c_s,i,t} \cdot \mu_{i,j,c_s,t} \cdot \beta_{i,j,t} \tag{4.8}$$

$$\sum_{\mathcal{M}_{i,j} \in \mathbb{M}} \psi_{i,j,c,t} \leq \Psi_c \tag{4.9}$$

Another important aspect of the model is the end-to-end communication latency. Each route $\mathcal{R}_{c_s,c_e,t}$ has some end-to-end latency $\Lambda_{c_s,c_e,t}$ such that $\Lambda_{c_s,c_e,t} = \Lambda_{c_e,c_s,t}$ due to bidirectionality. The end-to-end latency $\lambda_{i,j,t}$ between $\mathcal{M}_{i,j}$ and $\mathcal{A}_i$ at t is then given by (4.10) where $\tau_{c_s,i,t}$ is the $\Lambda_{c_s,c_e,t}$ of the $\mathcal{C}_{c_e}$ that satisfies $\rho_{i,c_e,t} = 1$.[2] The end-to-end latency $\lambda_{i,j,t}$ is then required by (4.11) to satisfy some upper-bound beyond which $\mathcal{A}_i$ would fail in executing its real-time tasks.

$$\lambda_{i,j,t} = \sum_{\mathcal{C}_{c_s} \in \mathbb{G}} \mu_{i,j,c_s,t} \cdot \tau_{c_s,i,t} \tag{4.10}$$

[2] As $\mathcal{C}_{c_e}$ always exists at any t due to $\mathcal{A}_i$ being exactly in one cell at any t, it follows that $\Lambda_{c_s,c_e,t}$ is always defined at any t due to the presence of $\mathcal{R}_{c_s,c_e,t}$ at any t. Hence, if $\Lambda_{c_s,c_e,t} > 0$ for every cell pair $(\mathcal{C}_{c_s}, \mathcal{C}_{c_e})$ and for time point t, then $\tau_{c_s,i,t} > 0$ also for every cell $\mathcal{C}_{c_s}$ and time point t.

$$\lambda_{i,j,t} \leq \lambda_{i,j}^{\max} \tag{4.11}$$

We now use the framework just shown to define the total QoS function. Specifically, we define in (4.12) the total QoS experienced by $\mathcal{M}_{i,j}$ at t in terms of a quality function $\mathcal{Q}_{i,j}^{+}$ and a migration cost $\mathcal{Q}_{i,j,t}^{-}$. The $\mathcal{Q}_{i,j}^{+}$ quantifies the QoS of $\mathcal{M}_{i,j}$ as a function of the given bandwidth for computation $\alpha_{i,j,t}$ and communication $\beta_{i,j,t}$ and the experienced end-to-end latency $\lambda_{i,j,t}$ in a manner that is specific to $\mathcal{M}_{i,j}$. The $\mathcal{Q}_{i,j,t}^{-}$, however, follows from the previous definitions and is given in (4.13) where the negative sign is justified by the migration's adverse effect on the QoS.

$$\mathcal{Q}_{i,j,t} = \mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t}) + \mathcal{Q}_{i,j,t}^{-} \tag{4.12}$$

$$\mathcal{Q}_{i,j,t}^{-} = -\mathcal{E}_{i,j} \cdot h_{i,j,t} \tag{4.13}$$

4.3 Formulation of the Optimization Problem

The model is formulated as a mixed-integer program (MIP) optimization problem in terms of (4.12), the different QoS experienced by every VM. Specifically, the MIP formulation maximizes (4.14) subject to (4.1)–(4.11) over a time horizon $\mathbb{H} \in (\wp(\mathbb{N}^{+} \cup \{0\}) \setminus \{\emptyset\})$, which is a finite subset of the naturals.

$$\sum_{t \in \mathbb{H}} \sum_{\mathcal{M}_{i,j} \in \mathbb{M}} \mathcal{Q}_{i,j,t} \tag{4.14}$$

Hence, due to the time horizon, the symbols t and t' in the MIP formulation refer to the members of $\mathbb{H}$ with $t \in \mathbb{H}$ and $t' \in (\mathbb{H} \setminus \{\min \mathbb{H}\})$, while the time $t = 0$ refers to the time $t_0 = \min \mathbb{H}$ in the MIP formulation. We now discuss how to turn our non-linear MIP into a mixed-integer linear program (MILP) by getting rid of its non-linear forms: the products of decision variables in (4.2), (4.6), and (4.8) and the expression $\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$ in (4.12). To that end, we adapt established techniques [138] as shown next. To turn (4.2), which involves the product of two binary variables,

into a linear form, we introduce the binary decision variable $z_{i,j,c,t'} \in \{0,1\}$ with $t' \in \mathbb{N}^+$ (positive naturals) and require that (4.15)–(4.17) hold. Since it can be shown by means of a truth table that $z_{i,j,c,t'} = \mu_{i,j,c,t'-1} \cdot \mu_{i,j,c,t'}$ if and only if (4.15)–(4.17) hold, the MILP formulation replaces (4.2) with (4.15)–(4.18).

$$z_{i,j,c,t'} \leq \mu_{i,j,c,t'-1} \tag{4.15}$$

$$z_{i,j,c,t'} \leq \mu_{i,j,c,t'} \tag{4.16}$$

$$z_{i,j,c,t'} \geq \mu_{i,j,c,t'-1} + \mu_{i,j,c,t'} - 1 \tag{4.17}$$

To turn (4.6), which involves the product of a real and a binary variables, into a linear form, we define A to be $\max_{\mathcal{M}_{i,j} \in \mathbb{M}} \alpha_{i,j}^{\max}$ and require that (4.19)–(4.21) hold. Since it can be shown that $\phi_{i,j,c,t} = \mu_{i,j,c,t} \cdot \alpha_{i,j,t}$ if and only if (4.19)–(4.21) hold, the MILP formulation replaces (4.6) with (4.19)–(4.21). Similarly for (4.8), we define B to be $\max_{\mathcal{M}_{i,j} \in \mathbb{M}} \beta_{i,j}^{\max}$ and introduce the real decision variable $\omega_{i,j,c_s,t}$ while requiring that (4.19)–(4.21) hold when $\omega_{i,j,c_s,t}$, B, $\mu_{i,j,c_s,t}$, and $\beta_{i,j,t}$ replace $\phi_{i,j,c,t}$, A, $\mu_{i,j,c,t}$, and $\alpha_{i,j,t}$, respectively. The MILP formulation then replaces (4.8) with $\psi_{i,j,c,t} = \sum_{\mathcal{C}_{c_s} \in \mathbb{G}} \eta_{c,c_s,i,t} \cdot \omega_{i,j,c_s,t}$ and the additional constraints.

$$h_{i,j,t} = \begin{cases} 0 & \text{, if } t = 0 \\ 1 - \sum\limits_{\mathcal{C}_c \in \mathbb{G}} z_{i,j,c,t} & \text{, otherwise} \end{cases} \tag{4.18}$$

$$0 \leq \phi_{i,j,c,t} \leq A \cdot \mu_{i,j,c,t} \tag{4.19}$$

$$\phi_{i,j,c,t} \leq \alpha_{i,j,t} \tag{4.20}$$

$$\phi_{i,j,c,t} \geq \alpha_{i,j,t} - A\,(1 - \mu_{i,j,c,t}) \tag{4.21}$$

To turn $\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$ into a linear form, we assume that the function $\mathcal{Q}_{i,j}^{+}$ is additively separable so that $\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t}) = \sum_{k=1}^{3} U_{i,j}^{(k)} f_{i,j}^{(k)}\left(v_{i,j,t}^{(k)}\right)$ with $v_{i,j,t}^{(1)} = \alpha_{i,j,t}$, $v_{i,j,t}^{(2)} = \beta_{i,j,t}$, and $v_{i,j,t}^{(3)} = \lambda_{i,j,t}$. This assumption can be

broadened to include non-separable functions [139]. Each $f_{i,j}^{(k)}$ is then approximated by a piecewise linear continuous function $\tilde{f}_{i,j}^{(k)}$ defined in (4.22), which segments the domain of $f_{i,j}^{(k)}$ into $n_{i,j}^{(k)}$ possibly-unequal intervals and approximates $f_{i,j}^{(k)}$ in every interval by a linear function. By introducing $n_{i,j}^{(k)} \in \mathbb{N}^+$ pairs of real $\delta_{i,j,t}^{(k),l}$ and binary $b_{i,j,t}^{(k),l}$ decision variables, if (4.23)–(4.26) hold with $b_{i,j,t}^{(k),n_{i,j}^{(k)}+1} = 0$, then $\sum_{k=1}^{3} U_{i,j}^{(k)} \tilde{f}_{i,j}^{(k)}\left(v_{i,j,t}^{(k)}\right)$ has (4.27) as its linear form. The term $\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$ in (4.12) is then replaced with (4.27) while adding (4.23)–(4.26) as constraints.

$$\tilde{f}_{i,j}^{(k)}\left(v_{i,j,t}^{(k)}\right) = \begin{cases} m_{i,j}^{(k),1}\ \left(v_{i,j,t}^{(k)} - L_{i,j}^{(k),0}\right) + C_{i,j}^{(k)} \\ \qquad\qquad , \text{ if } v_{i,j,t}^{(k)} \in \left[L_{i,j}^{(k),0}, L_{i,j}^{(k),1}\right] \\ \qquad\qquad \vdots \\ m_{i,j}^{(k),n_{i,j}^{(k)}}\ \left(v_{i,j,t}^{(k)} - L_{i,j}^{(k),n_{i,j}^{(k)}-1}\right) \\ \qquad\qquad + \tilde{f}_{i,j}^{(k)}\left(L_{i,j}^{(k),n_{i,j}^{(k)}-1}\right) \\ \quad , \text{ if } v_{i,j,t}^{(k)} \in \left(L_{i,j}^{(k),n_{i,j}^{(k)}-1}, L_{i,j}^{(k),n_{i,j}^{(k)}}\right] \end{cases} \tag{4.22}$$

$$v_{i,j,t}^{(k)} = L_{i,j}^{(k),0} + \sum_{l=1}^{n_{i,j}^{(k)}} \delta_{i,j,t}^{(k),l} \tag{4.23}$$

$$0 \leq \delta_{i,j,t}^{(k),l} \leq b_{i,j,t}^{(k),l}\left(L_{i,j}^{(k),l} - L_{i,j}^{(k),l-1}\right) \tag{4.24}$$

$$\delta_{i,j,t}^{(k),l} \geq b_{i,j,t}^{(k),l+1}\left(L_{i,j}^{(k),l} - L_{i,j}^{(k),l-1}\right) \tag{4.25}$$

$$b_{i,j,t}^{(k),l} \geq b_{i,j,t}^{(k),l+1} \tag{4.26}$$

$$\sum_{k=1}^{3} U_{i,j}^{(k)} \left(\sum_{l=1}^{n_{i,j}^{(k)}} m_{i,j}^{(k),l} \delta_{i,j,t}^{(k),l} + C_{i,j}^{(k)}\right) \tag{4.27}$$

Finally, our formulation is summed up in terms of its parameters and variables in Table 4.1.

Table 4.1: The MILP parameters and decision variables.

Parameters

$\mathbb{A}$, $\mathbb{M}$, $\mathbb{G}$	The sets of agents $\mathcal{A}_i$, VMs $\mathcal{M}_{i,j}$, and cells $\mathcal{C}_c$.
$\mathcal{R}_{c_s,c_e,t}$	A bidirectional route $\mathcal{C}_{c_s} \leftrightarrow \mathcal{C}_{c_e}$ (i.e., $\mathcal{R}_{c_s,c_e,t} = \mathcal{R}_{c_e,c_s,t}$).
$\Lambda_{c_s,c_e,t}$	$\mathcal{R}_{c_s,c_e,t}$ end-to-end latency ($\Lambda_{c_s,c_e,t} = \Lambda_{c_e,c_s,t} \in \mathbb{R}^{\geq 0}$).
Φ_c, Ψ_c	$\mathcal{C}_c$ computation & communication capacities in $\mathbb{R}^{\geq 0}$.
$\mathcal{E}_{i,j}$	$\mathcal{M}_{i,j}$ migration cost in $\mathbb{R}^{\geq 0}$.
M	The cap in $\mathbb{N}^+$ on concurrent migration count at any t.
$\alpha_{i,j}^{\min}$, $\alpha_{i,j}^{\max}$	$\mathcal{M}_{i,j}$ computation bandwidth lower & upper bounds in $\mathbb{R}^{\geq 0}$.
$\beta_{i,j}^{\min}$, $\beta_{i,j}^{\max}$	$\mathcal{M}_{i,j}$ communication bandwidth lower & upper bounds in $\mathbb{R}^{\geq 0}$.
$\lambda_{i,j}^{\max}$	The upper bound on $\mathcal{M}_{i,j} \leftrightarrow \mathcal{A}_i$ end-to-end latency in $\mathbb{R}^{\geq 0}$.
pos_i	The function that maps any time t to the cell $\mathsf{pos}_i(t) \in \mathbb{G}$ where the agent $\mathcal{A}_i$ is in at time t.

With $l \in \mathbb{N}^+$ and $\tilde{f}_{i,j}^{(1)}$, $\tilde{f}_{i,j}^{(2)}$, and $\tilde{f}_{i,j}^{(3)}$ being defined in (4.22) as the piece-wise linear functions that approximate the contributions of $\alpha_{i,j,t}$, $\beta_{i,j,t}$, and $\lambda_{i,j,t}$, respectively, in the quality function $\mathcal{Q}_{i,j}^{+}$:

$U_{i,j}^{(1)}$	$\tilde{f}_{i,j}^{(1)}(\alpha_{i,j,t})$ weight to approximate $\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$.
$U_{i,j}^{(2)}$	$\tilde{f}_{i,j}^{(2)}(\beta_{i,j,t})$ weight to approximate $\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$.
$U_{i,j}^{(3)}$	$\tilde{f}_{i,j}^{(3)}(\lambda_{i,j,t})$ weight to approximate $\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$.
$C_{i,j}^{(1)}$	$\tilde{f}_{i,j}^{(1)}(\alpha_{i,j,t})$ offset to approximate $\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$.
$C_{i,j}^{(2)}$	$\tilde{f}_{i,j}^{(2)}(\beta_{i,j,t})$ offset to approximate $\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$.
$C_{i,j}^{(3)}$	$\tilde{f}_{i,j}^{(3)}(\lambda_{i,j,t})$ offset to approximate $\mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$.
$L_{i,j}^{(1),0}, L_{i,j}^{(1),l}$	The piece-wise interval endpoints of $\tilde{f}_{i,j}^{(1)}$.
$L_{i,j}^{(2),0}, L_{i,j}^{(2),l}$	The piece-wise interval endpoints of $\tilde{f}_{i,j}^{(2)}$.
$L_{i,j}^{(3),0}, L_{i,j}^{(3),l}$	The piece-wise interval endpoints of $\tilde{f}_{i,j}^{(3)}$.
$m_{i,j}^{(1),l}$	The piece-wise gradient of $\tilde{f}_{i,j}^{(1)}$ in $\left(L_{i,j}^{(1),l-1}, L_{i,j}^{(1),l}\right]$.
$m_{i,j}^{(2),l}$	The piece-wise gradient of $\tilde{f}_{i,j}^{(2)}$ in $\left(L_{i,j}^{(2),l-1}, L_{i,j}^{(2),l}\right]$.
$m_{i,j}^{(3),l}$	The piece-wise gradient of $\tilde{f}_{i,j}^{(3)}$ in $\left(L_{i,j}^{(3),l-1}, L_{i,j}^{(3),l}\right]$.

Variables

$\mu_{i,j,c,t}$	One (zero) if $\mathcal{M}_{i,j}$ is (not) in $\mathcal{C}_c$ at time t.
$\alpha_{i,j,t}$	$\mathcal{M}_{i,j}$ computation bandwidth at time t.
$\beta_{i,j,t}$	$\mathcal{M}_{i,j}$ communication bandwidth at time t.

4.4 Evaluation

We evaluate our MILP formulation's effectiveness and scalability in a realistic ultra-dense network where many robots and services are deployed in an indoor environment (i.e., an industrial scenario). For clarity of illustration and without loss of generality, we use different edge networks, different MILP-solving strategies, and different MILP parameters while considering that in an individual scenario, the agents and the cells have the same types, the same type of links statically interconnect the cells, the MEC servers are of the same type with the same software stack, every server and every link have the same capacities, respectively; every VM migration has the same cost, and every VM has the same bounds on the computation and communication bandwidth and latency and the same quality function.

In all scenarios, we assume that every end-to-end communication latency depends on two factors: 1) the intra-cell latency along the wireless link used by an agent to connect to the network, which, for simplicity, is assumed to be the same for every agent in every cell at any time, and 2) the inter-cell latency along the links used to connect a pair of cells, which, for simplicity, is assumed to be the same for every cell pair at any time.

Specifically, Section 4.4.1 uses the small edge network shown in Figure 4.1a with mesh topology, while Section 4.4.2 and Section 4.4.3 use the large edge network shown in Figure 4.1b with both star and mesh topologies. Furthermore, taking 4.1b as an N-by-N grid of cells roamed by $4N$ agents, Section 4.4.3 also has further scenarios to evaluate larger values of N. While the scenarios evaluated in Section 4.4.1 are solved with a time limit of 1 hour, the scenarios evaluated in Section 4.4.2 and Section 4.4.3 are solved until their MILP solutions are within 10% of the optimum. And while different scenarios use different sets of MILP parameters, the different sets are derived from the following common assumptions.

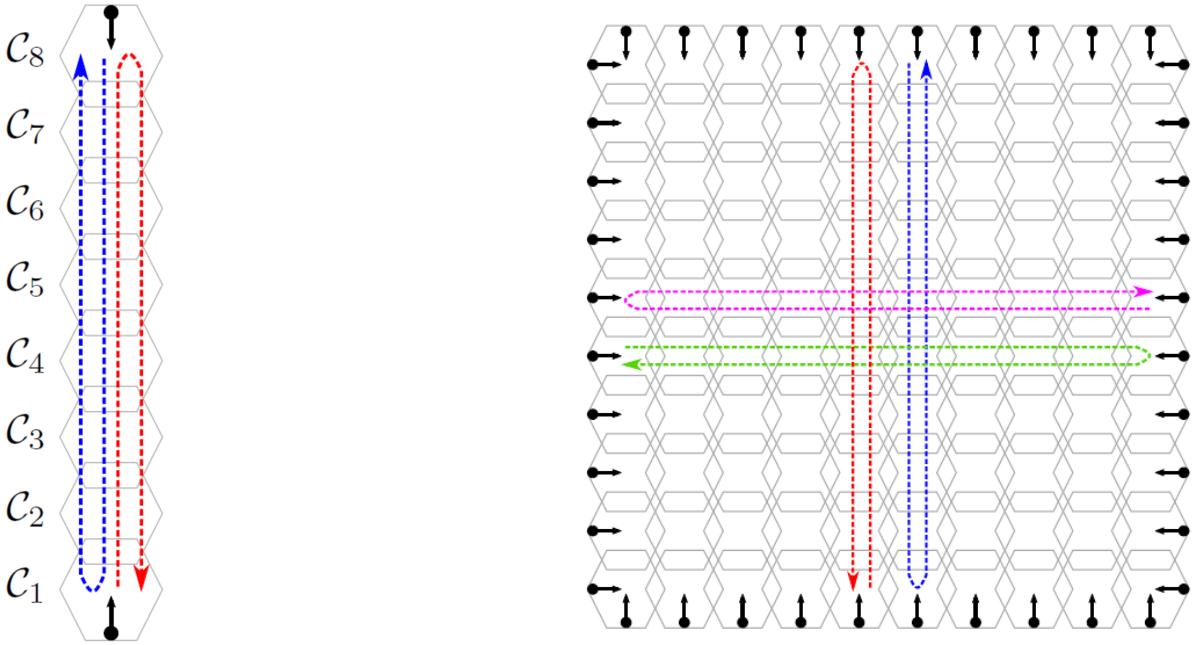
Every scenario assumes a certain network shown in Figure 4.1 to derive their respective $\mathbb{G}$. As shown in Figure 4.1, each row/column has two agents that start at the opposite ends facing each other to move forward at the same speed to the opposite edges only to restart by turning around, and due to having the same speed, every agent enters the next cell at the next time point. Every scenario uses these assumptions to obtain their respective $\mathbb{A}$ and $\rho_{i,c,t}$. To obtain their respective $\mathcal{R}_{c_s,c_e,t}$, every scenario assumes that in the star topology, all cells are connected to one aggregation point so that every end-to-end latency assumes one of two distinct values, while in the mesh topology, every cell pair is connected in the Manhattan scheme[3]. Each cell is connected to adjacent cells in a structured grid-like pattern, similar to the layout of streets in Manhattan. As a result, the end-to-end latency in the mesh topology increases proportionally to the number of cells on the route, mirroring the distance traveled along the grid-like network.

Additionally, every scenario assumes that in the edge network every communication bandwidth Ψ_c is at 1 Gbps (gigabits/s) and the intra-cell latency is at 2 ms while the latency along each inter-cell link is at 3 ms so that $\Lambda_{c_s,c_e,t} = 2 + 3(|\mathcal{R}_{c_s,c_e,t}| - 1)$ (e.g., every end-to-end latency in the star topology is two if not 5 ms). Every scenario then assumes that no limit exists on the number of concurrent migrations, and hence, they derive their respective M to be $|\mathbb{M}|$ (limiting M to different percentages of $|\mathbb{M}|$ is planned in our future work).

Finally, CPLEX CC8ATML 20.1.0 for GNU/Linux (`ibm.com/analytics/cplex-optimizer`) is used as the MILP solver on Ubuntu 16.04.7 on a Lenovo E40-80 laptop with 16 GiB RAM, no swap, and a 4-core Intel Core i3-5010U (2×64-bit 2.1-GHz cores, two threads/core). As its development

[3]If $\mathcal{C}_{(x,y)}$ is $\mathcal{C}_c$ at $(x,y) \in (\mathbb{N}^+)^2$, $\mathcal{R}_{(x_1,y_1),(x_2,y_2),t} = \{\mathcal{C}_{(x,y_1)} \in \mathbb{G} \mid \min\{x_1,x_2\} \leq x \leq \max\{x_1,x_2\}\} \cup \{\mathcal{C}_{(x_2,y)} \in \mathbb{G} \mid y_1 \leq y \leq y_2\}$ for $y_1 \leq y_2$.

environment (`oplide`) uses extra time and memory, the solver is run directly as `cplex -c "read i.lp" "$prm" mipopt "write o.sol"`. As `cplex` accepts a problem in the LP format, we first translate the formulation in Section 4.3 into a GMPL model[4] accepted by another MILP solver, GLPK (`gnu.org/software/glpk`). Then, for every scenario written in GMPL, we run GLPK as `glpsol --check --wlp i.lp -m model.glp -d data.glp` to translate the GMPL model (`model.glp`) and the scenario (`data.glp`) into an LP-format file (`i.lp`) without solving the MILP (`--check`).



(a) A small 8-by-1 two robot's network

(b) A large 10-by-10 40 robot's network

Figure 4.1: Two different networks 4.1a and 4.1b are used in different subsections: 4.1a the 8-by-1 mesh network used in the scenarios evaluated in Section 4.4.1, showing the two agents as black circles, their initial headings at time t_0 with arrows, and their trajectories over the next 14 time points with dashed U-arrows, and 4.1b the 10-by-10 network used in the scenarios evaluated in Section 4.4.2 and 4.4.3 with both the star and mesh topologies, showing the 40 agents, their initial headings at time t_0, and the trajectories For clarity, there are only four agents over the next 14 time points.

[4]We make the model available at `archive.org/details/model-202108`.

4.4.1 System Behavior in the Time Domain

Our formulation effectiveness is first shown by the effect of distinct quality functions on the small mesh network shown in Figure 4.1a. The network is small, which makes the resulting plots easy to analyze. Furthermore, mesh topology is used instead of star topology because mesh topology is more complex than star topology in that the link of a cell may have to bear the traffic between a pair of other cells. This complexity makes it easier to analyze the effect of quality functions that maximize the available link bandwidth.

The effect of distinct quality functions is shown using scenarios that make the solver favor some decision variables only by the quality functions. The quality functions obtained by distinct weights $\left(U_{i,j}^{(1)}, U_{i,j}^{(2)}, U_{i,j}^{(3)}\right)$ are used with the same Eight cells, 15-time points, two agents, 3 VMs/agent, Φ_c=100 GIPS (gigainstructions/s), $\alpha_{i,j}^{\min}$=15 GIPS (any MEC server can host all VMs), $\alpha_{i,j}^{\max}$=90 GIPS, $\beta_{i,j}^{\min}$ = 150 Mbps (any link can route all communication channels), $\beta_{i,j}^{\max}$ = 900 Mbps, $\lambda_{i,j}^{\max}$=23 ms (migration is optional), $C_{i,j}^{(1)}$=$C_{i,j}^{(2)}$=0, $C_{i,j}^{(3)}$=1, $\left(L_{i,j}^{(1),0}, L_{i,j}^{(1),1}\right)$=$\left(\alpha_{i,j}^{\min}, \alpha_{i,j}^{\max}\right)$, $\left(L_{i,j}^{(2),0}, L_{i,j}^{(2),1}\right)$=$\left(\beta_{i,j}^{\min}, \beta_{i,j}^{\max}\right)$, $\left(L_{i,j}^{(3),0}, L_{i,j}^{(3),1}\right) = \left(2, \lambda_{i,j}^{\max}\right)$, $m_{i,j}^{(1),1} = \frac{1}{\alpha_{i,j}^{\max} - \alpha_{i,j}^{\min}}$, $m_{i,j}^{(2),1} = \frac{1}{\beta_{i,j}^{\max} - \beta_{i,j}^{\min}}$, $m_{i,j}^{(3),1}$=$\frac{-1}{\lambda_{i,j}^{\max} - 2}$, and $\mathcal{E}_{i,j}$=80% $\max_{\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t}} \mathcal{Q}_{i,j}^{+}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$. The scenarios are solved in an hour by setting `prm="set timelimit 3600"`.

Figure 4.2 shows the effect of distinct quality functions on the positions of the VMs overtime on the left part, on the MEC server, average processing $\overline{load}_c^1 = \frac{\sum_{\mathcal{M}_{i,j} \in \mathbb{M}, t \in \mathbb{H}} \phi_{i,j,c,t}}{|\mathbb{M}|+|\mathbb{H}|}$ on the middle part with the mean $\mu_1 = \frac{\sum_{\mathcal{C}_c \in \mathbb{G}} \overline{load}_c^1}{|\mathbb{G}|}$ shown at the top of each plot, on the mesh-network link, average traffic $\overline{load}_c^2 = \frac{\sum_{\mathcal{M}_{i,j} \in \mathbb{M}, t \in \mathbb{H}} \psi_{i,j,c,t}}{|\mathbb{M}|+|\mathbb{H}|}$ on the right part with the mean $\mu_2 = \frac{\sum_{\mathcal{C}_c \in \mathbb{G}} \overline{load}_c^2}{|\mathbb{G}|}$ shown at the top of each plot and on the average bandwidth of computation $\frac{\sum_{\mathcal{M}_{i,j} \in \mathbb{M}, t \in \mathbb{H}} \alpha_{i,j,t}}{|\mathbb{M}|+|\mathbb{H}|}$ and communication $\frac{\sum_{\mathcal{M}_{i,j} \in \mathbb{M}, t \in \mathbb{H}} \beta_{i,j,t}}{|\mathbb{M}|+|\mathbb{H}|}$ and la-

tency $\frac{\sum_{\mathcal{M}_{i,j}\in\mathbb{M},t\in\mathbb{H}}\lambda_{i,j,t}}{|\mathbb{M}|+|\mathbb{H}|}$ in the table at every row heading, which shows the distinct weights. The weight of 10 used in a row makes the solver favor $\alpha_{i,j,t}/\beta_{i,j,t}/\lambda_{i,j,t}$ if $U_{i,j}^{(1)}/U_{i,j}^{(2)}/U_{i,j}^{(3)}$ is 10. Placing VMs in the MEC server in the cell where their agent is results in the lowest latency and traffic born by intermediary links at the cost of higher migration frequency and lower computation bandwidth as VMs have to follow their agents and some servers have to host multiple VMs, Figure 4.2 shows that our formulation is effective at implementing distinct quality functions, e.g., migration frequency is highest for $U_{i,j}^{(3)}=10$ to minimize $\lambda_{i,j,t}$ but lowest for $U_{i,j}^{(1)}=10$ as $\alpha_{i,j,t}$ is highest when each server hosts just one VM.

4.4.2 System Key Performance Indicators (KPIs)

Our formulation effectiveness is then shown on the VM migration frequency, outage count, and average computation bandwidth and latency (system KPIs) for different VM migration costs $\mathcal{E}_{i,j}$ and MEC server computation capacities Φ_c by scenarios that use the network shown in Figure 4.1b with star/mesh topology and the same 100 cells, 19 time points, 40 agents, 3 VMs/agents, and the same QoS bounds and quality functions based on [12, 140–142]: $(\alpha_{i,j}^{\min},\alpha_{i,j}^{\max})=(11,13)$ GIPS, $(\beta_{i,j}^{\min},\beta_{i,j}^{\max})=(9,11)$ Mbps, $\lambda_{i,j}^{\max}=15$ ms, $U_{i,j}^{(1)}=U_{i,j}^{(2)}=U_{i,j}^{(3)}=1$, $C_{i,j}^{(1)}=C_{i,j}^{(2)}=-1$, $C_{i,j}^{(3)}=1$, $L_{i,j}^{(1),0}=L_{i,j}^{(2),0}=L_{i,j}^{(3),0}=0$, $\left(L_{i,j}^{(1),1},L_{i,j}^{(1),2}\right)=\left(\alpha_{i,j}^{\min},\alpha_{i,j}^{\max}\right)$, $\left(L_{i,j}^{(2),1},L_{i,j}^{(2),2}\right)=\left(\beta_{i,j}^{\min},\beta_{i,j}^{\max}\right)$, $\left(L_{i,j}^{(3),1},L_{i,j}^{(3),2}\right)=\left(5,\lambda_{i,j}^{\max}\right)$, $m_{i,j}^{(1),1}=\frac{1}{\alpha_{i,j}^{\min}}$, $m_{i,j}^{(2),1}=\frac{1}{\beta_{i,j}^{\min}}$, $m_{i,j}^{(3),1}=0$, $m_{i,j}^{(1),2}=\frac{1}{\alpha_{i,j}^{\max}-\alpha_{i,j}^{\min}}$, $m_{i,j}^{(2),2}=\frac{1}{\beta_{i,j}^{\max}-\beta_{i,j}^{\min}}$, $m_{i,j}^{(3),2}=\frac{-1}{\lambda_{i,j}^{\max}-5}$.

Each scenario is solved to get a solution that is within 10% of the optimum by `prm="set mip tolerances mipgap 0.1"`. To highlight our formulation effectiveness, we use baseline scenarios that always migrate the VMs to the MEC server in the cell where their agent is with the same $(\alpha_{i,j}^{\min},\alpha_{i,j}^{\max})=(0,11)$ GIPS and $(\beta_{i,j}^{\min},\beta_{i,j}^{\max})=(0,9)$ Mbps so that their solu-

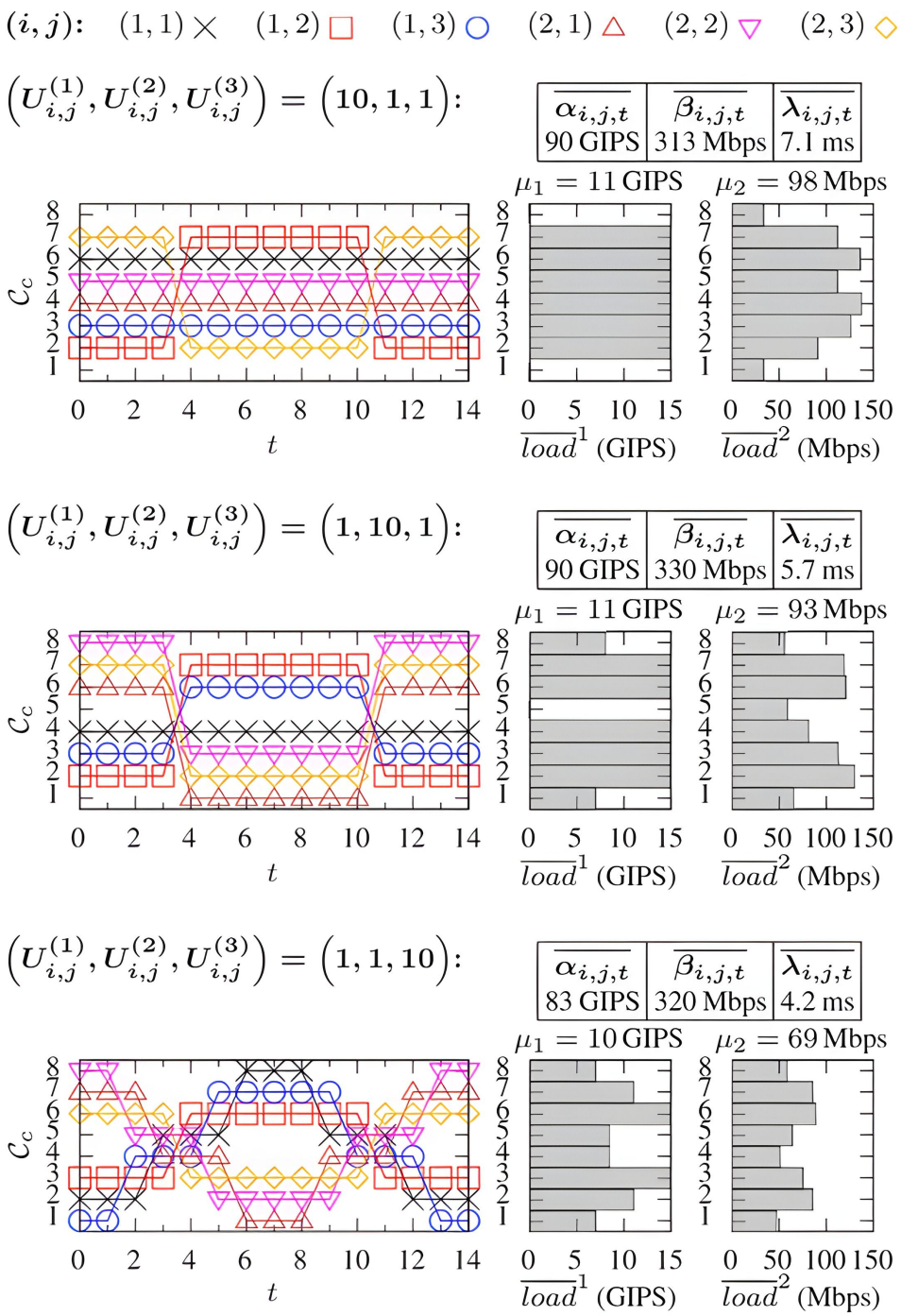


Figure 4.2: The effectiveness of different quality functions as evaluated in Section 4.4.1 on the network shown in Figure 4.1a.

tions give every VM the lowest latency but the highest migration frequency and possibly some outage times, each occurring at time t and at cell $\mathcal{C}_c$ if

$\alpha_{i,j,t} < 11\,\text{GIPS}$ for the least possible number of VMs on the server in $\mathcal{C}_c$ when $|\{\mathcal{M}_{i,j} \in \mathbb{M} \,|\, \mu_{i,j,c,t} = 1\}| \times 11\,\text{GIPS} > \Phi_c$. Figure 4.3 shows the system KPIs attainable by the optimal solutions for the baseline scenarios and by the ≥90%-optimal solutions for the other scenarios. Every ≥90%-optimal solution has no service outage by (4.4), (4.5), and (4.11) and sets $\beta_{i,j,t} = 11\,\text{Mbps}$ as $\Psi_c = 1\text{Gbps} > \frac{\Phi_c}{11\,\text{GIPS}} \times 11\text{Mbps}$, while the baseline solutions are plotted as one line in each KPI as they are equal despite the various topologies and migration costs.

Figure 4.3 shows that our formulation is effective for the system KPIs as every ≥90%-optimal solution migrates much less often, especially in the star topology, which has no outage and gives a much higher computation bandwidth, even when the server is very constrained at 25 GIPS regardless of the topology, all of these with latency that is very acceptable in the robotic domain.

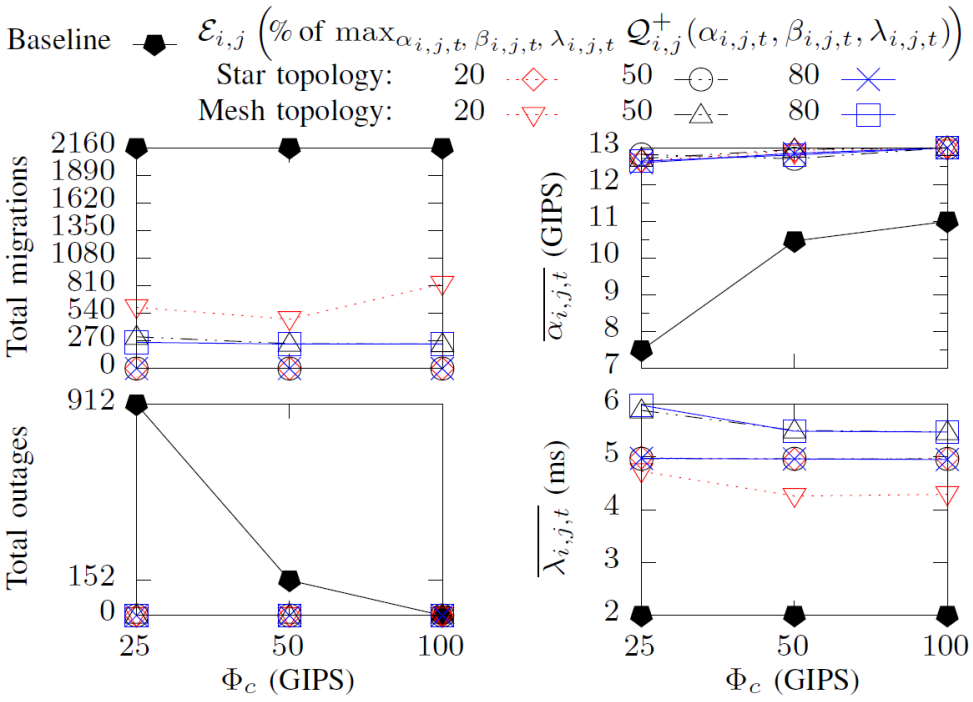


Figure 4.3: The effectiveness of our MILP formulation as evaluated in Section 4.4.2 on the network shown in Figure 4.1b.

4.4.3 System Complexity

Our formulation scalability in handling complex systems is shown in Figure 4.4 by the time taken to get the ≥90%-optimal solutions plotted in Figure 4.3 and in Figure 4.5 by the time taken to get the ≥90%-optimal solutions for the mesh-network scenarios described in the previous section with $\Phi_c = 50\,\text{GIPS}$ and $\mathcal{E}_{i,j} = 80\%\max_{\alpha_{i,j,t},\beta_{i,j,t},\lambda_{i,j,t}} \mathcal{Q}^+_{i,j}(\alpha_{i,j,t},\beta_{i,j,t},\lambda_{i,j,t})$ when the scenarios use different time horizon lengths $|\mathbb{H}|$, grid sizes N, which mean different counts of cells $|\mathbb{G}|$ and agents $|\mathbb{A}|$, and VMs/agent counts, which mean different $|\mathbb{M}|$. Figure 4.4 shows three important points about our formulation scalability: 1) the mesh topology takes less time than the star topology to solve, 2) the more constrained the MEC server is, the (possibly exponentially) longer the solution is obtained, and 3) compared to the previous point, migration cost has no significant effect on the solution time. Furthermore, Figure 4.5 shows two important points about

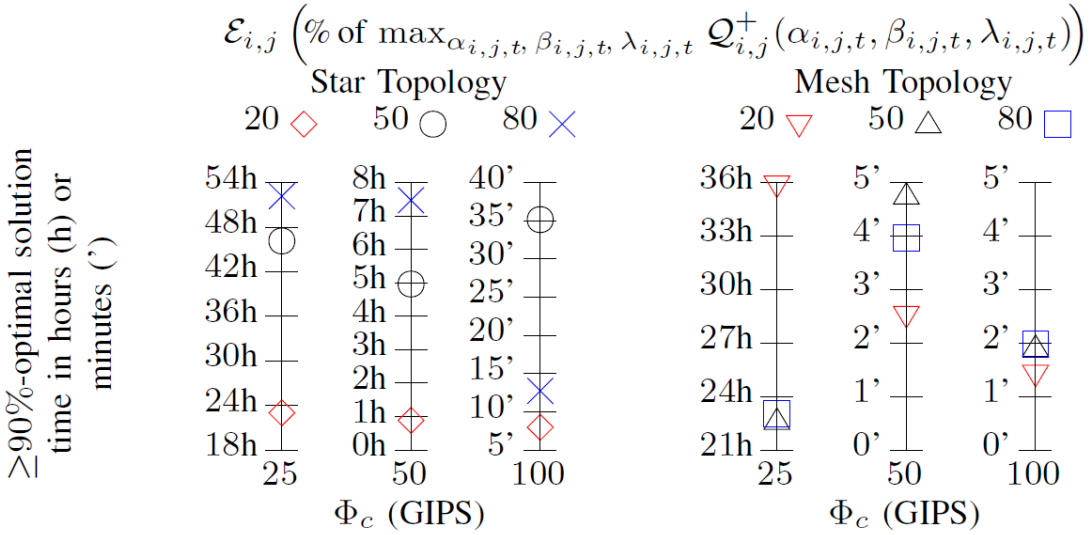


Figure 4.4: The effectiveness of our MILP formulation as evaluated in Section 4.4.3 on the network shown in Figure 4.1b.

our formulation scalability: 1) the fiercer the MEC server is contested, the (possibly exponentially) longer the solution is obtained (e.g., for 6 VM-

s/agent, the server-to-VM ratio is 100:240, but for $N = 26$, the ratio is 676:312, and hence, the servers are contested fiercer in the former than in the latter), and 2) memory becomes the main limitation as the number of cells, and hence the number of end-to-end routes increases.

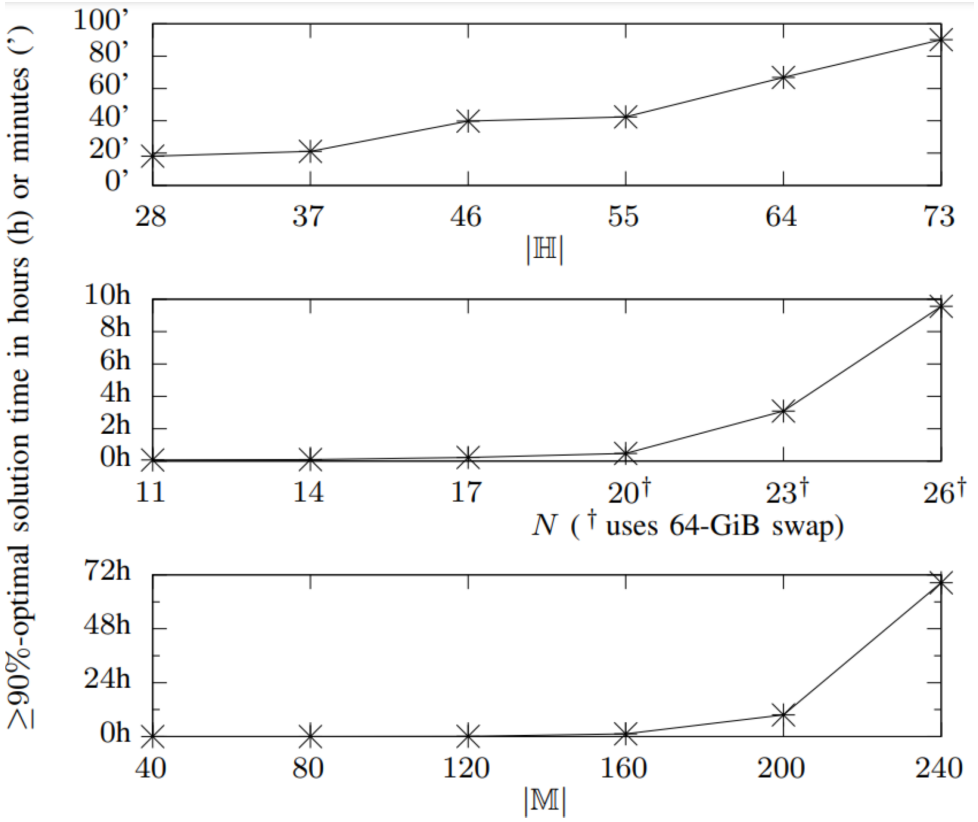


Figure 4.5: The scalability of our MILP formulation as evaluated in Section 4.4.3 on the mesh network shown in Figure 4.1b when $\Phi_c = 50\,\text{GIPS}$ and $\mathcal{E}_{i,j} = 80\% \max_{\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t}} \mathcal{Q}^+_{i,j}(\alpha_{i,j,t}, \beta_{i,j,t}, \lambda_{i,j,t})$.

4.4.4 Discussion

In this work, we have considered the problem of allocating resources at the edge of a B5G network to real-time services optimally by formulating a MILP whose decision variables are the amount of computation and communication resources and the MEC servers to execute the VMs providing the services at each time point. Our optimization framework applies equally well to star and mesh topologies. Using state-of-the-art optimization tools allows us to treat problems of reasonable size in the number of cells and agents when the agent trajectories are known up-front, and the optimization can be performed offline before starting the system operations. When the size of the problem grows or when the system is highly dynamic and requires online optimization, heuristic approaches are needed to produce high-quality, sub-optimal solutions.

Chapter 5

Architectural Exploration and Design for Ultra-Reliable Low-Latency Indoor Robotics Systems

In this work, the plan is to modify the core structure of ARCHEX by re-creating the MILP optimization model and creating the dynamicity by introducing a time dimension. The idea is to create robot trajectories and path constraints and let the optimization tool run so the connections between entities can change dynamically at each time step and the mapping of services to architecture components can be migrated to achieve the desired performance. Previously, ARCHEX was developed only for the architectural exploration of static scenarios, but our goal is to overcome this limitation and integrate a network simulator in the loop, i.e., OMNeT++. In this technique, the optimization of network resources is carried out in a loop by solving multiple constraints in ARCHEX to get an optimal solution. The new solution obtained will be used in the simulation to validate the dynamic parameters of the network. After the completion of both optimization and simulation, the network design and planning are considered suitable for deployment if the solution satisfies the desired QoS requirements. However, if the achieved results do not meet the desired network

requirements; then, extra constraints must be added to the ARCHEX MILP model, and the steps must be repeated in a loop until they produce an optimal solution for the network design.

5.1 Introduction

The evolution of mobile communication in modern times goes beyond the limits of traditional communication systems by enabling ultra-reliable low-latency communication (URLLC), whose minimum latency is 1 ms [143, 144] and by making the network intelligent [145]. Network providers aim to minimize costs while ensuring specific levels of service quality, like adequate computation and communication bandwidth and minimal latency for different services, as this is crucial for meeting customer needs effectively [40]. Various commercial optimization tools are available that can handle the complexity of network design; however, in this work, we consider an open-source design-space exploration (DSE) tool such as ARCHEX [13, 35], which is designed as a MATLAB toolbox, following a specific process outlined in Figure 5.1 of how it works:

- Input Files: ARCHEX starts by taking input files containing a problem description (expressing system properties and objectives) and a library file (listing available components and their attributes).
- Translation to YALMIP: The implementation translates these inputs into YALMIP [146], a MATLAB toolbox for formulating mixed-integer linear programming (MILP) problems.
- Solving the MILP Problem: YALMIP then solves the MILP problem using various supported solvers like GLPK[1] and CPLEX[2].

[1] `https://www.gnu.org/software/glpk/`
[2] `https://www.ibm.com/analytics/cplex-optimizer`

- Output: Finally, the implementation produces an optimized network architecture using MATLAB.

This process ensures network providers can efficiently design and optimize their networks to meet diverse service requirements.

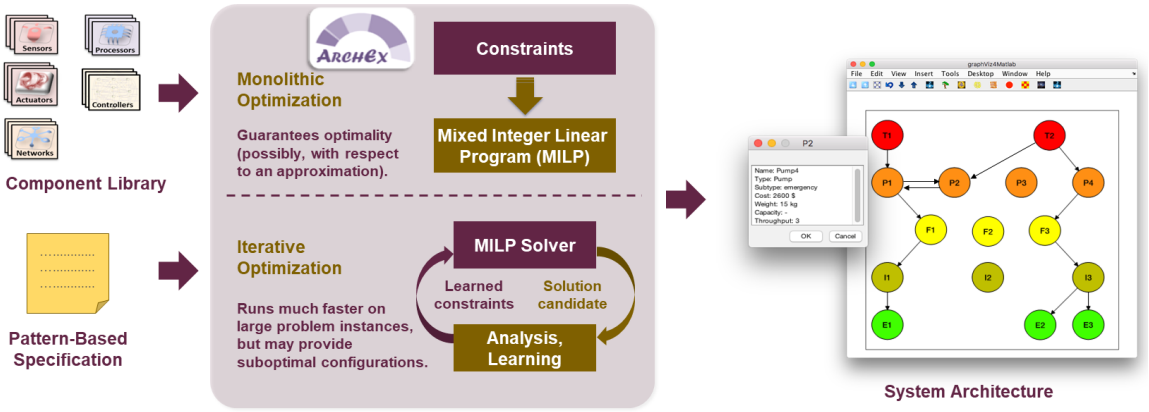


Figure 5.1: The processing flow of ARCHEX [14]

When designing a high-reliability and low-latency indoor network for robots in a large factory, the designer begins by selecting various network components like servers, APs, and services. These components are then placed on a factory model, such as a floor plan. Next, the designer uses a DSE tool to find the most cost-effective off-the-shelf products for each chosen component while still meeting the QoS requirements of the services. This ensures the network is efficient and reliable for the factory's operations.

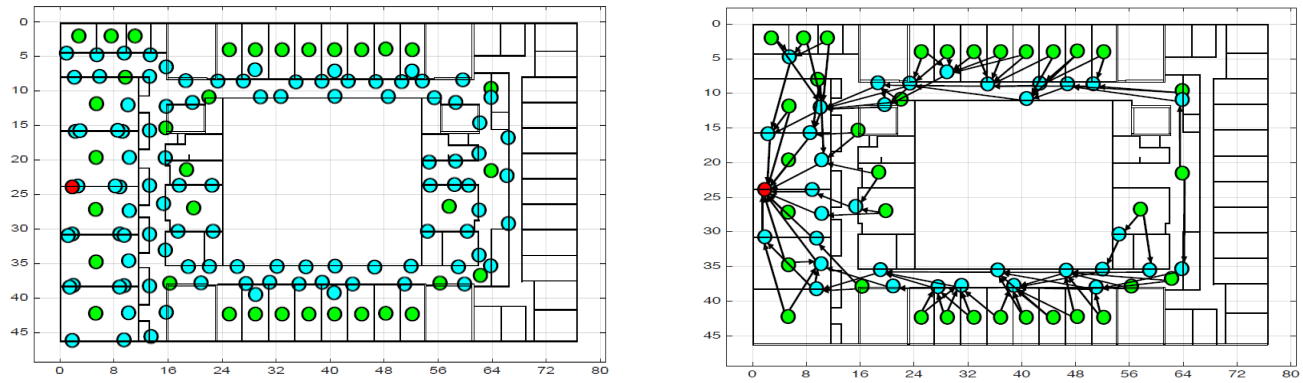
This, however, is not the only possibility because, as demonstrated using ARCHEX, the designer can indeed obtain other design answers (e.g., which path between a robot $\Rightarrow$ service are redundant) and ask other design questions [126]. For instance, instead of manually placing the chosen components on the factory model, the designer can specify the desired received signal strength (RSS) and let the tool automatically put the components on the model. To demonstrate the design effectiveness of ARCHEX,

mainly to highlight the potential benefits of using ARCHEX in designing URLLC-based networks, we adapt the methodology of a wireless sensor network (WSN) system as an example, which is concerned with the design of an indoor static network topology that considers a fixed physical placement of nodes and their fixed physical links in an indoor setting and the selection of network components to be deployed. The architectural design is demonstrated in two examples: data collection and localization [121].

In data collection, multiple sensors are used to collect the data, which is then forwarded through a routing device called a relay to a base station or sink node, where the collected data is monitored and processed. The control algorithms may be fed with the collected data from sensors and can be used to manage a set of actuators to operate windows, blinds, heaters, and room lights. In the second example, i.e., the localization network, the nodes can be used to determine their position or location from a base station depending on the RSS and possibly to track a mobile device or user.

The authors express the system requirements, including link quality, energy consumption, routing, and localization, as a MILP constraint and allow the ARCHEX to solve the MILP problem to find an optimal design. The authors implemented the data collection example as an indoor WSN for periodic data collection, which consists of end devices (sensors) that measure or detect physical environmental phenomena. These one or more base stations collect and process the sensor data and relay that forwards the messages towards the base station. The example considers a total number of 136 nodes in the Template ($\mathcal{T}$), with 35 sensors (in green) and one base station (in red), whereas the remaining nodes represent candidate locations for relays whose positions are permanently fixed. Figure 5.2a and 5.2b show the architectural design of the data collection example.

In the localization example, the environment considers 150 candidate



(a) The floor plan input to ARCHEX

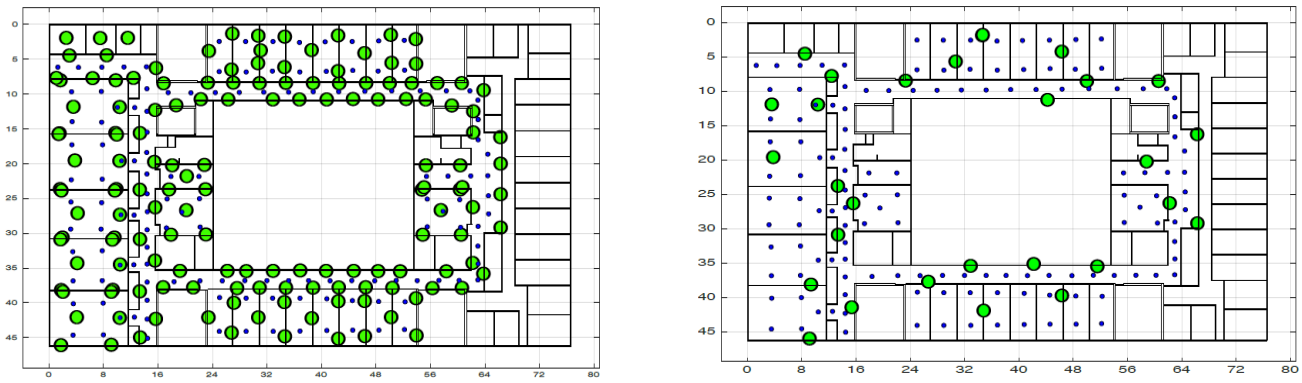
(b) The floor plan output from ARCHEX

Figure 5.2: Wireless Sensor Network data collection example in ARCHEX [121]

node positions and 135 evaluation (mobile node) locations for the same building floor. The example assumes a star topology where anchor nodes that have to be allocated by the tool communicate directly with the mobile device; the latter estimates its position using a set of distance measurements obtained from the anchors. In the example, localization constraints for minimum reachable devices are implemented, requiring the mobile node to receive signals from at least three distinct anchors at every test point. Furthermore, the same constraint enforces that only reliable links with a minimum RSS of -80 dBm are selected. This also contributes to decreasing the ranging error, which rapidly grows for more considerable path losses and unstable signals. The localization example implemented in ARCHEX is shown in Figure 5.3a, and 5.3b.

Although most of the literature is concerned with optimizing a given architecture, we are interested in the earlier phase of architectural design. Nevertheless, several aspects of a real deployment are either too difficult to model for the optimizer or lead to excessive run times due to the combinatorial nature of the optimization problem.

Simulation, instead, can easily handle details such as protocol messages and run-time collisions but does not scale to a large design space. There-



(a) Template of candidate nodes for localization

(b) Generated nodes placement locations for localization

Figure 5.3: Wireless Sensor Network localization example in ARCHEX [121]

fore, the idea we pursue in this work is to combine these two techniques. For instance, Finn et al. link the optimization algorithm that generates the candidate architecture of an aircraft environmental control system over a discrete space with the MODELICA simulator to perform the analysis over a continuous space [95]. Moin et al. synthesize the topologies for body area networks (BANs) under energy and reliability constraints through an iterative in-the-loop optimization technique based on a discrete-event network simulator [100].

In this work, we follow a similar approach and complement a symbolic MILP optimization strategy with a simulation environment, where several key aspects can easily be estimated, accounting for lower-level details that would be otherwise too expensive to handle. The objective is to generate the architecture starting from a relatively simple description of the environment and the application requirements, expressed as performance and quality of service constraints. The motivation is to target the tool's static nature and overcome this limitation by designing a dynamic scenario.

Additionally, the contributions of this work are to create:

- A MILP-based novel case study of a dynamic nature for an indoor

robotic scenario with a time dimension `t` in the problem formulation where objects and connectivity can change during execution.

- A new set of specification patterns for QoS requirements with variables supporting computation, communication, latency requirements, and server_capacity, denoted as α, β, λ, and Φ.

- Integrate ARCHEX with the OMNeT++ discrete-event simulator [147], automatically generating the simulation configuration files. The technique significantly improves design automation, where the optimizer generates a candidate solution through different patterns and simulation results are automatically retrieved.

5.2 System Model & Mixed-integer linear program formulation

We formalize the exploration as a MILP optimization problem, adapting to the baseline methodology of ARCHEX. We extend the tool structure by re-creating the basis constraints and implementing a new group of application constraints. Here, we briefly recall the formulation and refer the reader to our previous work for details [126]. The architecture model, or template, is a directed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, which represents a network of interconnected components. $\mathbb{V}$ is a set of components (nodes) $V_0, \ldots, V_{|V|}$, `|V|` being the cardinality of $\mathbb{V}$ and $\mathbb{E}$ is a set of edges $e_{ij} \in \mathbb{E}$ present in the network. Both nodes and edges in the graph are labeled with types, terminal variables, and attributes corresponding to those from the library $\mathbb{L}$.

The edges $e_{ij} \in \mathbb{E}$ of $\mathbb{G}$ are binary variables representing a connection between nodes $_i$ and v_j, where $e_{ij} = 1$ or 0 indicates the presence or absence of a connection between the source and destination. Similarly, a node v_i

is represented by a binary variable δ_i, which evaluates to 1 if at least one incoming edge e_{ji} or outgoing edge $e_{ij} = 1$. The components and edges in the template are "virtual": the optimizer will retain only those necessary to satisfy the application constraints. A library $\mathbb{L}$ contains a set of "real" components assigned to the template nodes during the mapping process. Nodes and edges in the graph are labeled with different attributes, such as type, cost, delay, etc., corresponding to those from the components library $\mathbb{L}$.

We denote with $\mathbb{M} : \mathbb{V} \to \mathbb{L}$ the map that associates each virtual component of the template with a real component in the library. We represent this map with a binary variable $m_{ij} \in \mathbb{M}$, which is $= 1$ if $v_i \in \mathbb{V}$ is mapped to $l_j \in \mathbb{L}$, else $= 0$. So, given a template $\mathcal{T} = (\mathbb{V}, \mathbb{E})$ and a library of components $\mathbb{L}$, we use the optimization to find a topology $\mathbb{E}^*$ and a map $\mathbb{M}^*$ to satisfy a set of requirements (e.g., interconnection, routing, link quality, and QoS) while minimizing a cost function. Next, we formulate the system requirements and application constraints for, but not limited to, QoS, routing, and mapping constraints in terms of mixed-integer linear constraints.

5.2.1 Basis Constraints

A set of generic basis constraints and an objective function, such as cost minimization, define the topology selection and mapping problem. Topology selection consists of choosing the components from the template required to minimize the objective function, and mapping is the process of assigning a library component to the selected elements in the template. These constraints formalize the exploration problem and ensure the process follows a set of rules. For instance, they impose that if a virtual component is used, then it is actually mapped to a library component. At the same time, it must be mapped to only one library component to avoid

ambiguities. Figure 5.4 shows a basis of the exploration problem.

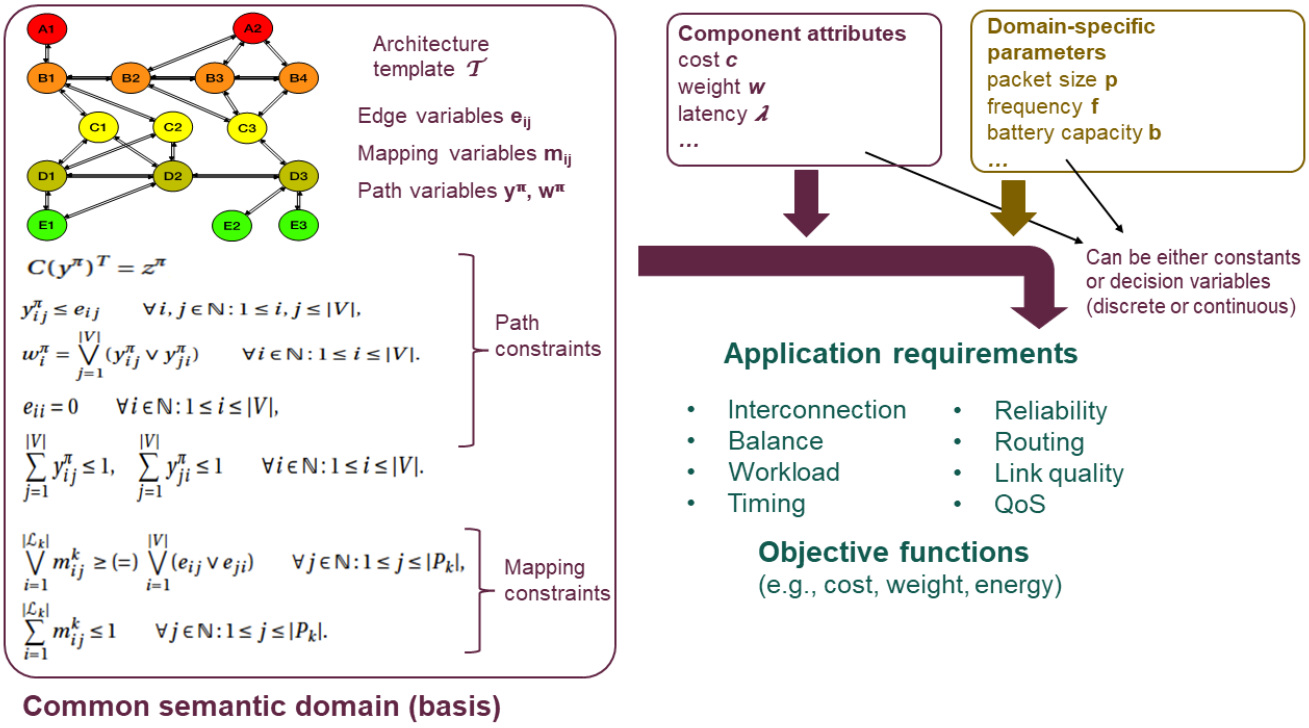


Figure 5.4: A set of generic basis constraints and exploration problems in ARCHEX

5.2.2 Path Constraints

In the components architecture, sources must be linked to destinations through specific routes, known as paths. These paths must meet specific criteria called routing constraints. These constraints might require certain paths to be present or absent, specify the types of nodes they must include, set limits on their number and length, or define how they relate to each other (such as totally disjoint, partly disjoint, or equal).

A network path, denoted as π, is a sequence of distinct nodes $v_0, \ldots, v_n$. We use paths to define the interactions among the components. In our scenarios, a path connects an agent (a robot) to each of its required services, running on a server through the wireless infrastructure. Formally, a path is associated with binary variables y^π_{ij} having the value 1 if the edge e_{ij}

connects the node pair (v_i, v_j) within the path π, and 0 otherwise. A set of constraints is generated to let the optimizer decide how to create a path consistent with the given template topology.

Similarly, each node $v \in \mathbb{V}$ is associated with a binary variable w^π, where $w_i^\pi = 1$ if the node v belongs to the path π, or 0 otherwise. The balance equation (5.1) is an integer linear constraint that ensures the consistency of assigned values to the path and node variables (y^π, w^π) by the given topology:

$$\mathbb{C}(y_t^\pi)^T = \mathtt{z}_t^\pi \qquad \mathtt{t} \in \mathbb{R} \tag{5.1}$$

Here, the matrix $\mathbb{C}$ is the incidence matrix, with dimensions $|V| \times |E|$. The matrix stores the values 1 or -1, indicating whether an edge leaves from or enters a node, respectively, and 0 otherwise. z^π is a column vector of size $|V|$, where $z_{src}^\pi = 1$, $z_{dst}^\pi = -1$, and 0 otherwise. A constraint is created to specify the relation between e_{ij} and y_{ij}^π.

$$y_{ijt}^\pi \leq e_{ij} \quad \forall i, j \in \mathbb{N} : 1 \leq i, j \leq |V| \quad \mathtt{t} \in \mathbb{R} \tag{5.2}$$

$$\sum_{j=1}^{|V|} y_{ij}^\pi \leq 1, \quad \sum_{j=1}^{|V|} y_{ji}^\pi \leq 1, \quad \forall i \in N : 1 \leq i \leq |V| \tag{5.3}$$

Constraint in (5.2) defines that the edge e_{ij} is present only if it is used by a path at any time $\mathtt{t}$. Constraint in (5.3) prevents loops in a path: each node along the path must have at most one predecessor and at most one successor.

$$y_{ij}^{\pi_1} + y_{ij}^{\pi_2} \leq 1 \tag{5.4}$$

$$\sum_i \sum_j y_{ij}^\pi \leq (\geq, =) N_{\text{hops}} \tag{5.5}$$

The constraint in (5.4) requires that two paths π_1 and π_2 be disjoint; they do not share any nodes or edges. On the other hand, the constraint in (5.5) is used to establish the maximum, minimum, or exact length of path

π in terms of the number of hops it contains. An example of a path (route) between a robot $\Longrightarrow$ service is shown in Figure 5.5.

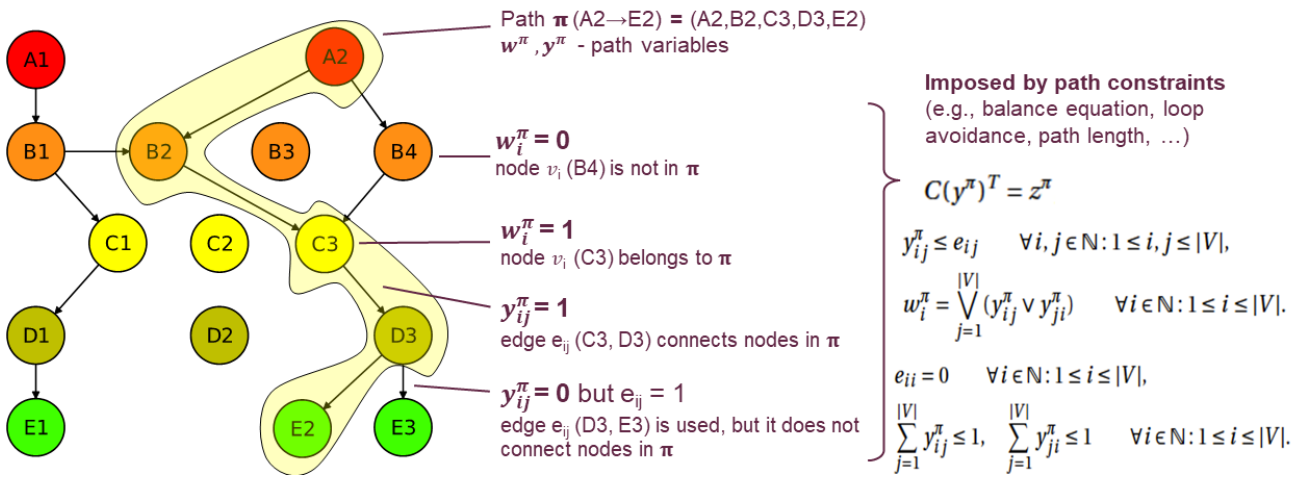


Figure 5.5: Example of a network path π^q , q = (A2, E2), and corresponding path variables w^{π^q} and y^{π^q} for nodes and edges of the graph [14]

5.2.3 Mapping Constraints

A mapping $\mathbb{M}$ is function denoted with $\mathbb{M} : \mathbb{V} \to \mathbb{L}$, the map associates each component of the template (a "virtual component") $v \in \mathbb{V}$ with one or many components in the library (a "real component") $l \in \mathbb{L}$. The mapping $\mathbb{M}$ is defined as a binary variable $m_{ij} \in \mathbb{M}$, which is equal to 1 if a virtual node $v_i \in \mathbb{V}$ is mapped to a real node $l_j \in \mathbb{L}$, else it is equal to 0.

$$\bigvee_{i=1}^{|\mathcal{L}_k|} m_{ij}^k = \bigvee_{i=1}^{|V|} (e_{ij} \vee e_{ji}) \qquad \forall j \in \mathbb{N} : 1 \le j \le |P_k| \qquad (5.6)$$

$$\bigvee_{i=1}^{|\mathcal{L}_k|} m_{ij}^k \le 1 \qquad \forall j \in \mathbb{N} : 1 \le j \le |P_k| \qquad (5.7)$$

Constraint (5.6) imposes that each node v of type `k` that is instantiated must be mapped to one of the components in $\mathbb{L}_k$, which represents the type of components in the library. Constraint (5.7) ensures that virtual

components are never mapped to multiple library components. Similar restrictions are enforced for all the types in $\mathcal{T}$. An example of how mapping constraints are imposed and how a virtual to library component mapping is performed in ARCHEX is shown in Figure 5.6.

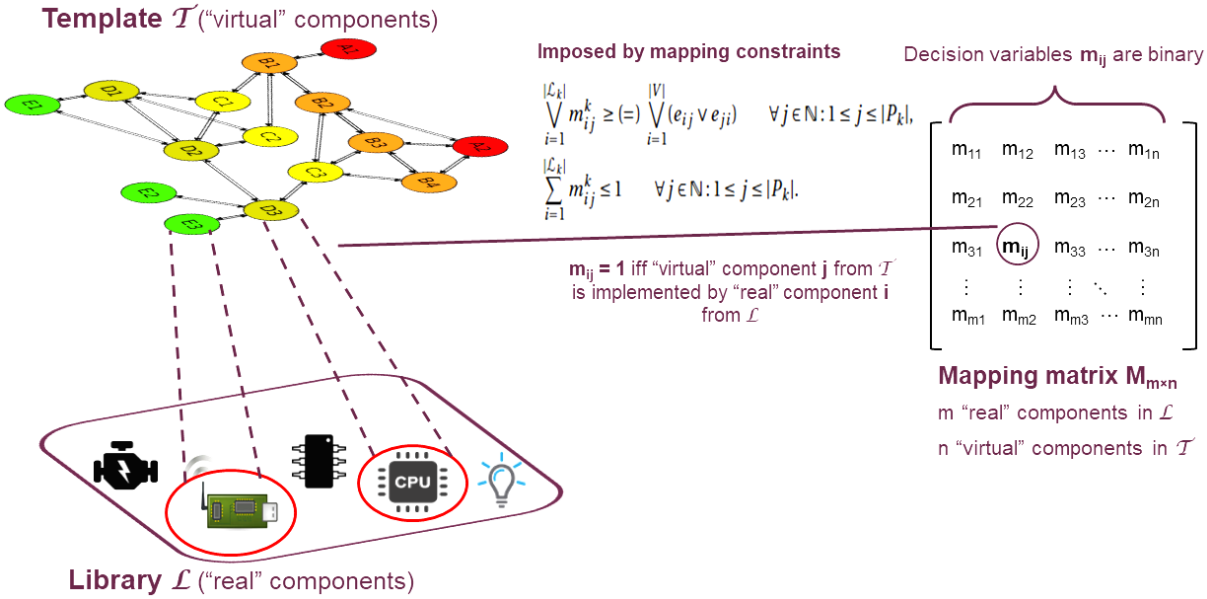


Figure 5.6: A virtual ($\mathcal{T}$) component mapping with real ($\mathbb{L}$) component [14]

5.2.4 QoS Constraints

We implemented multiple QoS constraints for computation, communication, latency (E2E delay), and server capacity, denoted as α, β, λ, and Φ. The optimization involves determining the optimal values for two real variables $\alpha_{i,j,t}$ and $\beta_{i,j,t}$ allocated to server i and service j at any given time `t` as given by the following equations:

$$\alpha_{i,j}^{\min} \leq \alpha_{i,j,t} \leq \alpha_{i,j}^{\max}, \tag{5.8}$$

$$\beta_{i,j}^{\min} \leq \beta_{i,j,t} \leq \beta_{i,j}^{\max}, \tag{5.9}$$

which define the minimum and maximum available resources. The path constraints above determine the association between the server and the

service. In addition, a constraint ensures compliance with the upper-bound threshold latency $\lambda_{i,j}^{\max}$ for each route that a robot is required to traverse:

$$\lambda_{i,j,t}^{\pi} \leq \lambda_{i,j}^{\max} \tag{5.10}$$

Finally, a constraint ensures compliance with the requirement that the computation capacity allocated by a server to the services ($\phi_{i,j}$) at time `t` does not exceed the maximum computation capacity of the server ($\Phi^{\max}$). Exceeding the server capacity would lead to the robot's failure to execute its real-time tasks.

$$\sum \phi_{i,j,t} \leq \Phi^{\max} \tag{5.11}$$

5.2.5 Cost Function

Each node and edge in the network $\mathcal{T}$ is assigned a cost value, representing monetary expenses or other parameters like idle time, energy consumption, or weight. We define cost functions that sum up the costs of all instantiated components (nodes) and connections (edges) in the network:

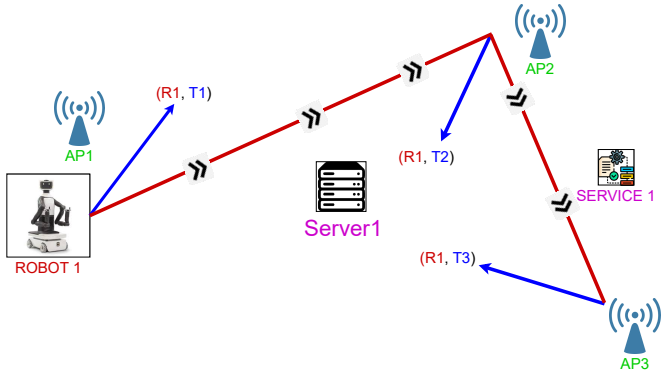
$$\sum_{i=1}^{|V|} \delta_i c_i + \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} e_{ij} \tilde{c}_{ij} \tag{5.12}$$

Here, c_i represents the cost of the node v_i, $\tilde{c}_{ij}$ represents the cost of the edge e_{ij}, and δ_i is a binary variable that equals one if the component is instantiated and zero otherwise.

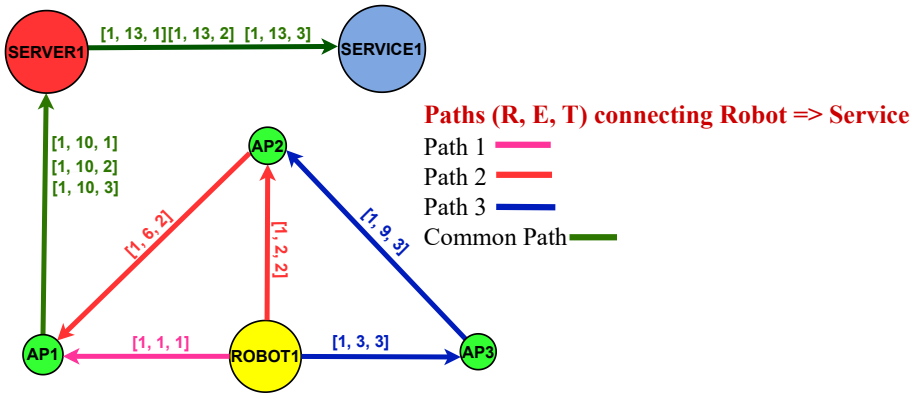
5.3 Evaluation

We evaluate the model for multiple scenarios, and for each scenario, we create a new template, library, and an SVG file. We aim to get an optimal solution and let the optimization tool decide on a feasible and unique path for the robot at every step. Starting from a simpler scenario as shown in

Figure 5.7, Figure 5.7a is the input SVG with the following nodes: robot = 1, AP = 3, server = 1, and service = 1, and the robot trajectories at different coordinates showing their direction and position at every discrete time step (for simplicity, all unwanted obstacles are removed). The APs are the medium of the path between the source and destination nodes, and at each time step, when the robot changes its position, it receives a minimum RSS and finds a path to the destination node, i.e., the service. Figure 5.7b shows the final graph generated by ARCHEX after the optimization runs successfully and finds the optimal solution.



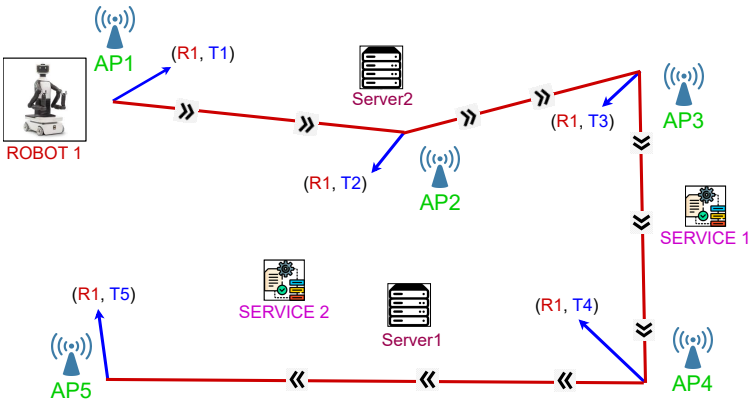
(a) The floor plan as an input SVG to ARCHEX.



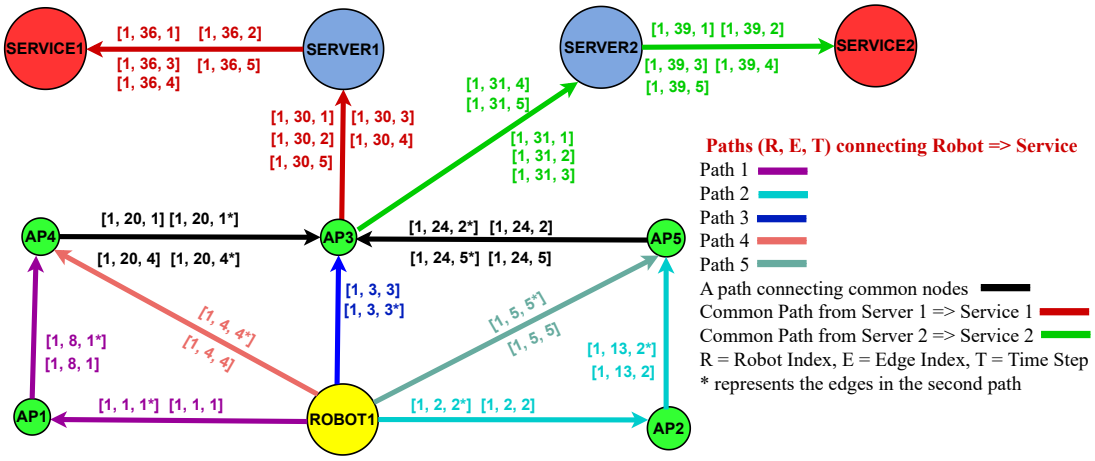
(b) The generated architecture from ARCHEX.

Figure 5.7: Scenario 1: one robot, three APs, and one server

Similarly, the second scenario is shown in Figure 5.8, the input SVG in Figure 5.8a has the following nodes: robot = 1, AP = 5, server = 2, and service = 2, and a robot with more trajectory points. After the optimization runs successfully, we get Figure 5.8b as the output in which the paths, node selection, and placement are generated by ARCHEX. Finally, the



(a) The floor plan as an input SVG to ARCHEX.

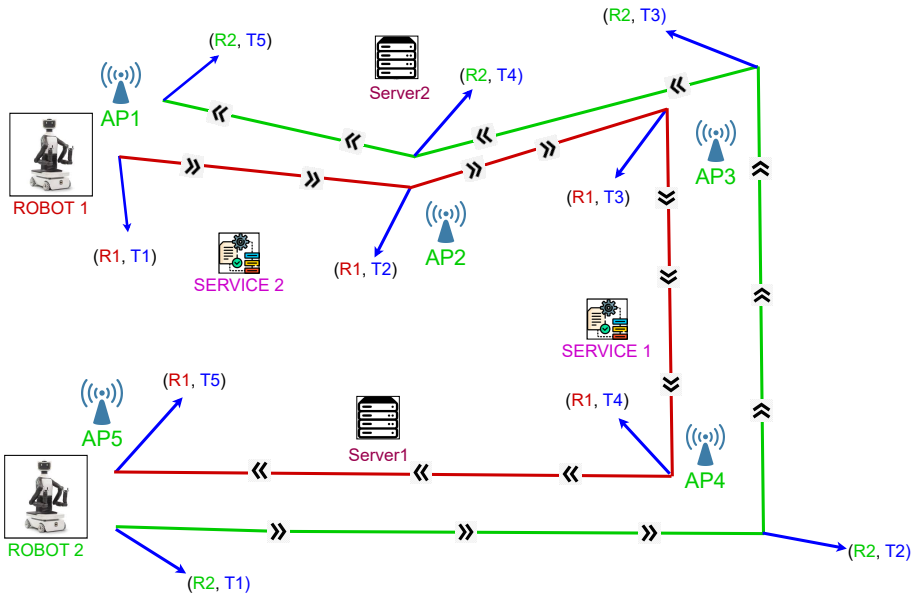


(b) The generated architecture from ARCHEX.

Figure 5.8: Scenario 2: one robot, five APs, and two servers

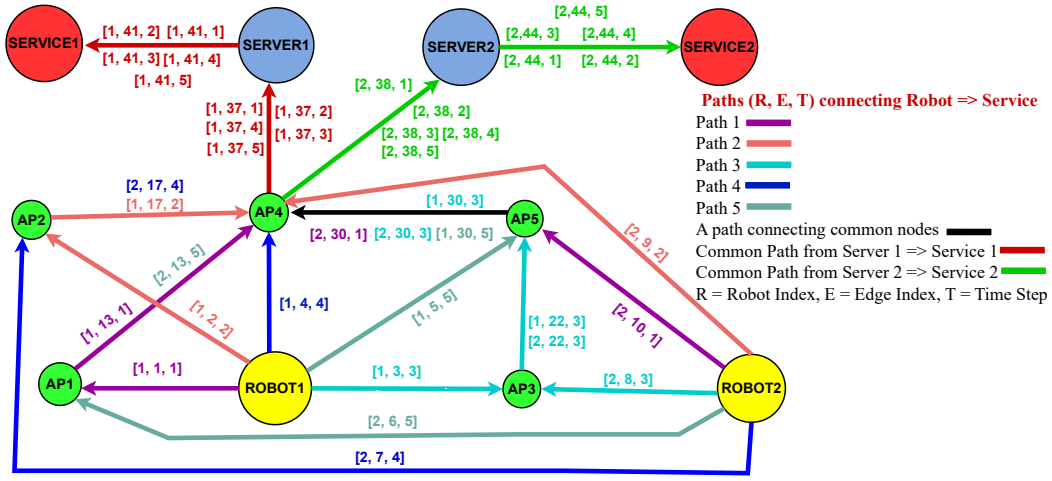
third scenario is shown in Figure 5.9, the input SVG in Figure 5.9a has the following nodes: robot = 2, AP = 5, server = 2, and service = 2, and clearly, each robot has a different trajectory. After the optimization runs

successfully, Figure 5.9b architecture is generated by ARCHEX.



- R1 (--------) represents the robot index and direction in the input SVG
- R2 (--------) represents the robot index and direction in the input SVG
- T1,..., T5 (----------) represents the position of robots at each time steps

(a) The floor plan as input SVG to ARCHEX.



(b) The generated architecture from ARCHEX.

Figure 5.9: Scenario 3: two robots, five APs, and two Server

In all scenarios, we computed a connection_matrix and a threshold_matrix,

which are 3×3 and 5×5 matrices for scenario 1 and 2, as shown in Table 5.1 and Table 5.2, respectively. While the third scenario's matrices are different as it has two robots, so instead, the connection_matrix and threshold_matrix are $5 \times 5 \times 2$, as seen in Table 5.3. The connection_matrix is computed from a path loss (PL) function [148] that uses the Euclidean distance and takes as an input a distance_threshold (`d`) (computed from the channel model and geometry helper class) and the two vectors of coordinate pairs (Robot and AP). The threshold_matrix is calculated by comparing the elements of the connection_matrix to a receive threshold value RX_threshold to see if the AP is in range. The threshold_matrix is then implemented with the path constraints defined in Section 5.2 and lets the optimizer find a unique path between the Robot and its associated service.

Table 5.1: Connection Matrix (the calculated received signal strength (RSS) through Euclidean distance formula between robots and APs at each time-step) and Threshold Matrix (the measured RSS if it is less than ($\leq$) to threshold value (-90dbm))

(a) Connection Matrix 1

Robot, Time	AP1	AP2	AP3
R1, T1	64.933	101.123	106.316
R1, T2	102.384	63.301	102.983
R1, T3	106.332	99.328	56.038

(b) Threshold Matrix 1

AP1	AP2	AP3
1	0	0
0	1	0
0	0	1

Table 5.2: Connection Matrix 2 and Threshold Matrix 2 fora scenario with one robot and five APs

(a) Connection Matrix 2

Robot, Time	AP1	AP2	AP3	AP4	AP5
R1, T1	62.80	96.74	108.30	107.21	98.98
R1, T2	94.60	68.19	100.74	101.73	100.99
R1, T3	106.92	100.00	57.48	99.98	108.97
R1, T4	106.09	101.34	98.71	56.90	105.22
R1, T5	98.76	101.29	109.32	106.18	55.69

(b) Threshold Matrix 2

AP1	AP2	AP3	AP4	AP5
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Table 5.3: Connection Matrix 3 and Threshold Matrix 3 for scenario with two robots: robot 1 and robot 2 and five APs

(a) The RSS between robot 1 and APs

Robot, Time	AP1	AP2	AP3	AP4	AP5
R1, T1	58.29	97.74	106.67	108.22	99.60
R1, T2	94.84	68.20	100.38	100.86	98.99
R1, T3	105.96	101.64	57.34	100.82	107.94
R1, T4	105.87	101.03	100.07	54.87	104.97
R1, T5	101.18	100.22	109.17	105.02	56.26

(b) Threshold Matrix 3 for robot 1

AP1	AP2	AP3	AP4	AP5
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

(c) The RSS between robot 2 and APs

Robot, Time	AP1	AP2	AP3	AP4	AP5
R2, T1	101.36	99.66	106.20	105.28	53.35
R2, T2	108.20	100.65	101.29	68.97	106.88
R2, T3	107.05	98.96	56.65	101.03	108.34
R2, T4	98.67	54.15	100.54	100.97	100.82
R2, T5	70.37	96.72	107.16	106.05	99.32

(d) Threshold Matrix 3 for robot 2

AP1	AP2	AP3	AP4	AP5
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0

5.4 Integrating a Network Simulator with ARCHEX

Many telecommunication service providers claim to provide optimization tools that can perform resource optimization and planning of a 5G network. Among others, the optimization tools commercially available include Meritech, Atollranplan Forsk, Ranplan, Actix-One, Accuver, and Mentor

and ASSET. These tools possess different feature sets but generally produce only post-network deployment optimization. Their proprietary licensing limits their usefulness in this research work to achieve the objectives and investigate the dynamic area of 5G and subsequent generations. Here, we are interested in designing the architecture instead, for which a tool, i.e., ARCHEX that supports efficient design, produces high-quality solutions, and provides correctness guarantees, is highly desirable. However, the current version of ARCHEX is limited to supporting only static scenarios and is therefore unsuitable for a dynamic environment, such as a robotic environment.

We plan to introduce the mobility of the robots by creating a time dimension where the connections between template components can change dynamically and implementing `no_server_overload` constraints to map services to resourceful servers. The idea is to overcome ARCHEX limitations by integrating a network simulator to handle the design problem and satisfy the desired QoS requirements. A static analysis for the definition of the system architecture and a run-time or simulation-based analysis for its evaluation are required to achieve this objective. Our proposed methodology uses a network simulator, such as OMNeT++, which must be tailored to our defined objectives by constructing the appropriate models and topology.

Moreover, simulators are also considered in almost every domain because of their lower risk, low cost, and effectiveness, e.g., Bouras et al. [108] highlight the importance of selecting a suitable simulator depending on the speed, cost, accuracy, modularity, multi-protocol support, and convenience; for example, many network simulators, particularly for 5G systems, are available such as Simu5G [111], Open-Air-Interface [112], Vienna 5G [113], 5G-K simulator [114], 5G-LENA [115], WiSE [116], and 5G air-interface [117] to perform testing and simulation of new algorithms and

network behavior, respectively. Although simulators are considered effective tools for realizing the complex scenarios of a real-time system, their capabilities are still limited because they are time-consuming, provide a limited number of configuration evaluations, and do not generate the best possible strategies.

To overcome the above-listed shortcomings of both simulators and ARCHEX, first, we obtain a feasible candidate architecture generated from ARCHEX; second, we verify and analyze the architecture for two critical network parameters, i.e., overall throughput and end-to-end latency (E2E). We performed simulation with an event-driven simulator, i.e., OMNeT++, as it is widely used by researchers in the modern-era telecommunication sector, where they can test and evaluate complex communication systems and new algorithms and also provides multiple libraries that support data processing and communication protocols, which are ideal for our case study.

5.5 Simulation-Assisted Architecture Design Space Exploration of Indoor Robotics Networks

This section explores the combination of static architectural optimization with dynamic simulation. We describe an architecture through abstract models of components and a set of constraints that drive the design space exploration process to a few promising solutions. These are evaluated through simulation to extract detailed performance parameters. We tested and validated the tool extensions, optimizing and analyzing two indoor robotics scenarios for critical performance parameters, i.e., overall throughput and end-to-end delay (E2E).

5.5.1 Automated Simulation

Given an optimal architecture selected by ARCHEX, all the configuration files, the simulation execution, and the results are handled automatically. While the temporal dimension is modeled through discrete space in the MILP optimization phase, we use simulation to evaluate the solution on a continuous space and to observe the dynamic behavior of the architectural solution in terms of throughput and end-to-end (E2E) delay.

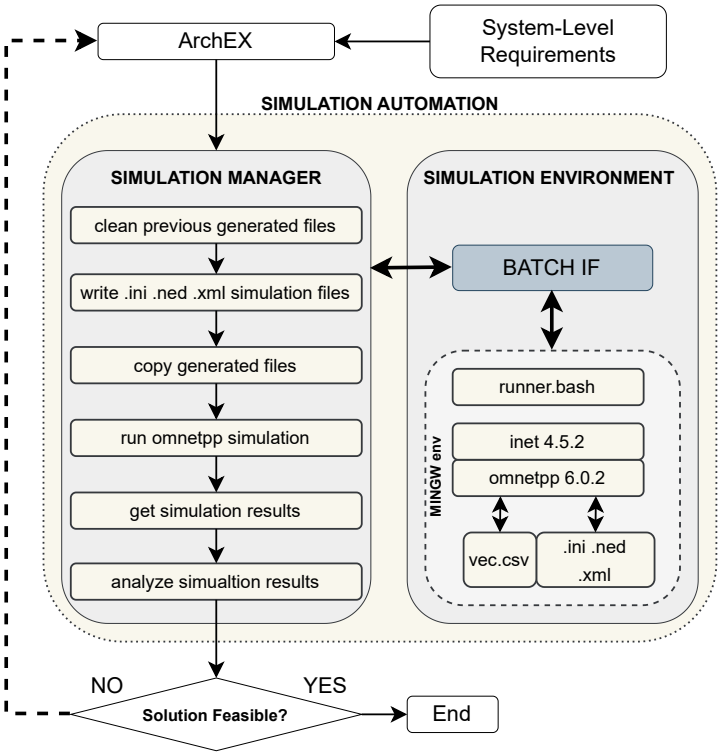


Figure 5.10: Framework for Automated Simulation

Having the topology information (coordinates of the APs and servers where the services are running) and the trajectory of the robots from the optimizer, a python script, denoted as `'Simulation Manager'` in Figure 5.10, is used to manage the automatic code generation of the configuration files, while the interfaces to manage OMNet++ simulations within the `MinGW` environment are implemented via batch scripts. This choice

maintains the design flexibility and independence between the automation software and the simulator, allowing us to modify the simulator environment for further work with minimal effort. In the simulation manager, the first step is to clean up the old simulation files that have been generated to ensure reliable results. Secondly, starting from ARCHEX outputs, all the files required by OMNet++ (i.e., `.ned` for topology, `.ini` for parameter configuration, and `.xml` for robot trajectories) are automatically generated.

The `runner.bash` is a batch script inside the simulation folder used to start and manage the simulation in the `MinGW` environment. In particular, it loads all the required `.dll` libraries and runs the simulation in command window mode (`'Cmdenv'`) to reduce the overhead given by the graphical user interface (`'Qtenv'`). After the simulation, the `.anf` results files are converted into `.csv` files (readable with `omnetppSimulationManager.py`) using the `opp_scavetool` OMNeT++ utility. The batch script RunSimulation.BAT acts as an interface between the Python code and the bash runner and is implemented to wait for the simulation process to complete. Finally, the `omnetppSimulationManager.py` retrieves Throughput and E2E delay by parsing the `.csv` files.

5.5.2 Tool Evaluation and Results

We begin our model evaluation by considering two scenarios in an indoor robotic environment. We aim to attain a feasible and optimal path through the APs so that the robots can reach the services at each time step. In the first scenario, we consider one robot, five access points, and two services, while in the second, we have two robots that compete for the resources. The latency bound is set to 1 ms. After the successful execution of the optimization problem, the architectures shown in Figure 5.8b and 5.9b are generated as the output, showing the node selection and the paths orchestrated by ARCHEX.

After each optimization phase, we perform the simulations through OMNeT++ 6.0.2 with the INET 4.5.0 extension. Table 5.4 reports the configuration parameters for all three scenarios. Notice that the configuration parameters for all three scenarios are similar except for the number of robots, APs, and servers. Switches enable communication between multi-

Table 5.4: Configuration parameters for scenarios 1 (Robot 1), scenario 2 (Robot1, Robot2), and scenario 3 (Robot0, Robot1)

Parameter	**Configuration**
Protocol	WiFi 802.11p
Robot Tx/Rx Power	0.52 mW
APs Tx/Rx Power	0.52 mW
Center Frequency	2.4 GHz
Network Bitrate	48 Mbps
Channel Bandwidth	20 MHz
Transport Protocol	UDP
Number of APs	**Scenario 1 and 2**: 5 **Scenario 3**: 15
Number of Robots	**Scenario 1**: R1 **Scenario 2**: R1, R2 **Scenario 3**: R0, R1
Number of Servers	**Scenario 1 and 2**: 2 **Scenario 3**: 15
Background Noise Power	-122 dBm

ple APs and model multi-hop transmission. Each robot sends radar data at 5.2 Mbps to the server hosting the requested service. All simulations are run on a ZBOOK with 32 GB of RAM and an Intel core-i7-11850h. We simulate the first 80 seconds of operation of each scenario, which take 46.05 s and 87.75 s, respectively, to complete.

The simulation scenarios for both scenarios are modeled with a mixed wired and wireless network with moving robots, APs, and servers, as shown in Figure 5.11. The simulation environment uses an XML file containing the robots' trajectories, as reported in Table 5.5. The required upper limit of 1 ms is met for the first scenario, and the average measured throughput

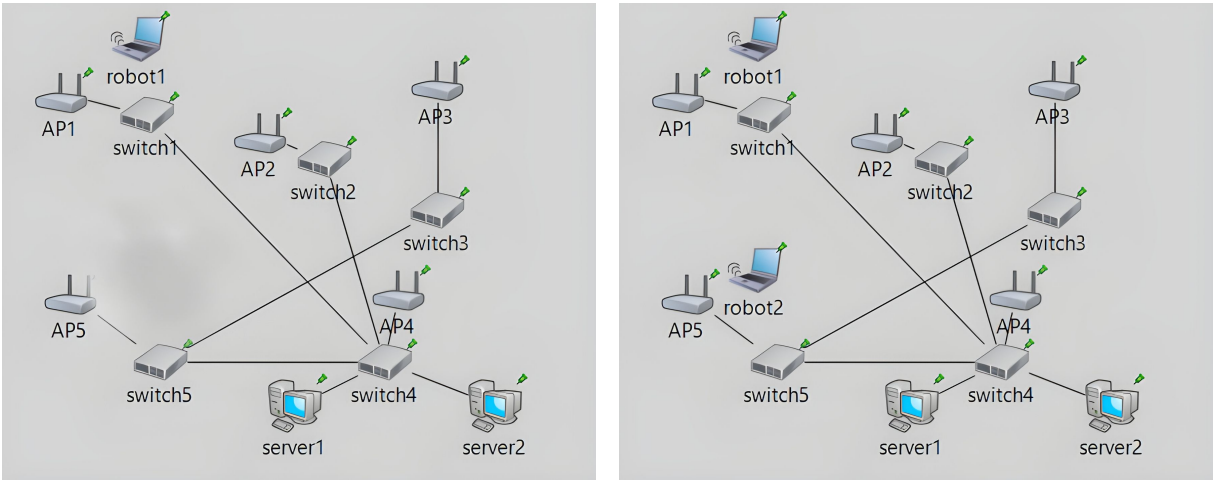


Figure 5.11: OMNeT++ user interface (Qtenv) for both scenarios with one and two robots

Table 5.5: Trajectories of Robot 1 and Robot 2 in scenario 2

Time Point	Start–End Time (s)	R1 Position (m)		R2 Position (m)	
		X	Y	X	Y
1	0.000–0.001	31.554	41.176	31.0417	437.585
2	0.001–20.0	388.251	113.903	685.710	435.900
3	20.0–40.0	720.912	21.051	740.912	25.0515
4	40.0–60.0	657.710	415.900	340.0	123.0
5	60.0–80.0	27.041	427.585	51.554	41.176

is about 5 Mbps. However, throughput is less than 5.2 Mbps because data is not transmitted during the authentication steps required to associate each robot with a new AP when a handover between two cells occurs.

For the second scenario, the average throughput is again around 5 Mbps, and the average upper limit of 1 ms for E2E delay is met for both robots. However, the maximum latency does not satisfy the constraints. Figure 5.12 shows the throughput and the E2E delay for both robots in time. While throughput is relatively constant (the AP can handle the aggregate throughput of the two robots), the E2E delay increases when a robot tries to reach a server with a longer path that traverses several access points using a multi-hop protocol. In addition, when the two robots transmit within the same cell to the same AP, the E2E delay increases to a maximum of

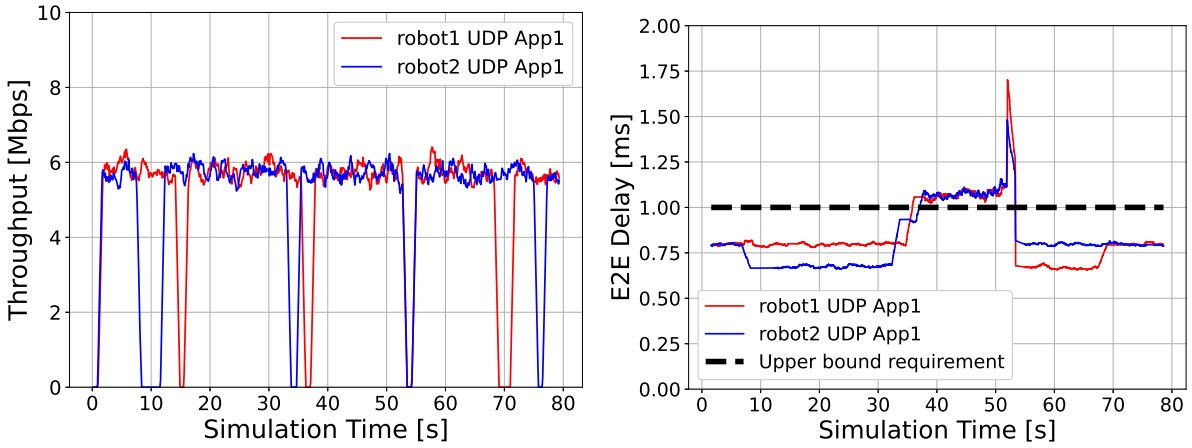


Figure 5.12: Second scenario: throughput and E2E delay

1.7 ms (for Robot1), which could result in an error for the robots performing real-time tasks. Not modeled in the original problem formulation, this behavior can be fed back to the optimizer as extra constraints to force alternative paths or the instantiation of additional APs. This allows us to improve the model and converge toward a stable solution. The simulation results are reported in Table 5.6.

Table 5.6: Throughput and E2E delay for the considered indoor robotics scenario

Scen.	Robot	Min E2E	Max E2E	Mean E2E	Mean Throughput
1	Robot1	0.65 ms	0.94 ms	0.82 ms	5.02 Mbps
2	Robot1	0.65 ms	1.70 ms	0.85 ms	5.11 Mbps
2	Robot2	0.66 ms	1.48 ms	0.83 ms	4.99 Mbps

A third scenario uses a more complex arrangement with 15 APs, each with a corresponding server. Two robots move around the space, connecting to three services (two for robot 0 and one for robot 1). The floor plan is shown in Figure 5.13 together with the robot trajectories and the location of the APs and the servers in the template. The figure also shows some representative communication paths. Each robot sends compressed camera data at 24 Mbps, corresponding to the maximum link capacity. In this case, we do not define specific bounds on latency and throughput.

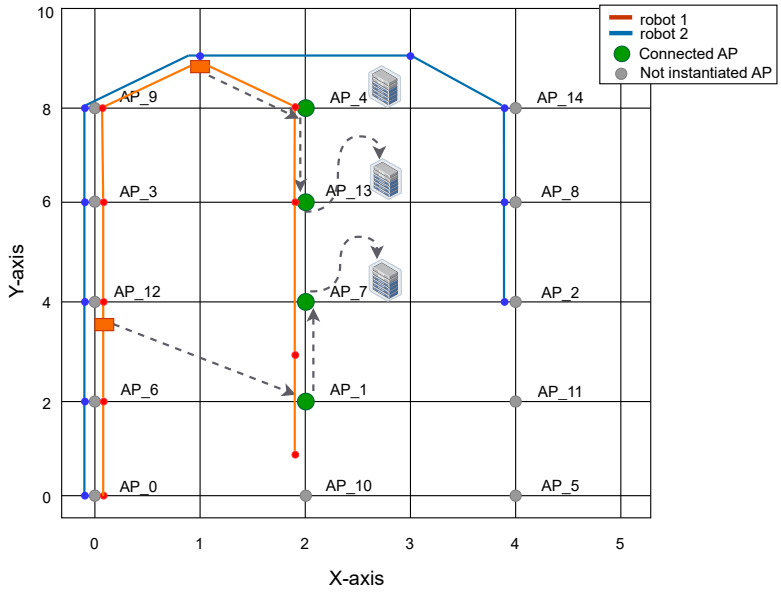


Figure 5.13: Third scenario floor plan and trajectories

The optimizer selects 4 out of the 15 APs located in the middle section of the deployment, guaranteeing complete coverage of the area. Three also connect to the corresponding servers to run the services. Like the first two scenarios, the third scenario is also modeled with a mixed wired and wireless network, as shown in Figure 5.14. Likewise, in scenario 2, the coor-

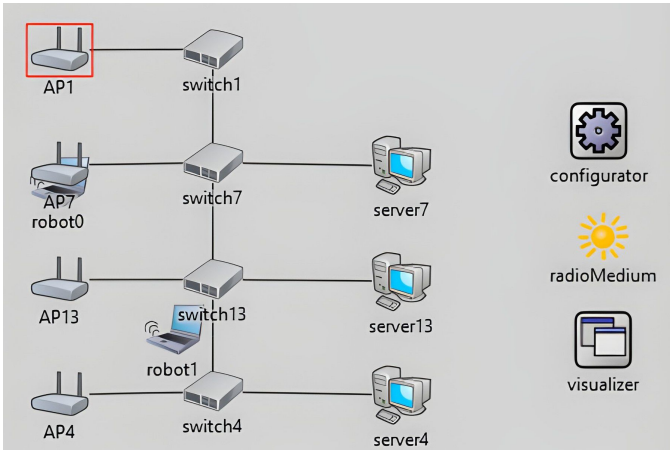


Figure 5.14: OMNeT++ user interface (Qtenv) for the third scenario with two robots

dinates of robots' trajectories stored in XML files are reported in Table 5.7.

Table 5.7: Trajectories of Robot 0 and Robot 1 in scenario 3

Time Point	Start–End Time (s)	R0 Position (m)		R1 Position (m)	
		X	Y	X	Y
1	0.000–0.001	0.0	0.0	0.0	0.0
2	0.001–40.0	0.0	200.0	0.0	200.0
3	40.0–80.0	0.0	400.0	0.0	400.0
4	80.0–120.0	0.0	600.0	0.0	600.0
5	120.0–160.0	0.0	800.0	0.0	800.0
6	160.0–200.0	100.0	900.0	100.0	900.0
7	200.0–240.0	200.0	800.0	300.0	900.0
8	240.0–280.0	200.0	600.0	400.0	800.0
9	280.0–320.0	200.0	300.0	400.0	600.0

Figure 5.15 shows the results of the simulation. We observe how the overall achieved communication throughput is significantly degraded at the beginning of the simulation when the robots move close to each other and share the same access point. While the optimizer does employ a chan-

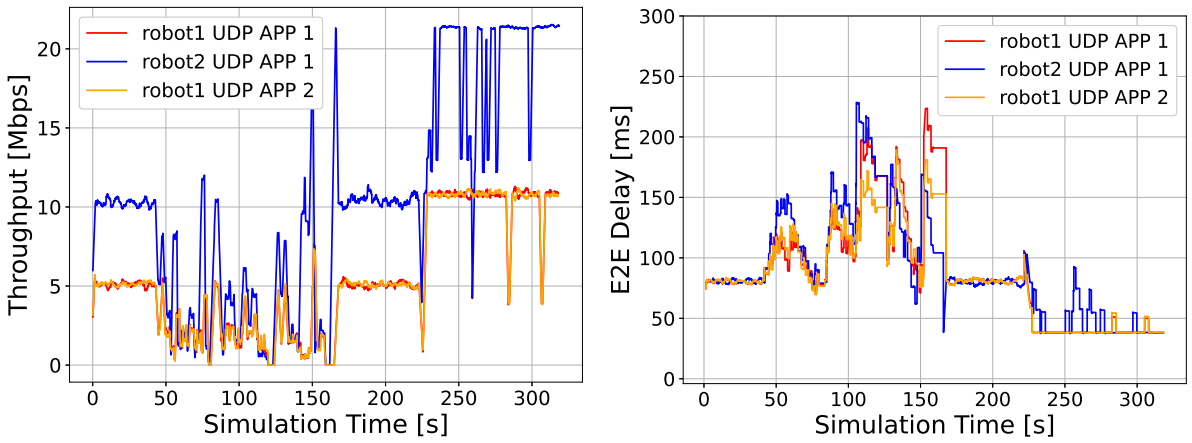


Figure 5.15: Third scenario: throughput and E2E delay

nel model to determine which APs are reachable at every time step and construct the paths [35], the simulator can more precisely account for both collisions and transmission errors, which are more frequent at the borders between cells, just before the handovers. This is visible in the 50–150 s time range, where the robots interfere with each other and several APs

are in view. A more precise communication scheduling could potentially address some of these issues.

In addition, the simulation shows that robot 1 experiences lower throughput since the resources are shared by the two services it uses. The E2E delay presents a similar, but opposite, evolution and points to the same critical sections. Note that throughput and latency need not be inverse of each other. In particular, robot 1 experiences latencies similar to robot 2 despite using two separate services. The simulation for this scenario runs for 776 seconds, highlighting the much higher complexity of performance evaluation due to the increased data transfer rate and longer simulated time.

This underscores the importance of combining the optimizer and the simulator. The optimizer selects only the most promising architectures, limiting a potentially expensive performance evaluation to fewer cases. Conversely, simulation leads the optimizer to exclude infeasible solutions, reducing the search space. The effect is, therefore, a conceivably much faster convergence toward a solution that satisfies the application requirements.

5.6 Discussions

This work presents a novel case study in which we re-formulated the MILP optimization problem and developed and implemented a dynamic scenario in ARCHEX. Unlike previous case studies in ARCHEX, we designed a network where a group of new patterns is developed and introduced to a dynamic feature, i.e., the time dimension and an objective function, to minimize the overall network cost by satisfying topology, mapping, and application constraints. Considering the reliability and QoS requirements, the formulated MILP optimization model considers path redundancy, where

robots and their associated services are always connected with resourceful servers through different paths at every time step. Nevertheless, several aspects of a real deployment are either too difficult to model for the optimizer or lead to excessive run times due to the combinatorial nature of the optimization problem.

Conversely, simulation easily handles details such as protocol messages and run-time collision but does not scale to a large design space. Therefore, the idea we pursue is to combine these two techniques. This technique combines abstract architectural models amenable to MILP optimization algorithms with performance and behavior models that reveal the detailed behavior of the system. The idea is to use the optimizer to find feasible candidate architectures to be evaluated by the simulator within a loop that converges to an optimal solution by feeding back information in the form of constraints to satisfy the network QoS requirements. In the future, we will extend our methodology with an automated iterative simulation in the loop (SIL) strategy, logically represented by the dotted line in Fig. 5.10, and evaluate the framework's scalability through more extensive case studies.

Chapter 6

Conclusion

Exploration into the upcoming generation of mobile networks endeavors to achieve the ambitious goal of collectively supporting a diverse array of heterogeneous services within a versatile and resilient communication and computing infrastructure. Within this context, the adept management of task offloading is paramount for ensuring the provision of high-quality services. Consequently, a demand arises for the optimal orchestration of computing and communication resources at the network edge. This important topic has been investigated extensively, with researchers explicitly addressing user mobility and proposing optimization algorithms to optimize server allocation to tasks while adhering to energy, latency, and communication delay constraints. Task offloading in mobile scenarios presents a complex challenge, requiring mobile users to be served by different servers at the edge as they move across network attachment points. These servers must possess positions and capabilities that align with the expected service level, making the overall management highly dynamic. In scenarios where dedicated virtual machines (VMs) or containers implement the offered services, such dynamic conditions necessitate frequent migration operations.

Following the challenges mentioned above, we provided our contributions in detail in Chapters 3 and 4; we provided an innovative approach

for the optimal and adaptable management of task offloading, specifically tailored for real-time applications. We consider the resource allocation challenge at the edge of a B5G network for real-time services by formulating a mixed-integer linear programming (MILP) model, wherein decision variables encompass computation and communication resources and multi-access edge computing (MEC) servers for executing VMs providing services at each time point. We propose a novel methodology in the context of URLLC-based applications, which are well represented in the robotic domain: AGVs (automated guided vehicles) touring a large logistic facility using the edge facilities for computation-intensive tasks. Network attachment points offer wireless connectivity to agents (e.g., AGVs) that require heterogeneous services. In a factory robotics environment, it is reasonable to assume that the agent or robot's mobility is predictable; hence, we model the Quality of Service (QoS) dynamics based on factors such as robot and VM positions, service requirements, link communication capabilities, and MEC server computing capabilities.

Unlike the current state-of-the-art, QoS is a function of computing and communication requirements, End-to-End (E2E) latency, and migration cost. The proposed approach is characterized by taking into account the constraints of the number of admitted VM migrations and the QoS lower and upper bounds expected by the agents. Leveraging state-of-the-art optimization tools enables the handling of problems of reasonable size regarding the number of cells and agents. The optimization can be performed offline before the initialization of system operations, assuming that agent trajectories are known in advance. Wireless connectivity is provided by network attachment points to agents, each with diverse service requirements. These attachment points establish connections to an edge network with computing capabilities facilitated by MEC servers. The agents subsequently link to one of the available MEC servers through edge links with

fixed communication capabilities.

Furthermore, in Chapter 5, we introduce a novel case study, where we reformulated the MILP optimization problem and implemented a dynamic indoor robotic scenario in a design-space exploration (DSE)-based optimization tool. Leveraging a scalable, modular, and open-source tool, i.e., ARCHEX, which proves its effectiveness in designing various cyber-physical systems (CPSs), particularly in wireless sensor networks (WSNs), the system designer can express the system architecture as a directed graph, defining nodes and interconnecting links with specific attributes from a predefined component library. Using a predefined pattern language, the system designer specifies system requirements and an objective function, subsequently running ARCHEX to derive an optimal system architecture. While existing literature primarily focuses on optimizing a given architecture, we aim to emphasize the earlier phase of architectural design. This involves determining the infrastructure placement, server deployment, sizing, and distribution of services. The objective is to automatically generate the architecture from a relatively simple description of the environment and application requirements expressed through performance and QoS constraints.

However, certain aspects of a real deployment pose challenges; they are either too difficult to model for the optimizer or lead to excessive run times due to the combinatorial nature of the optimization problem. We adopt a hybrid approach, combining a symbolic MILP optimization strategy with a simulation environment. Simulation excels at handling details such as protocol messages and runtime collisions but faces limitations in scaling to a large design space. In this part, our contributions span two dimensions. Firstly, we design and implement a new case study in ARCHEX, and unlike previous case studies, we devise a network with unique patterns, integrating a dynamic feature, i.e., the time dimension for a URLLC-based

indoor robotics scenario. The objective is to minimize the overall network cost while satisfying topology, mapping, and application constraints. The formulated MILP optimization model accounts for path redundancy, ensuring that robots and their associated services remain connected to resourceful servers through various paths at every time-step. Secondly, we verify the generated candidate architectural graph from ARCHEX and conduct an automated simulation using an event-driven simulator, i.e., OMNeT++. We successfully implemented multiple scenarios with multiple robots, access points, servers, and services and integrated the tool with an automatic simulation method. The resulting architecture is analyzed for two critical network parameters, i.e., overall throughput and E2E delay.

6.1 Future Work

When the size of the problem grows, and the system is highly dynamic and requires online optimization, heuristic approaches are needed to produce high-quality, sub-optimal solutions. This is one of the most promising research areas we reserve for future investigations, including futuristic scenarios where the base stations are mobile (e.g., aerial or terrestrial vehicles) and need an optimal position decision. We aim to implement more complex scenarios by increasing the number of robots, APs, servers, and services, evaluating the service migration, and performing fully automatic simulations in the loop with ARCHEX to verify the network architecture.

To extend the core structure of ARCHEX and perform the iterative simulation in loop (SIL), we will check through simulation at each optimization step whether the throughput and delay requirements are met and reformulate the constraints of the ARCHEX input problem when required. Considering that each robot has one or more associated services that always follow the robot's path, we will exclusively implement the service

migration strategy in the tool.

Dynamicity and iterative solving necessitate an overall rethinking of the optimization strategy. In particular, new patterns must be devised to aid the user in the specification of constraints that involve time and to represent different configurations of the system that may result from the migration of services between the network computation resources, guaranteeing the latency constraints. The transparency offered by the patterns and the efficiency of an iterative optimization are the essential ingredients for a methodology that is effective in proposing feasible and optimized solutions and, at the same time, makes it easy for the designer to specify the desired properties. The motivation for iterative optimization, i.e., online SIL, arises after multiple design examples are successfully implemented into the proposed architecture exploration framework, and that could be another considerable achievement. Additionally, the main challenge in exploiting the integration of any simulator with the optimization tool is to extract meaningful constraints that can guide the optimization tool in converging toward an optimal and feasible solution.

In an iterative optimization technique, the method starts by parsing the simulation output, and it must proceed by tracking the components involved in constraint violation through a process of abstraction to return information that can be used to exclude the solution in the succeeding optimizations. After completing network optimization and simulation, the network design and planning are considered suitable for deployment if the algorithm terminates with a feasible solution and satisfies the desired QoS requirements. However, if the results achieved do not meet the desired network requirements or the optimization problem is declared inconsistent, in that case, extra constraints must be generated after the simulation that can be fed into ARCHEX again and repeat the steps in a loop to modify and solve the problem until it produces an optimal solution for the network

design.

6.1.1 Future Applications Research Themes

The capabilities of 5G networks extend far beyond mobile broadband and support communication with unprecedented reliability, very low latencies, and massive IoT connectivity. This paves the way for the next era in industrial production, known as Industry 4.0, which aims to significantly improve the flexibility, versatility, usability, and efficiency of future smart factories and industrial plants. Industry 4.0 integrates the IoT and related services in industrial manufacturing and delivers seamless vertical and horizontal integration down the entire value chain and across all layers of the automation pyramid.

In this industrial context, robots have assumed a prevalent role as a means to deal safely with dangerous processes or as an aid for human activities. Specifically, to apply the architecture design methodology in a case study involving autonomous robots in an industrial context, one could likely address one of the following themes:

- Industrial Robotics: This application theme includes using robots in manufacturing, automation, and logistics industries. Robots can perform repetitive tasks such as assembly, packaging, and material handling.
- Service Robotics: This application theme involves using robots in service industries such as healthcare, retail, and hospitality. Robots can provide personalized services, assist with customer service, and provide medical care.
- Mobile Robotics: This application theme includes using robots in the transportation and delivery industries. Robots can be used to deliver goods and services, reducing the need for human intervention.

The primary manufacturing-domain use cases can be regrouped into several categories: motion control, control-to-control, mobile control panels, mobile robots, massive wireless sensor networks, remote access and maintenance, augmented reality, closed-loop process control, process monitoring, and plant asset management. Table 6.1 maps the various application areas to the use case categories [149].

Table 6.1: Areas of application and corresponding use cases

	Motion control	Control-to-control	Mobile control panels	Mobile robots	Massive wireless sensor networks	Remote access and maintenance	Augmented reality	Closed-loop process control	Process monitoring	Plant asset management
Factory Automation	X	X		X	X					
Process Automation				X	X			X	X	X
HMIs and production IT			X				X			
Logistics and warehousing	X			X						X
Monitoring and maintenance				X	X	X	X			

Bibliography

[1] M. A. Siddiqi, H. Yu, and J. Joung, "5g ultra-reliable low-latency communication implementation challenges and operational issues with iot devices," *Electronics*, vol. 8, no. 9, p. 981, 2019.

[2] K. Mallinson, "The path to 5g: as much evolution as revolution, may 2016," *URL: http://www. 3gpp. org/news-events/3gpp-news/1774-5g_wise-harbor (visited on 01/25/2018)*, 2016.

[3] Y. Duan, C. She, G. Zhao, and T. Q. Quek, "Delay analysis and computing offloading of urllc in mobile edge computing systems," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, IEEE, 2018.

[4] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, "5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view," *Ieee Access*, vol. 6, pp. 55765–55779, 2018.

[5] ETSI, "Listserv 16.5 listserv archives at list.etsi.org." `https://list.etsi.org/scripts/wa.exe?HOME`. (Accessed on 06/30/2023).

[6] M. A. Lema, A. Laya, T. Mahmoodi, M. Cuevas, J. Sachs, J. Markendahl, and M. Dohler, "Business case and technology analysis for 5g low latency applications," *IEEE Access*, vol. 5, pp. 5917–5935, 2017.

[7] Nokia, "5g use cases and requirements, white paper, nokia." https://www.hit.bme.hu/~jakab/edu/litr/5G/Nokia_5g_requirements_white_paper.pdf, July 2014. (Accessed on 03/05/2024).

[8] S. Chen and J. Zhao, "The requirements, challenges, and technologies for 5g of terrestrial mobile telecommunication," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 36–43, 2014.

[9] C. D. Warren, "Understanding 5g: Perspectives on future technological advancements in mobile." https://data.gsmaintelligence.com/research/research/research-2014/understanding-5g-perspectives-on-future-technological-advancemer Dec. 2014. (Accessed on 03/05/2024).

[10] IEEE, "Ieee 5g and beyond technology roadmap white paper5." https://futurenetworks.ieee.org/images/files/pdf/ieee-5g-roadmap-white-paper.pdf, October 2017.

[11] M. E. ISG, "Multi-access edge computing (MEC); phase 2: Use cases and requirements," Tech. Rep. RGS/MEC-0002v211TechReq, ETSI, 2018.

[12] L. Liu and Q. Fan, "Resource allocation optimization based on mixed integer linear programming in the multi-cloudlet environment," *IEEE Access*, vol. 6, pp. 24533–24542, 2018.

[13] D. Kirov, P. Nuzzo, R. Passerone, and A. Sangiovanni-Vincentelli, "Archex: An extensible framework for the exploration of cyber-physical system architectures," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2017.

[14] D. Kirov, *Optimization-Based Methodology for the Exploration of Cyber-Physical System Architectures.* PhD thesis, University of Trento, Italy, 2018.

[15] C.-P. Li, J. Jiang, W. Chen, T. Ji, and J. Smee, "5g ultra-reliable and low-latency systems design," in *2017 European Conference on Networks and Communications (EuCNC)*, pp. 1–5, IEEE, 2017.

[16] A. A. Esswie and K. I. Pedersen, "Opportunistic spatial preemptive scheduling for urllc and embb coexistence in multi-user 5g networks," *Ieee Access*, vol. 6, pp. 38451–38463, 2018.

[17] 3GPP, "Release 15." `https://www.3gpp.org/release-15`. (Accessed on 07/30/2023).

[18] ETSI, "Etsi, multi-access edge computing standards for mec." `https://www.etsi.org/technologies/multi-access-edge-computing`. (Accessed on 06/30/2023).

[19] Q.-V. Pham, L. B. Le, S.-H. Chung, and W.-J. Hwang, "Mobile edge computing with wireless backhaul: Joint task offloading and resource allocation," *IEEE Access*, vol. 7, pp. 16444–16459, 2019.

[20] ETSI, "Etsi white paper #11 mobile edge computing - a key technology towards 5g." `https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf`. (Accessed on 10/17/2023).

[21] M. Afrin, J. Jin, A. Rahman, Y.-C. Tian, and A. Kulkarni, "Multi-objective resource allocation for edge cloud-based robotic workflow in smart factory," *Future Generation Computer Systems*, vol. 97, pp. 119–130, 2019.

[22] F. Fellir, A. El Attar, K. Nafil, and L. Chung, "A multi-agent based model for task scheduling in cloud-fog computing platform," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp. 377–382, IEEE, 2020.

[23] A. Alreshidi, A. Ahmad, A. B. Altamimi, K. Sultan, and R. Mehmood, "Software architecture for mobile cloud computing systems," *Future Internet*, vol. 11, no. 11, 2019.

[24] L. Qingyang, Y. Long, Z. Zhiya, K. Hu, Y. Song, and J. Wang, "Research on energy internet architecture based on cloud computing platform," in *2020 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, pp. 164–168, 2020.

[25] Z. Bai, Y. Lin, Y. Cao, and W. Wang, "Delay-aware cooperative task offloading for multi-uav enabled edge-cloud computing," *IEEE Transactions on Mobile Computing*, 2022.

[26] A. Qadeer and M. J. Lee, "Hrl-edge-cloud: Multi-resource allocation in edge-cloud based smart-streetscape system using heuristic reinforcement learning," *Information Systems Frontiers*, pp. 1–17, 2023.

[27] M. Babar, M. S. Khan, F. Ali, M. Imran, and M. Shoaib, "Cloudlet computing: recent advances, taxonomy, and challenges," *IEEE access*, vol. 9, pp. 29609–29622, 2021.

[28] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, "A survey on cloudlets, mobile edge, and fog computing," in *2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pp. 139–142, IEEE, 2021.

[29] M. Chen, Y. Qian, J. Chen, K. Hwang, S. Mao, and L. Hu, "Privacy protection and intrusion avoidance for cloudlet-based medical data sharing," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1274–1283, 2020.

[30] M. Buvana, K. Loheswaran, K. Madhavi, S. Ponnusamy, A. Behura, and R. Jayavadivel, "Improved resource management and utilization based on a fog-cloud computing system with iot incorporated with classifier systems," *Microprocessors and Microsystems*, p. 103815, 2021.

[31] Y. Teng, M. Liu, F. R. Yu, V. C. M. Leung, M. Song, and Y. Zhang, "Resource allocation for ultra-dense networks: A survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2134–2168, 2018.

[32] N. Zhang, N. Cheng, A. T. Gamage, K. Zhang, J. W. Mark, and X. Shen, "Cloud assisted hetnets toward 5g wireless networks," *IEEE communications magazine*, vol. 53, no. 6, pp. 59–65, 2015.

[33] L. Liu and Q. Fan, "Resource allocation optimization based on mixed integer linear programming in the multi-cloudlet environment," *IEEE Access*, vol. 6, pp. 24533–24542, 2018.

[34] P. K. Gkonis, P. T. Trakadas, and D. I. Kaklamani, "A comprehensive study on simulation techniques for 5g networks: State of the art results, analysis, and future challenges," *Electronics*, vol. 9, 2020.

[35] D. Kirov, P. Nuzzo, A. L. Sangiovanni-Vincentelli, and R. Passerone, "Efficient encodings for scalable exploration of cyber-physical system architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[36] I. F. Akyildiz, A. Kak, and S. Nie, "6G and beyond the future of wireless communications systems," *IEEE Access*, vol. 8, pp. 133995–134030, 2020.

[37] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng, "When mobile terminals meet the cloud: computation offloading as the bridge," *IEEE Network*, vol. 27, no. 5, pp. 28–33, 2013.

[38] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile networks and Applications*, vol. 18, pp. 129–140, 2013.

[39] T. Prastowo, A. Shah, L. Palopoli, and R. Passerone, "Resource optimization in mec-based B5G networks for indoor robotics environment," in *Applications in Electronics Pervading Industry, Environment and Society (APPLEPIES)*, pp. 164–172, Springer, 2022.

[40] T. Prastowo, A. Shah, L. Palopoli, R. Passerone, and G. Piro, "Migration-aware optimized resource allocation in B5G edge networks," in *2022 IEEE 19th Annual Consumer Communications and Networking Conference (CCNC)*, pp. 106–113, 2022.

[41] A. Shah and R. Passerone, "Architectural exploration and design for ultra-reliable low-latency indoor robotics systems," in *Proceedings of the IEEE Consumer Communications & Networking Conference*, CCNC 2024, (Las Vegas, NV), January 6–9, 2024.

[42] C. Bianchi, A. Shah, C. Marangoni, and R. Passerone, "Toward simulation-assisted architecture design space exploration of indoor robotics networks," in *2024 IEEE 20th International Conference on Factory Communication Systems (WFCS)*, WFCS 2024, (Toulouse, France), April 17–19 2024.

[43] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, 2020.

[44] Q. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[45] T. Subramanya, D. Harutyunyan, and R. Riggio, "Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks," *Computer Networks*, vol. 166, p. 106980, 2020.

[46] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. S. Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. on Mob. Comp.*, 2020.

[47] G. Wang, F. Xu, and C. Zhao, "Multi-access edge computing based vehicular network: Joint task scheduling and resource allocation strategy," in *IEEE Int. Conf. on Comm. (ICC) Workshops*, pp. 1–6, 2020.

[48] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, and H. Duan, "Mobility-aware multi-user offloading optimization for mobile edge computing," *IEEE Trans. on Vehicular Tech.*, vol. 69, no. 3, pp. 3341–3356, 2020.

[49] S. Thananjeyan, C. A. Chan, E. Wong, and A. Nirmalathas, "Mobility-aware energy optimization in hosts selection for computation offloading in multi-access edge computing," *IEEE Open Journal of Com.Soc.*, 2020.

[50] C.-L. Wu, T.-C. Chiu, C.-Y. Wang, and A.-C. Pang, "Mobility-aware deep reinforcement learning with glimpse mobility prediction in edge computing," in *IEEE Int. Conf. on Comm. (ICC)*, pp. 1–7, 2020.

[51] B. Németh, N. Molner, J. J. Martín-Pérez, C. J. Bernardos, A. de la Oliva, and B. Sonkoly, "Delay and reliability-constrained VNF placement on mobile and volatile 5G infrastructure," *arXiv:2007.11870*, 2020.

[52] A. Dalgkitsis, P.-V. Mekikis, A. Antonopoulos, and C. Verikoukis, "Data driven service orchestration for vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–1, 2020.

[53] X. Yu, M. Guan, M. Liao, and X. Fan, "Pre-migration of vehicle to network services based on priority in mobile edge computing," *IEEE Access*, vol. 7, pp. 3722–3730, 2019.

[54] P. Roy, A. Tahsin, S. Sarker, T. Adhikary, M. A. Razzaque, and M. M. Hassan, "User mobility and quality-of-experience aware placement of virtual network functions in 5G," *Computer Communications*, vol. 150, pp. 367–377, 2020.

[55] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on markov decision process," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1272–1288, 2019.

[56] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, IEEE, 2016.

[57] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha, "The role of cloudlets in hostile environments," *IEEE Pervasive Computing*, vol. 12, no. 4, pp. 40–49,, 2013-10.

[58] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Transactions on Mobile Computing*, vol. 14, no. 12, pp. 2516–2529,, 2015-12-01.

[59] W. Balid, H. Tafish, and H. H. Refai, "Intelligent vehicle counting and classification sensor for real-time traffic surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1784–1794, 2017.

[60] M. Emara, M. C. Filippou, and D. Sabella, "Mec-assisted end-to-end latency evaluations for c-v2x communications," in *2018 European Conference on Networks and Communications (EuCNC)*, pp. 1–9, IEEE, 2018.

[61] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled v2x service placement for intelligent transportation systems," *IEEE Transactions on Mobile Computing*, 2020.

[62] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.

[63] L. Yang, D. Yang, J. Cao, Y. Sahni, and X. Xu, "Qos guaranteed resource allocation for live virtual machine migration in edge clouds," *IEEE Access*, vol. 8, pp. 78441–78451, 2020.

[64] D. Fontanelli, L. Greco, and L. Palopoli, "Soft real-time scheduling for embedded control systems," *Automatica*, vol. 49, no. 8, pp. 2330–2338, 2013.

[65] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "Practical solutions for QoS-based resource allocation problems," in *Proceedings of the 19th IEEE Real-Time Systems Symposium*, pp. 296–306, 1998.

[66] F. M. de Chamisso, D. Cancila, L. Soulier, R. Passerone, and M. Aupetit, "Lifelong exploratory navigation: an architecture for safer mobile robots," *IEEE Design & Test*, vol. 38, pp. 57–64, October 2021.

[67] V. Magnago, L. Palopoli, R. Passerone, D. Fontanelli, and D. Macii, "Effective landmark placement for robot indoor localization with position uncertainty constraints," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, pp. 4443–4455, November 2019.

[68] A. O. Akmandor and N. K. Jha, "Keep the stress away with soda: Stress detection and alleviation system," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 3, no. 4, pp. 269–282, 2017.

[69] T. J. Y. James, K. Otto, and K. Wood, "A comparison of design decisions made early and late in development," *ICED*, 08 2017.

[70] H. Neema, Z. Lattmann, P. Meijer, J. Klingler, S. Neema, T. Bapty, J. Sztipanovits, and G. Karsai, "Design space exploration and manipulation for cyber-physical systems," in *IFIP First International Workshop on Design Space Exploration of Cyber-Physical Systems (IDEAL'2014), Springer-Verlag Berlin Heidelberg*, p. 8, 2014.

[71] K. Lynch, R. Ramsey, G. Ball, M. Schmit, and K. Collins, "Ontology-driven metamodel validation in cyber-physical systems," in *Information Technology: New Generations*, pp. 1255–1258, Springer International Publishing, 2016.

[72] R. Passerone, I. B. Hafaiedh, S. Graf, A. Benveniste, D. Cancila, A. Cuccuru, S. Gérard, F. Terrier, W. Damm, A. Ferrari,

L. Mangeruca, B. Josko, T. Peikenkamp, and A. Sangiovanni-Vincentelli, "Metamodels in Europe: Languages, tools, and applications," *IEEE Design and Test of Computers*, vol. 26, pp. 38–53, May/June 2009.

[73] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Sangiovanni-Vincentelli, "Languages and tools for hybrid systems design," *Foundations and Trends® in Electronic Design Automation*, vol. 1, no. 1–2, pp. 1–193, 2006.

[74] A. Pinto, L. P. Carloni, R. Passerone, and A. L. Sangiovanni-Vincentelli, "Interchange formats for hybrid systems: Abstract semantics," in *Hybrid Systems: Computation and Control,* 9^{th} *International Workshop, HSCC06* (J. P. Hespanha and A. Tiwari, eds.), vol. 3927 of *Lecture Notes in Computer Science*, (Santa Barbara, CA, USA), pp. 491–506, Springer, March 29–31, 2006.

[75] A. Pinto, A. L. Sangiovanni-Vincentelli, L. P. Carloni, and R. Passerone, "Interchange formats for hybrid systems: Review and proposal," in *Hybrid Systems: Computation and Control, 8th International Workshop, HSCC05* (M. Morari and L. Thiele, eds.), vol. 3414 of *Lecture Notes in Computer Science*, (Zurich, Switzerland), pp. 526–541, Springer, March 9–11, 2005.

[76] P. Terway, K. Hamidouche, and N. K. Jha, "DISPATCH: design space exploration of cyber-physical systems," *CoRR*, vol. abs/2009.10214, 2020.

[77] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1.* Wiley, 2003.

[78] S. Karris, *Introduction to Simulink with Engineering Applications.* Orchard Publications, 2006.

[79] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, "Taming heterogeneity - the ptolemy approach," *Proc. of the IEEE*, vol. 91, no. 1, pp. 127–144, 2003.

[80] A. Davare, D. Densmore, L. Guo, R. Passerone, A. L. Sangiovanni-Vincentelli, A. Simalatsar, and Q. Zhu, "METROII: A design environment for cyber-physical systems," *ACM Transactions on Embedded Computing Systems*, vol. 12, pp. 49:1–49:31, March 2013.

[81] L. Guo, Q. Zhu, P. Nuzzo, R. Passerone, A. L. Sangiovanni-Vincentelli, and E. A. Lee, "Metronomy: a function-architecture co-simulation framework for timing verification of cyber-physical systems," in *Proceedings of the International Conference on Hardware/-Software Codesign and System Synthesis*, CODES14, (New Delhi, India), pp. 24:1–24:10, ACM, New York, NY, USA, October 12–17, 2014.

[82] D. Densmore, R. Passerone, and A. L. Sangiovanni-Vincentelli, "A platform-based taxonomy for ESL design," *IEEE Design and Test of Computers*, vol. 23, pp. 359–374, May 2006.

[83] A. Pinto, A. Bonivento, A. L. Sangiovanni-Vincentelli, R. Passerone, and M. Sgroi, "System level design paradigms: Platform-based design and communication synthesis," *ACM Transactions on Design Automation of Electronic Systems*, vol. 11, pp. 537–563, July 2006.

[84] R. Passerone, J. R. Burch, and A. L. Sangiovanni-Vincentelli, "Refinement preserving approximations for the design and verification of heterogeneous systems," *Formal Methods in System Design*, vol. 31, pp. 1–33, August 2007.

[85] H. T. T. Le and R. Passerone, "Refinement-based synthesis of correct contract model decompositions," in *Proceedings of the* 12^{th} *ACM-*

IEEE International Conference on Formal Methods and Models for System Design, MEMOCODE14, (Lausanne, Switzerland), pp. 134–143, October 19–21, 2014.

[86] R. Passerone, Íñigo Íncer Romeo, and A. L. Sangiovanni-Vincentelli, "Coherent extension, composition, and merging operators in contract models for system design," *ACM Transactions on Embedded Computing Systems*, vol. 18, pp. 86:1–86:23, October 2019.

[87] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, and K. G. Larsen, "Contracts for system design," *Foundations and Trends® in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[88] L. Dal Lago, O. Ferrante, R. Passerone, and A. Ferrari, "Dependability assessment of SOA-based CPS with contracts and model-based fault injection," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 360–369, January 2018.

[89] T. T. H. Le, R. Passerone, U. Fahrenberg, and A. Legay, "Contract-based requirement modularization via synthesis of correct decompositions," *ACM Transactions on Embedded Computing Systems*, vol. 15, no. 2, pp. 33:1–33:26, 2016.

[90] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming Dr. Frankenstein: Contract-based design for cyber-physical systems," *European Journal of Control*, vol. 18, no. 3, pp. 217–238, 2012.

[91] D. Cancila, R. Passerone, T. Vardanega, and M. Panunzio, "Toward correctness in the specification and handling of non-functional attributes of high-integrity real-time embedded systems," *IEEE Transactions on Industrial Informatics*, vol. 6, pp. 181–194, May 2010.

[92] T. T. H. Le, R. Passerone, U. Fahrenberg, and A. Legay, "A tag contract framework for modeling heterogeneous systems," *Science of Computer Programming*, vol. 115–116, pp. 225–246, 2016.

[93] H. T. T. Le, R. Passerone, U. Fahrenberg, and A. Legay, "A tag contract framework for heterogeneous systems," in *Proceedings of the* 12^{th} *International Workshop on Foundations of Coordination Languages and Self Adaptive Systems*, FOCLASA13, (Malaga, Spain), pp. 204–217, September 11, 2013.

[94] T. T. H. Le, R. Passerone, U. Fahrenberg, and A. Legay, "Tag machines for modeling heterogeneous systems," in *Proceedings of the* 13^{th} *International Conference on Application of Concurrency to System Design*, ACSD13, (Barcelona, Spain), pp. 186–195, July 8–10, 2013.

[95] J. Finn, P. Nuzzo, and A. Sangiovanni-Vincentelli, "A mixed discrete-continuous optimization scheme for cyber-physical system architecture exploration," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 216–223, 2015.

[96] IBM, "Ilog cplex optimization studio." `https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer`.

[97] "The leader in decision intelligence technology - gurobi optimization." `https://www.gurobi.com/`.

[98] "Mosek." `https://www.mosek.com/products/mosek/`. (Accessed on 06/21/2023).

[99] G. Guan, W. Dong, Y. Gao, K. Fu, and Z. Cheng, "Tinylink: A holistic system for rapid development of iot applications," in *Proceedings*

of the 23rd Annual International Conference on Mobile Computing and Networking, pp. 383–395, 2017.

[100] A. Moin, P. Nuzzo, A. L. Sangiovanni-Vincentelli, and J. M. Rabaey, "Optimized design of a human intranet network," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2017.

[101] M. A. Mansoori and M. R. Casu, "Efficient training and hardware co-design of machine learning models," in *International Conference on Applications in Electronics Pervading Industry, Environment and Society (APPLEPIES)*, pp. 243–248, Springer, 2022.

[102] A. Tierno, G. Turri, A. Cimatti, and R. Passerone, "Symbolic encoding of reliability for the design of redundant architectures," in *Proceedings of the* 5^{th} *IEEE International Conference on Industrial Cyber-Physical Systems*, ICPS 2022, (Coventry, UK), May 24–26, 2022.

[103] A. Tierno, G. Turri, A. Cimatti, and R. Passerone, "Automatic design space exploration of redundant architectures," in *Proceedings of the International Conference on Applications in Electronics Pervading Industry, Environment and Society*, ApplePies 2021, (Pisa, Italy), September 21–22, 2021.

[104] A. Boulis, "Castalia: Revealing pitfalls in designing distributed algorithms in wsn," in *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, SenSys '07, (New York, NY, USA), p. 407–408, Association for Computing Machinery, 2007.

[105] F. Balarin and R. Passerone, "Specification, synthesis and simulation of transactor processes," *IEEE Transactions on Computer-Aided De-*

sign of Integrated Circuits and Systems, vol. 26, pp. 1749–1762, October 2007.

[106] O. Ferrante, R. Passerone, A. Ferrari, L. Mangeruca, and C. Sofronis, "BCL: a compositional contract language for embedded systems," in *Proceedings of the* 19^{th} *IEEE International Conference on Emerging Technologies and Factory Automation*, ETFA14, (Barcelona, Spain), pp. 1–6, September 16–19, 2014.

[107] O. Ferrante, R. Passerone, A. Ferrari, L. Mangeruca, C. Sofronis, and M. D'Angelo, "Monitor-based run-time contract verification of distributed systems," in *Proceedings of the* 9^{th} *IEEE International Symposium on Industrial Embedded Systems*, SIES14, (Pisa, Italy), June 18–20, 2014.

[108] C. Bouras, G. Diles, A. Gkamas, and A. Zacharopoulos, "Comparison of 4g and 5g network simulators," in *The Fifteenth International Conference on Wireless and Mobile Communications (ICWMC 2019)*, (Rome Italy), pp. 13–18, IARIA XPS Press, 2019.

[109] I. Minakov, R. Passerone, A. Rizzardi, and S. Sicari, "A comparative study of recent wireless sensor network simulators," *ACM Transactions on Sensor Networks*, vol. 12, no. 3, pp. 20:1–20:39, 2016.

[110] I. Minakov and R. Passerone, "PASES: An energy-aware design space exploration framework for wireless sensor networks," *Journal of Systems Architecture*, vol. 59, pp. 626–642, September 2013.

[111] G. NARDINI, G. STEA, A. VIRDIS, and D. SABELLA, "Simu5g: a system-level simulator for 5g networks," in *SIMULTECH 2020*, (online streaming), pp. 68–80, INSTICC, SciTePress, 2020.

[112] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5g research," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 33–38, 2014.

[113] S. Pratschner, B. Tahir, L. Marijanovic, M. Mussbah, K. Kirev, R. Nissel, S. Schwarz, and M. Rupp, "Versatile mobile communications simulation: The vienna 5g link level simulator," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, p. 226, 2018.

[114] J. Baek, J. Bae, Y. Kim, J. Lim, E. Park, J. Lee, G. Lee, S. I. Han, C. Chu, and Y. Han, "5g k-simulator of flexible, open, modular (fom) structure and web-based 5g k-simplatform," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, (Las Vegas, NV, USA), pp. 1–4, IEEE, 2019.

[115] N. Patriciello, S. Lagen, B. Bojovic, and L. Giupponi, "An e2e simulator for 5g nr networks," *Simulation Modelling Practice and Theory*, vol. 96, p. 101933, 2019.

[116] C.-K. Jao, C.-Y. Wang, T.-Y. Yeh, C.-C. Tsai, L.-C. Lo, J.-H. Chen, W.-C. Pao, and W.-H. Sheen, "Wise: A system-level simulator for 5g mobile networks," *IEEE Wireless Communications*, vol. 25, no. 2, pp. 4–7, 2018.

[117] S. Martiradonna, A. Grassi, G. Piro, and G. Boggia, "Understanding the 5g-air-simulator: a tutorial on design criteria, technical components, and reference use cases," *Computer Networks*, vol. 177, p. 107314, 2020.

[118] M. Törngren, F. Asplund, S. Bensalem, J. McDermid, R. Passerone, H. Pfeifer, A. Sangiovanni-Vincentelli, and B. Schätz, "Characteri-

zation, analysis, and recommendations for exploiting the opportunities of cyber-physical systems," in *Cyber-Physical Systems* (H. Song, D. B. Rawat, S. Jeschke, and C. Brecher, eds.), Intelligent Data Centric Systems, ch. 1, pp. 3–14, Academic Press, Elsevier, September 2016.

[119] M. Törngren, S. Bensalem, J. McDermid, R. Passerone, A. L. Sangiovanni-Vincentelli, and B. Schätz, "Education and training challenges in the era of cyber-physical systems: beyond traditional engineering," in *Proceedings of the Workshop on Embedded and Cyber-Physical Systems Education*, WESE 2015, (Amsterdam, The Netherlands), pp. 8:1–8:5, ACM, October 8, 2015.

[120] M. Berno, J. J. Alcaraz, and M. Rossi, "On the allocation of computing tasks under QoS constraints in hierarchical MEC architectures," in *4th Int. Conf. on Fog and Mobile Edge Comp. (FMEC)*, pp. 37–44, 2019.

[121] D. Kirov, P. Nuzzo, R. Passerone, and A. L. Sangiovanni-Vincentelli, "Optimized selection of wireless network topologies and components via efficient pruning of feasible paths," in *Proceedings of the* 55^{th} *Design Automation Conference*, (San Francisco, CA), June 24–28, 2018.

[122] W.-C. Chien, S.-Y. Huang, C.-F. Lai, H.-C. Chao, M. S. Hossain, and G. Muhammad, "Multiple contents offloading mechanism in ai-enabled opportunistic networks," *Computer Comm.*, vol. 155, pp. 93–103, 2020.

[123] T. Dlamini, Á. F. Gambín, D. Munaretto, and M. Rossi, "Online resource management in energy harvesting BS sites through predic-

tion and soft-scaling of computing resources," in *IEEE Symp. on Personal, Indoor, and Mobile Radio Comm.*, pp. 1820–1826, 2018.

[124] Z. Cheng, Q. Wang, Z. Li, and G. Rudolph, "Computation offloading and resource allocation for mobile edge computing," in *IEEE Symp. Series on Comp. Intelligence (SSCI)*, pp. 2735–2740, 2019.

[125] B. Yang, X. Cao, J. Bassey, X. Li, T. Kroecker, and L. Qian, "Computation offloading in multi-access edge computing networks: A multi-task learning approach," in *IEEE Int. Conf. on Comm.*, pp. 1–6, 2019.

[126] P. Nuzzo, N. Bajaj, M. Masin, D. Kirov, R. Passerone, and A. L. Sangiovanni-Vincentelli, "Optimized selection of reliable and cost-effective safety-critical system architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2109–2123, 2020.

[127] H. Peng, Q. Ye, and X. S. Shen, "SDN-based resource management for autonomous vehicular networks: A multi-access edge computing approach," *IEEE Wireless Comm.*, vol. 26, no. 4, pp. 156–162, 2019.

[128] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "LORM: Learning to optimize for resource management in wireless networks with few training samples," *IEEE Transactions on Wireless Communications*, vol. 19, pp. 665–679, Jan 2020.

[129] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in-vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11158–11168, 2019.

[130] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for mobile edge computing," in *Symp. on Mod. & Opt. in Mob., Ad Hoc, & Wi. Net. (WiOpt)*, pp. 1–6, IEEE, 2018.

[131] L. Ferdouse, A. Anpalagan, and S. Erkucuk, "Joint communication and computing resource allocation in 5G cloud radio access networks," *IEEE Trans. on Vehicular Tech.*, vol. 68, no. 9, pp. 9122–9135, 2019.

[132] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on emerging topics in computing*, 2019.

[133] L. Chen, D. Yang, M. Nogueira, C. Wang, D. Zhang, *et al.*, "Data-driven C-RAN optimization exploiting traffic and mobility dynamics of mobile users," *IEEE Transactions on Mobile Computing*, 2020.

[134] Z. Hong, H. Huang, S. Guo, W. Chen, and Z. Zheng, "QoS-aware cooperative computation offloading for robot swarms in cloud robotics," *IEEE Trans. on Vehicular Tech.*, vol. 68, no. 4, pp. 4027–4041, 2019.

[135] R. Passerone, D. Cancila, M. Albano, S. Mouelhi, S. Plosz, E. Jantunen, A. Ryabokon, E. Laarouchi, C. Hegedűs, and P. Varga, "A methodology for the design of safety-compliant and secure communication of autonomous vehicles," *IEEE Access*, vol. 7, pp. 125022–125037, 2019.

[136] F. Malandrino, C. F. Chiasserini, G. Einziger, and G. Scalosub, "Reducing service deployment cost through VNF sharing," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2363–2376, 2019.

[137] M. Afrin, J. Jin, A. Rahman, Y.-C. Tian, and A. Kulkarni, "Multi-objective resource allocation for edge cloud-based robotic workflow in smart factory," *Future Generation Computer Systems*, vol. 97, pp. 119–130, 2019.

[138] S. P. Bradley, A. C. Hax, and T. L. Magnanti, *Applied Mathematical Programming*, ch. 9. Addison-Wesley, 1977.

[139] J. P. Vielma, S. Ahmed, and G. Nemhauser, "Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions," *Operations Research*, vol. 58, no. 2, pp. 303–315, 2010.

[140] G. Brown *et al.*, "Ultra-reliable low-latency 5G for industrial automation," *Technol. Rep. Qualcomm*, vol. 2, p. 52065394, 2018.

[141] "Fujitsu server: Fujitsu global." `https://www.fujitsu.com/global/products/computing/servers/`, May 2023. (Accessed on 05/26/2023).

[142] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Comm. Surveys & Tutorials*, vol. 20, no. 4, pp. 3098–3130, 2018.

[143] S. Z. Asif, *5G Mobile Communications: Concepts and Technologies*. Boca Raton, FL: CRC Press, 2018.

[144] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.

[145] F. H. Fitzek, P. Seeling, T. Höschele, and B. Jacobfeuerborn, "On the need of computing in future communication networks," in *Computing*

in Communication Networks: From Theory to Practice (F. H. Fitzek, F. Granelli, and P. Seeling, eds.), pp. 3–45, UK: Academic Press, 2020.

[146] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *2004 IEEE International Conference on Robotics and Automation*, (Washington, DC, USA), pp. 284–289, IEEE, 2004.

[147] A. Virdis and M. Kirsche, *A Practical Introduction to the OMNeT++ Simulation Framework*, pp. 3–51. Cham: Springer International Publishing, 2019.

[148] D. Kirov, R. Passerone, and M. Donelli, "Statistical characterization of the 2.4 GHz radio channel for WSN in indoor office environments," in *Proceedings of the* 21^{st} *IEEE International Conference on Emerging Technologies and Factory Automation*, ETFA16, (Berlin, Germany), September 6–9, 2016.

[149] 5G-ACIA, "5g-acia publications, white papers, positioning papers, and case studies on specific-topics white papers." `https://5g-acia.org/insight/whitepapers/`, December 2023.

Appendix A

List of Abbreviations in Thesis

Abbreviation	Full-text
Chapter 1	
5G	Fifth-generation
B5G	Beyond 5G
MEC	Multi-access edge computing
URLLC	Ultra-Reliable Low-Latency Communication
eMBB	Enhanced Mobile Broadband
mMTC	Massive Machine Type Communication
M2M	Machine-to-Machine Communication
ITU	International Telecommunication Union
IoT	Internet of Things
AGV	Autonomous guided vehicles
UAV	Unmanned aerial vehicles
QoS	Quality of Service
AR and VR	Augment Reality and Virtual Reality
LTE	Long term evolution
AP	Access Point
ITS	Intelligent Transportation Systems
UHD	Ultra-high-definition
MILP	Mixed-integer linear program
ARCHEX	Architectural Exploration
DSE	Design-space exploration
CPS	Cyber-physical systems
MEC	Multi-access edge computing
BS	Base Station
3GPP	Third generation partnership project
End-to-End	E2E

ETSI	European Telecommunication Standards Institute
QoE	Quality of Experience
M2M	Machine-to-Machine
MDP	Markov decision process
VM	Virtual machine
V2X	Vehicle-to-everything communication
NSGA-II	Non-Dominated Sorting Genetic Algorithm-II
VRU	Vulnerable road users
CAM	Cooperative awareness messages
FC	Fog computing
CC	Cloud computing
MCC	Mobile cloud computing
LC	Local cloud
LAN	Local area network
RSS	Received signal strength
SVG	Scalable vector graphics
V2X	Vehicle to everything
AR and VR	Augment reality and Virtual reality
WSN	Wireless sensor network
RPL	Reconfigurable production line
EPN	Electrical Power Network (Aircraft Power Distribution in ARCHEX
HMI	Human-machine interactive
LQ	Link Quality
Chapter 2	
MBRLLC	Mobile broadband reliable low latency communications
mURLLC	Massive ultra-low latency communications
HCS	Human-centric services
MDP	Markov Decision Process
VM	Virtual Machine
MOO	Multi-objective optimization
BO	Bayesian Optimization
HLS	High-level synthesis
GA	Genetic Algorithm
NN	Neural Network
BAN	Body area network
Chapter 3	
5G	Fifth-generation
B5G	Beyond 5G
MEC	Multi-access edge computing
URLLC	Ultra-Reliable Low-Latency Communication
VM	Virtual machine

QoS	Quality of service
MILP	Mixed-integer linear program
RT	Real-time
RRUs	Remote Radio Units
GIPS	Giga instructions per second
Mbps	Megabits per second
Chapter 4	
5G and B5G	Fifth-generation and beyond 5G
MBRLLC	Mobile broadband reliable low latency communications
mURLLC	Massive ultra-low latency communications
HCS	Human-centric services
KPIs	Key performance indicators
VMs	Virtual machines
MEC	Multi-access edge computing
QoS	Quality of service
MILP	Mixed-integer linear program
AGVs	Automated guided vehicles
Chapter 5	
QoS	Quality of service
MEC	Multi-access edge computing
URLLC	Ultra-reliable low-latency communication
E2E	End-to-End
DSE	Design-space exploration
AGV	Automated guided vehicles
MILP	Mixed-integer linear program
RSS	Received signal strength
WSN	Wireless sensor networks
AP	Access points
SVG	Scalable vector graphics
BANs	Body area networks
CPS	Cyber-physical systems
SIL	Simulation in loop
PL	Path loss