ABSTRACT

Title of Dissertation: Resource Allocation in Next-Generation Mobile Networks

Anousheh Gholami Ghavamabad
Doctor of Philosophy, 2022

Dissertation Directed by: Professor John S. Baras
Department of Electrical and Computer Engineering

The increasing heterogeneity of the mobile network infrastructure together with the explosively growing demand for bandwidth-hungry services with diverse quality of service (QoS) requirements leads to a degradation in the performance of traditional networks. To address this issue in next-generation mobile networks (NGMN), various technologies such as software-defined networking (SDN), network function virtualization (NFV), mobile edge/cloud computing (MEC/MCC), non-terrestrial networks (NTN), and edge ML are essential. Towards this direction, an optimal allocation and management of heterogeneous network resources to achieve the required low latency, energy efficiency, high reliability, enhanced coverage and connectivity, etc. is a key challenge to be solved urgently. In this dissertation, we address four critical and challenging resource allocation problems in NGMN and propose efficient solutions to tackle them.

In the first part, we address the network slice resource provisioning problem in NGMN for delivering a wide range of services promised by 5G systems and beyond, including enhanced mobile broadband (eMBB), ultra-reliable and low latency (URLLC), and massive machine-type

communication (mMTC). Network slicing is one of the major solutions needed to meet the differentiated service requirements of NGMN, under one common network infrastructure. Towards robust mobile network slicing, we propose a novel approach for the end-to-end (E2E) resource allocation in a realistic scenario with uncertainty in slices' demands using stochastic programming. The effectiveness of our proposed methodology is validated through simulations.

Despite the significant benefits that network slicing has demonstrated to bring to the management and performance of NGMN, the real-time response required by many emerging delay-sensitive applications, such as autonomous driving, remote health, and smart manufacturing, necessitates the integration of multi-access edge computing (MEC) into network sliding for 5G networks and beyond. To this end, we discuss a novel collaborative cloud-edge-local computation offloading scheme in the next two parts of this dissertation. The first part studies the problem from the perspective of the infrastructure provider and shows the effectiveness of the proposed approach in addressing the rising number of latency-sensitive services and improving energy efficiency which has become a primary concern in NGMN. Moreover, taking into account the perspective of application (higher layer), we propose a novel framework for the optimal reservation of resources by applications, resulting in significant resource savings and reduced cost. The proposed method utilizes application-specific resource coupling relationships modeled using linear regression analysis. We further improve this approach by using Reinforcement Learning to automatically derive resource coupling functions in dynamic environments.

Enhanced connectivity and coverage are other key objectives of NGMN. In this regard, unmanned aerial vehicles (UAVs) have been extensively utilized to provide wireless connectivity in rural and under-developed areas, enhance network capacity, and provide support for peaks or unexpected surges in user demand. The popularity of UAVs in such scenarios is mainly owing

to their fast deployment, cost-efficiency, and superior communication performance resulting from line-of-sight (LoS)-dominated wireless channels. In the fifth part of this dissertation, we formulate the problem of aerial platform resource allocation and traffic routing in multi-UAV relaying systems wherein UAVs are deployed as flying base stations. Our proposed solution is shown to improve the supported traffic with minimum deployment cost.

Moreover, the new breed of intelligent devices and applications such as UAVs, AR/VR, remote health, autonomous vehicles, etc. requires a novel paradigm shift from traditional cloud-based learning to a distributed, low-latency, and reliable ML at the network edge. To this end, Federated Learning (FL) has been proposed as a new learning scheme that enables devices to collaboratively learn a shared model while keeping the training data locally. However, the performance of FL is significantly affected by various security threats such as data and model poisoning attacks. Towards reliable edge learning, in the last part of this dissertation, we propose trust as a metric to measure the trustworthiness of the FL agents and thereby enhance the reliability of FL.

Resource Allocation in Next Generation Mobile Networks

by

Anousheh Gholami Ghavamabad

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor John S. Baras, Chair/Advisor
Professor Richard J. La
Professor Behtash Babadi
Professor Andre L. Tits
Professor Bruce L. Golden, Dean's Representative

Dedication

To my Pedar, Maadar, Azi and Reihaneh.

Acknowledgments

I owe my gratitude to all the people who have made this dissertation possible. First and foremost, I would like express my deepest gratitude to my advisor, Professor John S. Baras, for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the past five years. His patience, motivation, guidance and immense knowledge helped me in all aspects of my academic life, and it has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank Professor Richard La, Professor Behtash Babadi, Professor Andre Tits, and Professor Bruce L. Golden for serving on my dissertation committee. Their feedback during various stages has been instrumental in the improvement and completion of this work. I am greatly thankful to Mrs. Kim Edwards, for her generous and timely assistance with the administrative aspects of my work.

I would also like thank my colleagues during my internships at NEC Laboratories of America, in particular Dr. Srimat Chakradhar, Kunal Rao, Wang-Pin Hsiung, Oliver Po, and Murugan Sankaradas. They provided me with wonderful industrial research experience and tremendous help regarding the technical soundness of the second part of this dissertation.

My colleagues and labmates at university have enriched my graduate life in many ways and deserve a special mention. First of all, I owe a debt of gratitude to Professor Chrysa Papagianni for her truly indispensable mentorship. Her experience, deep knowledge and insightful comments

have guided me along this path significantly. I am specially indebted to my labmate and forever friend, Nariman Torkzaban, for his continuous support, and the valuable discussions and collaborations we had the result of which are included in this thesis. I would also like to thank my old office-mates Siddharth Bansal, Omkar Ninawe, Dr. Usman Amin Fiaz, and Dr. Fatima Alimardani, who made the office an enjoyable place to work.

I owe my deepest thanks to my lovely family - my mother and father, Azi, Reihaneh, Mohammad, Hesam, and my sweet niece, Jana - who have always stood by me, and have pulled me through against impossible odds at times even from a far distance. Words cannot express the gratitude I owe them. I would also like to thank my long-time friends, Pegah, Shadi, and Shaghayegh who are like family to me. I would be certainly at fault not to thank my cute kitty Lola, for choosing me as her favorite human and making my days delightful even by sleeping on my keyboad.

This dissertation is based upon research supported by the Office of Naval Research (ONR) grant N00014-17-1-262, Leidos Corporation, and DARPA under Agreement No. HR00111990027.

It is impossible to remember all, and I apologize to those I've inadvertently left out.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

NGMN	Next- generation mobile network
SDN	Software-defined networking
NFV	Network function virtualization
QoS	Quality of service
QoE	Quality of experience
MEC	Mobile edge computing (multi-access edge computing)
MCC	Mobile cloud computing
NTN	Non-terrestrial network
eMBB	Enhanced mobile broadband
URLLC	Ultra-reliable, and low latency
mMTC	Massive machine-type communication
InP	Infrastructure provider
SP	Service provider
MNO	Mobile network provider
CN	Core network
RAN	Radio access network
TN	Transport network
VMNO	Virtual mobile network provider
SFC	Service function chain
VM	Virtual machine
VNF	Virtual network function
NAT	Network address translation
5GC	5G core
SMF	Session management function
AMF	Access and mobility function
UPF	User plane function
UE	User equipment
C-RAN	Cloud-RAN
NG-RAN	Next-generation RAN
RRH	Remote radio heads
BBU	Baseband unit
CU	Centralized unit
DU	Distributed unit
PRB	Physical resource block

gNB	Next generation NodeB
MD	Mobile device
VR/AR	Virtual/augmented reality
ITS	Intelligent transportation system
UAV	Unmanned aerial vehicles
LoS	Line-of-sight
MANET	Mobile ad hoc network
FL	Federated learning

Chapter 1: Introduction

As mobile networks became heterogeneous, dynamic, and capable of organizing themselves, efficient and effective management of network resources to achieve a high quality of service (QoS) requires keeping track of and optimizing a far larger number of network elements and the interactions between them. The 5G main service classes of enhanced mobile broadband (eMBB), ultra-reliable and low-latency (uRLLC), and massive machine-type communication (mMTC), demand very diverse and sometimes extreme requirements. Moreover, with the advent of new services such as extended reality (XR), holographic communications, eHealth, etc., it is expected that the next-generation mobile networks (NGMN) will be ultra-large-scale, highly dynamic, and extremely heterogeneous. As a result, the resource management and optimization in NGMN will become even more challenging calling for advancements in different domains. To this end, various technologies such as software-defined networking (SDN), network function virtualization (NFV), mobile edge/cloud computing (MEC/MCC), non-terrestrial networks (NTN), and edge machine learning (ML) are arising as enablers for 5G networks and beyond. Thanks to the flexibility that NFV brings to network resource management and the relatively very short reconfiguration time of software-defined networks, network slicing offers a dynamic and on-demand infrastructure resource provisioning that can support several virtual and isolated networks on top of shared physical infrastructure. However, a challenging practical issue in network slicing is the uncertainties

in demands for resources required by different services. In this dissertation, we propose a novel approach to handling the E2E network slicing problem under demand uncertainty using two-stage stochastic programming. A two-timescale resource provisioning algorithm is developed to allocate RAN and core network resources based on the dynamics in each of these domains.

Moreover, the explosion of intelligent and latency-sensitive applications such as AR/VR, remote health, and autonomous driving, has resulted in a paradigm shift from mobile cloud computing (MCC) to mobile edge computing (MEC) which enables the deployment and execution of distributed computing capabilities at the network edge. In such a heterogeneous infrastructure, compute and network resources realized through network slices are available at different tiers, making the problem of service design and optimization very challenging. The problem becomes even more challenging considering the recent advancements in the service orchestration framework from monolithic application architecture to microservices architecture. A microservices-based application encompasses multiple independent modules (components/functions) with arbitrary interconnections, usually modeled as a graph. Each module requires different resources such as computing and storage, and the overall performance of the application relies on the allocated resources to all the application components. Thus, the resource management of multi-component applications in NGMN is a very challenging problem. This problem can be viewed from the perspective of InP or application developers, each with specific decision scope and objective. Taking into account the perspective of InP, we propose a collaborative cloud-edge-local computation offloading system for multi-component applications realized through microservices architecture. The proposed scheme is capable of reducing the delay and energy cost while satisfying infrastructure capacity and hard energy constraints and effectively accommodates both computationally intensive and latency-sensitive mobile applications. Moreover, from the perspective of application (higher

layer), we propose two novel frameworks based on mathematical optimization and reinforcement learning (RL) for the optimal reservation of resources by applications in a dynamic environment resulting in significant resource savings and reduced costs for service providers. In this solution, we introduce a novel concept of resource coupling, which captures the relationships between the usage of different pairs of resources and their impact on application performance. The introduced coupling functions are exploited in the resource reservation of applications requested to the InPs and delivered via network slices.

Furthermore, the NGMN is envisioned to support 3D network coverage due to integrating terrestrial and non-terrestrial technologies (e.g., UAV-assisted and satellite communications). The opportunity to realize UAV-mounted flying BSs that can relocate themselves in an on-demand manner to boost coverage, connectivity, and spectral efficiency has been extensively explored in recent years. Aerial Platforms (APs) can be deployed to provide wireless connectivity in rural and under-developed areas, enhance network capacity and provide support for peaks or unexpected surges in user demand. Considering that the UAVs potentially form a multi-hop aerial network, capacity and connectivity constraints have significant impacts on the end-to-end network performance. To this end, the problem of UAV placement, relocation, and traffic routing in a UAV-assisted mobile network is explored in this dissertation. Similar to the terrestrial network, APs can benefit from SDN/NFV and more specifically, the network slicing characteristic of 5G and 6G networks by enabling virtually isolated on-board processing systems on UAVs, HAPs, and satellites, making an extension of 5G/6G network slices to remote areas possible.

On the other hand, the new breed of intelligent devices and high-stake applications such as drones, and AR/VR, autonomous systems require a novel paradigm change from cloud-based

ML to distributed, low-latency, and reliable ML at the network edge (referred to as edge ML). Federated learning (FL) is a new learning framework that enables edge devices to collaboratively learn a shared prediction model while keeping all local training data on device, decoupling the ability to do ML from the need to store the data in the cloud. The main advantages of FL especially in edge computing environments are smarter models, lower latency, and less power consumption, all while ensuring privacy. Moreover, in addition to providing an update to the shared model, the improved model on edge devices can be used immediately, powering experiences personalized by the way the users use their mobile equipment. However, the performance of FL is significantly affected by various security threats such as data and model poisoning attacks. Towards reliable edge ML, in the last part of this dissertation, we propose trust as a metric to measure the trustworthiness of the FL agents and thereby enhance the security of FL.

1.1 Contributions and Organization of the Dissertation

This dissertation is organized into five chapters, covering different and interrelated resource allocation problems in NGMN. Chapter 2 addresses the problem of mobile network slicing. While the literature on the resource allocation problem of network slicing in the RAN and core domains is rich, we consider the end-to-end mobile network slicing problem. Moreover, in contrast to most of the existing works with the assumption of full knowledge about slice demands, we assume a realistic scenario with demand uncertainty. Thus, we formulate the end-to-end resource allocation problem for mobile network slicing with demand uncertainty. We propose a novel robust two-timescale framework to provision resources over core and radio access networks using two-stage stochastic programming. Our simulations show the effectiveness of the proposed

methodology in supporting the observed real-time traffic demand in a robust manner.

In Chapter 3, we study the resource orchestration problem of next-generation cloud and edge computing systems for the deployment of multi-component applications realized through microservices architecture. Taking into account the perspective of the infrastructure providers and considering generic applications with arbitrary structure and usage domain, we illustrate for the first time that a collaborative cloud-edge-local computation offloading scheme for multi-component applications results in a higher acceptance ratio with lower average energy and delay, compared to the baseline solutions relying on edge-local or cloud-local offloading. Our solution is based on solving a MILP and in order to deal with the time-complexity of the proposed scheme, we employ a heuristic algorithm which is shown to perform in close proximity to the MILP-based solution while being scalable.

In Chapter 4, the resource orchestration of multi-component applications is studied from the viewpoint of application developers. We propose a novel optimization method for the resource reservation problem of applications, above a network slice abstraction, that achieves significant resource savings while maintaining application performance requirements. The proposed solution incorporates a novel concept, namely resource coupling, that captures the relationship between the usage of different resources (such as network and compute) and their impact on the application performance. While we present the practical results of the proposed methodology for real-life application examples of intelligent transportation systems and watchlist applications, our framework is generic and could be applied to any application deployed in a microservices architecture. Compared to the existing solutions which ignore the resource couplings, our method saves significant network and computation as the infrastructure condition changes. In a further step, we extend our solution to an online setup in a dynamic environment (changing video scenes for video analytics

applications) by using RL. We show that the RL-based solution is capable of automatically deriving the nonlinear resource couplings and environmental conditions, and by doing so, it outperforms our first method.

In Chapter 5, we study the design and resource allocation problem of multi-UAV relaying systems in NGMN. In contrast to many existing frameworks for aerial platforms which consider only a single-hop or a two-hop aerial communication, we allow multi-hop aerial paths to carry the ground traffic. Moreover, we consider the end-to-end traffic guarantee by imposing capacity and routing constraints in our formulation while most of the existing solutions study the UAV placement problem with connectivity constraints/objectives and ignore the end-to-end traffic delivery. We propose a joint UAV placement and traffic routing solution that deploys the minimum number of UAVs as flying base stations to satisfy the traffic demand of a ground mobile network. We demonstrate that the proposed solution improves the average supported traffic compared to the state-of-the-art.

In Chapter 6, we address the problem of reliable edge learning in NGMN. Focusing on the FL framework as a key enabler for intelligent edge networks, we propose a trust-aware decentralized FL scheme wherein trust is used as a metric to measure the reliability of model updates shared by an agent in the FL training process. Our proposed solution is shown to effectively capture the malicious behavior of agents attacking the learning model through model poisoning attacks.

In Chapter 7, we conclude the dissertation and discuss future directions

Chapter 2: Resource Allocation for Mobile Network Slicing

2.1 Overview

Network softwarization based on network function virtualization (NFV) and software-defined networking (SDN) is promised to realize programmable control and agile management of network resources in NGMN [2]. While NFV enables the virtualization and segmentation of the underlying physical infrastructure devices, SDN introduces a novel way of controlling the routing of data packets through a logically centralized controller and by decoupling the network data and control planes. This new paradigm differs from that of traditional networks which use proprietary hardware devices for different network functions (e.g. routers and switches). Building on SDN and NFV, *network slicing* enables the segmentation of the physical infrastructure into a number of logically isolated sub-networks resulting in significant network management and equipment cost reductions [3]. One of the key applications of *network slicing* is the realization of the promised use cases by the fifth generation (5G) of mobile networks as shown in Fig. 2.1. To support a wide range of services in the categories of enhanced mobile broadband (eMBB), ultra-reliable and low latency (uRLLC) and massive machine-type communication (mMTC) in 5G networks, dedicated resources must be provisioned for each service in a dynamic and efficient way. In the paradigm of 5G mobile systems enhanced by *network slicing*, mobile network operators (MNOs) manage and set up network slices and provide service providers (SPs) with customized and scalable network

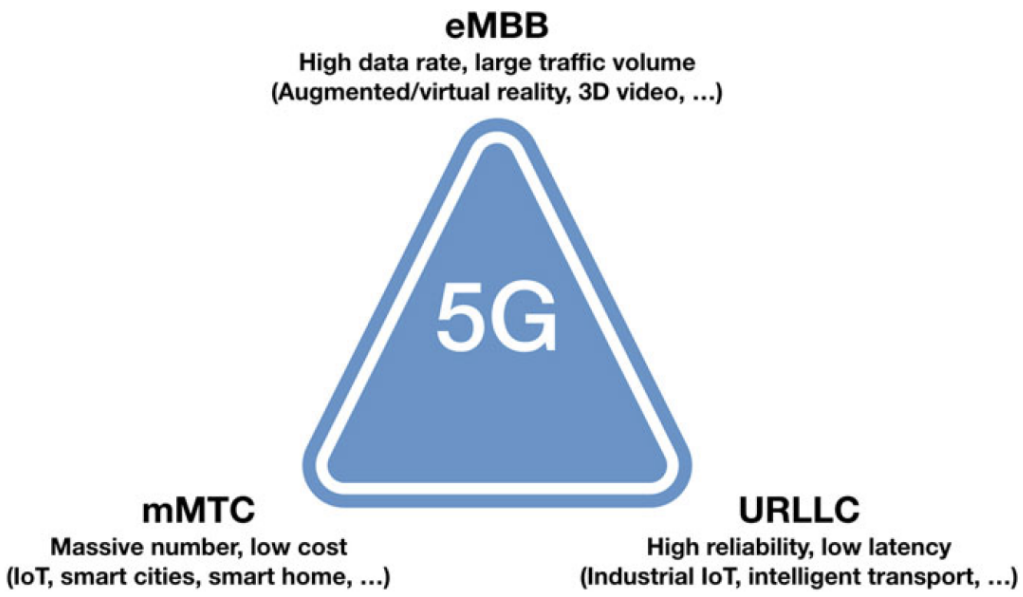


Figure 2.1: Envisioned 5G Use Cases Realized Through Network Slicing

slices [4]. The SPs, namely *tenants*, dedicate network slices to the customers of different services with specific QoS requirements in a sustainable way. A network slice spans across different parts of the network such as radio access network (RAN), core network (CN), and transport network (TN), forming an end-to-end (E2E) sub-network. 3GPP has put efforts into incorporating *network slicing* in the specification of both RAN and CN domains [5]. In 5G, RAN slicing introduces the capability of sharing physical network infrastructure among mobile virtual network operators (VMNOs). This is in contrast to the traditional static allocation of resources such as frequency and power by reserving them on the fly with *network slicing* based on the user demand and radio channel variations [6].

A network slice corresponding to a sub-network consists of service function chains (SFCs), each including multiple virtual network functions (VNFs) with interconnections between them. A VNF is a service running usually in a virtual machine (VM) on a common virtualization platform. Examples of VNFs include virtualized routers, firewalls, and network address translation (NAT)

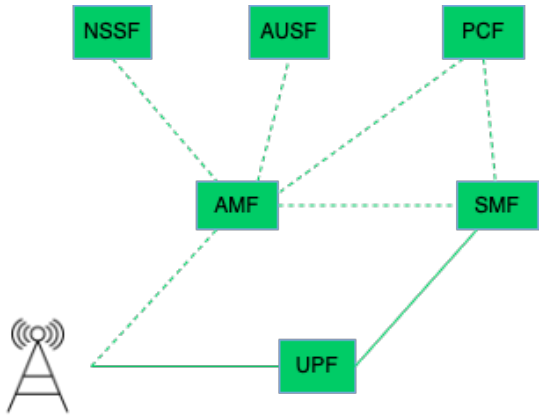


Figure 2.2: An Example of a Network Slice

services. Fig.2.2 depicts an example of a 5G network slice SFC. It includes 5G core (5GC) functions such as session management function (SMF), access and mobility function (AMF), and user plane function (UPF). Design and optimization of E2E network slices requires resource provisioning across heterogeneous physical and virtual network infrastructure, each with specific technical constraints. Moreover, with the proliferation of advanced and diverse digital services such as hologram video streaming and multisensory extended reality, the sixth-generation (6G) mobile system is envisioned to realize massive and extremely heterogeneous *network slicing*, where tenancy would be extended further to the end user [7]. In such architectures with slicing-aware user equipment (UE), new challenges are introduced. For instance, UE mobility should be considered by the SPs as part of slice setup and management [8]. In Fig. 2.3, the deployment of 5G network slices a shared infrastructure is shown. Each VNF is placed on an infrastructure node and interconnections between VNFs representing data communication requirements are mapped to substrate paths (single or multi-hop paths). Moreover, the RAN domain is enhanced by cloud-RAN (C-RAN) architecture [9] which is a promising solution to reach the ambitious capacity and energy efficiency goals of next-generation RAN (NG-RAN) systems. In C-RAN, the RAN is decomposed into two parts: the distributed remote radio heads (RRHs) and the centralized

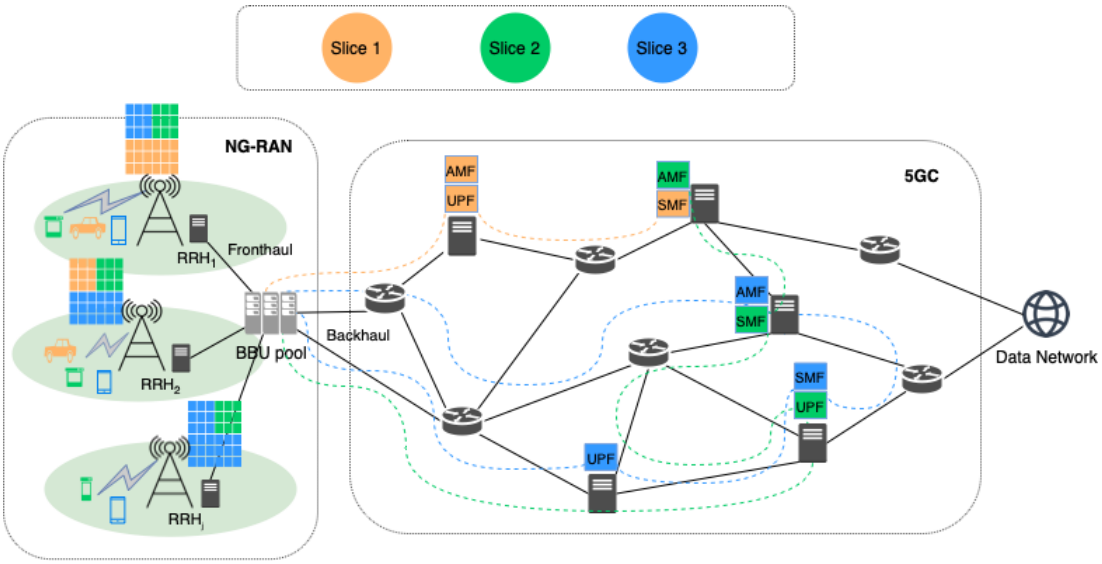


Figure 2.3: End-to-end Resource Provisioning of Network Slices

baseband units (BBUs) pool. RRHs provide basic radio functionalities (encompassing lower-level network functionalities such as RF and low PHY) and cover UE over a large area. On the other hand, BBUs manage the network resources cooperatively and handle the required signal processing (corresponding to higher-level functionalities such as high PHY, MAC, RLC, PDCP, RRC, and SDAP). Further improvements are proposed in stand-alone 5G systems by breaking BBUs into centralized unit (CU) (which includes high PHY, MAC, and RLC) and distributed unit (DU) (that handles PDCP, RRC, and SDAP RAN protocol layers). The BBU pool performs dynamic resource allocation according to the real-time demands based on the virtualized resources in cloud computing.

Although *network slicing* enhances the service agility, assurance, and security in NGMN, dynamic and efficient network slice orchestration and optimization face critical challenges. One of the most challenging issues is uncertainties in the demand for different services supported through network slices. Today, resource provisioning for network slices is typically done in a best-effort manner [10]. However, there is no guarantee that resources allocated to different

slices will be sufficient to support a fluctuating number of users with varying demands. *Elasticity* which refers to the availability of resources in an adaptive way according to user demands plays a critical role in avoiding resource under-utilization or over-utilization [11]. The slice demand variations as well as the dynamics in resource availability may degrade the slice QoS compared to the service level agreement (SLA) promised by the SPs [12].

The main focus of this chapter is the E2E network slice resource provisioning taking into account demand uncertainties. We assume that the forecasted traffic patterns are given from existing prediction solutions such as [13], and propose a novel resource provisioning method for E2E network slicing using stochastic programming. Stochastic programming [14] is a powerful tool to address optimization under uncertainties. We aim at formulating the joint resource allocation in the domain of NG-RAN and mobile packet core (5GC) in a dynamic environment and then adjusting the RAN slice resource allocation as needed. Indeed, the proposed adaptive RAN slicing is triggered more frequently (shorter life cycle) compared to the 5GC segment, due to the existence of more dynamic factors in the wireless network, such as user mobility and channel condition changes. In accordance with this adaptation requirement, we propose a two-timescale approach for the E2E network slicing resource provisioning. The objective of the proposed formulation is to minimize the total provisioning cost, while satisfying capacity constraints and distinct QoS requirements of network slices.

The rest of this chapter is organized as follows. In section 2.2, we present the literature review. Section 2.3 provides an overview of stochastic programming which builds the necessary mathematical background for the rest of the chapter. Section 2.4 describes system model and problem formulation. The proposed solution is presented in Section 2.5. Numerical results are provided in Section 2.6. We highlight the conclusions and future directions in Section 2.7.

2.2 Related Work

In order to realize network slicing in NGMN, infrastructure providers need to perform efficient resource management to meet the QoS and QoE requested by different slices. To this end, various resource management mechanisms have been investigated in the literature, including (i) admission control, (ii) resource allocation, (iii) resource scheduling, and (iv) resource orchestration [15]. In the first and second problems which are the main focus of this chapter, the decision on whether a requested network slice is accepted or rejected according to various predefined policies and assigning resources to the admitted requests are addressed. Authors in [16] focus on developing an admission control scheme for effective allocation of network resources for sliced 5G network considering inter-slice and intra-slice priorities, to improve QoE, enhance resource utilization efficiency and improve the UEs' throughput. In [17], the E2E resource orchestration problem of 5G networks is addressed. In [18], the resource allocation problem in 5G networks leveraging both network slicing and MEC is studied. However, all of the above works assume that the demand for different slices is known in advance and optimize the resource allocation or admission control problem given this unrealistic assumption. In practice, however, the task of accurately estimating future traffic behavior is challenging due to the dynamic nature of NGMN services [19]. Authors in [20] propose an energy-efficient solution for the joint online admission control and resource allocation problem using Γ-Robustness concept to consider the uncertainties in computation and communication demands. The main drawback of this method is that they only consider the core network slicing problem and ignore the correlation between demand for different slices. Similarly, a robust model in proposed in [21] for the network slice design problem under demand uncertainty. In this work, the authors consider both uncorrelated and correlated

traffic case which resembles the spatial demand correlations, but they only study the core network design problem. Moreover, they do not consider delay constraints and the differentiation between different slices based on delay requirements. In this chapter, we study the resource allocation problem of E2E mobile network slicing with demand uncertainties and consider both traffic and delay as QoS requirements.

2.3 Background: Stochastic Programming

In this section, we provide an overview of stochastic programming based on [14]. *Stochastic linear programs* (SLP) are linear programs (with continuous decision variables) in which some problem data are considered to be uncertain. The data with uncertainty can be represented as random variables, with accurate probabilistic description given as probability measures. Since the value of the random variables are known only after the random experiment, the set of optimization decision variables in SLP is divided into two groups: (i) first-stage decision variables, and (ii) second-stage decision variables. While the first-stage variables are determined before the random experiment, the second-stage decisions are taken after the random experiment becomes known. Let $\boldsymbol{x} \in \mathbb{R}^{n_1 \times 1}$ and $\boldsymbol{\xi} = \boldsymbol{\xi}(\omega)$ denote the vector of first-stage optimization variables and data random variables respectively with $\omega \in \Omega$ representing a random event. The vector of second-stage decision variables is denoted by $\boldsymbol{y}(\omega, \boldsymbol{x})$. We assume that the probability distribution $\mathcal{F}$ on Ω is known in the first stage. The classical two-stage SLP with *fixed recourse* is formulated as

follows:

$$minimize \quad \boldsymbol{c}^T\boldsymbol{x} + \mathbb{E}_{\mathcal{F}}[min\ \boldsymbol{q}(\omega)^T\boldsymbol{y}(\omega,\boldsymbol{x})] \tag{2.1}$$

$$s.t. \quad A\boldsymbol{x} = \boldsymbol{b} \tag{2.2}$$

$$T(\omega)\boldsymbol{x} + W\boldsymbol{y}(\omega,\boldsymbol{x}) = \boldsymbol{h}(\omega) \tag{2.3}$$

$$\boldsymbol{x}, \boldsymbol{y}(\omega,\boldsymbol{x}) \geq \boldsymbol{0}. \tag{2.4}$$

where $\boldsymbol{c} \in \mathbb{R}^{n_1}, \boldsymbol{b} \in \mathbb{R}^{m_1\times 1}, A \in \mathbb{R}^{m_1\times n_1}$. When the random event ω is realized $T(\omega) \in \mathbb{R}^{m_2\times n_2}, \boldsymbol{h}(\omega) \in \mathbb{R}^{m_2\times 1}, \boldsymbol{q}(\omega) \in \mathbb{R}^{n_2\times 1}$ become known. In particular $\boldsymbol{\xi}(\omega) = (\boldsymbol{q}(\omega), \boldsymbol{h}(\omega), T_1(\omega), \ldots, T_{m_2}(\omega))$ represents the random data vector where $T_1(\omega), \ldots, T_{m_2}(\omega)$ are the rows of $T(\omega)$. The objective function (2.1) contains a deterministic term $\boldsymbol{c}^T\boldsymbol{x}$ and the expectation of the term $\boldsymbol{q}(\omega)^T\boldsymbol{y}(\omega,\boldsymbol{x})$ over all realizations of the random event ω. For a given ω, let $Q(\boldsymbol{x},\boldsymbol{\xi}(\omega))$ denote the value of the second-stage problem, defined as:

$$Q(\boldsymbol{x},\boldsymbol{\xi}(\omega)) = min_{\boldsymbol{y}\geq 0}\{\boldsymbol{q}(\omega)^T\boldsymbol{y} | W\boldsymbol{y} = \boldsymbol{h}(\omega) - T(\omega)\boldsymbol{x}\} \tag{2.5}$$

Moreover, the second-stage expected value function is defined as:

$$\mathcal{Q}(\boldsymbol{x}) = \mathbb{E}_{\mathcal{F}}[Q(\boldsymbol{x},\boldsymbol{\xi}(\omega))] \tag{2.6}$$

Then, the *deterministic equivalent program* (DEP) corresponding to (2.1)-(2.4) is defined as:

$$min \quad \boldsymbol{c}^T\boldsymbol{x} + \mathcal{Q}(\boldsymbol{x}) \tag{2.7}$$

$$s.t. \quad A\boldsymbol{x} = \boldsymbol{b} \tag{2.8}$$

$$\boldsymbol{x} \geq \boldsymbol{0} \tag{2.9}$$

The DEP formulation illustrates the major difference between deterministic linear programs and a SLP and its difficulty. Given $\mathcal{Q}(\boldsymbol{x})$, a SLP is an ordinary nonlinear program. However, the computation of $\mathcal{Q}(\boldsymbol{x})$ requires the evaluation of $Q(\boldsymbol{x}, \boldsymbol{\xi}(\omega))$ for all $\boldsymbol{x}$ and ω which is not possible in practice. The problem becomes more challenging considering integer first-stage or second-stage variables, i.e. $\boldsymbol{x} \in \mathbb{Z}_+^{n_1}$, $\boldsymbol{y} \in \mathbb{Z}_+^{n_2}$. In the next section, we present our system model for the problem of network slicing with demand uncertainties followed by our formulation based on two-stage stochastic mixed-integer programming.

2.4 System Model and Problem Formulation

We consider the resource provisioning problem of mobile network slicing across both RAN and core components. In this section, the network and slice models are presented followed by problem formulation.

2.4.1 Infrastructure Network Model

The infrastructure (a.k.a. substrate network) encompasses next-generation nodeBs (gNBs) supporting 5G NR and core nodes implementing 5GC components. In a mobile network empowered

by NFV and SDN, the substrate nodes are general purpose servers that host different VNFs. Fig 2.3 illustrates such an infrastructure. Let $G = (V, E)$ denote the substrate network graph, where V is the set of substrate nodes and E denotes the set of infrastructure links. The set of gNB and non-gNB nodes are denoted by V_{gNB} and V_{-gNB}, respectively, and $V = V_{gNB} \cup V_{-gNB}$. Each gNB is characterized by its maximum supported traffic which is computed based on its available bandwidth (physical resource blocks or PRBs) and MIMO antenna configuration. Non-gNBs are general-purpose servers providing essential capabilities for running core VNFs. A substrate node is characterized by its available CPU, storage, and RAM resources. We define the set of resources as $\mathcal{T} = \{CPU, STO, RAM\}$, and the vector $W_j = (W_j^{CPU}, W_j^{STO}, W_j^{RAM})$ representing the capacity of the substrate node $j \in V$. These amounts correspond to the initial capacity of the node reduced by the amount allocated to previously accepted concurrent requests. Moreover, the cost of each infrastructure node j being used by a VNF consists of two parts: (i) a fixed part C_j paid for each VNF deployed on node j, (ii) a variable part associated with the per-unit usage of each resource which depends linearly on the amount of the resource consumed by a VNF. The per-unit cost corresponding to CPU, storage and memory of node j are denoted by C_j^{CPU}, C_j^{STO} and C_j^{RAM}.

We assume that the available spectrum of gNB $j \in V_{gNB}$ divided into a set of W_j^r PRBs can be allocated to different users of different slices [22]. Furthermore, the set of infrastructure link $E = E_{FH} \cup E_{BH}$ where E_{FH} and E_{BH} stand for the links between gNB nodes and the core network and the remaining link, respectively. Each substrate link $e \in E$ has a bandwidth capacity and propagation delay denoted by W_e^{BW} and τ_e^{PRO} in respective order. Moreover, a per-unit bandwidth cost is associated with the infrastructure link e denoted by C_e^{BW}. In order to meet the traffic transmission between two VNFs of a network slice, we consider a set of substrate

paths denoted by $\mathcal{P}$. Let $\mathcal{P}(i \rightarrow j)$ denote the set of substrate path between i and j. Therefore, $\mathcal{P} = \cup_{i,j \in V, i \neq j} \mathcal{P}(i \rightarrow j)$.

We consider an environment in which UEs send their request for different services to SPs. For the sake of simplicity, we assume that each UE requests at most one service. This can be readily extended to the case with multiple requests per UE by using UE duplication. Each service is realized through network slices managed by SPs. The SPs meet their customers requested services by dedicating network slices with specified QoS requirements deployed on the substrate network which is owned and managed by InPs. Let $\mathcal{U}$ denote the set of UEs distributed across the considered geographical area. We assume that each UE is served by a gNB according to the nearest association rule. In the nearest association rule, a UE is assigned to the gNB achiving the maximum SNR.

2.4.2 Network Slice Model

In this section, we present the network slice model. We assume that a slice request is modeled as one or multiple SFCs, each consisting of a number of VNFs (e.g. gNB , AFM, UPF, SMF) and virtual links (VLs) between them. The set of network slices is represented by $\mathcal{K} = \{1, ..., K\}$. Let the undirected graph $G'_k = (V'_k, E'_k)$ represent the kth slice SFC. Receiving a number of slice/SFC requests, the InP embeds them to the substrate network and allocates sufficient amount of different resources, while making sure that the requirements of the SFCs are satisfied and the network resources are used in an efficient manner. Similar to the substrate network, we assume that the nodes of slice k SFC consist of two types of nodes, gNB, and non-gNB. Therefore, $V'_k = V'_{k,gNB} \cup V'_{k,-gNB}$ where $V'_{k,gNB}$ and $V'_{k,-gNB}$ correspond to the set of

gNBs and non-gNB functions of slice k. Moreover, $E'_k = E'_{k,FH} \cup E'_{k,BH}$ where $E'_{k,FH}$ and $E'_{k,BH}$ denote the fronthaul and backhaul VLs, respectively. We assume that each instance of the SFC corresponding to slice k has QoS requirements expressed as network-level UE throughput and maximum E2E tolerable latency. In order to guarantee the performance requirement of different slices, we assume that the network-level performance metric of a slice is translated to cell-level radio resource requirement [23]. Let $\underline{R}_j^k$ denote the average required number of PRBs for slice $k \in \mathcal{K}$ users in the geographical region covered by gNB $j \in V_{gNB}$. The value of $\underline{R}_j^k$ depends on the overall cell load, mobility patterns, antenna design, channel condition, modulation and coding scheme (MCS), and slice performance requirements. The details of such a translation mechanism is out of the scope of this chapter and solutions such as [24], [25] are proposed for the translation algorithm. In accordance with the slice management architecture of [26], the overall network slice performance is decomposed into core and RAN domain requirements. Thus, $\underline{R}_j^k$ captures the RAN throughput and delay performance requirements. Moreover, the value of $\underline{R}_j^k$ is dynamically determined as the condition of different affecting parameters vary. Slice isolation is a critical feature of *network slicing* which prevents any impact on the performance of a network slice as the condition in other slices changes. To guarantee slice isolation, we use an SLA parameter $\overline{R}^k$ defined as the maximum allowed number of PRBs for slice $k \in \mathcal{K}$. We assume that the traffic flow of each $e' \in E'_k$ is routed through substrate paths (single or multi-hop). Therefore, we represent the set of substrate paths considered for slice k edges by $\mathcal{P}_k$, where $\mathcal{P}_k \subseteq \mathcal{P}$.

Let $\mathcal{P}'_k$ denote the set all paths for slice $k \in \mathcal{K}$. The $\mathcal{P}'_k$'s constituting paths are for either CP or UP data flows. For instance, the path gNB-AMF-SMF of the network slice example shown in Fig. 2.2 is a CP path while gNB-UPF is a UP path. Let $\mathcal{P}'_{k,UP}$ and $\mathcal{P}'_{k,CP}$ denote the set of paths in G'_k which are composed of UP and CP functions. Thus $\mathcal{P}'_k = \mathcal{P}'_{k,UP} \cup \mathcal{P}'_{k,CP}$. In terms

of QoS requirements, each network slice is characterized by its maximum tolerable UP and CP latencies denoted by d_k^{UP}, d_k^{CP}.

In order to guarantee the demands of all users requesting different network slices, multiple instances of each slice may be needed to be deployed. Let $\mathcal{U}_k \subseteq \mathcal{U}$ denote the set of UEs requesting slice $k \in \mathcal{K}$. We define $u_j^{k,i}$ to be equal to 1 if user i of slice k ($i \in \mathcal{U}_k$) is covered by gNB $j \in V_{gNB}$ and 0 otherwise. Moreover, the parameter I_k indicates the number of instances of slice k needed to be deployed in order to support the demand for slice k. In addition, each VNF instance of the node $j' \in V_k'$ requires CPU, storage, and RAM resources. The required amount of these resources depends on the number of UEs and the resource sharing factor which determines how much of the workload of a VNF can be shared among multiple UEs. Let the vector $R_{j'} = (R_{j'}^{CPU}, R_{j'}^{STO}, R_{j'}^{RAM})$ denote the required per-unit amount of CPU, STO and RAM resources for VNF j'. Similarly, each VL $e' \in E_k'$ is characterized by its per-unit bandwidth requirement $R_{e'}^{BW}$ to meet the demand for data transmission between the two end-points VNFs of e'. In addition, the number of resources allocated to the VNFs and VLs of a network slice must be dimensioned according to the traffic demand that is going through them [27]. We define χ_k^t to be the scaling factor of resource ν ($\nu \in \mathcal{T}$ for VNFs and $\nu = BW$ for VLs) for slice k which depends on the slice load and sharing factor. Therefore, χ_k^ν is proportional to $u_j^{k,i}$ and I_k.

During the resource provisioning phase of network slices, the true values of $u_j^{k,i}$ and I_k are not known. However, given the historical data on the demand and mobility patterns of UEs, the SP and InP are capable of forecasting the statistical characteristics of $u_v^{k,i}$ and I_k. Let $\{\mathcal{F}_k(x,y), k \in \mathcal{K}\}$ denote the spatial demand density function of network slices across the considered geographical area. A key feature of the proposed resource provisioning approach is that the demand distribution function can be very generic and different methods can be used

Table 2.1: System Model Parameters

Network Parameters	Description
$G = (V, E)$	Substrate network graph
V_{gNB}, V_{-gNB}	Set of substrate gNB and non-gNB nodes
E_{FH}, E_{BH}	Set of fronthaul and backhaul links
$\mathcal{P}_k$	Set of substrate paths considered for slice k
W_j^r	Available spectrum of gNB j
$\mathcal{U}_k, U_k$	Set and number of users of slice k
$W_j^{CPU,STO,RAM}$	CPU, Storage and RAM capacities of the substrate node j
W_e^{BW}	BW capacity of the substrate link e
Slice Parameters	**Description**
$\mathcal{K}, K$	Set and number of requested slices
$G'_k = (V'_k, E'_k)$	The kth slice SFC graph
$V'_{k,gNB}, V'_{k,-gNB}$	Set of virtual RAN and core VNFs
$E'_{k,FH}, E'_{k,BH}$	Set of fronthaul and backhaul virtual links
$\mathcal{P}'_k$	Set of slice k E2E paths
$\overline{R}_j^k$	Slice k maximum allowed radio resource in gNB j
$\underline{R}_j^k$	Slice k radio resource requirement in gNB j
$R_{j'}^{CPU,STO,RAM}$	CPU, Storage and RAM resource requirements of the VNF j'
$R_{e'}^{BW}$	BW requirement of the virtual link e'
d_k^{UP}, d_k^{CP}	UP and CP latency requirement of slice k

to predict $\mathcal{F}$ functions. Moreover, the usage of spatial probability distributions can potentially resemble a flash-crowd or a traffic surge where some spatial correlation exists between different traffic demands for a network slice. There exist also different methods to obtain the density functions. For instance, the prior art in [28], [29], [30], [31] have proposed different ML methods such as GMM, Holt-Winters forecasting, neural networks, and maximum likelihood approach to predict traffic demands that are utilized in different network resource management problems.

A summary of the used notations in this chapter is provided in Table 2.1.

2.4.3 Problem Formulation

Given the substrate and slice graph models, we formulate the problem of E2E network slicing under demand uncertainty as a two-stage SMIP. We define the following decision variables for the problem formulation:

- A set of binary variables $\boldsymbol{x}_k$, where $x_j^{k,j'}$ is equal to 1 if the VNF j' of the kth slice is placed on the substrate node j.
- A set of binary variables $\boldsymbol{y}_k$, where $y_p^{k,e'}$ is equal to 1 if the VL e' of the kth slice is mapped to substrate path $p \in \mathcal{P}_k$.
- A set of continuous decision variables $\boldsymbol{z}_k$, where z_j^k represents the fraction of gNB j spectrum allocated to slice k UEs.

It is important to note that the value of the vector $\boldsymbol{z} = (z_1^1, ..., z_{|V_{gNB}|}^1, ..., z_1^K, ..., z_{|V_{gNB}|}^K)$ represents the RAN slicing policy defined as the fraction of each cell available radio resource that is assigned to each slice. For instance, in the case of having three network slices, the resulted RAN slice allocation for a gNB can be $20\%, 50\%$, and 30%. Once this spectrum assignment policy is determined, the actual radio resource allocation in terms of the scheduled PRBs for the users of each slice to fulfill the RAN performance requirement of the corresponding slice is obtained from a separate problem that is out of the scope of this chapter. In this regard, different solutions have been proposed in the literature. For instance, authors in [32] proposed a RAN resource allocation solution given the spectrum assignment policy, with the objective of minimizing the interference between different mobile virtual network operators.

We model the resource provisioning problem for the requested slices as a two-stage SMIP,

described in the following sections. We refer to this method as stochastic network slicing resource provisioning, i.e. **SNSR**.

2.4.3.1 First-Stage Problem

The first-stage objective in **SNSR** is the minimization of the total provisioning cost which consists of node and link deployment costs for all slices. Let $\mathcal{C}_k^N$ and $\mathcal{C}_k^L$ denote the node and link costs corresponding to slice k, respectively. Consequently, we have:

$$\mathcal{C}_k^N(\boldsymbol{x}_k) = \sum_{j \in V} \sum_{j' \in V_k'} C_j^k x_j^k + \sum_{j \in V} \sum_{j' \in V_k'} \sum_{\nu \in \mathcal{T}} \gamma_\nu R_{j'}^\nu C_j^\nu x_j^{k,j'} \tag{2.10}$$

$$\mathcal{C}_k^L(\boldsymbol{y}_k) = \sum_{p \in \mathcal{P}} \sum_{e \in p} \sum_{e' \in E_k'} \gamma_{BW} R_{e'}^{BW} C_e^{BW} y_p^{k,e'} \tag{2.11}$$

where

$$x_j^k = \begin{cases} 1 & \text{if } \sum_{j' \in V_k'} x_j^{k,j'} \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

Hence, the first-stage objective is:

$$\mathcal{C}_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k \in \mathcal{K}} \mathcal{C}_k^N(\boldsymbol{x}_k) + \sum_{k \in \mathcal{K}} \mathcal{C}_k^L(\boldsymbol{y}_k) \tag{2.13}$$

where $\gamma_\nu, \nu \in \mathcal{T} \cup \{BW\}$ are weight constants to balance between different objective terms of (2.10) and (2.11). In a valid network slicing solution, each gNB VNF is placed at substrate gNB nodes only and each non-gNB VNF runs on a substrate non-gNB node. Thus, the resulting VNFs

deployment must satisfy the following constraints:

$$\sum_{j \in V_{-gNB}} x_j^{k,j'} = 1, \ \forall k \in \mathcal{K}, j' \in V'_{k,-gNB} \tag{2.14}$$

$$x_j^{k,j'} = 0, \ \forall k \in \mathcal{K}, j \in V_{-gNB}, j' \in V'_{k,gNB} \tag{2.15}$$

The flow conservation constraints are guaranteed by equations (2.16) and (2.17).

$$\sum_{\substack{p \in \mathcal{P}_k(i \to j) \\ q \in \mathcal{P}_k(j \to i) \\ j \in V}} y_p^{k,e'} - y_q^{k,e'} = x_i^{k,i'} - x_i^{k,j'}, \ \forall k \in \mathcal{K}, e' \in E'_{k,BH}, src(e') = i', dst(e') = j' \tag{2.16}$$

$$\sum_{\substack{p \in \mathcal{P}_k(i \to j) \\ q \in \mathcal{P}_k(j \to i) \\ j \in V}} y_p^{k,e'} - y_q^{k,e'} = (x_i^{k,i'} - x_i^{k,j'}) I_k, \ \forall k \in \mathcal{K}, i \in V_{-gNB}, e' \in E'_{k,FH}, src(e') = i', dst(e') = j' \tag{2.17}$$

Finally, the domain constraints are as follows:

$$x_j^{k,j'}, y_p^{k,e'}, \in \{0,1\} \ \forall k \in \mathcal{K}, j \in V, j' \in V'_k, p \in \mathcal{P}_k, e' \in E'_k \tag{2.18}$$

2.4.3.2 Second-Stage Problem

The goal of the second-stage problem is to minimize the total cost of gNBs' radio resources allocated to different network slices. The second-stage objective is:

$$\mathcal{C}_2(\boldsymbol{z}) = \sum_{k \in \mathcal{K}} \sum_{j \in V_{gNB}} C_j^k z_j^k W_j^r \tag{2.19}$$

The processing delay of VNF $j' \in V_k'$ on an infrastructure node is represented by $\tau^{k,j'}$. Given the VL mapping decision variables $y_p^{k,e'}$, the data transmission latency corresponding to the infrastructure edge $e \in E$ given its aggregated load is calculated as:

$$\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : e \in p} \sum_{e' \in E_k'} y_p^{k,e'} \frac{\chi_k^{BW} R_{e'}^{BW}}{W_e^{BW}}$$

Thus, the latency of the VL $e' \in E_k'$ mapped to the path $p \in \mathcal{P}_k$ is:

$$\sum_{e \in p} \left[\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : e \in p} \sum_{e' \in E_k'} y_p^{k,e'} \frac{\chi_k^{BW} R_{e'}^{BW}}{W_e^{BW}} \right]$$

Therefore, the latency of VL e'' is calculated as follows:

$$\sum_{p \in \mathcal{P}_k} y_p^{k,e''} \left[\sum_{e \in p} \left[\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : e \in p} \sum_{e' \in E_k'} y_p^{k,e'} \frac{\chi_k^{BW} R_{e'}^{BW}}{W_e^{BW}} \right] \right]$$

Thus, the maximum UP/CP latency of slice k instances are guaranteed to be lower than the

required latency d_k^{UP}/d_k^{CP} in the following constraint:

$$\sum_{e'' \in p'} \left[\sum_{p \in \mathcal{P}_k} y_p^{k,e''} \left[\sum_{e \in p} \left[\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : e \in p} \sum_{e' \in E'_k} y_p^{k,e'} \frac{\chi_k^{BW} R_{e'}^{BW}}{W_e^{BW}} \right] \right] \right] + \sum_{j' \in p'} \left[\sum_{j \in V} x_j^{k,j'} \tau^{k,j'} \right] \leq d_k^{UP/CP},$$

$$\forall p' \in \mathcal{P}'_{k,UP/CP}, k \in \mathcal{K}$$

The above constraint ensures that the maximum latency of each slice is less than its tolerable latency by enforcing the inequality for all paths of G'_k. By doing so, the latency over the *critical path* defined as the path inducing the maximum latency among all paths is enforced to be less than the tolerable latency. This constraint is a quadratic constraint. Thus, we linearize it by introducing a set of additional continuous variables $\eta_p \geq 0$ and $\eta_p^{k,e'} \geq 0$, defined as the latency of path $p \in \mathcal{P}$ and the latency of VL $e' \in E'_k$ of slice k on path p, respectively. Therefore, the latency constraint is converted to the following set of constraints:

$$\sum_{e \in q} \left[\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : e \in p} \sum_{e' \in E'_k} y_p^{k,e'} \frac{\chi_k^{BW} R_{e'}^{BW}}{W_e^{BW}} \right] - \eta_q = 0, \ \forall q \in \mathcal{P} \tag{2.20}$$

$$\zeta y_p^{k,e'} + \eta_p - \eta_p^{k,e'} \leq \zeta, \ \forall p \in \mathcal{P}_k, e' \in E'_k, k \in \mathcal{K} \tag{2.21}$$

$$\sum_{e' \in p'} \sum_{p \in \mathcal{P}_k} \eta_p^{k,e'} + \sum_{j' \in p'} \sum_{j \in V} x_j^{k,j'} \tau^{k,j'} \leq d_k^{UP/CP}, \ \forall p' \in \mathcal{P}'_{k,UP/CP}, k \in \mathcal{K} \tag{2.22}$$

where ζ is a big constant. We also add the term $\epsilon \mathcal{D}$ to the objective function where $\mathcal{D}(\boldsymbol{\eta}) = \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k} \sum_{e' \in E'_k} \eta_p^{k,e'}$ and ϵ is a very small value in order to make sure that the main objective of minimizing the resource provisioning cost is not affected by adding $\mathcal{D}$. By doing so, it is ensured that the value of $\eta_p^{k,e'}$ is positive if and only if $y_p^{k,e'} = 1$.

Given the network-level performance translation to cell-level metric $\underline{R}_j^k$, the RAN slice

resource allocation is enforced by constraints (2.23).

$$\sum_{i=1}^{U_k} u_j^{k,i} \underline{R}_j^k \leq x_j^{k,j'} z_j^k W_j^r, \ \forall k \in \mathcal{K}, j \in V_{gNB}, j' \in V'_{k,gNB} \tag{2.23}$$

Constraints (2.23) is nonlinear due to the term $x_j^{k,j'} z_j^k W_j^r$. We use the big-M method and convert (2.23) to the following:

$$\sum_{i=1}^{U_k} u_j^{k,i} \underline{R}_j^k \leq z_j^k W_j^r + (1 - x_j^{k,j'})M, \ \forall k \in \mathcal{K}, j \in V_{gNB}, j' \in V'_{k,gNB} \tag{2.24}$$

where $M \in \mathbb{R}$ is a large number. Moreover, in order to satisfy the slice isolation constraint and given the maximum allowable number of PRBs allocated to each slice at each gNB, $\overline{R}_j^k$, we formulate the slice isolation constraint using the following inequality:

$$z_j^k W_j^r \leq \overline{R}_j^k x_j^{k,j'}, \ \forall k \in \mathcal{K}, j \in V_{gNB}, j' \in V'_{k,gNB} \tag{2.25}$$

Moreover, for each infrastructure node and link, we have capacity constraints as follows:

$$\sum_{k \in \mathcal{K}} \sum_{j' \in V'_k} x_j^{k,j'} R_{j'}^{\nu} \chi_k^{\nu} \leq W_j^{\nu}, \ \nu \in \mathcal{T}, \forall j \in V \tag{2.26}$$

$$\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : e \in p} \sum_{e' \in E'_k} y_p^{k,e'} R_{e'}^{BW} \chi_k^{BW} \leq W_e^{BW}, \ \forall e \in E \tag{2.27}$$

Furthermore, the capacity constraint of gNB $j \in V_{gNB}$ is enforced by the following inequality:

$$\sum_{k=1}^{K} z_j^k = x_j^{k,j'}, \ \forall j \in V_{gNB}, j' \in V'_{k,gNB} \tag{2.28}$$

The gNB placement constraint is enforced by equation (2.29).

$$x_j^{k,j'} \geq \mathbb{I}\{\sum_{i=1}^{U_k} u_j^{k,i} \geq 1\},\ \forall k \in \mathcal{K}, j \in V_{gNB}, j' \in V'_{k,gNB} \tag{2.29}$$

Finally, the domain constraints of the second-stage problem are:

$$\begin{aligned} & z_j^k \in [0,1],\ \forall k \in \mathcal{K}, j \in V_{gNB} \\ & \eta_p^{k,e'}, \eta_p \geq 0,\ \forall k \in \mathcal{K}, e' \in E'_k, p \in \mathcal{P}_k \end{aligned} \tag{2.30}$$

2.4.3.3 E2E Network Slice Resource Provisioning Problem (**SNSR**)

The E2E network slicing problem with demand uncertainties is formulated as a two-stage SMIP presented below:

$$minimize \quad \phi_1 \mathcal{C}_1(\boldsymbol{x}, \boldsymbol{y}) + \phi_2\, \mathbb{E}_{\boldsymbol{\xi}}[min\, \mathcal{C}_2(\boldsymbol{z}) + \epsilon \mathcal{D}(\boldsymbol{\eta})] \tag{2.31}$$

$$\text{s.t.} \qquad (2.14) - (2.30) \tag{2.32}$$

The objective (2.31) minimizes the summation of the core slice provisioning cost and the expectation of the second stage objective which is the RAN slice provisioning cost. ϕ_1, ϕ_2 are constant parameters that determine the balance between the objectives of the first and second stage problems. The above problem is interpreted as follows:

- The InP must provisions the core network resources for different network slices, represented

by the decision variables $\boldsymbol{x}, \boldsymbol{y}$, before the actual value of the random vector

$$\boldsymbol{\xi} = (\mathcal{U}_1, \ldots, \mathcal{U}_K, u_1^{1,1}, \ldots, u_1^{1,U_1}, \ldots, u_{|V_{gNB}|}^{K,\mathcal{U}_K}, \chi_1^{\nu}, \ldots, \chi_K^{\nu}, \nu \in \mathcal{T} \cup \{BW\})$$

is known.

- After the value of the random vector $\boldsymbol{\xi}$ becomes known, the RAN resource provisioning and delay decision variables, denoted by $\boldsymbol{z}, \boldsymbol{\eta}$, are determined.

The expectation term in (2.31) requires an integration over the high-dimensional random vector $\boldsymbol{\xi}$. To tackle this challenge, we use the sample average approximation technique and replace the expectation in (2.31) with the sample average. The samples can be viewed as historical observed data or can be generated using Monte Carlo sampling techniques.

2.4.4 Deterministic Equivalent Reformulation

In this section, we introduce the deterministic reformulation of the modeled **SNSR** and then propose a solution based on the sample average approximation (SAA). The SAA method is an approach for solving stochastic optimization problems by using Monte Carlo simulation. Suppose that H i.i.d observations of the random variables $u_j^{k,i}$, U_k, and χ_k^{ν} are available from sampling $\mathcal{F}$. We denote them by $\tilde{u}_{j,h}^{k,i}$,$\tilde{U}_{k,h}$, $\tilde{\chi}_{k,h}^{\nu}$, $h = 1, \ldots, H$. For each realization, we define a separate set of second-stage decision variables $z_{j,h}^{k}, \eta_{p,h}, \eta_{p,h}^{k,e'}$. Thus, we convert **SNSR** to its

sampled deterministic equivalent problem, defined as follows:

$$minimize \quad \phi_1 \mathcal{C}_1(\boldsymbol{x}, \boldsymbol{y}) + \phi_2 \sum_{h=1}^{H} \frac{1}{H} \left(\mathcal{C}_2(\boldsymbol{z}_h) + \epsilon \mathcal{D}(\boldsymbol{\eta}_h) \right) \tag{2.33}$$

$$s.t. \qquad (2.14) - (2.18) \tag{2.34}$$

$$\sum_{e \in q} \left[\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : e \in p} \sum_{e' \in E'_k} y_p^{k,e'} \frac{\tilde{\chi}_{k,h}^{BW} R_{e'}^{BW}}{W_e^{BW}} \right] - \eta_{q,h} = 0, \ \forall q \in \mathcal{P}, h \in \mathcal{H} \tag{2.35}$$

$$\zeta y_p^{k,e'} + \eta_{p,h} - \eta_{p,h}^{k,e'} \leq \zeta, \ \forall p \in \mathcal{P}_k, e' \in E'_k, k \in \mathcal{K}, h \in \mathcal{H} \tag{2.36}$$

$$\sum_{e' \in p'} \sum_{p \in \mathcal{P}_k} \eta_{p,h}^{k,e'} + \sum_{j' \in p'} \sum_{j \in V} x_j^{k,j'} \tau^{k,j'} \leq d_k^{UP/CP}, \ \forall p' \in \mathcal{P}'_{k,UP/CP}, k \in \mathcal{K}, h \in \mathcal{H} \tag{2.37}$$

$$\sum_{k \in \mathcal{K}} \sum_{j' \in V'_k} x_j^{k,j'} R_{j'}^{\nu} \tilde{\chi}_{k,h}^{\nu} \leq W_j^{\nu}, \ \forall \nu \in \mathcal{T}, j \in V, h \in \mathcal{H} \tag{2.38}$$

$$\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : e \in p} \sum_{e' \in E'_k} y_p^{k,e'} R_{e'}^{BW} \tilde{\chi}_{k,h}^{BW} \leq W_e^{BW}, \ \forall e \in E, h \in \mathcal{H} \tag{2.39}$$

$$\sum_{i=1}^{\tilde{U}_{k,h}} \tilde{u}_{j,h}^{k,i} \underline{R}_j^k \leq z_{j,h}^k W_j^r + (1 - x_j^{k,j'}) M, \ \forall k \in \mathcal{K}, j \in V_{gNB}, j' \in V'_{k,gNB}, h \in \mathcal{H} \tag{2.40}$$

$$z_{j,h}^k W_j^r \leq \overline{R}_j^k x_j^{k,j'}, \ \forall k \in \mathcal{K}, j \in V_{gNB}, j' \in V'_{k,gNB}, h \in \mathcal{H} \tag{2.41}$$

$$\sum_{k=1}^{K} z_{j,h}^k \leq 1, \ \forall j \in V_{gNB}, h \in \mathcal{H} \tag{2.42}$$

$$x_j^{k,j'} \geq \mathbb{1}\{\sum_{i=1}^{\tilde{U}_{k,h}} \tilde{u}_{j,h}^{k,i} \geq 1\}, \ \forall k \in \mathcal{K}, j \in V_{gNB}, j' \in V'_{k,gNB}, h \in \mathcal{H} \tag{2.43}$$

$$z_{j,h}^k \in [0,1], \ \forall k \in \mathcal{K}, j \in V_{gNB}, h \in \mathcal{H} \tag{2.44}$$

We refer to the above problem as **DET_SNSR**$(\mathcal{H})$. The first term in the objective function (2.33) is the same as the first term of the objective function (2.31). The second term is the weighted average of the second stage objective function over the H realized samples of the random variables χ_k, $u_{i,j}^k$ and U_k. Constraints (2.35)-(2.44) replace their counterparts. As the

sample size H increases and under certain mild conditions as discussed in [33], the solution to **DET_SNSR**$(\mathcal{H})$ converges to that of the original **SNSR** problem.

2.5 Two-Timescale Resource Allocation Algorithm

We propose a two-timescale resource allocation scheme that utilizes the resource provisioning methodology discussed in Section 2.4. In this approach, the resource allocation problem during the lifecycle of network slices is addressed through two long and short time slots. We assume that the lifecycle of a network slice is divided into a number of long time slots denoted by T. During each long time slot, the demand distribution for network slices is known and fixed. Let $\mathcal{F}^T = \{\mathcal{F}_1^T, \ldots, \mathcal{F}_K^T\}$ denote the demand distribution of K network slices in the long time slot T. Given $\mathcal{F}^T$, we solve the resource provisioning problem proposed in Section 2.4, **DET_SNSR**$(\mathcal{H})$. We further divide each long time slot into N_T short time slots. At each short time scale t, an observation of the demands is realized characterized by values of $\hat{\boldsymbol{\xi}} = (\hat{\mathcal{U}}_k, \hat{u}_j^{k,i}, \hat{\chi}_k^{\nu}, k \in \mathcal{K}, i \in \hat{\mathcal{U}}_k, j \in V_{gNB}, \nu \in \mathcal{T} \cup \{BW\})$. We define a new variable $\sigma_j^k, k \in \mathcal{K}, j \in V_{gNB}$ as the fraction of unsupported demand requested for slice k in gNB j. Given the observed realization at each short time slot, and the solution to the first-stage problem $\boldsymbol{x}, \boldsymbol{y}$, we adjust the RAN resource provisioning decision by solving the following optimization problem:

$$minimize \quad \sum_{k\in\mathcal{K}}\sum_{j\in V_{gNB}} \theta(\mathcal{C}_2(\boldsymbol{z})) + (1-\theta)\sigma_j^k \tag{2.45}$$

$$(1-\sigma_j^k)(\sum_{i=1}^{\hat{U}_k}\hat{u}_j^{k,i})\underline{R}_j^k \leq z_j^k W_j^r + (1-x_j^{k,j'})M,\ \forall k\in\mathcal{K}, j\in V_{gNB}, j'\in V'_{k,gNB} \tag{2.46}$$

$$\left(\frac{\sum_{j\in V_{gNB}}(1-\sigma_j^k)\sum_{i=1}^{\hat{U}_k}\hat{u}_j^{k,i}}{\hat{U}_k}\right)\hat{\chi}_k^\nu \leq \chi_{k,prov}^\nu,\ \forall k\in\mathcal{K}, \nu\in\mathcal{T}\cup\{BW\} \tag{2.47}$$

$$z_j^k W_j^r \leq \overline{R}_j^k x_j^{k,j'},\ \forall k\in\mathcal{K}, j\in V_{gNB}, j'\in V'_{k,gNB} \tag{2.48}$$

$$\sum_{k\in\mathcal{K}} z_j^k \leq 1,\ \forall j\in V_{gNB} \tag{2.49}$$

$$z_j^k, \sigma_j^k \in [0,1],\ \forall k\in\mathcal{K}, j\in V_{gNB} \tag{2.50}$$

We refer to this LP as the radio network slicing resource allocation, denoted by **RNSR**$(\boldsymbol{x}, \boldsymbol{y}, \hat{\boldsymbol{\xi}})$. The objective of (2.45) minimizes the weighted summation of RAN resource allocation cost and total unsupported traffic. The weight parameter $\theta \in [0,1]$ is used to balance between the two objective terms. Constraint (2.46) ensures that the radio resource reservation for a network slice is greater than the supported traffic on each gNB. M is a big positive real value used for the big-M method, enforcing constraint (2.46) for a tuple (j', k, j) only if $x_j^{k,j'} = 1$, i.e. a RAN VNF of slice k is instantiated on gNB j. Constraint (2.47) guarantees the availability of core network resources to support the admitted traffic of each slice. The slice isolation constraint is enforced by (2.48). Constraint (2.49) ensures the capacity constraint for each gNB, and the domain constraints are expressed by (2.50).

Figure 2.4 illustrates our proposed two-timescale resource allocation scheme. Given the demand distribution $\mathcal{F}^T$ for each long time slot, we solve an instance of **DET_SNSR**$(\mathcal{H})$ that

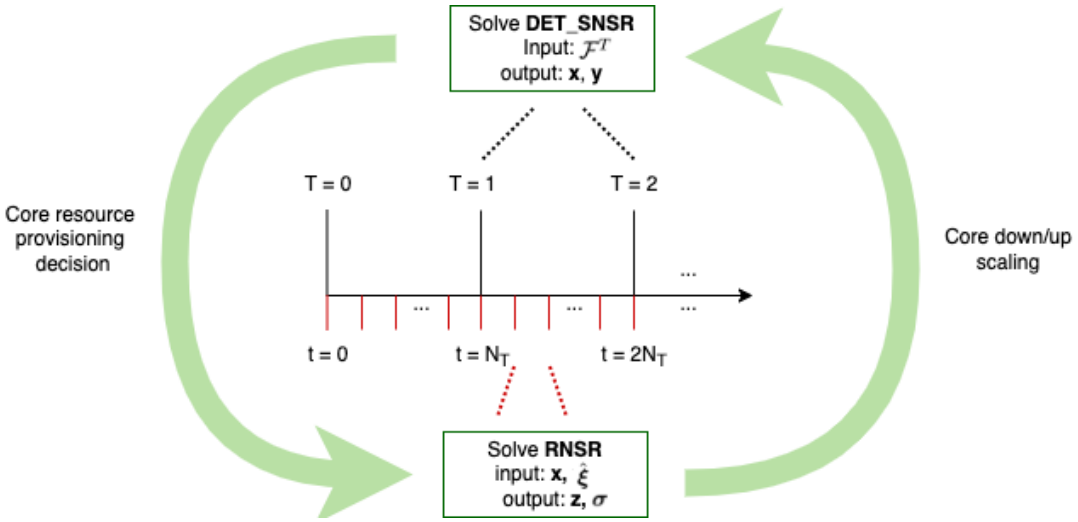


Figure 2.4: Two-Timescale Resource Provisioning Scheme

provides the resource provisioning solution for the core network. Given the core network slicing solution and a realization of the slices' demand points at each short time slot, the $\mathbf{RNSR}(\boldsymbol{x}, \boldsymbol{y}, \hat{\boldsymbol{\xi}})$ problem is solved to obtain the E2E resource allocation solution. The instantaneous demand for different network slices at each short time slot is used to scale down/up the core network slices if needed.

2.6 Numerical Results

In this section, we present the evaluation of the proposed two-timescale methodology through simulations.

2.6.1 Simulation Setup

The evaluation environment is implemented in Java. We consider a mobile network encompassing 7 servers at four levels of hierarchy and 6 gNB nodes connected to the third-level servers as illustrated in Fig. 2.6. L1, L2, L3 and L4 servers have CPU (cycle/s), Storage (GB) and RAM

(GB) capacities of $(72, 144, 288)$, $(36, 72, 144)$, $(18, 36, 72)$, and $(6, 12, 24)$ respectively. The backhaul and fronthaul links have the capacity of 2Gbps and 1Gbps respectively. For the sake of simplicity, the resource scaling factor χ_k^ν is assumed to be the same for all resource types $t \in \mathcal{T}$ and be equal to the number of users requesting slice k. Each VNF requires $(0.1\chi_k, 0.2\chi_k, 0.4\chi_k)$ units of CPU, STO and RAM resources. We use the CPLEX commercial solver for solving the **DET_SNSR** model using the branch-and-bound method. The **RNSR** problem is also solved by CPLEX using the Simplex method. All experiments are carried out on an Intel Xeon processor at 2.3 GHz with 8GB memory.

We assume that the UEs are distributed in a square area of dimension $10\ Km \times 10\ Km$. The demand densities $\mathcal{F}^T$ are obtained by applying bivariate kernel density estimation (KDE) to the synthetic data generated for each slice demand in the considered geographical area. Due to the unavailability of spatially distributed and correlated datasets for network slice demands, we generate synthetic data. We use Matérn cluster point process [34] to generate the UE locations for each slice. In this process, a number of parent points are first generated using a Poisson point process and for each parent point (representing a cluster center), another Poisson point process is used to generate the UE distribution within a radius of the parent point. The clusters of users represent the hot spot area for different network slices. We assume that the demand distributions change hourly and a 24-hour ($T \in \{0, \ldots, 23\}$) periodic demand pattern is considered. This can be adjusted depending on the environmental dynamics such as channel conditions or the user mobility behavior. In other words, a shorter/longer duration for the long times slots can be adopted if the dynamics in the traffic distribution for a slice are fast/slow. Fig. 2.5 illustrates an example of a demand distribution for a network slice obtained by using kernel density estimation method. The simulation parameters for network slices are shown in Table 2.2.

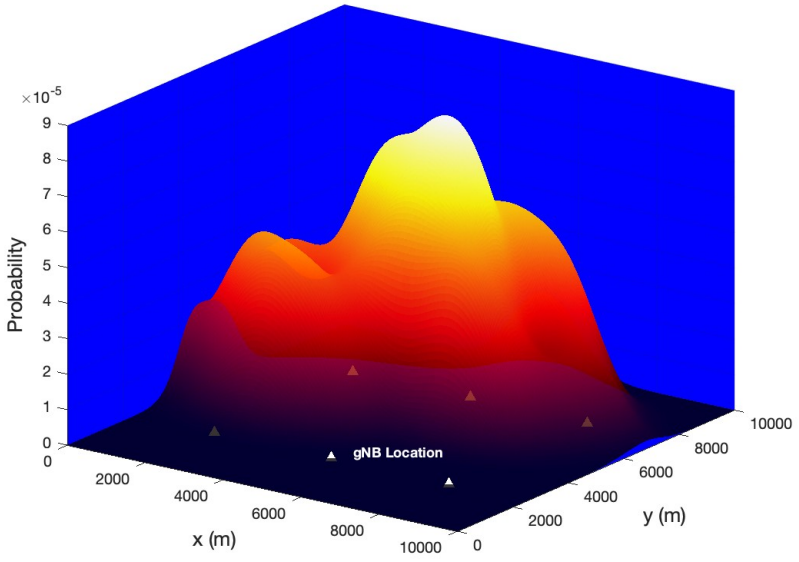


Figure 2.5: An Example of a Slice Demand Probability Distribution

Table 2.2: System Model Parameters

Slice Type	d_k^{UP}/d_k^{CP}	mean value of $\underline{R}_j^k$	$\overline{R}_j^k$
eMBB	100/20 ms	50	0.6
URLLC	25/5 ms	10	0.2
mMTC	300/60 ms	5	0.2

2.6.2 Provisioning Cost vs. Robustness

In this section, we evaluate the performance of the proposed **SNSR** formulation in terms of the resource provisioning cost and robustness. We define robustness as the percentage of the realized traffic demands for which the obtained solution of **DET_SNSR** is feasible. Figure 2.7 illustrates the relative cost and robustness as the number of sampled realization changes from $H = 1$ to $H = 30$. As expected, increasing the value of H enhances the robustness of the solution at the expense of increasing the resource provisioning cost.

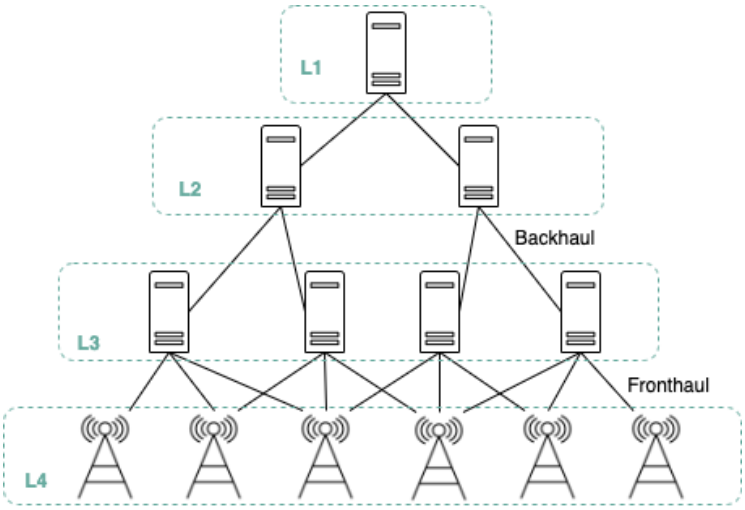


Figure 2.6: Simulation Infrastructure

2.6.3 Performance of the Two-Timescale Solution

In this section, we present the performance of the proposed two-timescale resource provisioning scheme, in terms of latency and acceptance ratio. We consider a duration of 60 short time slots and show the obtained latency and acceptance ratio for different slices. The results are averaged over 5 runs. Figure 2.8 and 2.9 illustrate the UP and CP latency for the URLLC, eMBB, and mMTC slices for the number of realizations (H parameter) equal to $5, 10$, and 30. We observe that in all cases, the latencies are below the specified QoS boundaries shown in Table 2.2.

The percentage of accepted slice demands (supported traffic) defined as $\sum_{j \in V_{gNB}} \sigma_j^k / |V_{gNB}|$ is shown in Fig. 2.10, 2.11, and 2.12 for eMBB, URLLC, and mMTC slices respectively. In this experiment, we change the number of realizations from 1 to 30. It is observed that as the number of realizations in problem **DET_SNST** increases, the acceptance ratio enhances for all slices.

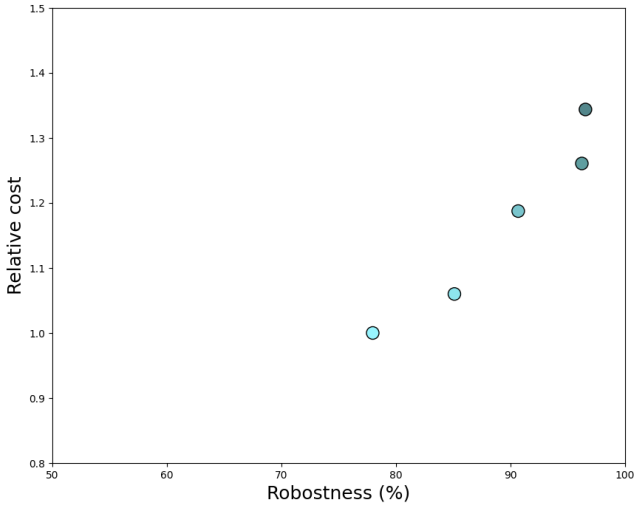


Figure 2.7: Relative Provisioning Cost vs. Robustness

2.7 Conclusion

E2E Network slicing is an emerging technology to carry flexible resource provisioning for various services with diverse QoS requirements in NGMN. While the resource provisioning for network slices is typically done in a best-effort manner, in this chapter, we studied the problem of mobile network slicing under demand uncertsinties. To this end, we propose a two-timescale framework including long and short time slots for the E2E network slicing by exploiting stochastic programming. In this method, we solve the network slice resource provisioning problem across the core network and RAN at each long time slot, modeled as a two-stage stochastic mixed integer linear program. We then adjust the RAN slices and scale down/up the core network slices if necessary, at each short time slot. We show the effectiveness of our proposed method through simulation.

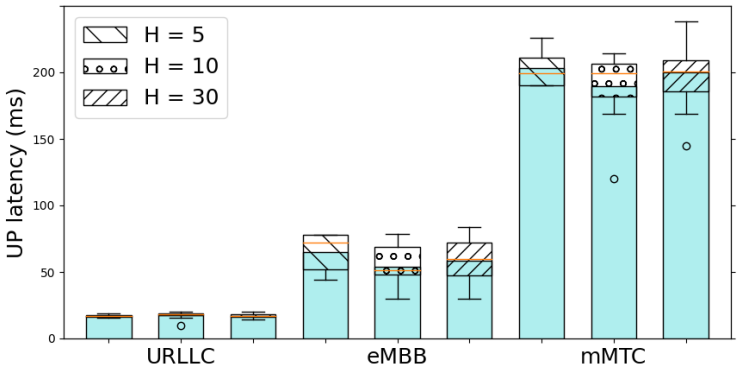


Figure 2.8: UP Latency for URLLC, eMBB and mMTC Slices

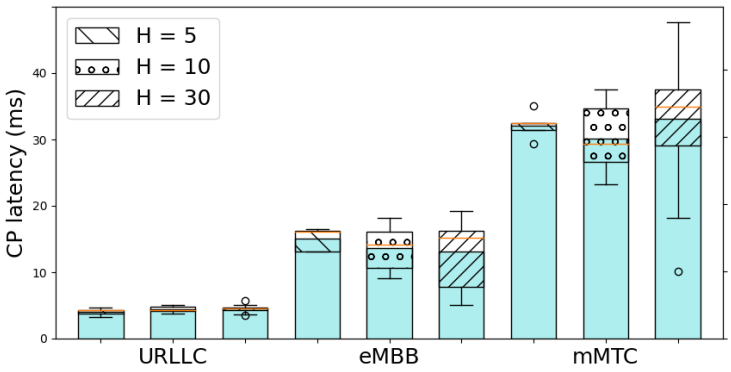


Figure 2.9: CP Latency for URLLC, eMBB, and mMTC Slices

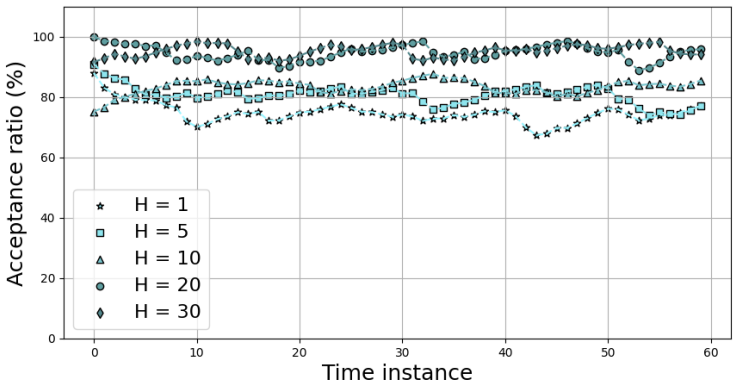


Figure 2.10: Average Acceptance Ratio for eMBB Slice

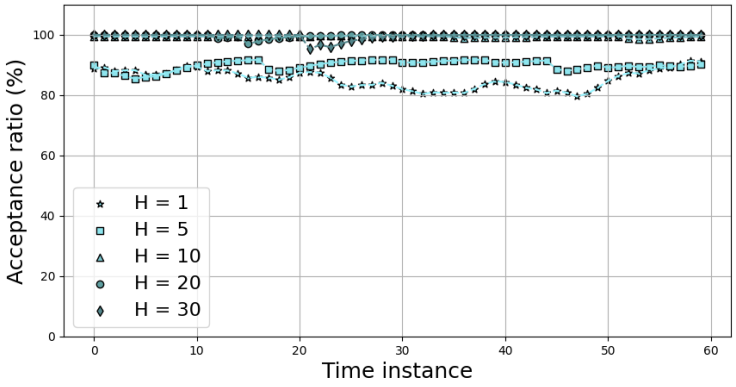


Figure 2.11: Average Acceptance Ratio for URLLC Slice

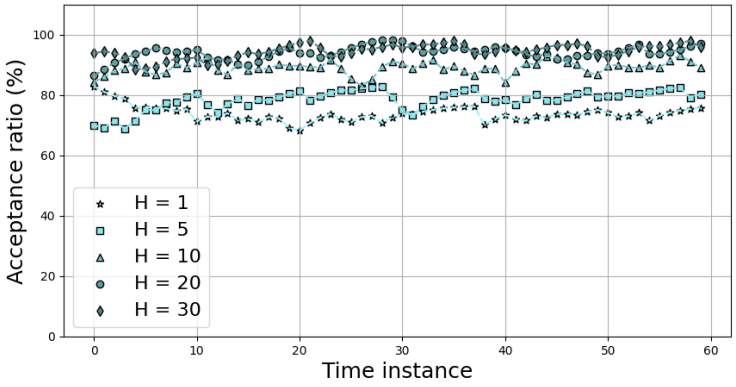


Figure 2.12: Average Acceptance Ratio for mMTC Slice

Chapter 3: Collaborative Cloud-Edge-Local Computing for Multi-Component Applications: An Infrastructure Provider Perspective

3.1 Overview

Driven by the needs of 5G to support a wide range of services in the domain of eMBB, URLLC, and mMTC, mobile cloud computing (MCC) has become a key solution to offload the heavy computation load of next-generation applications by pushing substantial amounts of computing functionality of mobile devices (MDs) to a remote cloud data center. However, the high E2E latency in MCC resulting from the large distance between an MD and the data center violates the low-latency requirement of many emerging applications such as autonomous driving, and virtual/augmented reality (VR/AR), and online gaming. This has led to a shift in the computing paradigm, namely mobile edge computing (MEC) [35]. The capability of utilizing the idle computation space distributed at the network edge has empowered MEC to become a promising element of 5G and 6G to realize a broad range of heterogeneous and computationally heavy use cases with reduced energy consumption end latency. On the other hand, edge servers have limited resources compared to the central cloud in MCC and can easily become overloaded. Therefore, it is usually not feasible to offload all the heavy computation loads to the MEC resources only, and a collaborative cloud-edge-local computation offloading scheme both enhances

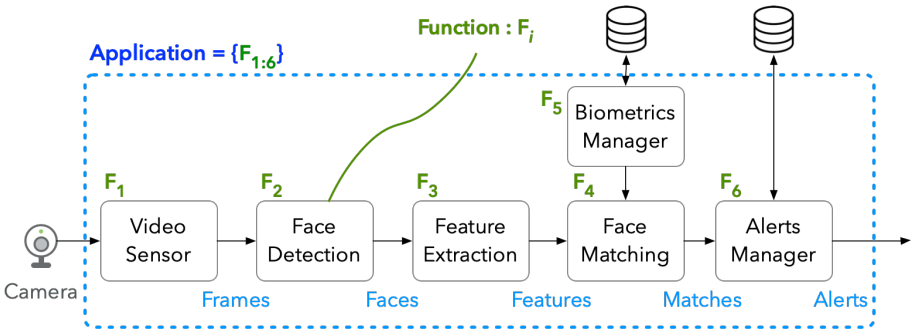


Figure 3.1: Video Surveillance Use Case: Watchlist Application

users' QoE and reduces providers' costs [36, 37]. While the problem of computation offloading and resource orchestration in a two-tiered infrastructure encompassing only two computing entities, either cloud-local or edge-local, has been extensively explored to date, there are less considerable works available on the collaborative multi-tiered computation offloading problem. Moreover, with the recent paradigm shift in the application deployment model from a monolithic service environment to the microservices architecture, most of the existing solutions for the resource orchestration of applications in MCC, MEC, or collaborative systems need to be re-examined [38]. In a microservices architecture, an application consists of multiple components, a.k.a. functions, with arbitrary component dependencies. Each component can be processed at any available infrastructure node including MD, edge servers, or the remote central server. Thus, the deployment of a multi-component application entails (i) deployment of individual functions, and (ii) management of data communication between various functions. It is worth noting that in a multi-tiered architecture, different kinds of networking capabilities such as 5G, MAN, and WAN, may exist at different tiers. Fig. 3.1 shows the pipeline of a microservices-based watchlist application, for video surveillance use cases. The application includes video ingress, face detection, feature extraction, face matching, biometrics manager, and alerts manager functions. Each individual function for this application has demands for different resources such as compute,

storage, bandwidth for the incoming streams, etc. The overall performance of the application depends on the allocated resources to all the microservices. This introduces a coupling relationship between the usage of different resources (such as networking and computation) and the application performance which should be considered in the resource orchestration phase for optimal output. While computation and communication are decoupled in 5G and it is not expected to be addressed in the future releases of 5G, 6G is foreseen to bring a tight integration of computing and communication in a distributed cloud platform [39]. However, the resource optimization of multi-component applications in such a complex infrastructure taking into account the resource coupling relationships is very challenging. An instance of such applications and their deployment in a collaborative cloud-edge-local environment is shown in Fig. 3.2. In this example, components $\{0, 1, 11\}$, $\{2, 4, 5, 6, 9\}$ and $\{3, 7, 8, 10\}$ are executed locally, at the edge server, and at the central cloud, respectively. In addition to the component assignment, network resources should be provisioned for the transmission of the data between two interacting components in multi-component applications. For instance, network bandwidth across a path from MD to the central cloud is reserved in the example of Fig. 3.2 to accommodate the data communication between components 0 and 3.

A multi-tiered infrastructure opens up several possibilities for the deployment of multi-component applications. In this chapter, we address the design and optimization of a cloud-edge-local collaborative offloading scheme for multi-component applications [40]. This problem can be studied from two viewpoints, (i) the InP perspective, and (ii) the application perspective. Each of these viewpoints has specific assumptions and objectives. While we study the first problem in this chapter, next chapter is dedicated to the second viewpoint. Given the requests for different services, the InP must answer the following questions:

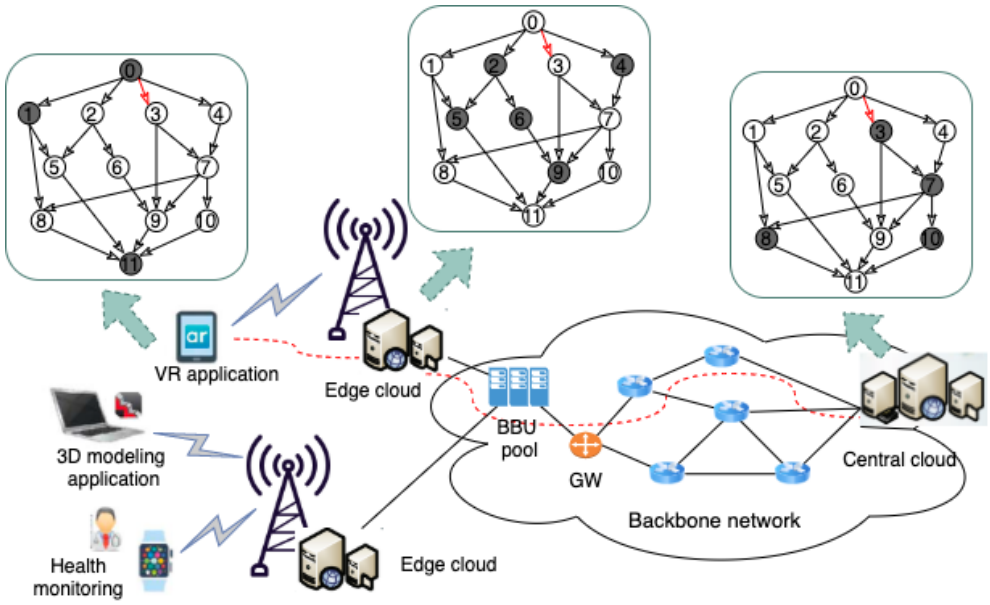


Figure 3.2: An Example of an Application Graph Deployment on a Collaborative Cloud-Edge-Local Computing System

- For each application request, which components are computed locally, offloaded to the edge, or to the central cloud in order to better utilize available computing, storage, and networking resources and achieve a higher level of QoS?

- How should the networking resources be provisioned in order that the data communication between different application components is realized?

- How the above decisions can be optimized such that the total deployment cost is minimized while adhering to the resource capacity and application QoS hard constraints?

Our objective for the collaborative cloud-edge-local computation offloading is to minimize the total cost defined as consumed energy and end-to-end applications' latency. The energy demand of the communications industry is projected to increase from 200-300 TWh in 2017 to 1200 or even 3000 TWh by 2025 [41], and the global network electricity bill is growing rapidly by 10% each year. Today's network devices are often powered 24/7 for a high availability

guarantee of network services. While energy efficiency in cloud data centers and for an MD with a limited battery has been broadly explored, it is largely left uninvestigated in multi-tiered computing environments due to the complicated interactions between MDs, edge servers, and the centralized cloud [42]. On the other hand, thanks to the flexibility of NFV and the very short reconfiguration time in software-defined mobile networks (SDMN), we are able to exploit the global network status information of the SDN controllers in order to re-route traffic in a dynamic manner and save energy by turning on just a minimal amount of network devices to carry the demanded traffic, instead of powering on all backbone network devices all the time. To this end, we formulate an optimization model for the resource allocation of multi-component applications in a collaborative cloud-edge-local offloading system. Our proposed approach is presented for a generic application and therefore can be used for any multi-component application with arbitrary component dependencies, as long as it can be modeled by a directed acyclic graph (DAG), which is applicable to most real-world applications. The formulated optimization problem is a MILP which impedes the effectiveness of the proposed approach in a dynamic and scalable manner due to time complexity. Thus, we also devise an efficient approximation algorithm based on LP relaxation and rounding (LPRR) to solve the problem efficiently.

The rest of this chapter is organized as follows. Section 3.3 presents the system model. Problem formulation and the proposed heuristic algorithm are discussed in Sections 3.4 and 3.5, respectively. We present the numerical results in Section 3.6. Finally, the main conclusion and future directions is Section 3.7.

We assume a collaborative cloud-edge-local environment as shown in Fig. 3.2, where the components of a mobile application can be computed locally, in edge servers, or in a central cloud. The proposed scheme is applicable to multi-component applications which [43] can be

classified into the following three major categories of data-partitioned-oriented, code-partitioned-oriented, and continuous-execution applications. The applications are realized through microservices architecture. From a practical point of view, two settings can be assumed: (i) an *offline* scheme in which all application requests are known in advance, and (ii) an *online* scheme where the applications arrive and depart the system over time and the requests are admitted or rejected upon arrival [44]. We consider an online setting that is more realistic and challenging. The goal is to optimize the resource allocation of the requested applications while minimizing the system-wide energy consumption and applications E2E latency.

The rest of this chapter is organized as follows. Section 3.2 provides an overview of the related works. We present system model and problem formulation in Section 3.3 and Section 3.4, respectively. The proposed heuristic solution is discussed in Section 3.5. Section 3.6 presents the numerical results. We conclude this chapter in Section 3.7.

3.2 Related Work

The related research on the MEC computation offloading can be divided into two categories, full offloading and partial offloading. Full offloading has been considered extensively in the literature such as in [45]. Partial offloading which traditionally deals with partitioning the considered task into two subtasks, one running locally and the other one remotely, has also been investigated in the existing works such as [46]. Recently, authors in [47] consider partitioning computation tasks into multiple subtasks each executed locally, at the edge or central cloud and model the latency minimization problem as a MILP with dual decomposition and matching-based algorithms proposed to derive near-optimal solutions. However, the the proposed solution is not applicable

to general subtask dependency. In the context of multi-component application which potentially extends partial offloading to arbitrary decomposition of applications, authors in [48] design an integer particle swarm optimization-based algorithm followed by a heuristic for the deployment of code-partitioned and MEC-enabled AR services. In [49], the problem of multi-component application placement in MEC systems is addressed considering the users mobility and network capabilities. The computation offloading problem in a collaborative environment has also been studied in the literature. Authors in [50] study computation offloading in a fog computing network, where the end users offload part of their tasks to a fog node and the fog node further offloads the task to neighboring fog nodes or a remote cloud server. An efficient collaborative task offloading scheme is proposed in [36] in which the MEC server collaborate with MDs and a remote cloud to provide better QoS. In contrast to the existing works targeting single task offloading in collaborative cloud-edge-local domains or multi-component application deployment in edge-local or cloud-local systems, we study the problem of multi-component application deployment in a collaborative multi-tiered environment.

3.3 System Model

We present the system model in this section. An edge cloud is defined as a pool of virtualized computing resources, usually co-located with a cellular BS. We consider a cellular system in which edge servers are co-located with BSs. The MDs within the coverage of a BS communicate with the corresponding BS (edge server) for offloading the necessary computation load. Moreover, edge servers are connected to a central cloud via multi-hop paths. Let V_N and $V_M = \{0, ..., M\}$ denote the set of backbone network devices and infrastructure computing

nodes, respectively. The indices 0, $\{1, ..., M-1\}$ and M stand for the MD, the edge servers, and the central cloud respectively.

3.3.1 Definitions

Substrate Graph: We model the physical infrastructure as a directed graph $G_S = (V_S, E_S)$, where $V_S = V_M \cup V_N$ and E_S denotes the substrate links. For each server $u \in V_M$, let f_u and W_u^{STO} denote its computation and residual storage capacity. Moreover, the data rate (in bit/sec) of the link $l \in E_S$ is represented by R_l.

Application Graph: We model the mobile application as a DAG, $G_A = (V_A, E_A)$, where the vertices in V_A denote the application components and an edge $e \in E_A$ represents the data dependency between two components. The required workload and storage of the node $i \in V_A$ and the data size (in bit) requirement of an edge $e \in E_A$ are denoted by $D_i^{CPU}, D_i^{STO}, D_e$, respectively. The application deployment process is considered as mapping the application graph G_A to the substrate graph G_S and it consists of two mappings: (i) *node mapping* which determines the assignment of the application components to substrate nodes, and (ii) *path mapping* that entails the assignment of the application edges to the substrate paths.

3.3.2 Computation Model

Let t_u^i and E_u^i denote the processing time and energy consumption of component i running on $u \in V_M$. Given the required workload of procedure i, t_u^i is expressed as:

$$t_u^i = \frac{D_i^{CPU}}{f_u}, \quad u \in \{0, 1, ..., M-1\} \quad i \in V_A \tag{3.1}$$

In addition to the CPU processing time, the queuing delay should also be accounted for by the substrate nodes with a partially smaller computation capacity. Therefore, a more holistic model addressing the queuing delay is stated as:

$$t_u^{i,k} = \sum_{j \leq k} \frac{D_{j,k}^{CPU}}{f_u}, \quad u \in \{1, ..., M-1\} \quad i \in V_A \tag{3.2}$$

where k stands for the processing order (based on FCFS) for application graphs components. For a component offloaded to the central cloud, the processing time can be ignored since the computation power of the cloud data center is relatively big compared to the local or edge servers, i.e $f_0 << f_u << f_M, u \in \{1, ..., M-1\}$. Following the model in [51], the energy consumption corresponding to the component i running on the substrate node u is expressed as:

$$E_u^i = \kappa D_i^{CPU} f_u^2 t_u^i \tag{3.3}$$

where $\kappa f_u^2 t_u^i$ is the energy consumption per CPU cycle and κ is a constant arising from the hardware architecture.

3.3.3 Communication Model

In this section, we introduce the communication model. Let t_l^e denote the transmission latency corresponding to mapping $e \in E_A$ to the substrate link $l \in E_S$, expressed as:

$$t_l^e = \frac{D_e}{R_l}, \quad \forall e \in E_A, l \in E_S \tag{3.4}$$

We assume that frequency division duplex (FDD) is used as the transmission mode and W_D and W_U denote the uplink and downlink channel bandwidths respectively. Hence according to the Shannon formula, the achievable data rate of the uplink/downlink (U/D) wireless links can be expressed as:

$$R_l^{D/U} = W_{D/U} log_2(1 + \frac{P_{s(l)}^{TX} d^{-\nu} |h_{D/U}|^2}{N_0}) \tag{3.5}$$

where $P_{s(l)}^{TX}$ is the transmission power of the transmitter of link l, and the uplink and downlink channels are assumed to be frequency-flat block-fading Rayleigh channels with a block length larger than the maximum latency requirement of the application. Throughout the section, we refer to the transmitter and receiver of the link l as $s(l), d(l)$. The path loss between MDs and BSs is modeled as $d^{-\nu}$ where d and ν are the corresponding distance and the path loss exponent respectively. Furthermore, the uplink and downlink channel fading coefficients are denoted by h_U and h_D modeled as circularly symmetric complex gaussian random variables.

Let $\mathcal{P}_S$ denote the set of K-shortest paths between any pair of nodes $u, v \in V_M, u \neq v$, i.e. $\mathcal{P}_S$ contains all K-shortest paths between MD and edge servers, MD and central cloud, and edge servers and central cloud. We also represent the total energy consumption corresponding to mapping the edge $e \in E_A$ to the substrate path $p_s \in \mathcal{P}_S$ as $E_{p_s}^e$. $E_{p_s}^e$ comprises the energy consumption for application components processing and data transmission between two components. Assuming that $P_l = P_{s(l)}^{TX} + P_{d(l)}^{RX}$ is the total consumed power of link l's transmitter and receiver,

$E^e_{p_s}$ is computed as follows:

$$E^e_{p_s} = \sum_{l \in p_s} P_l t^e_l + \sum_{u \in p_s} P_{on} f(u) \tag{3.6}$$

where $f(u)$ is an indicator function representing the required power P_{on} to turn on a network node that has been idle. Thus,

$$f(u) = \begin{cases} 0 & u \in V_S \text{ is active} \\ 1 & u \in V_S \text{ is idle} \end{cases} \tag{3.7}$$

3.4 Problem Formulation

In this section, we present the problem formulation. Given G_A, G_S and $\mathcal{P}_S$, we define the following decision variables for the problem formulation:

- A set of binary decision variables $\boldsymbol{x}$, where x^i_u equals 1 if the application node $i \in V_A$ is mapped to the substrate node $u \in V_M$.
- A set of binary decision variables $\boldsymbol{y}$, where $y^e_{p_s}$ is 1 if the application edge $e \in E_A$ is mapped to the substrate path $p_s \in \mathcal{P}_S$.

The system total consumed energy is computed as:

$$E(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \in V_A} \sum_{u \in V_M} E^i_u x^i_u + \sum_{e \in E_A} \sum_{p_s \in \mathcal{P}_S} E^e_{p_s} y^e_{p_s} \tag{3.8}$$

Let $\mathcal{P}_A$ denote the set of all directed paths in G_A and $p_A \in \mathcal{P}_A$. The overall latency of p_A denoted by $L_{p_A}(\boldsymbol{x}, \boldsymbol{y})$ equals the summation of its nodes (components) processing times and the data

transmission delay of its edges, i.e.:

$$L_{p_A}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{e \in p_A} (\sum_{p_s \in \mathcal{P}_S} t^e_{p_s} y^e_{p_s}) + \sum_{i \in p_A} (\sum_{u \in V_M} t^i_u x^i_u) \tag{3.9}$$

where $t^e_{p_s} = \sum_{l \in p_s} t^e_l + t^{prop}_l$ is the overall latency of mapping the edge $e \in E_A$ to the substrate path $p_s \in \mathcal{P}_S$, and t^{prop}_l is the propagation delay of link $l \in E_S$. Given $\mathcal{P}_A$, we define the *critical path* to be the path inducing the maximum latency among all paths of the application. It is important to note that since the latency incurred by an application component or edge is a function of the available resources of the selected substrate node and path for mapping, it is not trivial to determine the *critical path* in advance. Therefore, the overall latency of an application (the latency of its *critical path*) is given as:

$$L(\boldsymbol{x}, \boldsymbol{y}) = max_{p_A \in \mathcal{P}_A} L_{p_A}(\boldsymbol{x}, \boldsymbol{y}) \tag{3.10}$$

The constraints defining the feasible region of our optimization problem are defined below. Starting with the **mapping constraints**, we have:

$$\sum_{u \in V_M} x^i_u = 1, \quad \forall i \in V_A \tag{3.11}$$

$$x^1_0 = 1, \quad x^{|V_A|}_0 = 1 \tag{3.12}$$

$$\sum_{\substack{v \in V_M \\ p_s(u \to v) \in \mathcal{P}_S}} y^{e(i \to j)}_{p_s(u \to v)} - y^{e(i \to j)}_{p_s(v \to u)} = x^i_u - x^j_u \quad \forall e \in E_A, u \in V_M \tag{3.13}$$

Constraints (3.11) ensure that each application node is assigned to one substrate server. In (3.12), we enforce that the first and last components are computed locally. Constraints (3.13) guarantee the assignment of paths to the application edges for data communication between interacting components which are mapped to two different substrate nodes. The **MD energy budget constraint** is defined as follows:

$$\sum_{i \in V_A} E_0^i x_0^i + \sum_{v \in V_M} \sum_{e \in E_A} P_0^{TX} t_l^e y_{p_s(0 \to v)}^{e(i \to j)} + \sum_{v \in V_M} \sum_{e \in E_A} P_0^{RX} t_{p_s}^e y_{p(v \to 0)}^{e(j \to i)} \leq RE_0 \tag{3.14}$$

where RE_0 is the MD's residual energy. The **capacity constraints** are given below:

$$\sum_{i \in V_A} D_i^{STO} x_u^i \leq W_u^{STO}, \quad \forall u \in V_M \tag{3.15}$$

Finally, the **domain constrains** are given as:

$$x_u^i, y_{p_s}^e \in \{0, 1\}, \quad \forall i \in V_A, e \in E_A, u \in V_M, p_s \in \mathcal{P}_S \tag{3.16}$$

The objective is to minimize the weighted summation of the total consumed energy and application end-to-end latency. The problem formulation is as follows:

$$[P_1] \quad \text{minimize}_{\boldsymbol{x},\boldsymbol{y}} \quad \lambda \frac{E(\boldsymbol{x}, \boldsymbol{y})}{E_0} + (1 - \lambda) \frac{L(\boldsymbol{x}, \boldsymbol{y})}{L_0} \tag{3.17}$$

$$\textbf{s.t.} \quad (3.11) - (3.16)$$

Table 3.1: Table of the Used Notations

Parameter	Description
$G_S = (V_S, E_S)$	Substrate graph
$V_M = \{0, ..., M\}$	Set of infrastructure nodes for computation offloading
W_u^{STO}	Storage capacity of node $u \in V_M$
R^l	Data rate of link $l \in E_S$
t_l^{prop}	Propagation delay of $l \in E_S$
f_u	Computation capability (CPU-cycle) of node $u \in V_S$
$\mathcal{P}_S$	Set of K-shortest paths between nodes of V_M
$G_A = (V_A, E_A)$	Application graph
D_i^{CPU}, D_i^{STO}	Computing and storage requirement of node $i \in V_A$
D_e	Data size requirement of $e \in E_A$
$\mathcal{P}_A$	Set of all directed paths of G_A
t_u^i	Processing time of the procedure i running on node u
t_l^e	latency of D_e transmission on link l
E_u^i	Energy consumption of i running on u
E_p^e	Energy consumption of mapping edge $e \in E_A$ to path $p \in \mathcal{P}_S$
P_u^{TX}, P_u^{RX}	Transmission and reception power of node $u \in V_M$

where E_0 and L_0 are the total consumed energy and application latency when all application components are executed locally and are used to balance the two objective terms. λ is a non-negative constant that determines the tradeoff between energy and latency. In order to linearize (3.17), we use an auxiliary continuous variable z. It is straightforward to observe that $[P]$ is equivalent to the following MILP:

$$[P_1'] \quad \text{minimize}_{\boldsymbol{x},\boldsymbol{y}} \quad \lambda \frac{E(\boldsymbol{x},\boldsymbol{y})}{E_0} + (1-\lambda)\frac{z}{L_0} \tag{3.18}$$

s.t.

$$L_{p_A}(\boldsymbol{x},\boldsymbol{y}) \leq z, \forall p_A \in \mathcal{P}_A \tag{3.19}$$

(3.11) – (3.16)

Table 3.1 provides a summary of all used notations.

3.5 Heuristic Algorithm

Since the MILP model is NP-hard, we propose a heuristic algorithm based on LP relaxation and rounding, namely LPRR, to efficiently solve $[P_1']$ for large networks. In the algorithm, a sequence of relaxed LPs is solved until the decision on rejection or acceptance of the application request is made. At each round of LPRR, the optimal fractional mapping solutions for $\boldsymbol{x}$ and $\boldsymbol{y}$ are obtained by solving a reduced LP relaxation version of $[P_1']$ which outputs a node mapping decision for one of the application nodes. The algorithm terminates when all application nodes are mapped or the request is rejected. We denote the reduced relaxed LP solved at each iteration by **CCO_LP**$(\mathcal{X})$, where $\mathcal{X}$ denote the set of pairs (i, u) for which x_u^is are set to 1 at previous iterations. Thus, $\mathcal{X}$ is an empty set at the beginning of the algorithm, unless there are components in G_A that are required to be deployed at a specific tier due to other constraints. The proposed solution is shown in Algorithm 1.

Regarding the time-complexity of Algorithm 1, we note that in this algorithm, the LP **CCO_LP**$(\mathcal{X})$ is solved at most $|V_A|$ times, as at each iteration, one application component is mapped to the infrastructure network. Fast algorithms such as primal-dual Simplex method [52] and its variants have been proposed in the literature to solve LPs efficiently. Therefore, compared to the MILP of Section 3.4 which is computationally expensive, Algorithm 1 achieves polynomial time-complexity.

Algorithm 1 LPRR Algorithm for Collaborative Cloud-Edge-Local Computation Offloading

Input: G_S, G_A
Output: $\boldsymbol{x}, \boldsymbol{y}, z$
Initialize $\mathcal{X} \leftarrow \emptyset, Terminate \leftarrow False$
repeat
  $\{\boldsymbol{x}, \boldsymbol{y}\} \leftarrow$ Solve **CCO_LP**$(\mathcal{X})$ $\triangleright\ x_u^i = 1,\ \forall (i,u) \in \mathcal{X}$
  if problem infeasible **then**
    $Terminate \leftarrow True$
  end if
  $(i^*, u^*) = argmax\{x_u^i | i \in V_A, u \in V_M, (i,u) \notin \mathcal{X}$, (3.14), (3.15) not violated if $x_u^i = 1\}$
  if (i^*, u^*) exists **then**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{(i^*, u^*)\}$
  else
    $Terminate \leftarrow True$
  end if
until $Terminate == True$
if No solution exists **then**
  Request rejected
else
  Request accepted
  return $\boldsymbol{x}, \boldsymbol{y}, z$
end if

3.6 Numerical Results

In this section, we first describe the simulation environment and then proceed with evaluation results. We implemented our simulations in Java, using IBM ILOG CPLEX commercial solver to solve the MILP model with the branch-and-bound method. Our tests are carried out on a server with an Intel i5 CPU at 2.3 GHz and 8 GB of memory. We evaluate the performance of our collaborative cloud-edge-local (CEL) deployment approach on the Digex network topology available at the Topology Zoo [53]. A cloud data center is assumed to be located in San Francisco and edge servers are connected to BSs in Philadelphia, Boston, Miami, and Charlotte. We adopt the implementation of Yen's algorithm presented [54] for K-shortest path generation. 100 MDs are randomly distributed in a $1000m \times 1000m$ region around each BS where the BS is located

Table 3.2: Default Simulation Parameters

parameter	value
MD uplink/downlink BW	$10MHz$
MD uplink/downlink power	$50, 60dBm$
N_0, ν, κ, K	$-147dBm/Hz$, 2, 10^{-11}, 2
f_0, edge server f_u	1×10^9, $U(5,10) \times 10^9 cycles/s$
W_0^{STO}, W_u^{STO}	$\mathcal{U}(20,40)$, $\mathcal{U}(1,2) \times 10^4$
R_l	$10Mbps$ for wired links

at the center of the square region. The application requests arrive according to a Poisson process with an average rate of 3 requests per 100 time units. The lifetime of requests has an exponential distribution with an average of 1000 time units. The remaining simulation parameters are given in Table 3.2. We run the simulation for 200 applications requested by random MDs. The number of application components is uniformly distributed in $[4, 10]$. The dependency type of the application requested by each MD is randomly selected from (i) sequential, (ii) parallel, and (iii) layer-by-layer structures given in [55]. D_i^{CPU} and D_i^{STO} for the first and last components have distributions $\mathcal{U}(0.01, 0.03) \times 10^9$ cycles/s and $\mathcal{U}(1, 10)$ respectively. For the rest of the components, the values for CPU and storage are sampled according to $\mathcal{U}(0.1, 0.5) \times 10^9$ and $\mathcal{U}(10, 30)$. Moreover, $D_e \sim \mathcal{U}(50 - 100)Kb$. The performance of the proposed optimal CEL solution (CEL-CPLEX) and its approximation (CEL-LPRR) is compared with the following schemes and their approximations:

- Edge-local execution (EL): only edge servers are considered for offloading.
- Cloud-local execution (CL): only the central cloud is considered for offloading.

Fig. 3.3 illustrates the acceptance ratio for CEL-CPLEX, CEL-LPRR, EL-LPRR, and CL-LPRR. It is observed that the proposed CEL scheme outperforms the EL and CL strategies significantly, as it admits up to 23% and 29% more requests than EL-LPRR and CL-LPRR

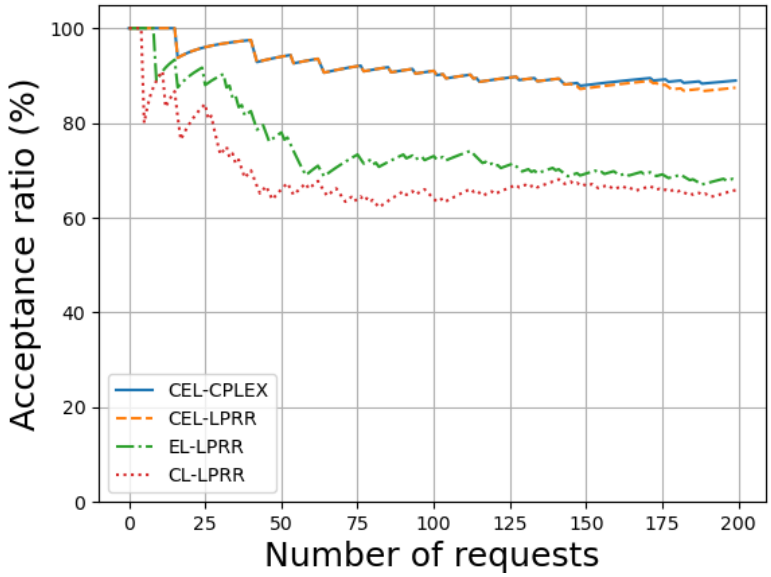


Figure 3.3: Applications Acceptance Ratio

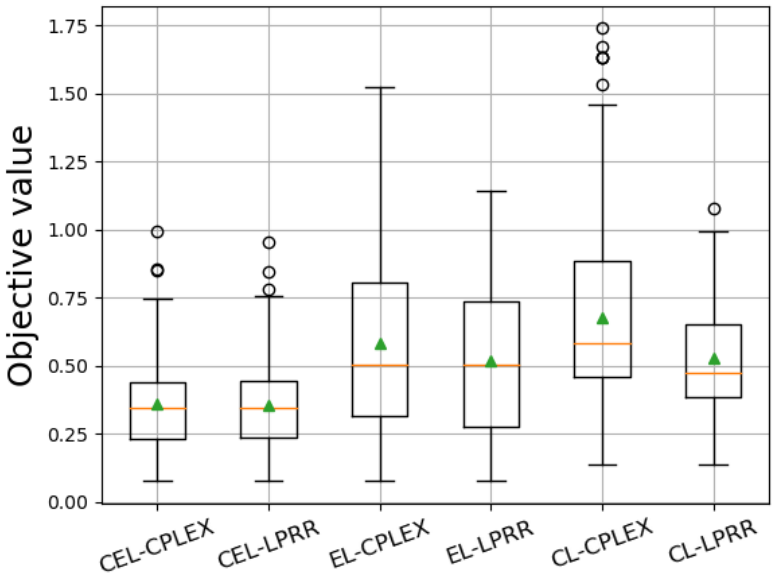


Figure 3.4: Objective Value of $[P_1']$

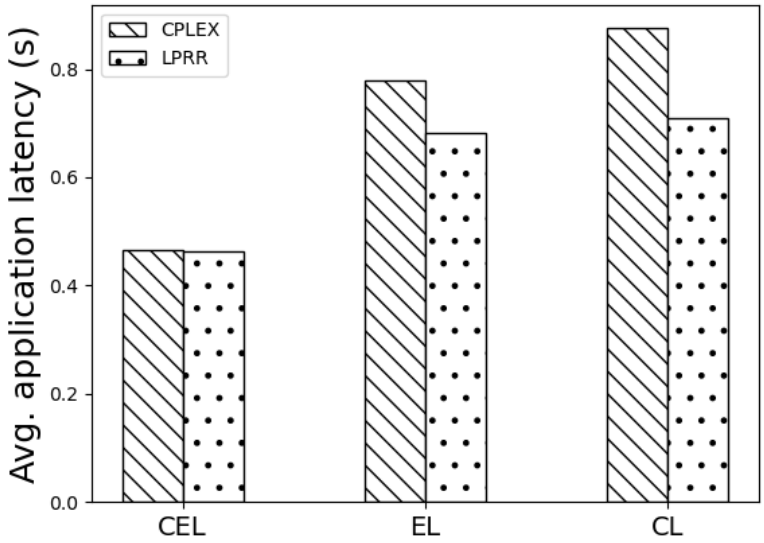


Figure 3.5: Average Application Latency

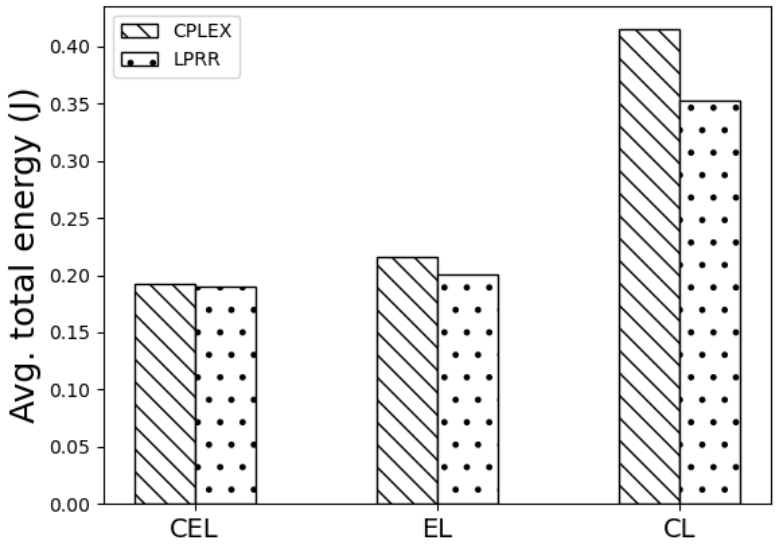


Figure 3.6: Average Total Energy

solutions respectively. Moreover, the maximum deviation of CEL-LPRR from CEL-CPLEX is 1.3%. Fig. 3.4 depicts the box plots corresponding to the objective function values for CEL-CPLEX, EL-CPLEX, CL-CPLEX, and their LPRR solutions for the set of accepted requests in each case. It is observed that CEL results in lower objective values compared to EL and CL approaches as expected. Moreover, CEL-LPRR is able to generate near-optimal solutions. It is important to note that CL-LPRR and EL-LPRR schemes have lower average objective values (denoted by green triangles) than CL-CPLEX and EL-CPLEX respectively since the optimal solutions found by the CPLEX solver admit more requests than the LPRR algorithm resulting in higher cost (energy and latency) per request. In Fig. 3.5 and 3.6, the average overall latency and total energy consumption of CEL are benchmarked against EL and CL solutions. It is observed that the proposed CEL scheme has lower average latency and energy consumption. The four figures together prove the efficiency of the CEL-LPRR approach.

3.7 Conclusion

In this chapter, we investigated the optimal computation offloading for multi-component applications in the collaborative cloud-edge-local systems. While the computation offloading problem for single tasks is extensively studied in the literature, the resource allocation problem of multi-component applications has remained largely uninvestigated. Moreover, we assume a collaborative cloud-edge-local system for the deployment of multi-component applications, which is not studied before. Our proposed scheme aims at minimizing the total consumed energy and the application end-to-end latency and is applicable to multi-component applications with arbitrary component dependencies. We formulated the problem as a MILP and applied the LP

relaxation and rounding technique to generate near-optimal solutions. The simulation results show the superior performance of our proposed solution in terms of acceptance ratio, consumed energy and end-to-end latency compared to two edge-local and central-local offloading baselines.

Chapter 4: Resource Orchestration of Multi-Component Applications: An Application-Side Perspective

4.1 Overview

As discussed in the previous chapter, a multi-tiered computing and networking system where critical services are offloaded to MEC and the delay-tolerant services are computed at the remote cloud has the potential to improve the application's performance and overall resource utilization. Taking into account the perspective of the application, the main focus of this chapter is to solve the resource orchestration problem of multi-component applications (realized through microservices architecture) in a collaborative cloud-edge-local computing system. We assume that there is a slice abstraction on top of the compute and network infrastructure and the application uses this slice abstraction to request network as well as compute slices. Thus, the underlying infrastructure components are untouched by the application. All requests for compute and network slices always go through slice abstraction, which may grant or deny requested slices depending on the resource conditions and demands at that time. Fig. 4.1 shows a multi-tiered compute and network fabric, and the scope of this section lies in the application layer on top of slice abstraction. In such a multi-tiered architecture, compute is available at various tiers like devices i.e. where data is produced, edge, and in the cloud. Similarly, different kinds of networking

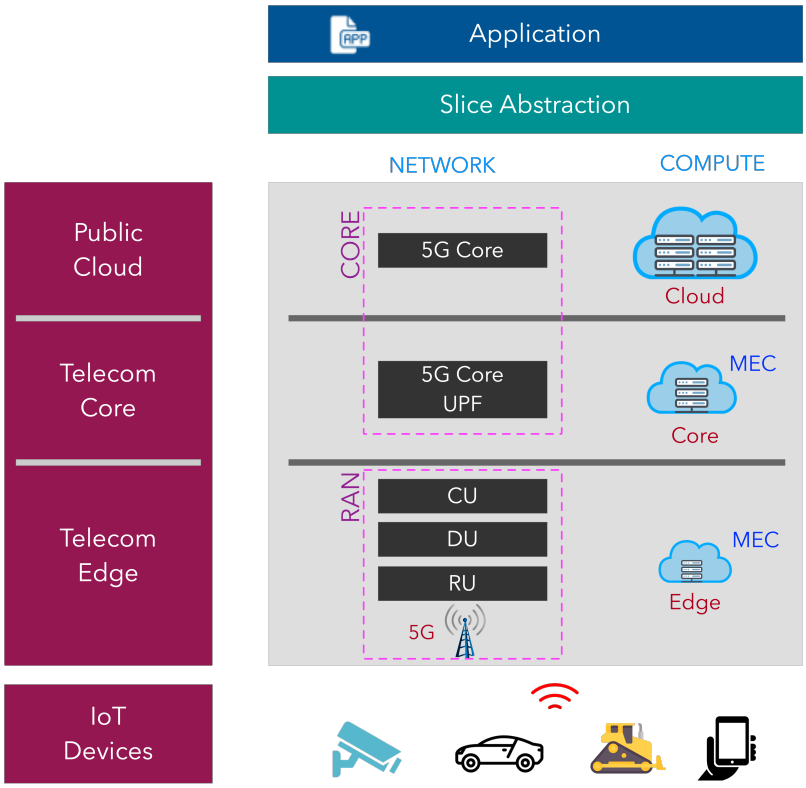


Figure 4.1: Multi-tiered Compute and Network Fabric

capabilities are available at different tiers, e.g. 5G connectivity between devices and edge servers, MAN between distributed edge resources, and WAN between edge and central cloud. In addition to the complexity of the infrastructure, the real-time state of different resources (e.g. available network and compute) is highly dynamic due to high variability in the compute (multi-tenancy, heterogeneity, etc.), and changing network (5G NR interference, link congestion, packet loss, etc.) conditions. Therefore, the problem of application deployment and optimization across multi-tiered compute and network fabric is even more challenging considering the real-time state of resources. To this end, we introduce the concept of resource coupling and show how an application can benefit from utilizing such coupling functions during the runtime [56].

The main goal of this chapter is to automatically manage the execution of microservices-based applications on a dynamic, heterogeneous, multi-tiered compute fabric in a 5G network

taking into account the coupling relationships between resources, when only application-level requirements are given, without knowing individual function-level requirements. The following are the main steps of our proposed methodology:

- We first identify and model the coupling between network and compute resource usage, and analyze the impact of the coupling on application performance by considering multiple real-world use cases.

- We propose a novel optimization formulation that captures the compute and network coupling relationship, and enables a principled consideration of resource allocation options to significantly reduce network and compute resource usage. Our proposed runtime system uses the new formulation, and utilizes the coupling to jointly optimize compute and network resources across different edge compute and network slices while ensuring consistent, high quality insights.

- We also propose an RL-based online method to dynamically adjust an application's compute and network resource reservations to minimize under-utilization of requested resources while ensuring acceptable service quality metrics. The RL method automatically captures the resource coupling relationships.

The rest of this chapter is organized as follows. We present the literature review in Section 4.2. In Section 4.3, we elaborate on resource coupling relationships through real-life application examples. Section 4.4 describes the optimization-based method. The RL-based method is presented in Section 4.5. Finally, in Section 4.6, we highlight the main conclusion and future directions.

4.2 Related Works

The work presented in [57], uses a distributed cross-domain resource orchestration (DIRECT) for cellular edge computing considering the radio and computing resources of a radio access network and multiple edge servers. The formulated resource orchestration problem takes the perspective of the network operator with the objective of maximizing the sum utility of network slices on all edge nodes. Assuming unknown utility functions, the authors propose a learning-assisted resource orchestration based on a probabilistic model with a gradient-based optimization solution. Authors in [58] show that DIRECT incurs overprovisioning due to ignoring the coupling between different edge resources. This coupling is modeled as a linear function used in a MILP to optimally instantiate joint network-MEC slices and prevent resource overprovisioning. In order to deal with the MILP time complexity, distributed algorithms are proposed to leverage the similarities among edge nodes and resource virtualization which can instantiate heterogeneous slices within a short distance from the optimum.

In [59], a multi-tier vehicular edge computing (VEC) system is considered which consists of three layers of data generation, vehicular edge computing, and remote cloud. The overall system involves the cooperation of local edge servers with the global cloud servers distributed over a geographical region. An ML-based prediction is utilized to perform a two-stage process for the offloading decision, a classification for predicting the offloading success, and a regression for the service time estimation. While the solution in [59] takes into account static datasets and models, the authors [60] in propose online multi-armed bandit (MAB) based task offloading schemes to avoid poor performance when the VEC environment conditions to which the static models are exposed differ from those used for model training. It is shown that the proposed

contextual bandit-based algorithm surpasses all other algorithms under the failure rate and QoE metrics besides achieving adequately comparable service time values. Different from [59] and [60], we consider an underlying realistic 5G system. Moreover, the workloads in the aforementioned works are only single tasks, which makes the proposed frameworks not applicable to microservices-based applications. Authors in [61] developed an emulation software named eXP-RAN, which allows experimenting with network slicing in virtualized RAN nodes and EC scenarios, with the key characteristics of slicing abstraction, service representation, and predictable performance. Compared with existing relevant tools such as EdgeCloudSim [62], eXP-RAN has the ability to monitor each network slice independently. However, the current version of eXP-RAN only implements the RAN slicing and it lacks 5G core network implementation. Therefore, its applicability for a case of a multi-tier computing framework is questionable. Regarding the microservices-based application deployment in a dynamic environment, the authors in [38] design an RL-based proactive scheme for the placement and migration of an already placed microservice in the MEC setup. In contrast to a conservative policy leading to wasteful resource allocation and a reactive on-demand policy causing high latency, the main contribution of this section is to learn and synthesize the optimal proactive prefetch, deployment, and migration schedule, given a microservice workflow by utilizing the user mobility. However, only sequential workflow structure (linear sequence of microservices) is considered in this study, and no back-end central cloud is assumed. Moreover, all the microservices invoked by a user can be co-located at one edge server.

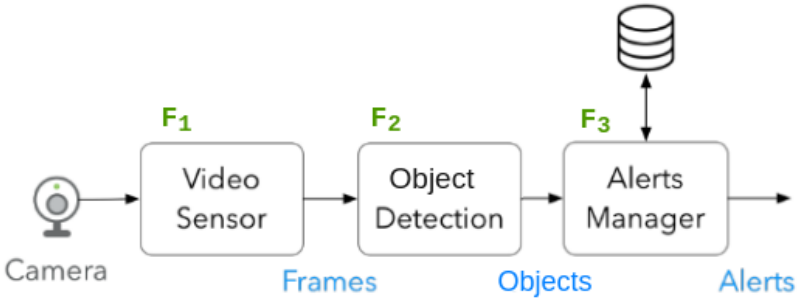


Figure 4.2: Intelligent Transportation System: Object Detection Application

4.3 Resource Coupling

In this section, we discuss the resource coupling relationship which is exploited in the resource allocation scheme. We consider two video analytics use cases in two different industry verticals. The first one is in video surveillance and the second one in ITS. Fig. 3.1 shows the structure of the video surveillance use case i.e. watchlist application, which uses face recognition technology to identify individuals seen in front of a camera. Fig. 4.2 shows object detection application, which is used in ITS to detect objects like cars and even people e.g. pedestrians, and build higher-level applications like accident prevention, safety alerting, traffic control, etc.

We now illustrate the impact of the network and compute resources on the performance of the above applications. In Fig. 4.3 and 4.4, the performance of the watchlist application in terms of face detection accuracy is evaluated for two sample videos as the allocated CPU cores to the face-detection function and the network bandwidth for the input streams of the video sensor ingress function vary. Fig. 4.5 and 4.6 show the performance of the object (person) detection application in terms of the detection score for two sample videos. In Fig. 4.7, the performance of the object detection application for a car detection task is illustrated for different compute and

network availabilities. It is important to note that in all cases, the performance of the application is controlled by both network and compute resource usage. As a result, in order to avoid resource overprovisioning and meet application requirements, this coupling effect should be considered when deploying the application. Thus, it is observed that although the general pattern is similar in different experiments (increasing compute and network improves the performance in most of the cases), the coupling relationship is application-specific and even within an application, it is non-linear and therefore not trivial.

In the following, we clarify the coupling relationships and how they can be utilized to optimize the resource allocation decision or enhance the application performance through three specific operation instances of experiment 1-1 (Fig. 4.3). Suppose that initially, the watchlist application is operating at point $P_1 = (0.5\,core, 10\,Mbps)$. If the network experiences congestion (network is a bottleneck) and the bandwidth drops from $10\ Mbps$ to $4\ Mbps$, the system is forced to operate at point $P_2 = (0.5\ core, 4\ Mbps)$, and the performance of the application drops from 83% to 71%. In this scenario, if there exists idle compute, the performance can be improved by increasing the allocated CPU cores to $1\ core$ and moving to $P_3 = (1\ core, 4\ Mbps)$. In an inverse order, if the system is initially deployed at P_3 with the performance of 83% and suddenly the available compute resource reduces (compute is a bottleneck), the performance degrades by 12% by moving from P_3 to P_2. In this case, allocating more network bandwidth to the incoming video stream changes the operating point to P_1, thus performance remains unchanged. In each of the aforementioned cases, there is a tradeoff between network and compute usage which can be exploited to avoid performance degradation by taking into account resource coupling relationships in the application resource orchestration and devising a joint network and compute resource allocation scheme.

Now consider a third case in which the system is operating at point $P_4 = (1\ core, 10\ Mbps)$ initially. It is observed that the allocated compute resource can be reduced to $0.5\ core$ by moving to $P_1 = (0.5\ core, 10\ Mbps)$ without affecting the performance, thus avoiding resource overprovisioning. The released CPU cores can be allocated to other services deployed on the same node, which enhances the resource utilization and the performance of other deployed applications. Another option is to reduce the network usage from $10\ Mbps$ to $4\ Mbps$ and reduce even more to $2\ Mbps$ by moving to point $P_5 = (1\ core, 2\ Mbps)$, without affecting the application accuracy. Further reduction of network bandwidth will result in significant accuracy degradation and thus should be avoided. Therefore, even though neither network nor compute resources become scarce, there may exist multiple paths or options to save on different resources and avoid overprovisioning while keeping the performance unchanged or within an acceptable range. The decision on which path (operational point) to opt for depends on the objective of the application manager and the state of the resources. In the following, we propose an optimization-based decision making process for application deployment incorporating the resource coupling relationships.

4.4 Optimization-based Resource Orchestration

4.4.1 System Model

In the sequel, we present the system model. The physical infrastructure consists of computing nodes distributed across multiple layers, at the edge, and at a central cloud. At each compute tier, compute slicing is possible for the allocation of the resources to different applications. Let $\mathcal{M}$ represent the set of compute nodes. Each compute node $m \in \mathcal{M}$ is characterized by the tuple

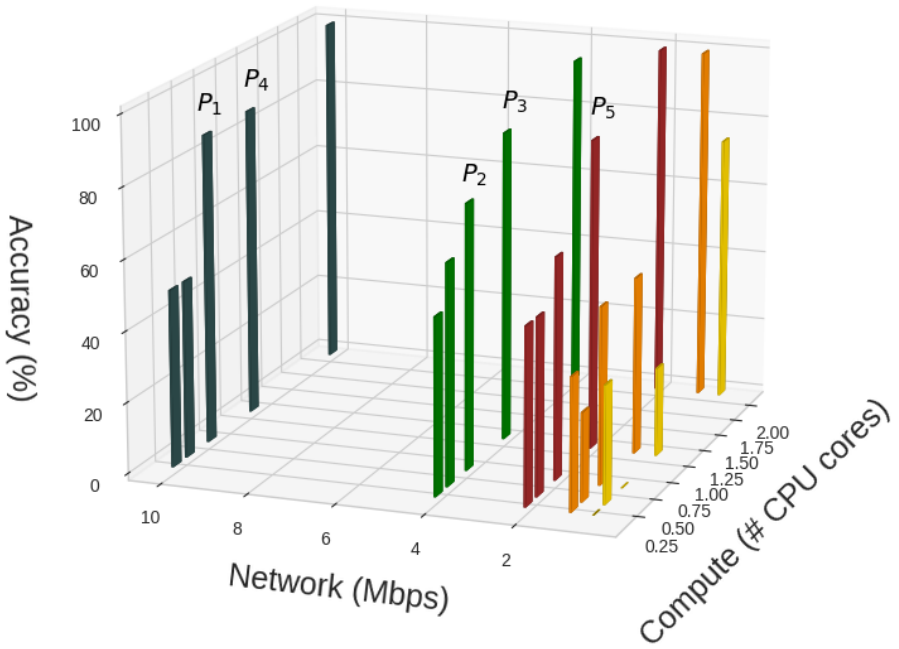


Figure 4.3: Exp 1-1 - Watchlist

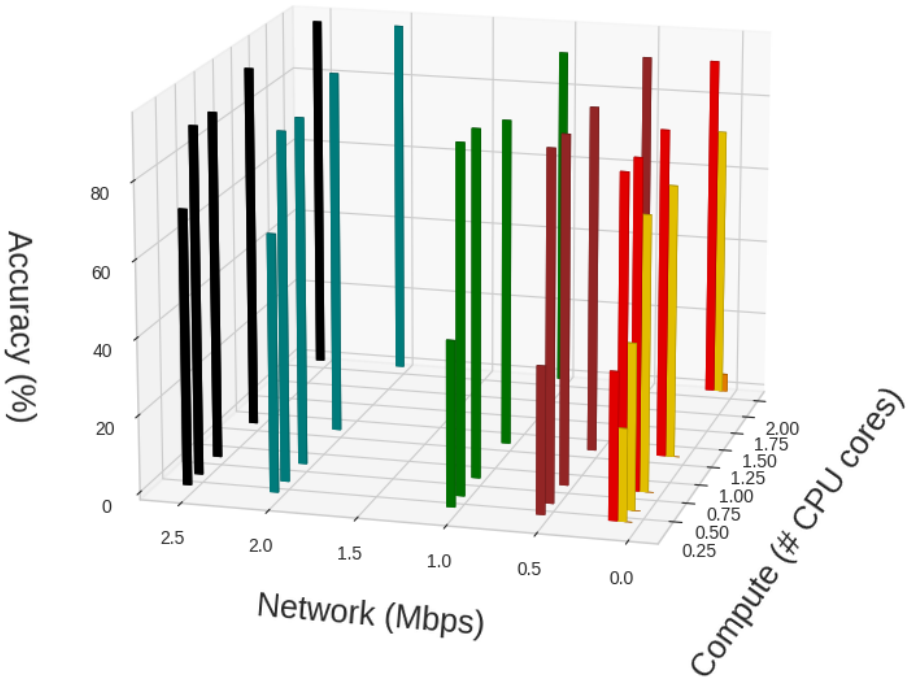


Figure 4.4: Exp 1-2 - Watchlist

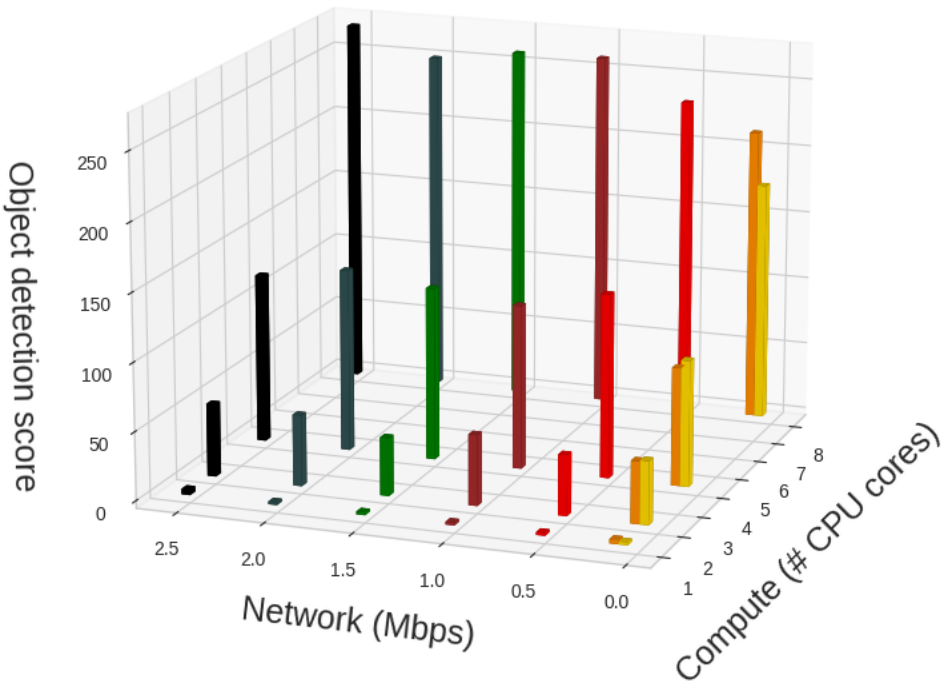


Figure 4.5: Exp 2-1 - Object (person) detection

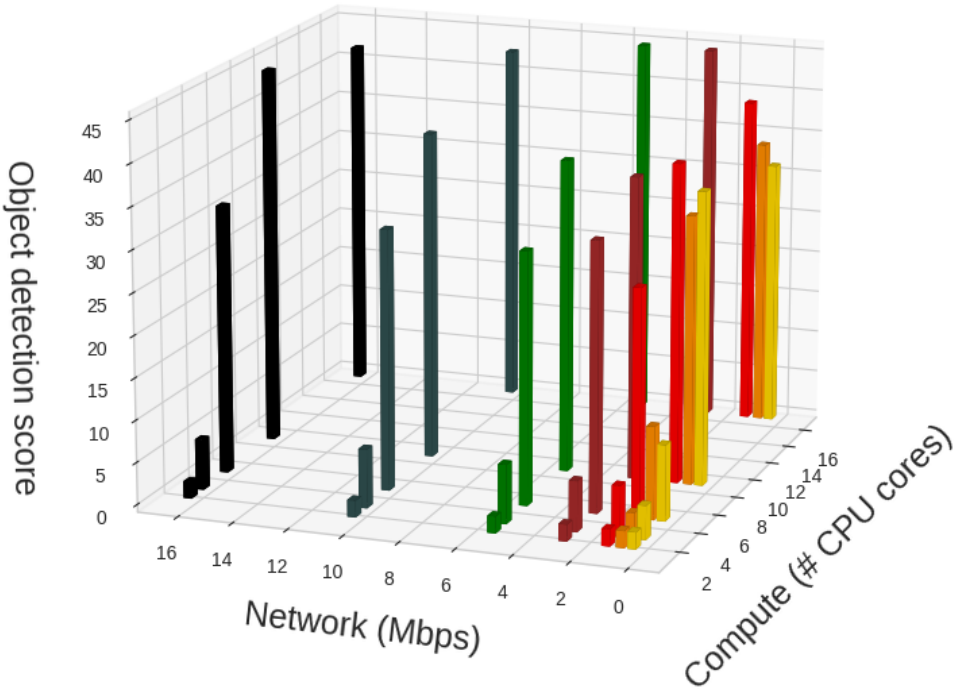


Figure 4.6: Exp 2-2 - Object (person) detection

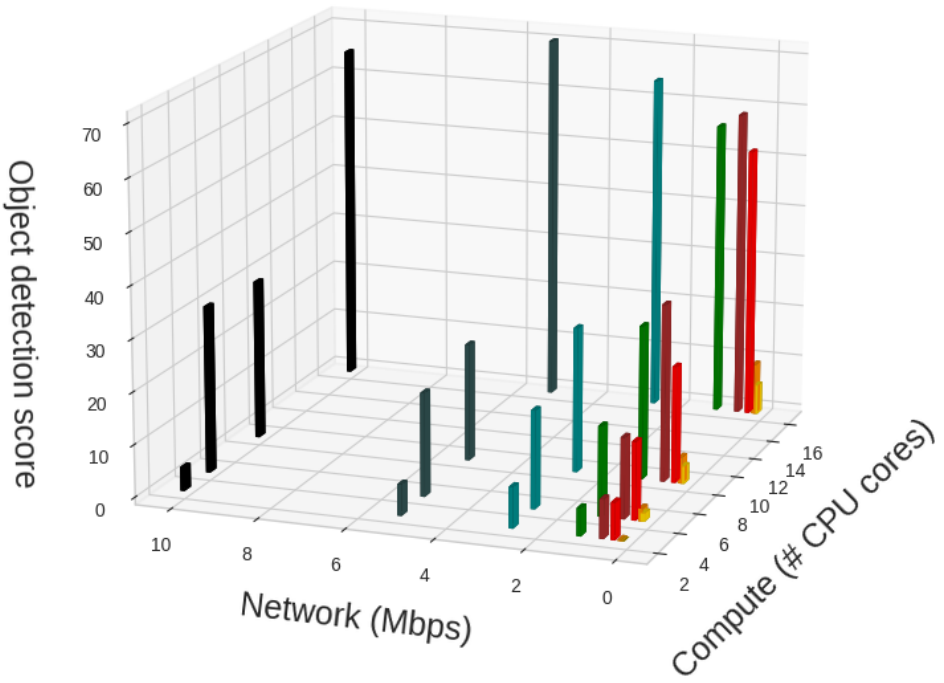


Figure 4.7: Exp 2-3 - Object (car) detection

$(\boldsymbol{g}_m, tier_m)$, where $\boldsymbol{g}_m$ and $tier_m$ denote the vector of available resources and the associated tier (e.g. IoT device, far edge, near edge, central cloud), respectively. Assuming that each node $m \in \mathcal{M}$ provides T different resources represented by set $\mathcal{T}$, the size of $\boldsymbol{g}_m$ is T. For instance, $\mathcal{T} = \{net, com\}$ specifies a scenario with the consideration of two resource types (network and computation) for nodes. In that case, $\boldsymbol{g}_m \in [0, \infty) \times [0, \infty)$. While our proposed approach can be easily extended to an arbitrary set of resources, in this section, we consider network and compute resources, i.e. $\mathcal{T} = \{com, net\}$.

We model an application as a set of microservices or functions and interconnections that represent the data dependency between functions. An application is specified by a tuple $\mathcal{R} = (\tau, \omega)$, where τ and ω stand for the required end-to-end delay and throughput of the application, respectively. Let $G = (V, E)$ be the graph representing the application, where V denotes the set of application functions and E represents the interconnections between functions. Furthermore,

$\mathcal{R}_v = (\tau_v, \omega_v)$ denotes the portion of the delay and throughput corresponding to node (function) v. Moreover, $tier^v$ denotes the tier on which function v should run, if such constraints exists for function v. For instance, there might exist constraints on some functions of the mobile application (such as in user-initiated applications) to run locally (on the user equipment). Given the function level performance metrics $\{\mathcal{R}_v, v \in V\}$, we assume that the rules defining the application level performance metrics are known. One challenge in this regard is to determine the set of functions contributing to each of the end-to-end application performance metrics (a.k.a the critical path or pipeline of the application). For the sake of simplicity, we assume that the knowledge about the contributing functions to each performance metric is available by the application developer similar to [63]. For instance, given the functions of the critical path of G as $V_{critical} \in V$, we can calculate the end-to-end application delay as $h_{delay}(\tau_1, ..., \tau_{|V|}) = \sum_{v \in V_{critical}} \tau_v$. Similarly, the throughput rule is computed as $h_{throughput}(\omega_1, ..., \omega_{|V|}) = min_{v \in V_{critical}} \omega_v$.

In order to successfully and optimally deploy an application given its end-to-end performance requirements, it is important to understand the coupling between the usage of different resources. Let p denote the desired application performance, e.g. p can be the detection accuracy in the watchlist application. To address the impact of the network and compute resources on the application performance, we define $f^{t,t'}_{v,v'}(x, p) : \mathbb{R} \rightarrow \mathbb{R}$, $v, v' \in V$, $t, t' \in \mathcal{T}$ as the minimum resource unit of type t' that should be allocated to function v' in order to achieve the application performance of p, given that x units of resource type t is allocated to function v. In fact, $f^{t,t'}_{v,v'}(.,.)$ reflects the coupling relationship between each pair of resources allocated to all pairs of application functions. Even for the same function i.e. when $v = v'$, the coupling relationship between different types of resources is reflected through the defined function as well. For instance, given that the input streams of the function F_2 of the watchlist application in Fig. 3.1 consume

x $Mbps$ network, the minimum number of CPU cores that should be allocated to F_2 in order to achieve an accuracy of p is equal to $f_{2,2}^{net,com}(x,p)$. As a numerical example, for the watchlist application of Fig. 3.1, it can be observed from Fig. 4.3 that $f_{1,2}^{net,com}(10,80) = 0.75$, i.e. in order to have the accuracy of 80% when the available network bandwidth for the video stream input of F_1 is 10 $Mbps$, it suffices to allocate 0.75 CPU core to the face detection function F_2.

We model the application deployment problem across a multi-tiered compute and network fabric as an optimization problem. We then discuss the usage of different models for coupling relationships.

4.4.2 Problem Formulation

The application resource allocation and performance optimization problem entails the assignment of microservices to the compute nodes in $\mathcal{M}$ (a.k.a. placement problem) and the allocation of different resources to each function, such that end-to-end application requirements (e.g. delay and throughput) are satisfied. We model this problem as a multi-objective optimization problem, with the objective of minimizing the total resource usage (equivalently, the deployment cost) and maximizing the application performance, by incorporating the resource coupling functions introduced in Section 4.3. By designing a joint optimization problem with two objective terms, the tradeoff between performance and resource usage illustrated in the examples of Section 4.3 is also captured. The following decision variables are defined for the problem formulation:

- $x_{v,m} \in \{0,1\}$: a binary decision variable for function placement which is equal to 1 if the function v of the application is assigned to the substrate node m for execution and 0 otherwise.

- $y^t_{v,m} \in [0, g^t_m]$: a continuous decision variable denoting the amount of resource type t of node m allocated to function v.

- $p \in [0, p_{max}]$: a continuous decision variable representing the application performance, e.g. the face recognition accuracy or object detection score.

The resulting optimization problem is as follows:

$$[P_2] \quad \text{minimize} \quad \eta \sum_{t\in\mathcal{T}} \sum_{v\in V} \sum_{m\in\mathcal{M}} y^t_{v,m} - (1-\eta)p \tag{4.1}$$

s.t.

$$\sum_{m\in\mathcal{M}} x_{v,m} f^{t,t'}_{v,v'}(y^t_{v,m}, p) \leq \sum_{m\in\mathcal{M}} x_{v',m} y^{t'}_{v',m}, \ \forall v, v' \in V, t, t' \in \mathcal{T} \tag{4.2}$$

$$y^t_{v,m} \leq g^t_m x_{v,m}, \ \forall t \in \mathcal{T}, m \in \mathcal{M}, v \in V \tag{4.3}$$

$$\sum_{v\in V} y^t_{v,m} \leq g^t_m, \forall t \in \mathcal{T}, m \in \mathcal{M} \tag{4.4}$$

$$\sum_{m\in\mathcal{M}|tier^v=tier_m} x_{v,m} = 1, \ \forall v \in V \tag{4.5}$$

$$\tau \geq h_{delay}(\tau_1, ..., \tau_{|V|}) \tag{4.6}$$

$$\omega \leq h_{throughput}(\omega_1, ..., \omega_{|V|}) \tag{4.7}$$

$$x_{v,m} \in \{0, 1\}, \forall v \in V, m \in \mathcal{M}$$

$$0 \leq y^t_{v,m} \leq g^t_m, \forall v \in V, m \in \mathcal{M}, t \in \mathcal{T}, \ 0 \leq p \leq p_{max} \tag{4.8}$$

In the objective function (4.1), η is a parameter between 0 and 1 used to control the balance between the two objective terms. In our experiments, we tested different values for η and selected a small value to promote a solution that primarily enhances the performance and minimizes the

total consumed resources. Constraints (4.2) ensure that the resources allocated to each application microservice is greater than or equal to the required minimum amount (given by the defined coupling functions) to potentially achieve the performance of p. For instance, the resource type t' allocated to function v' which is equal to $\sum_{m \in \mathcal{M}} x_{v',m} y^{t'}_{v',m}$ should be greater than or equal to $\sum_{m\mathcal{M}} x_{v,m} f^{t,t'}_{v,v'}(y^{t}_{v,m}, p)$ for all t, v. This constraint together with the objective of minimizing total used resources results in a solution that avoids resource overprovisioning. The set of inequalities in (4.3) and (4.4) enforce the infrastructure capacity constraints. Constraints (4.5) ensure that each function of an application is deployed at one infrastructure node. The application end-to-end performance requirements are guaranteed by constraints (4.6) and (4.7). Finally, the domain constraints are expressed in (4.8), where p_{max} is the maximum observed performance for a specific application in all resource allocation vectors. The optimization problem $[P_2]$ is a mixed integer nonlinear program (MINLP) owning to the constraints (4.2) and the integer variables, thus an $\mathcal{NP}$-hard problem. In the next section, we discuss the models for the coupling functions and solve a special case of $[P_2]$.

4.4.3 Modeling the Resource Coupling Relationships

In this section, we discuss different models that we can use for the coupling functions. The first one is a linear regression modeled as $f^{t,t'}_{v,v'}(y, p) = \alpha^{t,t'}_{v,v'} y + \beta^{t,t'}_{v,v'} p + \gamma^{t,t'}_{v,v'}$. The parameters $\alpha^{t,t'}, \beta^{t,t'}, \gamma^{t,t'}$ are obtained using the historical data collected in an offline manner. It is important to note that while we employ linear regression in this section for modeling resource couplings, it is possible to use other models such as a support vector regressor (SVR) or a multilayer perceptron (MLP) resulting in better prediction performance. However, the benefit of linear

Table 4.1: Performance Comparison of Linear Regression, SVR, and MLP Models for Resource Coupling Functions

Instance	MAE			MSE			RMSE		
	Lin	SVR	MLP	Lin	SVR	MLP	Lin	SVR	MLP
Exp 1-1, $f_{2,1}^{com,net}$	1.1	0.7	0.09	2.6	3.2	0.01	1.6	1.7	0.1
Exp 1-1, $f_{1,2}^{net,com}$	0.36	0.4	0.17	0.17	0.37	0.04	0.47	0.65	0.21
Exp 2-2, $f_{2,1}^{com,net}$	2.2	1.2	0.4	10.4	3.2	0.4	3.2	1.8	0.61
Exp 2-2, $f_{1,2}^{net,com}$	1.7	1.17	0.09	6.4	5.6	3.7	2.5	2.8	1.9
Exp 2-3, $f_{2,1}^{com,net}$	1.35	0.79	0.03	4.16	5.43	0.002	2.04	2.33	0.053
Exp 2-3, $f_{1,2}^{net,com}$	2.05	2.58	0.57	7.94	15.71	0.64	2.81	3.96	0.80

regression models is that if the placement variables $x_{v,m}$ are assumed to be known, the resource allocation problem $[P_2]$ becomes a linear program (LP) for which efficient algorithms exist to generate the optimal solution in polynomial time. Table 4.1 presents the performance of different regression models, for six coupling function examples of the watchlist and object detection applications, with coupling data shown in Fig. 4.3, 4.6 and 4.7. We use the polynomial kernel for the SVR model with γ parameter of 10 and the MLP has a hidden layer of size 100 and uses $relu$ as an activation function. It can be observed that the MLP regression model outperforms the SVR and linear regression models. However, the linear model is simple and useful for a special case of $[P_2]$ to become a LP as discussed earlier. In Section 4.4.4, we illustrate that the usage of the linear model for the coupling functions results in significant resource saving although it has limited prediction capability compared to SVR and MLP models.

4.4.4 Numerical Results

Our experimental setup is shown in Fig. 4.8, where we have wireless gateways from Multitech [64] and Access Point (AP) from Celona [65]. User Equipment (UE) connects over private 5G to AP. 5G core and MEC servers are in our internal LAN and the core is remotely

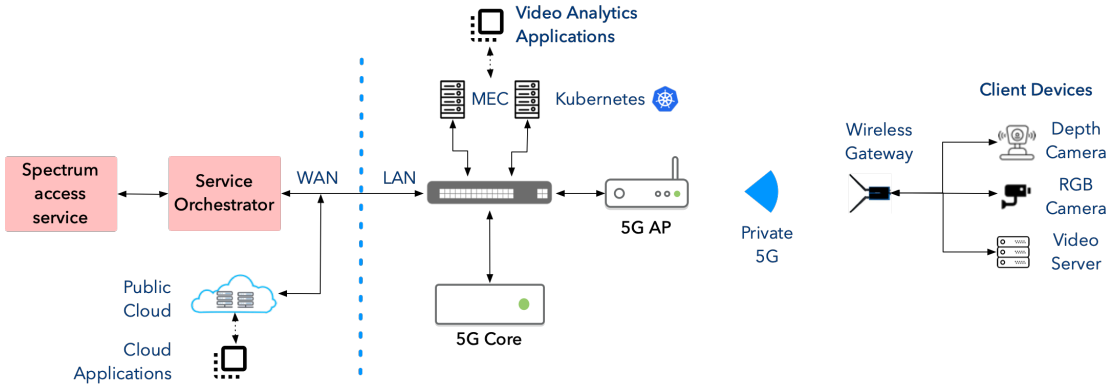


Figure 4.8: Experimental Setup

configured using Celona's Service Orchestrator. Control and Data plane traffic from AP is terminated at the core. In our MEC setup, we have one master and three worker node servers. The master node is equipped with a 10-core Intel core i9 CPU and the three worker nodes are equipped with 24-core Intel CPU and with NVIDIA RTX 2080 Ti GPUs. Kubernetes [66] cluster is set up on our MEC servers and both our use cases i.e. video surveillance (watchlist application) and intelligent transportation systems (object detection application) run within pods in Kubernetes. Each function runs as a separate pod and multiple replicas of these pods are created, as necessary. We stream videos from a video server using ffmpeg [67] and they are processed in MEC servers on a Kubernetes cluster, within pods. We use GNU Linear Programming Kit (GLPK) [68] solver for the optimization problem.

We present our experimental results for the two use cases i.e. video surveillance and intelligent transportation system. The baseline of the proposed resource orchestration framework for microservices-based applications (referred to as **ROMA**) against a static solution that ignores the introduced coupling relationships in the resource orchestration phase.

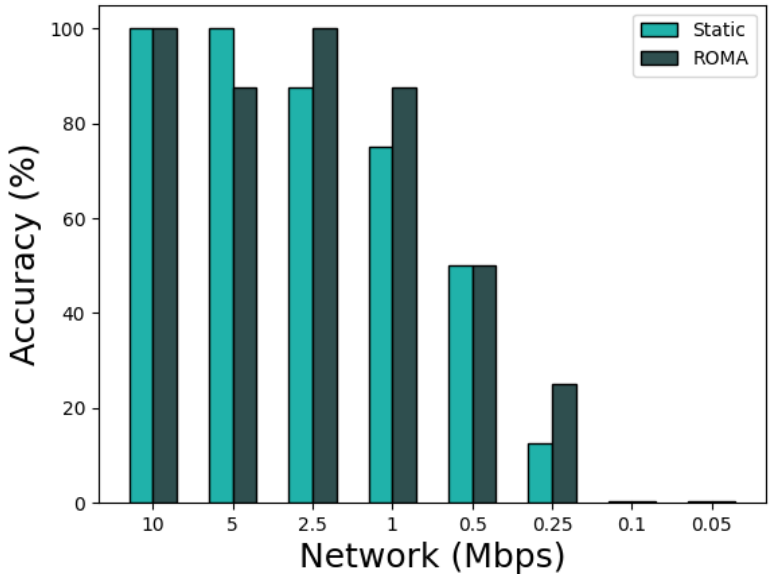


Figure 4.9: Performance of Watchlist Application: Detection Accuracy

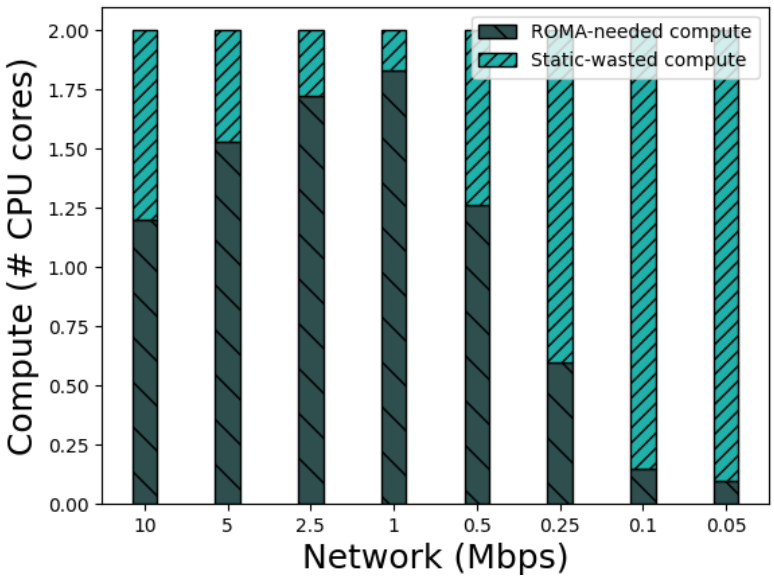


Figure 4.10: Performance of Watchlist Application: Resource Usage

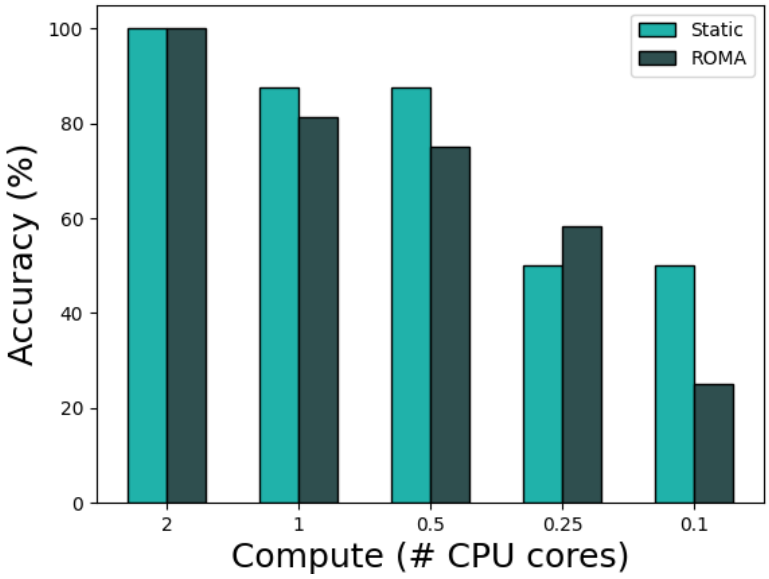


Figure 4.11: Performance of Watchlist Application: Detection Accuracy

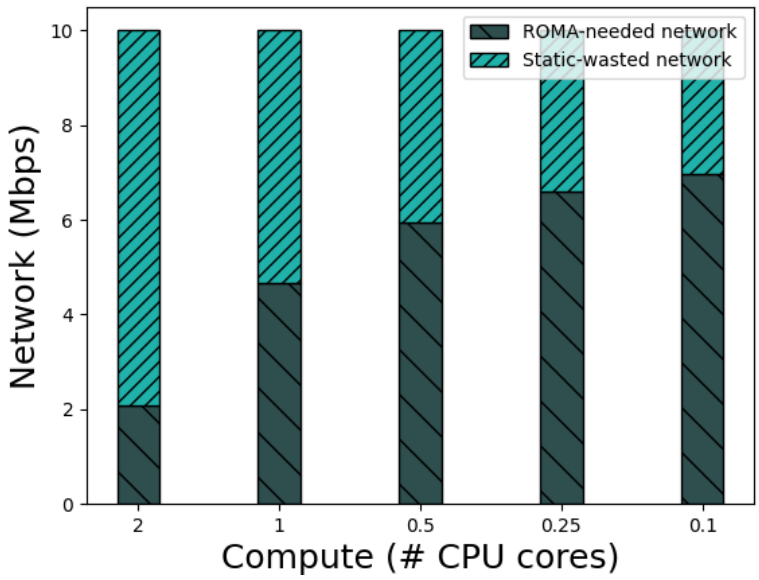


Figure 4.12: Performance of Watchlist Application: Resource Usage

4.4.4.1 Video Surveillance

For the video surveillance use case, we deploy the watchlist application depicted in Fig. 3.1. We consider a sample video including different people and compare the performance of two resource allocation strategies, **ROMA** and a static allocation in which the amount of computing allocated to the face detection function is fixed to 2 cores. The results are averaged over 3 runs. Fig. 4.9 shows the average application accuracy as the network experiences congestion and the bandwidth drops from 10 $Mbps$ to 0.05 $Mbps$. In Fig. 4.10, the compute resource usage is compared for the two schemes as the network bandwidth changes. It is observed that compared to the static resource allocation scheme, **ROMA** is able to reduce the compute resource usage up to 90%, hence preventing overprovisioning while maintaining the application accuracy, by exploiting the network-compute coupling relationship. It is important to note that the slight variation in the application accuracies in Fig. 4.9 is mainly due to the fact that different sets of frames may be processed in each case because some frames are dropped by the video ingress function until the CPU is released to process the next frame, and that is why we need to average the results over multiple runs. Similar results are shown in Fig. 4.11 and 4.12 when the compute resource changes, and the application accuracy and network resource usage are compared for **ROMA** and the static scheme. In particular, according to Fig 4.12, **ROMA** reduces the bandwidth up to 80% while the application accuracy remains acceptable.

4.4.4.2 ITS

In this experiment, we implement the object detection application shown in Fig. 4.2. We consider the used videos in Exp 2-2 and Exp 2-3 in Fig 4.6 and 4.7. The goal is to detect the

person or car objects in the videos. Fig. 4.13 demonstrates the application performance in terms of the object detection score, which we define next. The amount of used compute resource, as the network condition changes is also shown in Fig 4.14. Since not all the video frames are processed at each experiment instance, and because the number of objects differs in various video frames, we define the following weighted score (which is different from confidence score) for the object detection application:

$$score = \sum_{f \in FRAME} w_f \frac{TP_f}{GT_f} \tag{4.9}$$

where f, TP_f and GT_f denote the frame index, the number of true positives in frame f and the number of ground truth objects in frame f. Moreover, $FRAME$ denotes the set of processed frames. We use the intersection over union (IoU) metric to measure the overlap between the detected and ground truth bounding boxes. The IoU threshold is predefined as 0.5 and the predictions with an IoU of 0.5 and above are classified as TP. Fig. 4.13 and 4.14 illustrate that **ROMA** trades off a small degree of performance for significant compute resource saving by leveraging the network-compute coupling relationship in the resource allocation. The amount of compute resource usage is reduced up to 50% in this case. In Fig. 4.15 and 4.16, the object detection score and the network resource usage are shown respectively as the available compute resource varies. It is observed that **ROMA** outperforms the static scheme in terms of compute resource usage by saving up to 95% of the network bandwidth (in the case that the available number of CPU cores is 1), while the object detection score is comparable with the over-provisioned static solution. We now discuss the same results for the car detection application. According to Fig. 4.17 and 4.18, as the network bandwidth varies, **ROMA** is able to save on

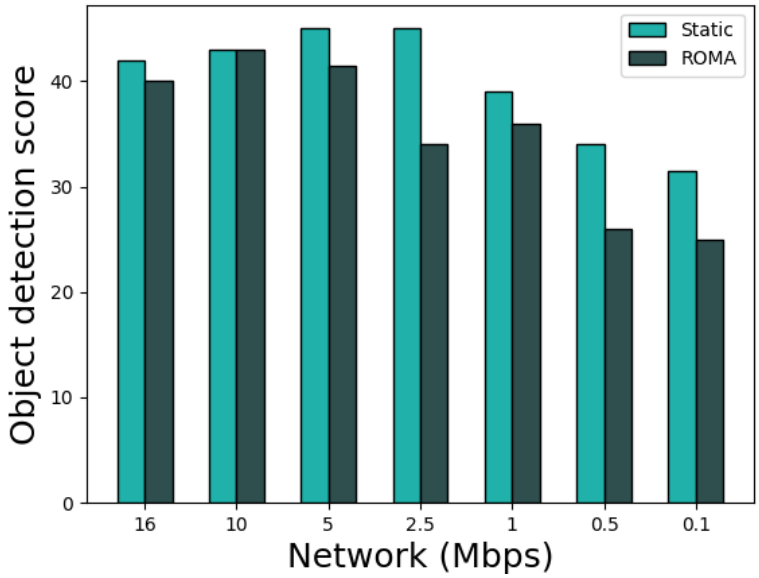


Figure 4.13: Performance of Object (person) Detection Application: Detection Score

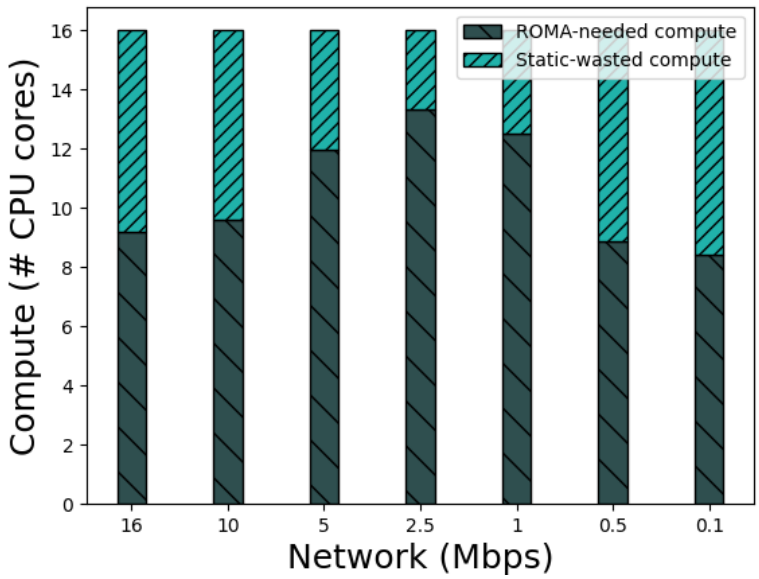


Figure 4.14: Performance of Object (person) Detection Application: Resource Usage

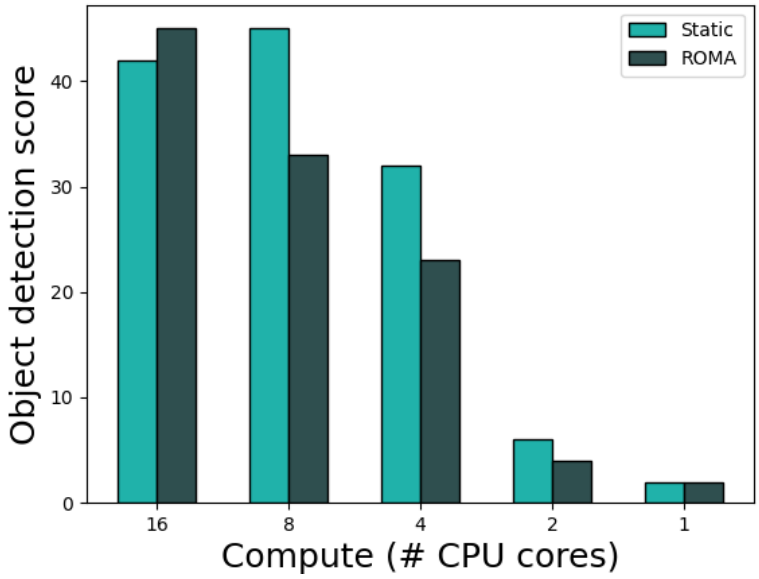


Figure 4.15: Performance of Object (person) Detection Application: Detection Score

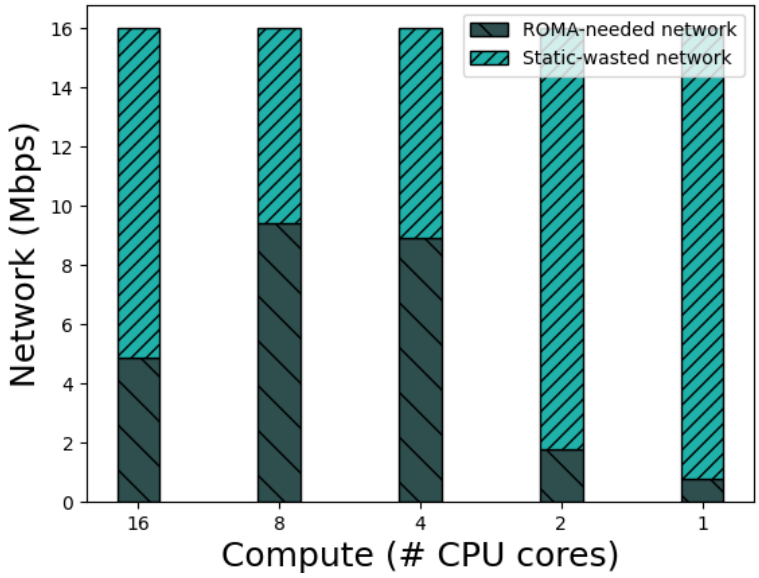


Figure 4.16: Performance of Object (person) Detection Application: Resource Usage

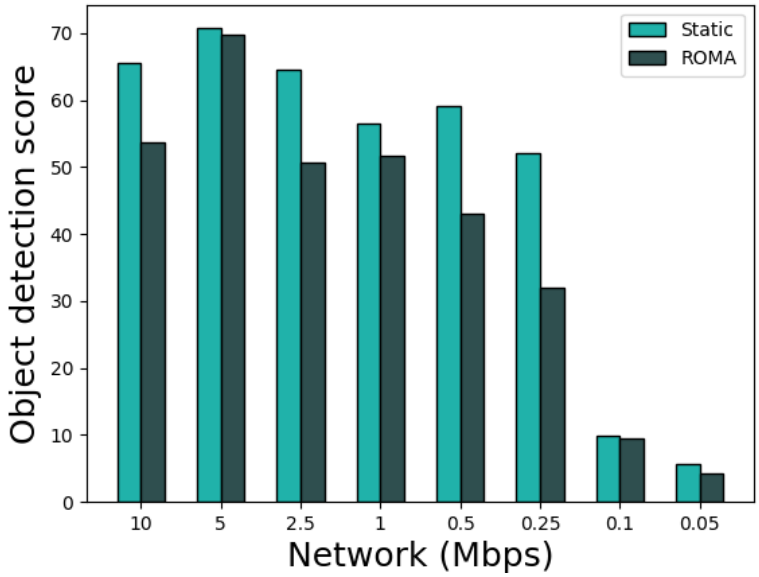


Figure 4.17: Performance of Object (car) Detection Application: Detection Score

the compute resource usage up to 44% compared to the static approach. The detection score remains within an acceptable range of the static solution except for the single case of 0.25 $cores$. From Fig. 4.19 and 4.20, as the compute resource changes from 16 $cores$ to 2 $cores$ for the car detection experiment, the network bandwidth is remarkably saved up to 75% in the case with 2 $cores$.

4.5 Dynamic Reservation of Resources by using Reinforcement Learning

In this section, we propose a Reinforcement Learning-based online method to dynamically adjust an application's compute and network resource reservations to minimize under-utilization of requested resources while ensuring acceptable service quality metrics. We observe that a complex application-specific coupling exists between the compute and network usage of an application. In the method presented in Section 4.4, an offline pre-processing is required to

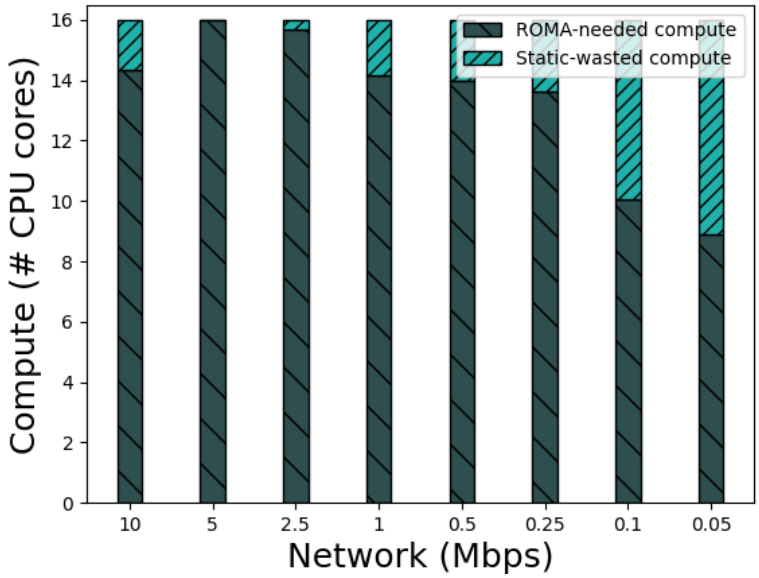


Figure 4.18: Performance of Object (car) Detection Application: Resource Usage

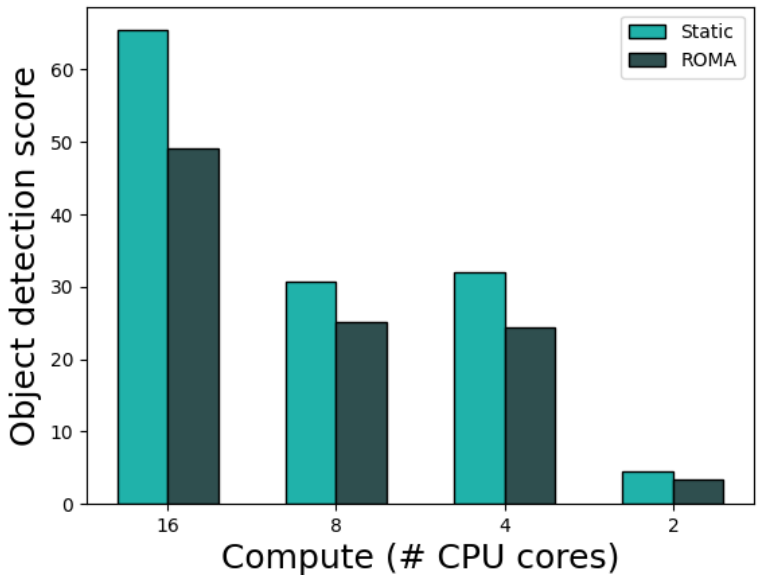


Figure 4.19: Performance of Object (car) Detection Application: Detection Score

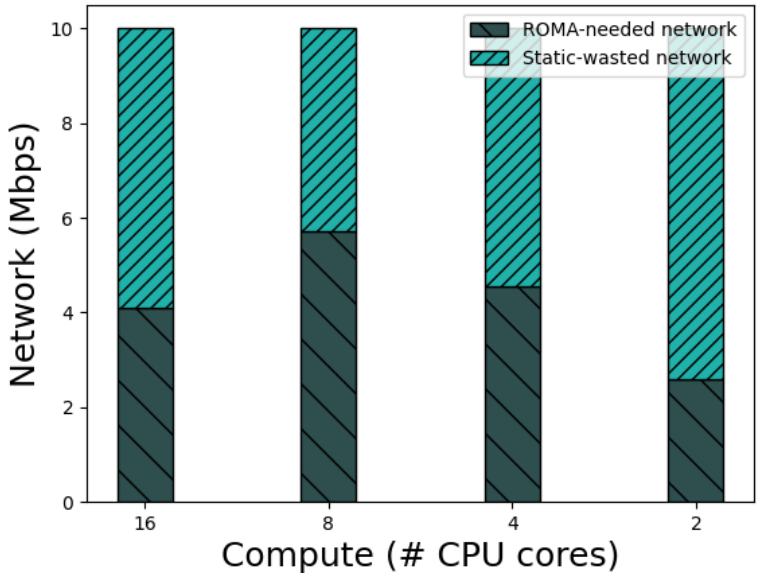


Figure 4.20: Performance of Object (car) Detection Application: Resource Usage

learn the application-specific coupling functions and then use them in the resource allocation phase. The coupling relationship derivation problem becomes even more severe at a large-scale deployment e.g. city-scale as shown in Fig. 4.21, when there are hundreds to thousands of IoT sensors, each continuously producing data stream, which needs to be transmitted over 5G for local/remote processing. In such scenarios, managing function-level resources for processing all these data streams in a dynamic environment is a very challenging task. Our proposed method in this section learns these coupling functions during the operation of the service, and dynamically modulates the compute and network resource requests to minimize under-utilization of reserved resources.

We now discuss the motivation behind dynamic resource reservation for microservices-based video analytics applications. Fig. 4.22a and Fig. 4.22b show the required compute and network resources for the object detection pipeline of Fig. 3.4 and for a real-world video that has a varying number of objects (traffic participants like vehicles, pedestrians, etc.) in different video

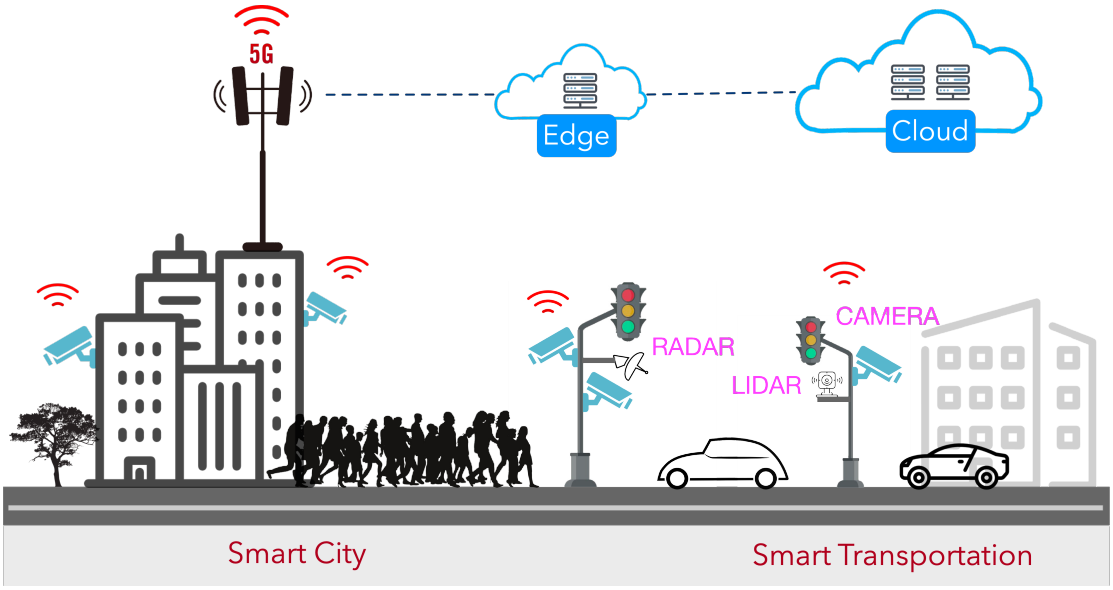
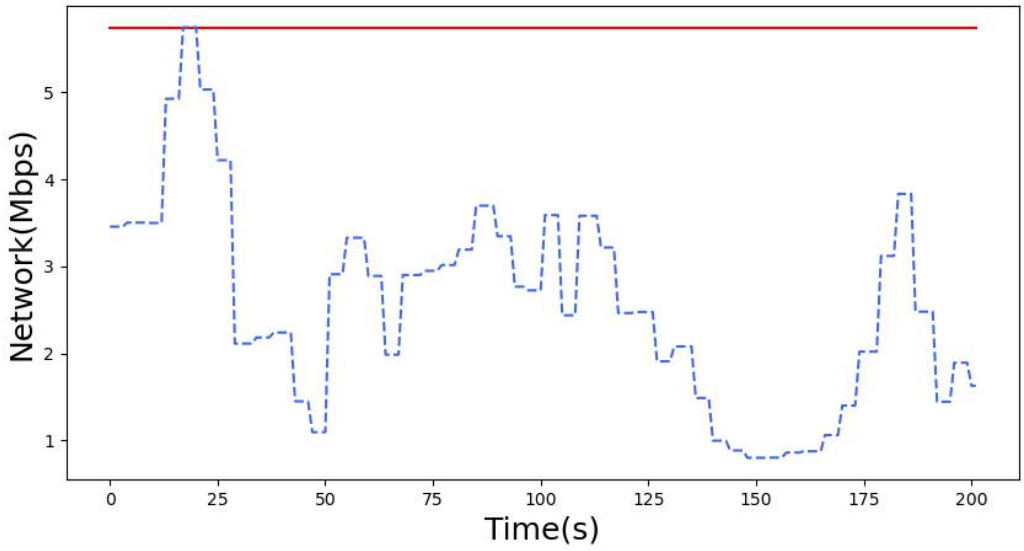


Figure 4.21: City-scale IoT Sensors Deployment

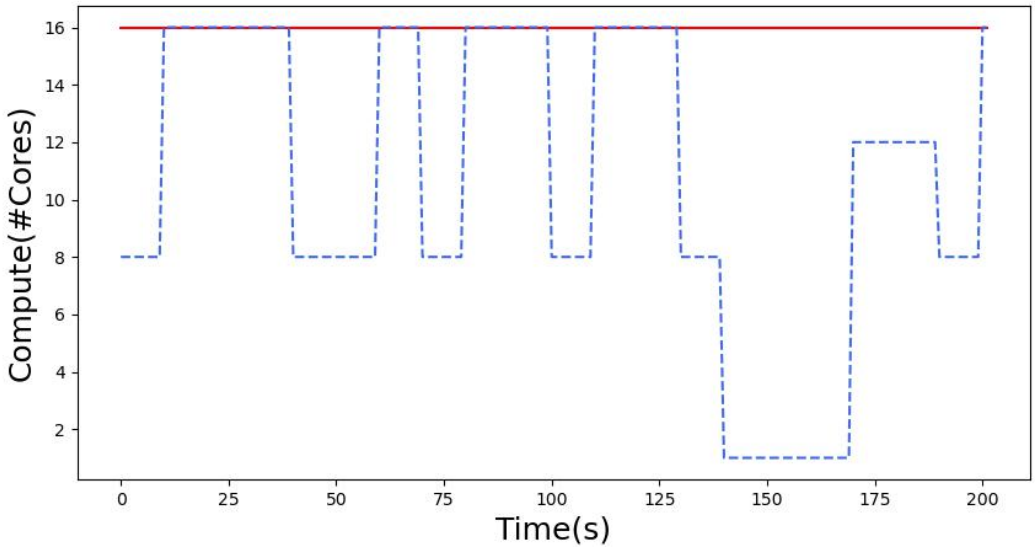
frames. We also show the number of cores allocated to the object detection microservice, and the network bandwidth required to stream the video. We denote the strategy of one-time fixed resource reservation (assuming infrastructure is able to support this) as ”static” in the rest of this section.

4.5.1 Impact of Environment and Stream Content on Resources Required

Fig. 4.22a shows that the required network bandwidth (in Mbps) for the transportation video stream varies over time. Typically, when the video stream does not have much variation across frames, the network bitrate drops, while if there is significant variation in the video stream from one frame to another, then the bitrate is high. For example, in nighttime conditions, when there is not much activity going on, the bitrate drops, while for the same camera, during the daytime when lots of people are walking around, the network bitrate goes high. This is an artifact of the way network cameras encode, compress, and stream videos. Such variation in the network



(a) Network Resource Requirement



(b) Compute Resource Requirement

Figure 4.22: Impact of Environment and Video Stream Content on Resources Required

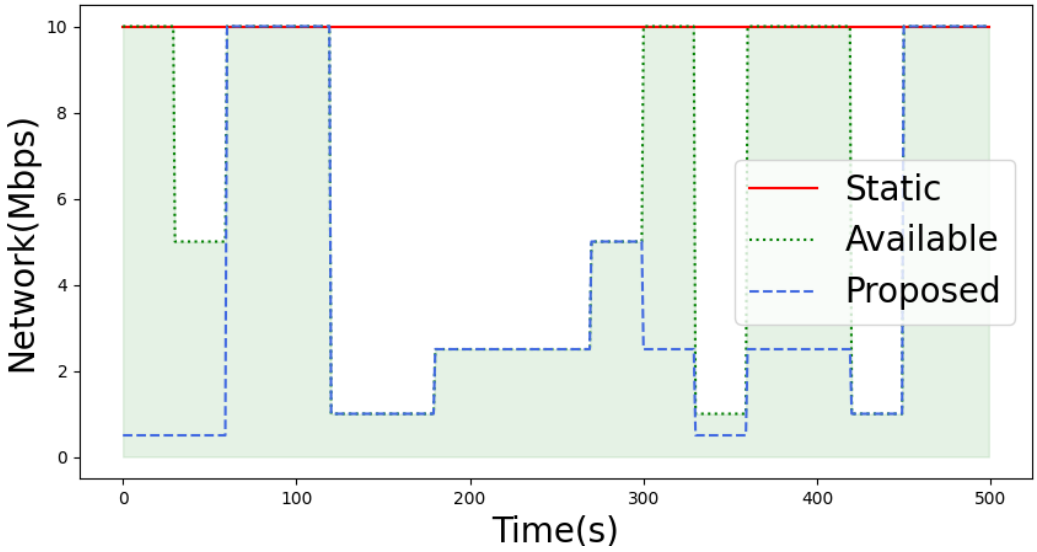
bitrate can be leveraged in appropriately reserving network resources. Instead of one-time, fixed network reservation, we can adjust the network resources dynamically as the environment and stream content i.e. scene in front of the camera changes.

As the network bandwidth usage varies, we also observe a variation in the compute resources required to process the video stream. Fig. 4.22b shows the minimum amount of compute resources required to achieve similar accuracy as over-provisioned, a fixed amount of compute resources. We observe that there is an opportunity to save on compute, without impacting application accuracy. For example, at about 150 seconds into the video stream, the network bitrate drops, and at that time, the amount of required compute also goes down. It does not benefit the application to reserve more compute resources because there is not much content to process. Thus, we can save on compute resources in reaction to changes in the environment and stream content, without impacting application accuracy.

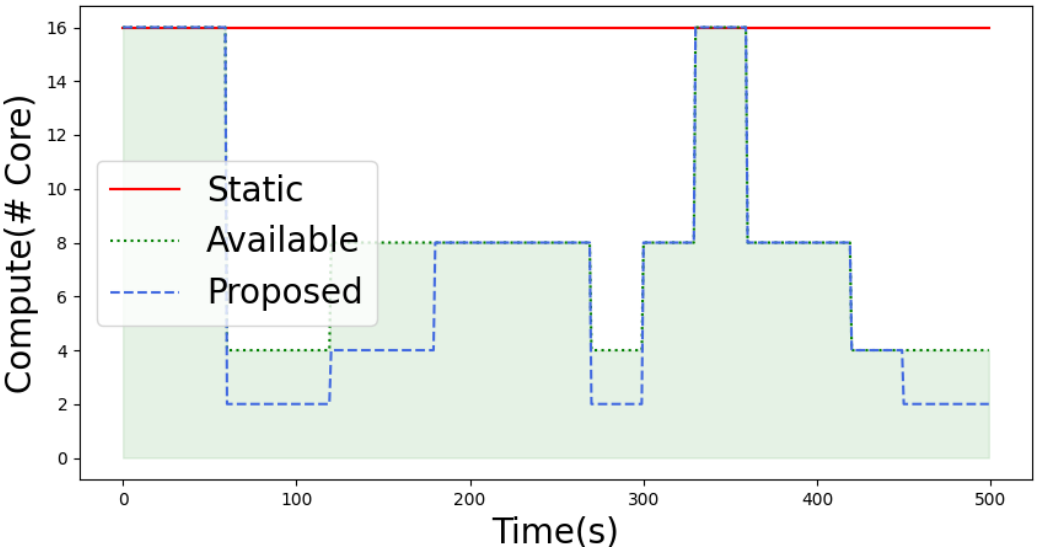
4.5.2 Impact of Dynamic Infrastructure on Resources Required

In Section 4.5.1, we studied the impact of environment and stream content on the resources required by an application by assuming that the infrastructure can adequately satisfy the resource requests at all times. However, in practice, this may not be the case always. Since the infrastructure is common and is shared across multiple applications, it is not always possible for the infrastructure to satisfy all resource requests. In this section, we study the impact of changes in infrastructure conditions on resource requests by an application.

In Fig. 4.23a and 4.23b, we show the case when infrastructure is not abundant for the network as well as compute resources. This infrastructure condition is depicted as “Available”,



(a) Network Resources Required vs. Resources Made available by Infrastructure



(b) Compute Resources Required

Figure 4.23: Impact of Dynamic Infrastructure on Resources Required

which is the maximum network or compute infrastructure available (shown in light green color in Fig. 4.23a and 4.23b). Now, the application can only reserve within these infrastructure limits. The number of resources required by the application (within the "Available" resources) is denoted as "Proposed". We observe that changes in infrastructure conditions directly impact the reservations that an application can make. For example, at around 120 seconds, when the available network from the infrastructure drops and even though the application would have desired to have a higher network reservation, the infrastructure is not able to provide it, then there is no point in reserving high compute, even though the infrastructure can provide it. Here we see that the "Proposed" compute resource goes down. Thus in these scenarios, reserving lower compute than what the infrastructure can provide, will save on the compute resource reservation. This is true vice versa as well i.e. if the compute that the infrastructure can provide drops, even though the application would have desired to be higher, then there is no point in reserving a high network because there isn't enough compute available to process the additional content. Thus, we see that infrastructure conditions do impact required resources and we can save on compute and/or network in reaction to changes in infrastructure conditions.

4.5.3 System Model

In this section, we present our system model. Given a multi-tiered infrastructure consisting of computing nodes at different tiers (edge, and central cloud), the reservation of the resources to different applications is realized through slicing. Let $\mathcal{M}$ and $\mathcal{T}$ denote the set of compute nodes and the set of different resource types on each node respectively. Each node $m \in \mathcal{M}$ is specified by $(\boldsymbol{g}_m, tier_m)$, where $\boldsymbol{g}_m = [g_m^t, t \in \mathcal{T}]$ is the vector of available resources, and $tier_m$ denotes

the associated tier.

We model an application as a set of functions or microservices and interconnections that represent the data dependency between functions. Let $G = (V, E)$ be the graph representing the application, where V denotes the set of functions in the application and E represents the interconnections (data dependency) between functions. Furthermore, $\mathcal{R}_v = (\omega_v, \mathcal{C}_v)$ denotes function v's requirements, where ω_v and $\mathcal{C}_v = (core_{min,v}, tier_v)$ are the networking and computing requirements. Our goal is to optimize the resource reservation for different functions and interconnections between them such that the overall application performance is maximized with a minimum amount of total resources.

4.5.4 Problem Formulation

Since the demands for different resources fluctuate over time as discussed in the previous section, a dynamic resource allocation approach is necessary to address the adjustments in the resource usage or placement decisions, taking into account the resource coupling relationships. We propose an RL-based orchestration system that automatically derives the resource coupling relationship and selects the best action periodically. We compared Q-learning and SARSA [69] algorithms and found that the learned model by SARSA has better performance. We assume that the available amount of resource t on node m is quantized into L levels, denoted by the set $\mathcal{G}_m^t = \{g_{m,1}^t, ..., g_{m,L}^t\}$. Let $y_{v,m}^t$ denote the amount of resource $t \in \mathcal{T}$ of node $m \in \mathcal{M}$ allocated to function $v \in V$. A valid resource allocation solution must satisfy node capacity constraints

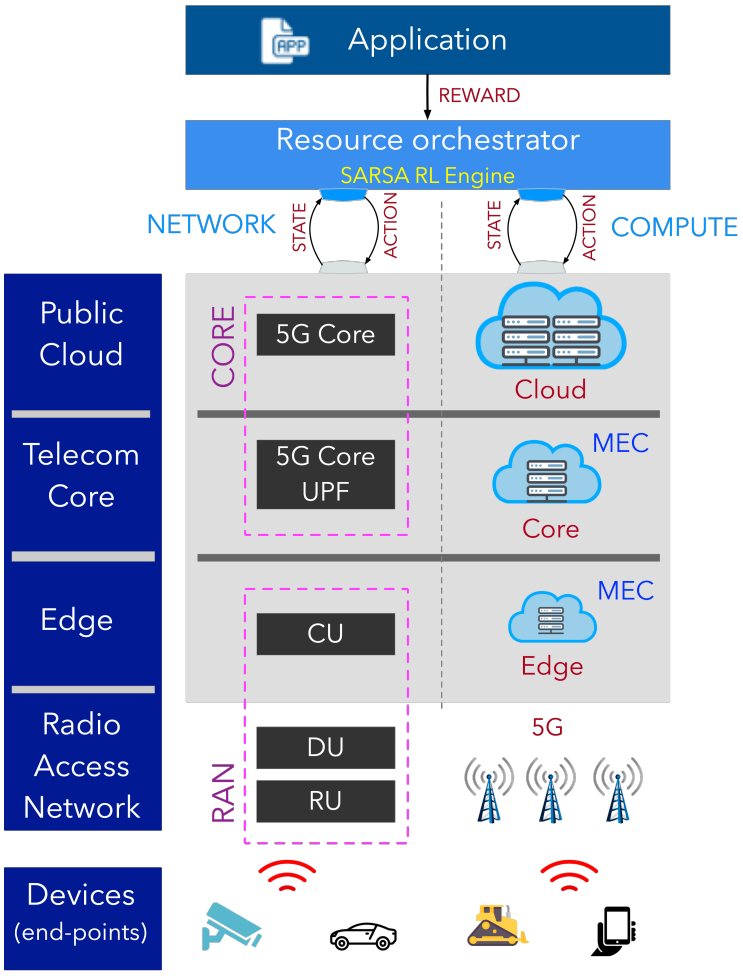


Figure 4.24: System Design for RL-based Resource Reservation

given by:

$$y_{v,m}^{t} \leq g_{m}^{t}, \quad \forall t \in \mathcal{T}, v \in V, m \in \mathcal{M} \tag{4.10}$$

We formulate the resource allocation problem as an episodic RL algorithm so that the infrastructure nodes' capacities are not violated. Our system design is shown in Fig. 4.24. We formulate the decision making process as an MDP, denoted by the tuple $< o, a, r >$, which is detailed as follows:

- **State representation** The state of the system at time step i is represented by the tuple

$o_i = (\{g_m^{t,i} \in \mathcal{G}_m^t, m \in \mathcal{M}, t \in \mathcal{T}\}, \{y_{v,m}^{t,i}, v \in V, m \in \mathcal{M}, t \in \mathcal{T}\})$. In this section, we assume that the function placement decisions are given according to heuristic solutions such as [63], and focus on the resource allocation problem. As a result, the size of state space is also reduced.

- **Action representation** An action is a valid resource reservation that determines the number of resources that an infrastructure node hosting an application microservice consumes. We define the action set to include $A = 5|V|$ actions capturing the five possible actions for each function: (a) increase/decrease the allocated network/compute resources, or (b) not change compute and network reservations.

- **Reward function** In RL, the learning agent improves its performance by constantly receiving rewards from the environment. To increase the probability of good actions, a positive reward is returned for valid actions. To this end, we define $d_i = \frac{\alpha p_{i+1}}{\sum_{t,v,m} \beta_t y_{v,m}^{t,i+1}} - \frac{\alpha p_i}{\sum_{t,v,m} \beta_t y_{v,m}^{t,i}}$ where p_i is the performance metric of the target application (object detection score defined in Section 3.6 for an object detection application). The parameters α and β_t's are used for the tradeoff between compute, network, and performance metric. d_i has a positive value only if the difference between the fraction of performance over total used resources from step i to $i+1$ is positive. The RL reward function is defined as follows:

$$r_i = r(o_i, a_i, a_{i+1}) = \begin{cases} d_i & \text{if (4.10)} \\ \text{-H} & \text{ow} \end{cases}$$

where H is a large positive number. In other words, the agent receives a penalty if it

reserves resources such that the capacity constraint for infrastructure nodes is violated.

We use SARSA algorithm which is an on-policy technique in which the Q-values are updated as follows:

$$Q(o_i, a_i) \leftarrow Q(o_i, a_i) + \alpha[r_i + \gamma Q(o_{i+1}, a_{i+1}) - Q(o_i, a_i)] \tag{4.11}$$

At each time step, the agent computes and stores the Q values in a Q-table which can be regarded as a long-term reward.

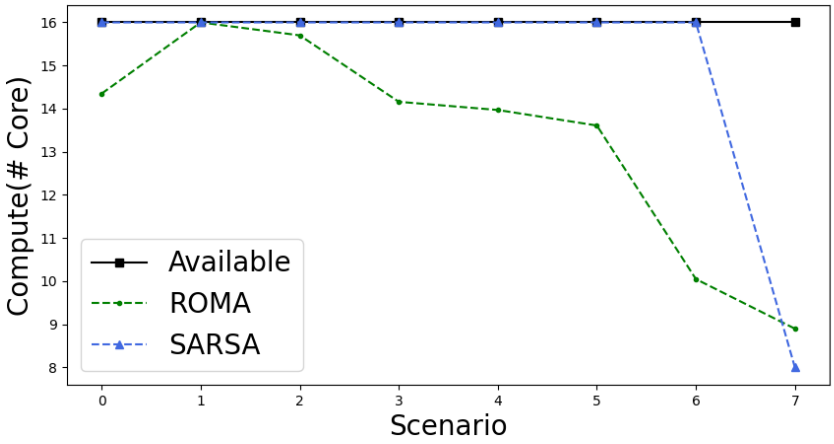
Algorithm 2 *SARSA method for Dynamic Resource Reservation of Multi-Component Applications*

```
Initialize Q(o, a)
for each episode do
    Choose a random state o_i
    Choose a_i from o_i derived from Q(o, a)
    for each step do
        Execute a_i, observe reward r_i and next state o_{i+1}
        Choose a_{i+1} from o_{i+1} derived from Q(o, a)
        Update Q(o_i, a_i) from (4.11)
    end for
end for
```

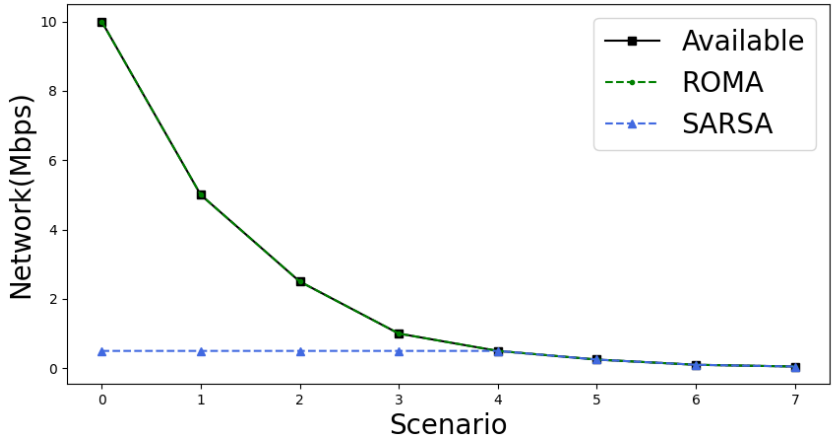
4.5.5 Numerical Results

We benchmark the RL resource orchestration solution, against a static resource reservation scheme, which ignores the coupling between resources and environmental changes. We also compare the performance of the RL solution with the ROMA solution presented in Section 4.4 in an offline setting. Furthermore, we show the effectiveness of our technique in an online setting on a real-world video analytics application. The experimental setup is the same as Section 4.4.

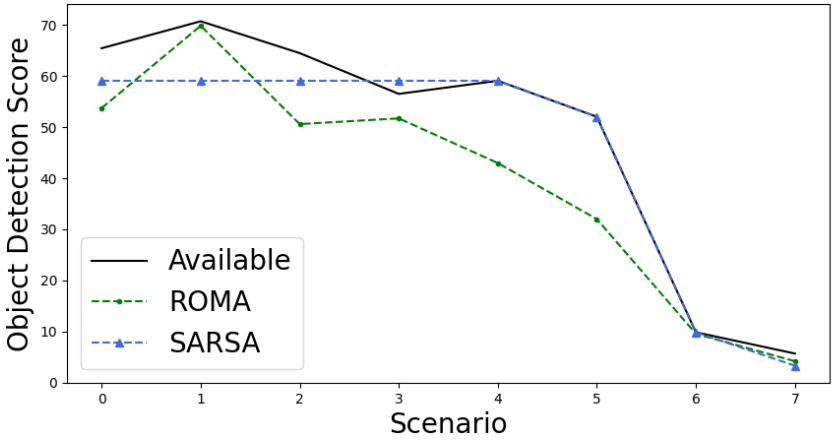
We present our results for the ITS use case. The goal is to detect vehicles or pedestrians in



(a) Exp 1: Compute Resource Reservation

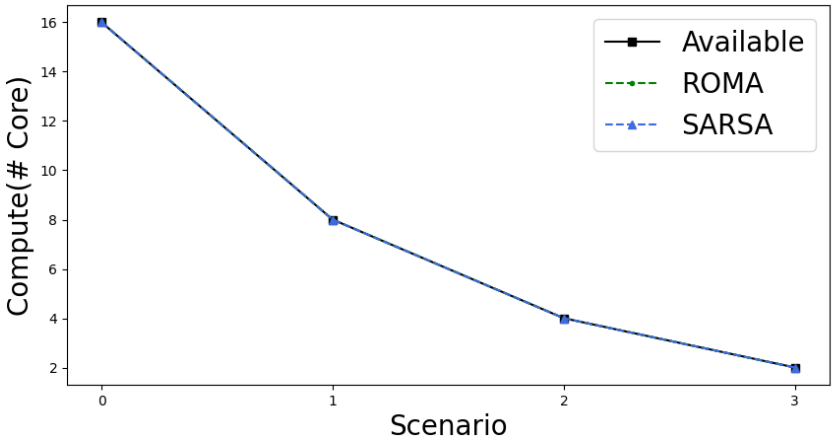


(b) Exp 1: Network Resource Reservation

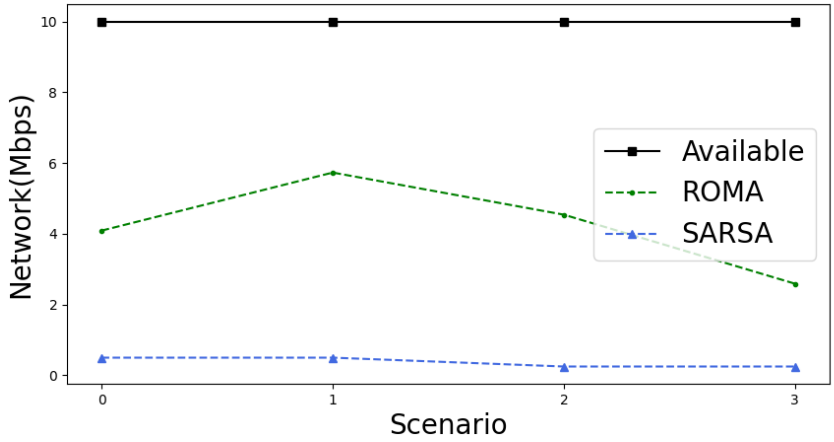


(c) Exp 1: Object Detection Score

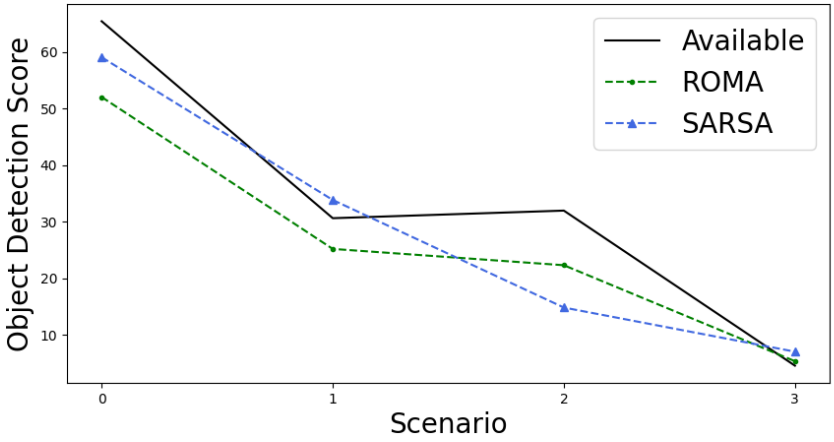
Figure 4.25: Performance of Object (car) Detection Application under Network Variation: SARSA vs. ROMA



(a) Exp 2: Compute Resource Reservation



(b) Exp 2: Network Resource Reservation



(c) Exp2: Object Detection Score

Figure 4.26: Performance of Object (car) Detection Application under Compute Variation: SARSA vs. ROMA

the video streams.

4.5.5.1 Performance Comparison: Available, ROMA and SARSA

We implement the proposed RL-based resource reservation approach (denoted as SARSA) for a car detection application and evaluate its capability to capture the non-linear resource coupling relationships and save on resource reservation. For application accuracy metric, we define a weighted object detection score (different from confidence score) as $object\ detection\ score = \sum_{f \in FRAME} w_f \frac{TP_f}{GT_f}$, where f is the frame index, $FRAME$ is the set of processed frames, TP_f is the number of true positive objects and GT_f is the number of objects in ground truth in frame f. For TP classification, we use intersection over union (IoU) metric to measure the overlap between detected and ground truth objects and detections with IoU over 0.5 are considered as TP. We consider two different experiments and in each experiment, we consider several scenarios with given available compute and network resources.

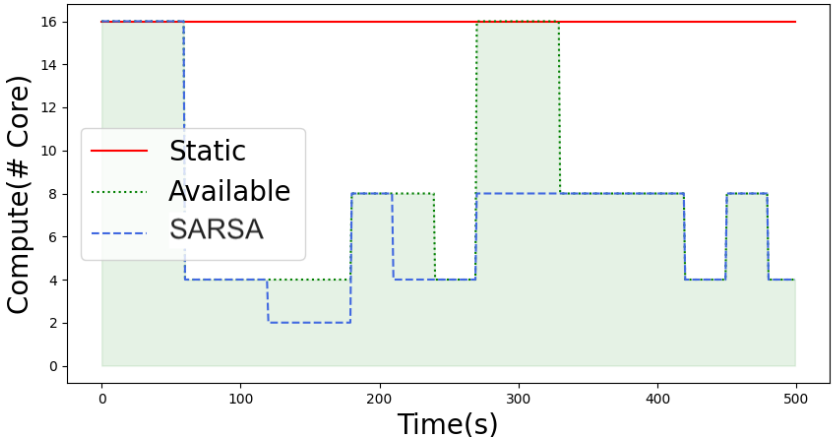
Experiment 1: In this experiment, we consider 8 scenarios in which the available compute is fixed and the available network is reduced gradually (network becomes bottleneck). Fig. 4.25a, 4.25b and 4.25c illustrate the compute and network reservation and the object detection score respectively. We see that across all scenarios, the object detection score (shown in Fig. 4.25c) for SARSA is comparable or slightly lower than "Available", which is when available compute and network is used. Now, for such good application accuracy, we see that SARSA reserves significantly less network resource (shown in Fig. 4.25b) than maximum available and when the available network is really low, SARSA also brings down the compute (scenario 7 in Fig. 4.25a) reservation. Compared to ROMA, SARSA has better accuracy, saves a lot on network

but looses slightly on compute. Thus, we see that for similar or slightly lower application accuracy compared to "Available", SARSA is able to save a lot on compute and network resource reservation.

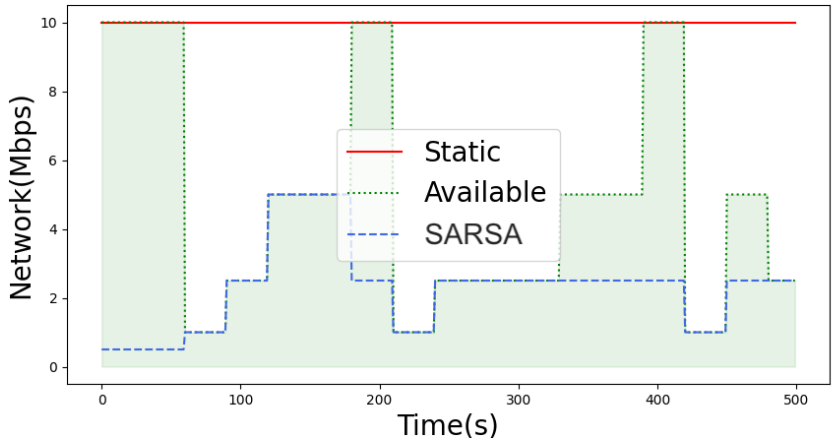
Experiment 2: In this experiment, we consider 4 scenarios in which we fix the available network resources and gradually reduce the available compute resources (compute becomes bottleneck). Fig. 4.26a, 4.26b and 4.26c illustrate the compute and network reservation and the object detection score respectively. Again, we see that SARSA has comparable or slightly lower object detection score (shown in Fig. 4.26c) than "Available". SARSA achieves this similar application accuracy at significantly lower network reservation (shown in Fig. 4.26b). The compute reservation by SARSA (and ROMA) is same as the "Available". Compared to ROMA, SARSA saves a lot on network resource reservation and most of the time gives better application accuracy, while compute reservation is the same. Thus, we see that for similar or slightly lower application accuracy compared to "Available", SARSA is able to save significant network reservation.

4.5.5.2 Performance of SARSA in an Online Setup

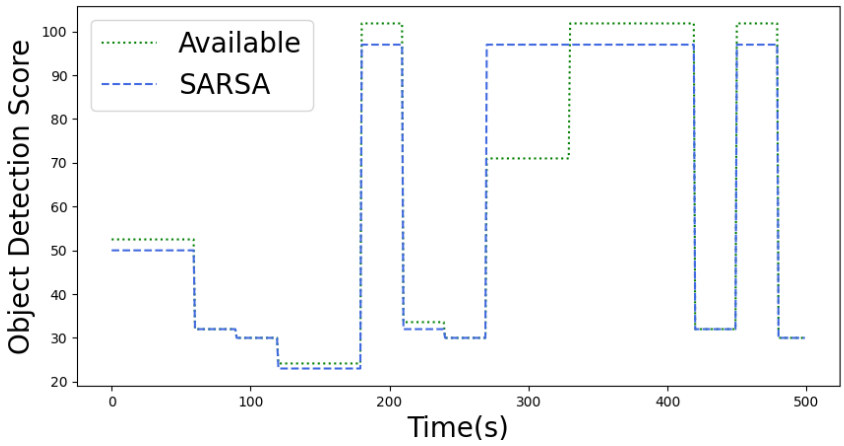
In this section, we evaluate performance of SARSA in an online setup where available compute and network varies over time based on a discrete uniform distribution on compute and network space. We run 15 experiments and observed that on average, SARSA reserves upto 93% less network and 65% less compute resources than "Available" compute and network resources. Fig. 4.27a, 4.27b, and 4.27c illustrate the compute and network reservation and the object detection score respectively for a single run. We see that for almost same object detection score (shown in Fig. 4.27c), SARSA saves on network and compute resource upto 50% and 95%,



(a) Compute Resource Reservation



(b) Network Resource Reservation



(c) Object Detection Score

Figure 4.27: Performance of SARSA for Object (person) Detection Application

respectively. Thus, in real-world deployment, we show that our Reinforcement Learning-based online technique is quite effective in capturing the compute and network coupling relationship and is able to significantly reduce network and compute resource reservation.

4.6 Conclusion

In this chapter, we introduce a novel method to solve the resource orchestration problem of multi-component applications from the perspective of application developers, based on resource coupling relationships. The introduced resource coupling modeling captures the inter-relation between the usage of different resources and its impact on an application performance. By exploiting resource coupling relationships, we propose a framework for the deployment of microservices-based 5G applications in a dynamic, heterogeneous, multi-tiered compute and network infrastructure. Our proposed solution ensure optimal deployment, such that end-to-end application requirements are met. By implementing two real-world IoT applications, one in video surveillance domain and another one in intelligent transportation systems domain, we show that our solution is able to reduce compute and network resource usage remarkably while maintaining the application performance.

Chapter 5: Mobile Network Performance Enhancement using Aerial Relaying Systems

5.1 Overview

Over the past decade, UAVs have been adopted in a broad range of application domains, due to their autonomy, high mobility, and low cost. Historically, UAVs have been primarily used in the military, usually deployed in hostile territory to reduce the risk for aircrew. Recent advances in UAV technologies have made them more affordable and accessible to civilian and commercial applications such as cargo transport, emergency search and rescue, precision agriculture, commercial package deliveries, etc. Moreover, UAVs are seen as a promising solution for next-generation wireless networks because of their inherent advantages, including flexible and fast deployment and reconfiguration, as well as a higher chance of having line-of-sight (LoS) links leading to less impaired communication channels compared to terrestrial wireless communication systems. According to a report from Federal Aviation Administration (FAA), the fleet of drones will be more than doubled from an estimated 1.1 million vehicles in 2017 to 2.4 million units by 2022 [70].

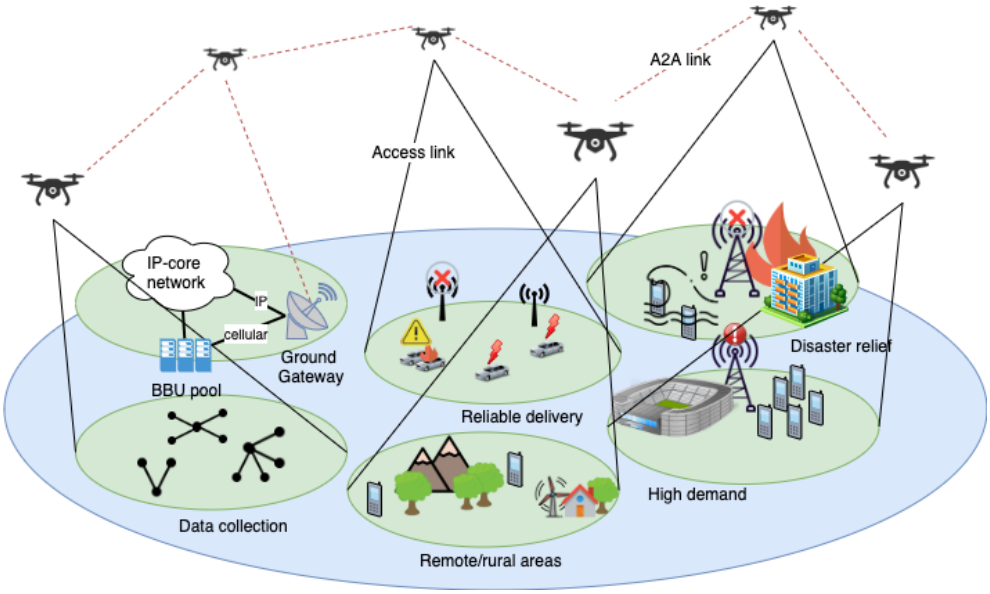


Figure 5.1: Application Scenarios of UAV-aided Networks

According to [70], UAV-aided wireless communications can fall into three representative categories of use cases; (i) UAV-aided ubiquitous coverage, (ii) UAV-aided information dissemination and data collection, and (iii) UAV-aided relaying. Focusing on the latter, communication relaying is an effective technique for network coverage extension and throughput maximization. However, a number of key challenges should be addressed in order to use UAVs as mobile relaying nodes, providing broadband communication to users (or user groups) without direct and/or reliable communication links (e.g., in disaster-hit or rural areas [71]). First, efficient algorithms should be devised to place UAVs in a 3D space. The mobility of the UAVs introduces new challenges to the network design problem compared to the traditional static relaying and fixed infrastructure schemes (e.g. WiFi access points). Moreover, in order to cover large geographical areas and because of the limited transmission range of UAVs, a swarm of UAVs is needed to route users' traffic demands through wireless multi-hop path(s). Due to the intermittent wireless links and frequent topology changes in such mobile ad hoc networks (MANETs), the traffic routing decision

should be considered together with the UAVs placement and relocation. More importantly, such decisions should be adaptive to the topology and traffic pattern changes in a timely manner.

In this chapter, we consider an aerial platform consisting of multiple UAVs that supports the traffic demand of a ground network [72], [71]. Multi-hop relaying in the next generation of wireless networks will not only facilitate the coverage of more UEs and the support of long-distance communications but also will be able to handle overloaded networks. Fig. 5.1 illustrates the architecture and application scenarios of such systems. In contrast to the majority of existing studies such as [73] and [74] that solely focus on a single UAV relay and the air-to-ground (A2G) access links with one-hop communication or at most two-hop communications considering also the UAV-to-BS links in UAV-aided cellular networks (e.g. [75]), we exploit a multi-hop aerial relaying platform. Moreover, the rate-constrained UAV-to-UAV (or Air-to-Air, A2A) communications and the connectivity between UAVs are considered in the proposed framework. Although facility placement and traffic routing are usually addressed sequentially as two separate problems, the meaningful interrelation between the two makes it more reasonable to approach them in a single model as shown in other problems such as [76]. Thus, another contribution of this section is jointly optimizing theUAV placement and the routing decisions, when the ground network is quasi-static. We also extend our approach to the case of a mobile ground network and consider the impact of the UAVs' speed, while most of the existing relaying schemes utilize static relay nodes due to the practical constraints on relay mobility and the need for high-throughput links. In order to reduce the system energy consumption and since the propulsion energy consumption of UAVs is typically significantly greater than the energy consumption for communications [77], we avoid unnecessary UAVs' relocation in subsequent snapshots in our solution. The problem is formulated as MILP. To reduce the time-complexity and enable real time re-positioning of the

mobile relays and routing decision, we propose an approximation algorithm using LP relaxation and a rounding procedure. The proposed approach assumes logically centralized network control i.e., SDN. The controller's global network view in an SDN architecture, renders centralized UAVs placement and adaptive routing strategies feasible [78]. The controller may be placed at a remote ground center, inside the ad hoc network devices as in [79] or at UAVs as shown in Fig 5.1.

The rest of this chapter is organized as follows. Section 5.2 presents the literature review. We explain system model and problem formulation in Section5.3 and Section 5.4. The numerical results are given in Section 5.5. Section 5.6 provides the chapter conclusion and future directions.

5.2 Related Work

Deployment of UAVs has extensively been a topic of research with different objectives, such as the maximization of the downlink (DL) throughput [80], [81] and DL received signal strength (RSS) [82]. In contrast, we explore a UAV-assisted communication system considering both uplink (UL) and DL traffic streams. Authors in [83], [84] investigate the usage of UAVs to maximize the covered area with respect to the UAV altitude, antenna gain and minimum received power of users. In [85], the coverage probability of a reference ground user is evaluated for a 3D UAV movement process characterized by the RWPM and uniform mobility models. In [86], authors propose a UAV-assisted cellular network and maximize the revenue, that is proportional to the number of covered users. However, in order to fully satisfy the QoS requirement of the users in a multi-hop wireless network, the end-to-end traffic delivery should be considered which is more challenging than a coverage problem. The required number of aerial UAVs is minimized in [1], [87]. Authors in [1] propose a UAV placement algorithm taking into account the connectivity

between UAVs and the clusters demands; however, the constraints on the UAVs mobility and the A2A links capacity are ignored. Authors in [88] and [78] considered multi-hop wireless backhauling in UAV-aided networks. In [88], the authors seek to form a multi-hop backhaul network in the sky connecting small ground base stations through the formation of a bidirectional tree structure. Different from [88], we consider both A2A and A2G links and jointly optimize the UAV placement and routing. In [78], the authors optimized the UAV placement, power and bandwidth allocation in an UAV-enabled multihop backhaul with fixed number of UAVs. In our case, we minimize the number of deployed UAVs in addition to imposing a constraint on the maximum number of UAVs. In contrast to [89] which investigates the trajectory design and power allocation strategies for a single fixed ground source-destination, we consider a general mobile ground network consisting of multiple traffic flows which make the UAV relays trajectory design and traffic routing more challenging and out of the scope of the set-up in [89]. Authors in [90] considering the placement and resource allocation problem of multi UAV relays for a ground network with multiple traffic flows; however they ignore the mobility of the ground nodes and how it affects the UAV locations and other decision variables.

5.3 System Model

In this section, the system model is presented followed by problem formulation. We discretize time and consider a directed graph $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$ representing the topology of the ground network at snapshot t. The vertex set $\mathcal{V}^t$ represents the wireless network nodes and the edge set $\mathcal{E}^t$ represents the wireless links. An ordered pair of nodes (u, v) belongs to the edge set $\mathcal{E}^t$ if and only if node v can receive data packets directly from node u. We assume that all node-

to-node communication is unicast, i.e. each packet transmitted by a node $u \in \mathcal{V}^t$ is intended for a unique $v \in \mathcal{V}^t$ where $(u, v) \in \mathcal{E}^t$. Each of the wireless links has a maximum capacity c_{uv}. For the sake of simplicity, the superscript t is dropped in the following.

5.3.1 Radio Propagation Model

We adopt the model proposed in [91] for the A2G propagation model, where two signal propagation groups are considered; Line-of-Sight(LoS) and Non-Line-of-Sight (NLoS). The latter corresponds to receivers with no Line-of-Sight but still having coverage via strong reflections and diffraction. Additional impairments to the radio channel are caused by scattering and shadowing from the man-made structures in the environment. The occurrence probability of LoS is given by:

$$p_{LoS} = \frac{1}{1 + aexp(-b(\frac{180}{\pi}tan^{-1}(\frac{h}{r_{n,l}}) - a))} \tag{5.1}$$

where a and b are constants depending on the environment, h is the UAV altitude and $r_{n,l}$ is the horizontal euclidean distance between the UAV l and the user equipment (UE) n. The probability of NLoS is $p_{NLoS} = 1 - p_{LoS}$ and the total A2G path loss (in dB) as a function of $r_{n,l}$ and h is:

$$\begin{aligned} L(h, r_{n,l}) &= p_{LoS}L_{LoS} + p_{NLoS}L_{NLoS}, \\ L_{LoS} &= 20log(\frac{4\pi f_c d_{n,l}}{c}) + \eta_{LoS}, \\ L_{NLoS} &= 20log(\frac{4\pi f_c d_{n,l}}{c}) + \eta_{NLoS} \end{aligned} \tag{5.2}$$

Where f_c is the carrier frequency, $d_{n,l} = \sqrt{h^2 + r_{n,l}^2}$ is the distance between UAV l and UE n, η_{LoS} and η_{NLoS} are respectively the average additional losses due to the environment.

We also assume that the A2A links are dominated by LoS components resulting in the following path loss model:

$$L(r_{u,v}) = 20log(\frac{4\pi f_c r_{u,v}}{c}) \tag{5.3}$$

Assuming that an interference-coordination mechanism among adjacent UAVs and users is available, the interference is negligible. Consequently, the coverage radii for the A2G and A2A channels, denoted by R_1 and R_2, are:

$$P_{UE} = L(h, R_1) + \gamma_{min} + \sigma_n^2 \tag{5.4}$$

$$P_{UAV} = 20log(\frac{4\pi f_c R_2}{c}) + \gamma_{min} + \sigma_n^2 \tag{5.5}$$

where h denotes the UAVs altitude. The QoS requirement is expressed in terms of the minimum received SNR at the receiver (γ_{min}), noise power (σ_n) and maximum transmission power of UAVs (P_{UAV}) and users (P_{UE}), where $P_{UE} \leq P_{UAV}$.

5.4 Joint UAV Placement and Traffic Routing

There are traffic demands between UEs given by a traffic demand matrix D, where the element d_{uv} denotes the amount of demand from the source UE u to the destination UE v. Due to

the limited transmission power of the UEs and in order to reduce the control traffic overhead required for traffic routing in MANETs, the ground network is partitioned into M clusters, denoted by C_i, $i = 1, ..., M$. Each cluster has a clusterhead (CH) which functions as a gateway, relaying the cluster's total traffic to the aerial platform. Let CH_i, $i = 1, ..., M$ denote the ith clusterhead. Given D, the inter-cluster traffic demand for a pair of clusters and from CH_i to CH_j is calculated by:

$$TD_{ij} = \sum_{u \in C_i, v \in C_j} d_{uv}, \quad \forall i, j \in \{CH_1, ..., CH_M\}$$

We denote by $i \rightarrow j$, a traffic flow originated from CH_i and destined to CH_j. The problem considered in this section entails the optimal placement of at most N_{max} available UAVs as relay nodes to support the traffic demand of the ground clusters. For each flow, $i \rightarrow j$, a collection of aerial multi-hop paths can be used to route the traffic demand of the flow. Multi-hop relaying in the next generation of wireless networks will not only facilitate the coverage of more UEs, but also will be able to handle overloaded networks.

Let $\mathcal{U} = \{u_i, i = 1, ..., |\mathcal{U}|\}$ denote the set of potential locations for UAV placement, where v_i stands for the ith location. Here, we assume that all UAVs are placed at the same altitude h; however, it is easy to extend the formulation to a 3D UAV placement. The following graphs are defined for the problem formulation:

Demand Graph We model the connectivity and traffic requirements of the ground clusters by a directed graph $\mathcal{G}_D = (\mathcal{V}_D, \mathcal{E}_D)$ where, $\mathcal{V}_D = \{CH_1, ..., CH_M\}$ is the set of all CHs, and $(i, j) \in \mathcal{E}_D$ if and only if the flow $i \rightarrow j$ exists for $i, j \in \mathcal{V}_D$.

Network Graph We introduce a directed graph $\mathcal{G}_P = (\mathcal{V}_P, \mathcal{E}_P)$ where $\mathcal{V}_P = \mathcal{V}_D \cup \mathcal{U}$ and $(u, v) \in \mathcal{E}_P$ if and only if $d_{u,v} \leq R_2$ for A2A links, and $d_{u,v} \leq R_1$ for A2G links. Given the network and

Table 5.1: System Model Parameters and Variables

Variables	Description
x_u	Binary decision variable of UAV placement at position u
f_{ij}^{uv}	The amount of (i,j) traffic d assigned to (u,v)
y_{ij}	Total unsupported traffic of the OD pair (i,j)
Parameters	**Description**
$\mathcal{G}_P = (\mathcal{V}_P, \mathcal{E}_P)$	The network graph of UAVs and ground cluster heads
$\mathcal{G}_D = (\mathcal{V}_D, \mathcal{E}_D)$	The demand graph
M	Number of ground cluster heads
$\mathcal{U}$	The set of UAV potential locations
N_{max}	Available number of UAVs
h	UAVs height
D	Traffic demand matrix of the ground network
TD_{ij}	Traffic demand between the CHs i and j
c_{uv}	Capacity of the link (u,v)

demand graphs $\mathcal{G}_P, \mathcal{G}_D$, we formulate the problem at hand considering the following decision variables:

- A set of binary variables $\boldsymbol{x}$, where x_u is set to 1 if a UAV is deployed at position $u \in \mathcal{U}$ and 0 otherwise.
- A set of continuous variables $\boldsymbol{f}$, where f_{uv}^{ij} is the amount of traffic from OD flow $i \rightarrow j$ assigned to the link $(u,v) \in \mathcal{E}_P$.
- A set of continuous variables $\boldsymbol{y}$, where y_{ij} denotes the traffic amount of the OD flow $i \rightarrow j$ that is not supported (not delivered).

A summary of the system model parameters and variables is given in Table 5.1. The proposed MILP formulation for the joint UAV placement and traffic routing (**UPR_MILP**) is as follows:

$$\text{minimize} \quad \phi \frac{\sum_{u \in \mathcal{U}} x_u}{N_{max}} + (1-\phi) \frac{\sum_{(i,j) \in \mathcal{E}_D} y_{ij}}{\sum_{(i,j) \in \mathcal{E}_D} TD_{ij}} \tag{5.6}$$

Feasibility Constraints:

$$f_{uv}^{ij} \leq x_u c_{uv} \quad \forall (i,j) \in \mathcal{E}_D, u \in \mathcal{U}, v \in \mathcal{V}_p \tag{5.7}$$

$$f_{uv}^{ij} \leq x_v c_{uv} \quad \forall (i,j) \in \mathcal{E}_D, v \in \mathcal{U}, u \in \mathcal{V}_p \tag{5.8}$$

$$\sum_{u \in \mathcal{U}} x_u \leq N_{max} \tag{5.9}$$

Flow Constraints:

$$\sum_{v \in \mathcal{V}_P} (f_{uv}^{ij} - f_{vu}^{ij}) = \begin{cases} 0 & \forall u \in \mathcal{V}_P \backslash \{i,j\}, (i,j) \in \mathcal{V}_D \\ TD_{ij} - y_{ij} & u = i, \forall (i,j) \in \mathcal{E}_D \\ -(TD_{ij} - y_{ij}) & u = j, \forall (i,j) \in \mathcal{E}_D \end{cases} \tag{5.10}$$

Capacity Constraints:

$$\sum_{(i,j) \in \mathcal{E}_D} f_{uv}^{ij} \leq c_{uv} \quad \forall (u,v) \in \mathcal{E}_P \tag{5.11}$$

Domain Constrains:

$$0 \leq f_{uv}^{ij} \quad \forall i,j \in \mathcal{V}_D, u,v \in \mathcal{V}_p \tag{5.12}$$

$$0 \leq y_{ij} \quad \forall (i,j) \in \mathcal{E}_D \tag{5.13}$$

$$x_u \in \{0,1\} \quad \forall u \in \mathcal{U} \tag{5.14}$$

The objective function (5.6) aims at jointly minimizing the cost of UAV deployment (reflected

as the number of deployed UAVs) and the total amount of requested traffic that can not be supported by the network (the total unsupported traffic). We normalize both metrics to be between 0 and 1 in order to avoid the known problem of different range values in Pareto Analysis (i.e. one metric has a large value and the other one has a small value). Since we have in our formulation two performance objectives (minimizing the number of deployed UAVs and minimizing the unsupported traffic), a full solution of the problem requires the complete tradeoff analysis between these two metrics and finding the Pareto Points or Pareto Frontier of this tradeoff problem. To arrive at equation (5.6), we employed what is known as the "scalarization method" for tradeoff analysis. This method is less computationally intensive. To fully understand the tradeoff between these two metrics using the scalarization method, we need to vary ϕ between 0 and 1. In this way, we can compute the convexified Pareto Frontier. Indeed in our experiments, we tested different values for ϕ and selected a relatively small value to promote a solution that primarily enhances the performance of the network by minimizing the unsupported traffic.

Constraints (5.7) and (5.8) guarantee that the amount of traffic assigned to an A2G link is nonzero only if a UAV is placed at the aerial end of the link. Constraint (5.9) limits the maximum number of deployed UAVs, while constraints (5.10) enforce flow conservation, i.e. the sum of all inbound and outbound traffic for the UAV relays should be zero. Moreover this constraint ensures that for each OD flow $i \rightarrow j$, the inbound (outbound) traffic to j (from i) is $TD_{ij} - y_{ij}$ (the amount of supported traffic). Constraints (5.11) ensure that the total traffic assigned to a link does not exceed its capacity. Finally, (5.12), (5.13) and (5.14) express the domain constraints.

5.4.1 Problem Formulation with UAV Mobility Constraints

In the case of mobile UEs or dynamic traffic patterns, **UPR_MILP** can be reapplied periodically in order to update the UAV positions. To consider the effect of UAV's maximum speed in a dynamic environment, we add mobility constraints to the optimization problem discussed in the previous section. The maximum speed of UAVs is represented by v_{max} and the time duration of a snapshot is denoted by ΔT. For each $u_i \in \mathcal{U}$, let $\mathcal{B}_i \subset \mathcal{U}$ denote the set of potential locations that the UAV deployed in u_i can reach in one snapshot, i.e. $u_j \in \mathcal{B}_i$ if $d_{i,j} \leq v_{max}\Delta T$.

Given $\mathcal{B}_i$ and the UAV placement decision variables at snapshot $t-1$ ($\boldsymbol{x}^{t-1}$), the UAV mobility constraints at snapshot t can be expressed as:

$$\sum_{u_j \in \mathcal{B}_i} x_j^t \geq \mathcal{I}_{\{x_i^{t-1}=1\}} \tag{5.15}$$

Moreover, in order to reduce the propulsion energy consumption of UAVs by avoiding unnecessary and less-effective UAV relocations in consecutive snapshots, we add another term to the objective function (5.6). The new objective function is:

$$(\phi \frac{\sum_{u \in \mathcal{U}} x_u}{N_{max}} + (1-\phi)\frac{\sum_{(i,j)\in\mathcal{E}_D} y_{ij}}{\sum_{(i,j)\in\mathcal{E}_D} TD_{ij}}) + \alpha(max_u |x_u^t - x_u^{t-1}|) \tag{5.16}$$

where α is a constant factor determining the balance between the two terms of the objective function. Instead of the maximum function in the new objective and in order to get rid of the

absolute value, we define a scalar variable z and add it to the objective function as follows:

$$(\phi\frac{\sum_{u\in\mathcal{U}} x_u}{N_{max}} + (1-\phi)\frac{\sum_{(i,j)\in\mathcal{E}_D} y_{ij}}{\sum_{(i,j)\in\mathcal{E}_D} TD_{ij}}) + \alpha z \tag{5.17}$$

and we add the following set of constraints to the optimization problem:

$$x_u^t - x_u^{t-1} \leq z \quad \forall u \in \mathcal{U} \tag{5.18}$$

$$x_u^{t-1} - x_u^t \leq z \quad \forall u \in \mathcal{U} \tag{5.19}$$

The resulting MILP is referred to as **MUPR_MILP** and is an NP-hard problem. However, the decision variables have to be determined in real-time, in response to the network changes. In the subsequent section, we employ an LP-relaxation to deal with the time-complexity of the **MUPR_MILP**. A greedy rounding approach is used to obtain the binary solution of the original problem.

5.4.2 Heuristic Algorithm

In this section, we propose a heuristic algorithm in order to handle the time complexity of **MUPR_MILP**. We derive the LP model of **MUPR_MILP** by relaxing the binary variables x_u^t or replacing the constraint sets (5.14) by:

$$x_u^t \in [0,1], \forall u \in \mathcal{U} \tag{5.20}$$

The resulting LP is represented by **UPR_LP**. We also define the set $\mathcal{X} \subseteq \mathcal{U}$ based on which the following LP denoted by **UPR_LP_reduced**($\mathcal{X}$) is defined:

$$\textbf{minimize} \quad (\phi \frac{\sum_{u \in \mathcal{U}} x_u^t}{N_{max}} + (1-\phi) \frac{\sum_{(i,j) \in \mathcal{E}_D} y_{ij}}{\sum_{(i,j) \in \mathcal{E}_D} TD_{ij}}) + \alpha z \tag{5.21}$$

$$\text{s.t} \quad (5.7) - (5.12), (5.18), (5.19) \tag{5.22}$$

$$x_u^t = 1, \quad \forall u \in \mathcal{X} \tag{5.23}$$

$$x_u^t \in [0,1] \quad \forall u \notin \mathcal{X} \tag{5.24}$$

We introduce a rounding-based decision-making process (DM-LP) to retrieve the binary decision variables of **MUPR_MILP** at each snapshot by solving a sequence of **UPR_LP_reduced**($\mathcal{X}$) problems iteratively. The proposed solution is shown in **Algorithm3**. The set $\mathcal{X}$ represents the locations chosen for UAV placement and is updated within each iteration (line (4)). The final $\mathcal{X}$ reflects the UAV placement decision. As explained in lines (7)-(9), UAVs are placed deterministically with the priority given to the neighboring locations of the deployed UAVs at the previous snapshot (reflected in the definition of the set S which is constructed by the union of the sets $\mathcal{B}_v$ for $v \in \mathcal{U} : x_v^{t-1} = 1$) in order to not violate the mobility constraints. For example, if two UAVs are placed at locations u_1, u_2 at snapshot $t-1$, the set $\mathcal{B}_1 \cup \mathcal{B}_2$ is first considered for UAV placement at snapshot t so that at least one UAV is placed at one of the locations of $\mathcal{B}_1$ (similarly for $\mathcal{B}_2$). Once all mobility constraints are satisfied, all the remaining potential UAV locations are considered for the placement of new UAVs. In both cases, a UAV is deployed at the position with maximum x value within each iteration (line (9) and (11)). The algorithm terminates when the addition of a new UAV does not reduce the objective function or makes the problem infeasible,

i.e. $x_u^t = 0, \forall u \in \mathcal{X}$ or equivalently, $x_{u^*}^t = 0$. Finally, the routing decisions are automatically obtained from the $\boldsymbol{f}$ solution of the last iteration. Moreover, the input $\boldsymbol{x}$ is $\mathbf{0}$ in the first snapshot meaning that any location in $\mathcal{U}$ can initially be considered for UAV placement. It is important

Algorithm 3 DM-LP for Joint UAV Placement and Routing

Input: $\mathcal{G}_p^t, \mathcal{G}_D^t, TD^t, \boldsymbol{x}^{t-1}$
Output: $\boldsymbol{x}^t, \boldsymbol{f}, \boldsymbol{y}$
Initialize $\mathcal{X} \leftarrow \emptyset, Terminate \leftarrow False$
repeat
  if not first iteration **then**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{u^*\}$
  end if
  $\{x_u^t, f_{uv}^{ij}\} \leftarrow$Solve **UPR_LP_reduced**$(\mathcal{X})$
  $S \leftarrow \cup_{\{v: x_v^{t-1}=1, v \notin \mathcal{X}\}} \{argmax_{k \in \mathcal{B}_v}\ x_k^t\}$
  if $S \neq \emptyset$ **then** ▷ Mobility constraints are not satisfied
    $u^* = argmax_{u \in S}\ x_u^t$
  else ▷ Mobility constraints are satisfied
    $u^* = argmax_{u \notin \mathcal{X}}\ x_u^t$
  end if
  if $x_{u^*}^t > 0$ **then**
    $x_{u^*}^t = 1$
  else
    $Terminate \leftarrow True$
  end if
until $Terminate == False$
return $\boldsymbol{x}^t, \boldsymbol{f}, \boldsymbol{y}$

to note that compared to **MUPR_MILP** which is intractable for large networks, the proposed approximation algorithm calls the LP solver at most $|N_{max}| + 1$ times. In the next section, we provide numerical results to evaluate the performance of the proposed approach.

5.5 Numerical Results

In this section, we benchmark out proposed decision-making process, DM-LP, against the exact solution, denoted as DM-MILP, and the connectivity-based approach proposed in [1],

namely DM-Conn. We also compare the performance of the mobile and static UAV deployment approaches. We use the CPLEX commercial solver for solving our MILP model using the branch-and-bound method, while the method used to solve the LP is primal-dual SIMPLEX. All experiments are carried out on an Intel Xeon processor at 2.3 GHz with 8GB memory. We consider a 10km x 10km square region and CHs are distributed according to a Matern cluster process [34] with the number of clusters changing between $2-11$. The cluster density mean and cluster radius are 10 and 1000m. We use the Reference Point Group Mobility (RPGM) model introduced in [92]. In this model, GUs in a cluster tend to coordinate their movement and the movement of each CH determines the behavior of the entire group. One example of such mobility is the movement of rescue teams during disaster relief. In our experiments, CHs move according to RWPM, and their speed is distributed uniformly according to $U(5,40)m/s$. We consider a grid with a total number of 100 points at height h as the potential UAV positions. The ground network flows are generated according to a Bernoulli distribution with the parameter 0.04 while the traffic demand for each pair is chosen with equal probability among the values 200, 400, and $600Kbps$. Moreover, c_{uv} is set to 5 and 10 Mbps, for A2G and A2A links respectively. Unless stated otherwise, simulation parameters are provided in Table 5.2.

Table 5.2: Simulation Parameters

Parameters	Description
$(a, b, \eta_{LoS}, \eta_{NLoS})$	$(9.61, 0.16, 1.0, 20.0)$ for urban environment
UAVs altitude h	2000m
Carrier frequency f_c	2GHz
Thermal noise power σ^2	-90dBm
SNR threshold γ_{min}	-4 dBm
E_{GU}, E_{UAV}	20dBm, 110dBm
v_{max}	55m/s
Snapshot Duration ΔT	25s

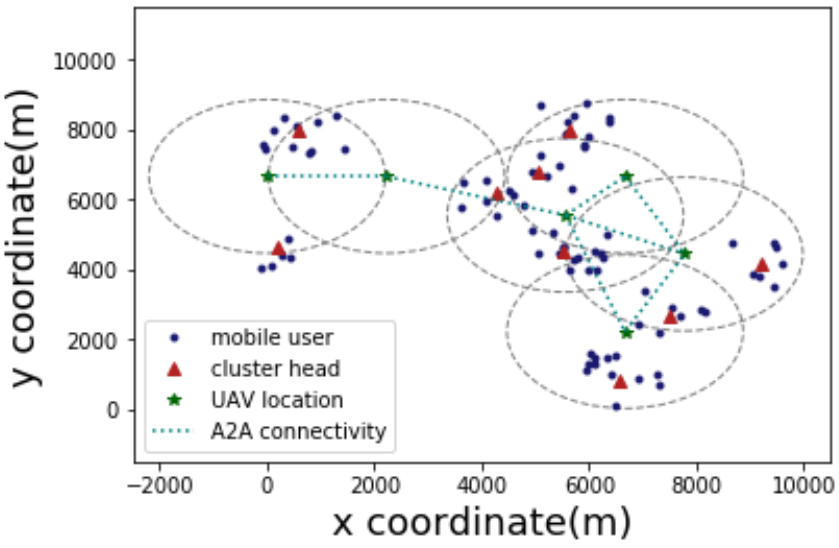


Figure 5.2: DM-MILP UAV Placement

5.5.1 Static Ground Network

In this experiment, we consider a fixed network with 9 clusters and a 10×10 UAV location grid.

Fig. 5.2-5.4 illustrates the UAV placement solution of DM-MILP, DM-LP, and DM-Conn strategies. It can be observed that all the clusters are covered by UAVs in all three cases. The number of deployed UAVs in DM-Conn is less than the other two approaches, since D-Conn only ensures the A2G, A2A connectivity, and A2G link capacity constraints, not the end-to-end traffic delivery. As a result, the supported traffic of DM-Conn is 67%, while both DM-MILP and DM-LP fully support the traffic demands in this example. This experiment highlights the need for joint UAV placement and traffic routing in multi-hop wireless networks.

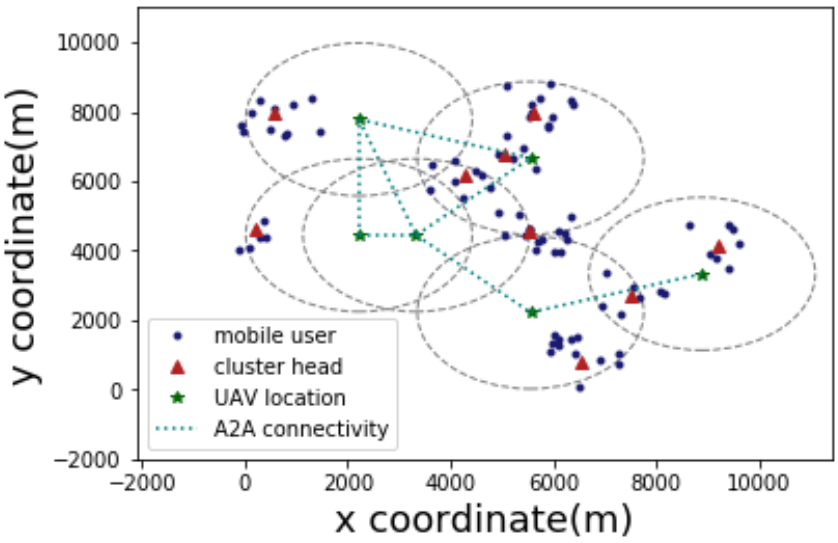


Figure 5.3: DM-LP UAV Placement

5.5.2 Mobile Ground Network

In this experiment, the proposed DM-LP is benchmarked against DM-MILP and a static UAV deployment in a dynamic ground netowrk. Fig. 5.5 depicts the number of relays deployed based on DM-MILP and DM-LP, as an indicator of the deployment cost. The results are averaged over 20 snapshots. There are no hard limits imposed on the maximum number of UAVs, i.e. their number is only constrained by the number of possible UAV positions on the grid. As a result, traffic demands are fully supported, while the difference in the average number of deployed UAVs is at most 2 more for the approximation algorithm. With regards to time complexity, as depicted in Fig. 5.6, DM-MILP follows an exponential growth, whereas the DM-LP method has an approximately linear time growth with respect to the number of clusters. Under the current evaluation environment, the real time operation of a network comprised of up to 6 clusters can be supported. However, DM-LP's linear time-complexity would guarantee real time support for

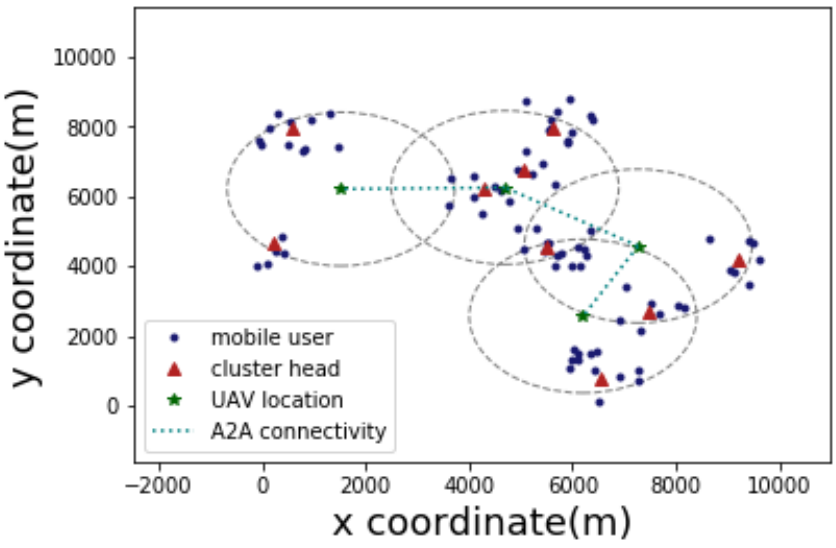


Figure 5.4: DM-Conn UAV Placement [1]

larger network instances with a more powerful system.

Fig. 5.7 shows the average total supported traffic per snapshot for a ground network of 10 clusters, following DM-LP, DM-MILP and a static UAV deployment where the UAVs locations are obtained from the solution of DM-MILP for the first snapshot. Fig. 5.8 depicts the profile of the average supported traffic for the same scenarios over snapshots. The results are averaged over 5 random networks. Overall, the deviation of the DM-LP from the optimal solution is on average 7% and 5% when $N_{max} = 5$ and $N_{max} = 6$ respectively. Note that 6 UAVs are enough to fully support the traffic demand of the generated network instances as DM-MILP achieves 100% traffic support in this experiment. This demonstrates the ability of the proposed LP-based scheme to generate good solutions, for a limited number of available UAVs. Moreover, a static UAV deployment with even an optimal initial deployment resulted in a maximum of 40% unsupported traffic, highlighting the need for a dynamic UAV deployment solution. The four figures together demonstrate that the DM-LP approach trades only a small degree of optimality for fast retrievable

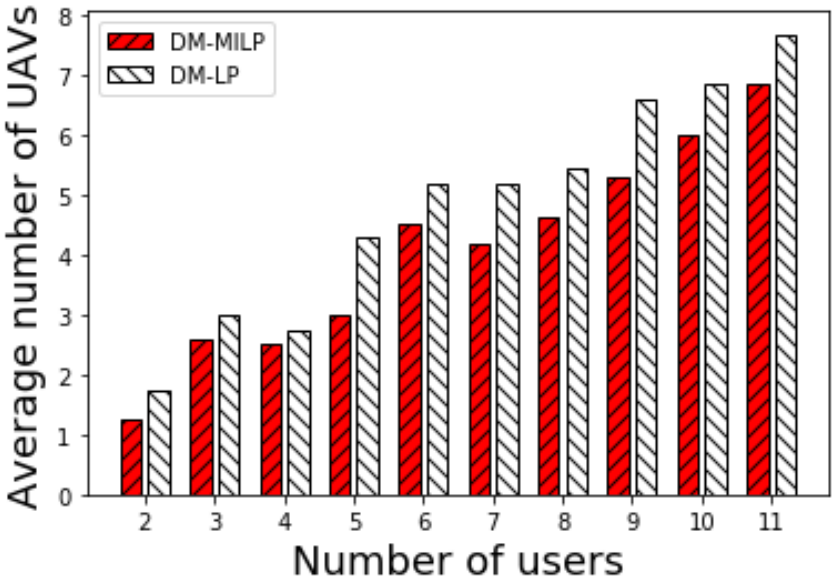


Figure 5.5: Number of UAVs: DM-LP vs. DM-MILP

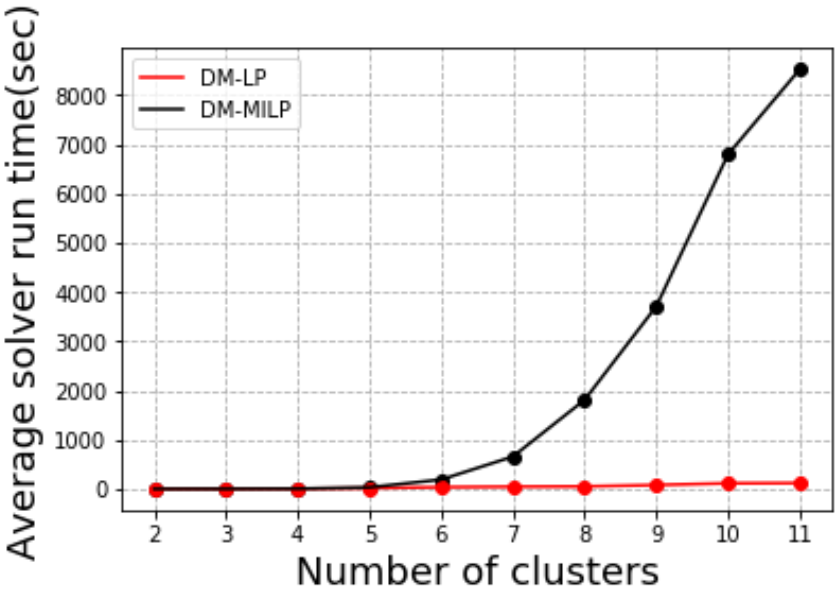


Figure 5.6: Solver Runtime: DM-LP vs DM- MILP

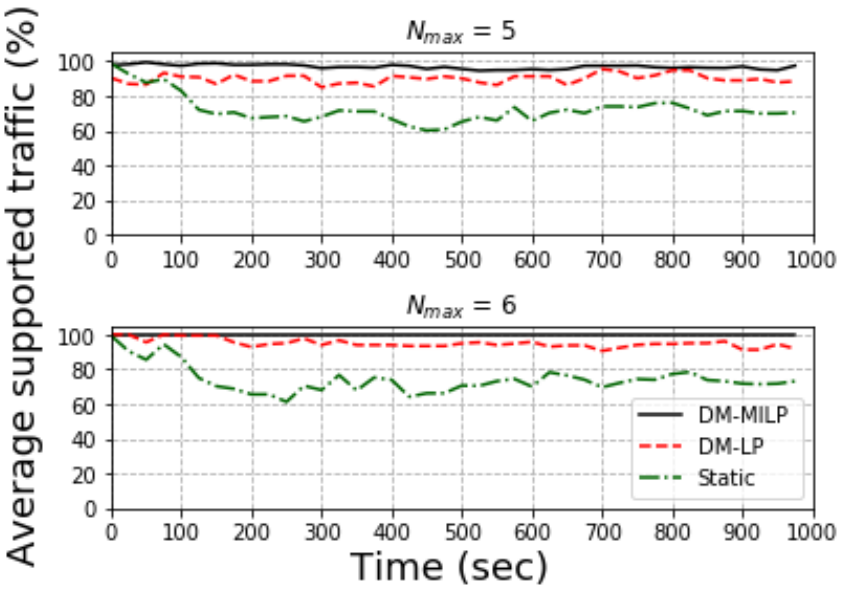


Figure 5.7: Average Percentage of Supported Traffic

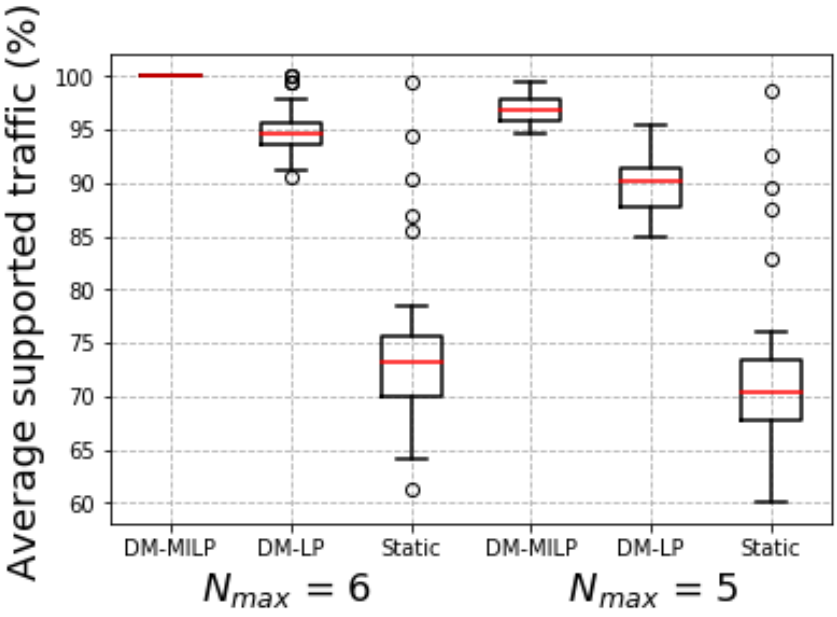


Figure 5.8: Average Percentage of Supported Traffic Profile

solutions.

5.6 Conclusion

In this chapter, we propose a framework for joint UAV placement and route optimization in a multi-hop UAV relaying communications system, taking into account the mobility of the ground nodes, the capacity of A2A and A2G links, UAVs mobility constraints and UAVs propulsion energy consumption. The proposed method is in contrast to many existing frameworks for aerial platforms which consider only a single-hop or a two-hop aerial communication, we allow multi-hop aerial paths to carry the ground traffic. Moreover, we consider the end-to-end traffic guarantee by imposing capacity and routing constraints in our formulation while most of the existing solutions study the UAV placement problem with connectivity constraints/objective and ignore the end-to-end traffic delivery. We model the problem as a MILP, and then propose an efficient LP-based approximation algorithm to effectively reduce the time complexity of our model, achieving a near-optimal solution. The numerical simulations provide insights on the effect the users' mobility and the dynamic relocation of UAVs on the decision making process and the service degradation. Among our future directions are to consider the control/management layer resource allocation problem and investigate the computation offloading and service placement problem together with the resource allocation in a mobile edge computing setup.

Chapter 6: Reliable Edge Learning

6.1 Overview

Traditional cloud-based machine learning wherein a remote data center performs training on a centralized dataset leads to insufficient performance quality for many emerging intelligent devices and applications. For instance, a controlled UAV in a smart factory or an autonomous vehicle should always be operational by sensing and reacting rapidly to hazardous situations in the environment. Moreover, it is very impractical for devices in NGMN to transmit huge volume of raw data to a remote cloud for central ML, due to communication inefficiency and privacy concerns. As a result, a novel paradigm change from centralized and cloud-based ML to a distributed, low-latency, and reliable ML at the network edge has been developed in recent years. Federated learning (FL) [93] is a new learning framework that enables multiple agents such as smartphones, sensors, robots, or drones to learn a model (e.g. a neural network) collaboratively, without exchanging their local data, thereby preserving data privacy to a great extent. To this end, two FL setups have been proposed in the literature, namely *centralized FL* and *decentralized F*. In the *centralized FL*, a central server (e.g. a server located at the network edge) collects the local model updates from all or a subset of agents, aggregates them and generates a global model which is then fed back to the agents. The process is repeated until a desired accuracy level or an upper bound on the number of iteration (training time) is achieved. As a leading algorithm in this

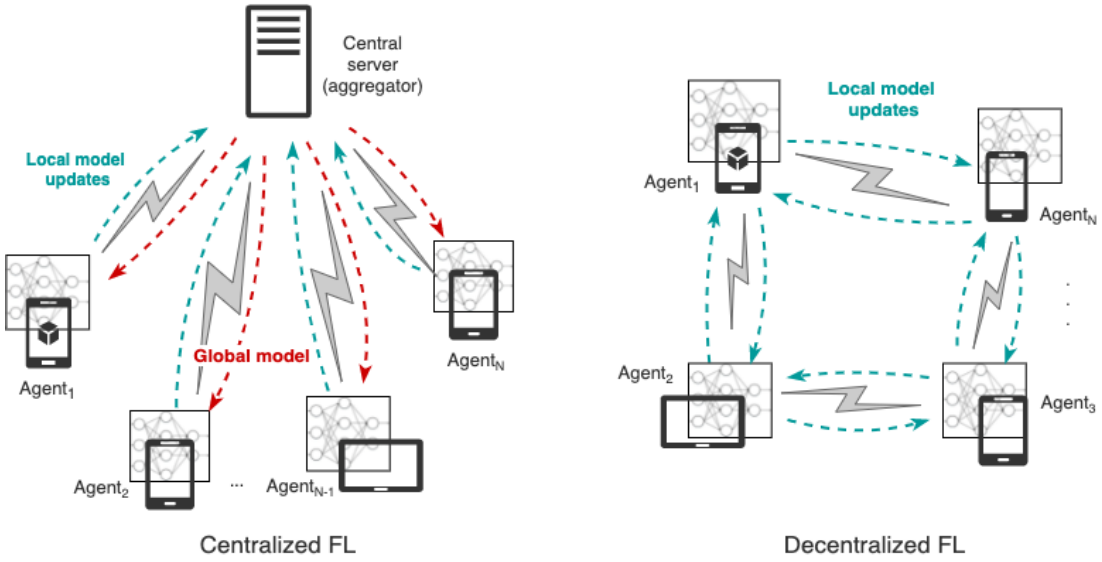


Figure 6.1: Decentralized vs. Centralized FL Model

setting, *Federated Averaging (FedAvg)* [94] runs stochastic gradient descent (SGD) in parallel on a small subset of the total agents and averages the local updates only once in a while. In particular, at each iteration of the algorithm, *FedAvg* locally performs E epochs of SGD on M devices, where E is a small constant and M is a small fraction of the total devices in the network. The devices then communicate their updated model to a central edge-cloud usually co-located with the BS or wireless access point. The edge-cloud aggregates the received model updates using a weighted average. The resulting model, namely global model is then shared with the participating devices.

Despite enhancing the agent's privacy and network communications efficiency through exchanging the model updates only instead of the agents' local data, the main drawbacks of centralized FL are the lack of scalability, connectivity, and the existence of single-point-of-failure. Moreover, next-generation networks are expected to be enhanced by new forms of decentralized and infrastructure-less communication schemes and device-to-device (D2D) multi-

hop connections such as in UAV-aided networks [72]. Within this scope, novel approaches are required to address decentralized (server-less) FL. While a number of research activities have focused on distributed learning algorithms [95], due to the special features of an FL setup in which the data is generated locally and remains decentralized and because of the communication efficiency considerations, many existing approaches developed for distributed learning are not applicable to decentralized FL. Authors in [96] extend the centralized FL approach for massively dense IoT networks that do not rely on a central server, where the agents perform training steps on their local data using SGD and consensus-based methods. At each consensus step, agents transmit their local model update to their one-hop neighbors. Each agent fuses the received messages from its neighbors and then feeds the result to SGD. Fig. 6.1 shows the architecture of a decentralized FL setting in contrast to a centralized FL setting. The fact that the information is crowd-sourced by the FL agents in decentralized FL, highlights the need to establish trust relationships between the FL agents. More specifically, apart from ensuring the security of communications between the FL agents, answering the following questions is of paramount importance: (i) whether an agent refuses to share its information with the FL process due to privacy concerns or conflict of interest? (ii) whether an agent manipulates the received data before processing? (iii) whether an agent intentionally or unintentionally, shares incorrect information with the rest of the network? etc. [97] [98]. In other words, it is essential to establish to what extent each agent of the network and its model updates can be trusted.

This chapter addresses the problem of securing the *decentralized FL* mechanism for the first time. In this context, we propose a *trusted decentralized FL* algorithm in which trust is interpreted as a relation between different network entities that may interact or collaborate in groups towards achieving various specific goals [99]. These relations are set up and updated

based on the evidence generated when the agents collaborate within a previous protocol. If the collaboration has been contributive towards achievement of the specific goal (positive evidence), the parties accumulate their trust perspective towards one another, otherwise (negative evidence), trust will decrease between them. Trust estimates have input to decisions such as access control, resource allocation, agent participation, and so on. The method by which trust is computed and aggregated within the network may depend on the specific application. In [98], the authors enumerate the central differences in the terminology of trust computation and aggregation. In this section, we propose an attack-tolerant consensus-based FL algorithm by incorporating the trust concept into the consensus step.

The rest of this chapter is organized as follows. We provide an overview of the related works in section 6.2. The system model is discussed in section 6.3, including the trust aggregation framework and the trust evaluation method. Section 6.4 describes the proposed trusted decentralized FL. Performance evaluation is presented in section 6.5 and we highlight our conclusions in section 6.6.

6.2 Related Work

In the context of centralized FL, several attempts have been made in the literature to address the security issues. The vulnerability of FL framework to data poisoning and model poisoning attacks has been recently studied in the literature. Sun et al. [100] study data poisoning attack strategies on federated multi-task learning framework and propose an optimal attack strategy based on the communication protocol and bi-level optimization which is shown to effectively damage the performances of real-world applications. Compared to data poisoning attacks, model

poisoning attacks have been shown to be more effective especially in large scale scenarios [101]. Authors in [102] demonstrated that FL is vulnerable to backdoor attack through model replacement by a malicious participant. In [103], a protocol is proposed to guarantee the confidentiality of users' local gradients and to verify the integrity of the aggregated results generated by the server. Authors in [104] proposed a reliable user selection scheme for centralized federated learning tasks based on the concept of reputation and consortium blockchain in order to defend against unreliable model updates. In [105], a Byzantine-resilient secure aggregation framework is proposed for secure federated learning in a centralized FL scenario. Fang et al. [106] showed that the existing aggregation rules which are claimed to be robust against Byzantine failures of some FL agents are vulnerable to local model poisoning attacks and proposed two defense mechanisms for data poisoning attacks. However, their proposed solutions are not effective enough in all tested cases. As far as our knowledge goes, no efforts have been made towards studying effective defense mechanisms against model poisoning attacks in decentralized FL framework.

6.3 System Model

We consider a network consisting of a set of N agents (benign and malicious) denoted by $\mathcal{N} = \{1, ..., N\}$. Each agent $i \in \mathcal{N}$ has D_i data samples and the total number of samples is D. The dth sample is represented by $(\mathbf{x}_d, \mathbf{y}_d)$ where $\mathbf{x}_d \in \mathbb{R}^{D_{in} \times 1}, \mathbf{y}_d \in \mathbb{R}^{D_{out} \times 1}$, $d = 1, ..., D$. We also assume that $\mathcal{P}_i$ denotes the set of indices of data points collected by agent i. Moreover, the datasets collected by different agents have the same distribution (i.i.d assumption). The local dataset $\mathcal{D}_i = \{(\mathbf{x}_d, \mathbf{y}_d), d \in \mathcal{P}_i\}$ is used to train a local model $\mathcal{M}_i$ parameterized by $\mathbf{w}_i$.

Following the decentralized FL model proposed in [96], we consider agents performing

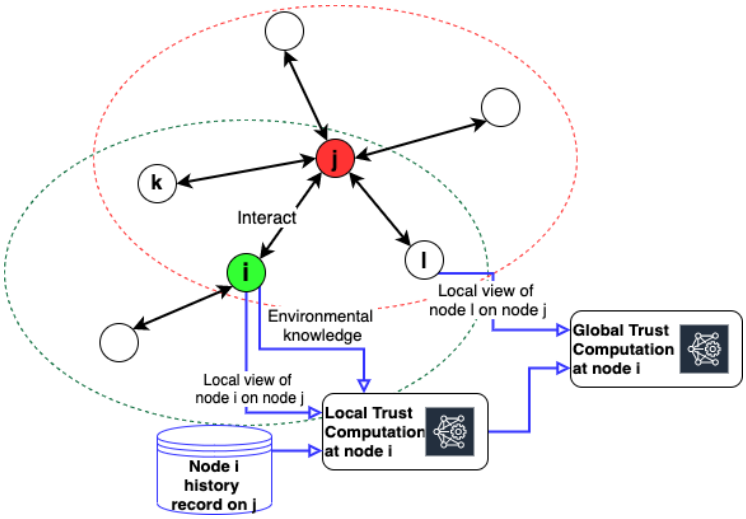


Figure 6.2: Decentralized Trust Aggregation Framework

training steps on their local dataset using SGD and consensus-based methods. At each consensus step (time instance) k, agents transmit their local model update to their one-hop neighbors. Let $\mathbf{m}_i^{(k)}$ denote the message that agent i sends to its neighbors at step k. For a benign agent, $\mathbf{m}_i^{(k)} = \mathbf{w}_i^{(k)}$, while an adversary sends a message different from the update computed by running SGD on its local data in order to poison the global model. Each agent fuses the received messages from its neighbors and then feeds the result to SGD. In the next subsections, we first elaborate on the trust evaluation and aggregation methods based on which the proposed trusted decentralized FL algorithm is presented.

6.3.1 Trust Aggregation Framework

We model the network of agents at time instance k, as an undirected graph $\mathcal{G}^{(k)} = (\mathcal{N}^{(k)}, \mathcal{L}^{(k)})$ where $\mathcal{N}^{(k)}$ is the set of agents and for $n, m \in \mathcal{N}^{(k)}$, $\mathcal{L}^{(k)}$ contains all links $(m, n)^{(k)}$ where agents m, and n can communicate with one another at time instance k. We denote this graph as the *communication graph* at time instance k. Let $\mathcal{N}_i^{(k)}$ be the set of neighbors of node i at time step

k. Apart from the communication relationship, we also define *local trust* relationships between nodes $i, j \in \mathcal{N}^{(k)}$ that is obtained at each node by processing the first-hand evidence extracted from previous interactions between its immediate neighbors. The agents also hold *global trust* estimates of one another that is formed by the local exchange of the local observations (local trust estimates) and is usually more accurate. Let $\tau_{ij}^{(k)}$, and $t_{ij}^{(k)}$ be the local and global view of node i on trustworthiness of the node j at time instance k in respective order. We may ignore the index k whenever doing so does not lead to confusion. In the next two subsections, we will formally state the definition and the mathematical models capturing local and global trusts. Fig. 6.2 depicts the schematic of the trust aggregation framework.

6.3.2 Local Trust Model

To formalize the definition of local trust, let us define $X_{ij}^{(k)}$ to be a random variable denoting the reputation that node j has in the perspective of node i in time instance k. $X_{ij}^{(k)}$ follows a Beta distribution [107] with parameters $\alpha_{ij}^{(k)}$, and $\beta_{ij}^{(k)}$. Moreover, define $r_{ij}^{(k)} = \alpha_{ij}^{(k)} - 1$, and $s_{ij}^{(k)} = \beta_{ij}^{(k)} - 1$ that determine the number of times up to round k, that node j's behavior is benign and malicious in perspective of node i, in respective order. The details of how $r_{ij}^{(k)}$, and $s_{ij}^{(k)}$ are obtained depends on the specific scenario and is to be explicitly mentioned in the next sections. We let $\tau_{ij}^{(k)}$ to be the expected value of the reputation random variable in the Beta system $X_{ij}^{(k)}$. Formally, we have:

$$f_{X_{ij}^{(k)}}(x;\alpha_{ij}^{(k)},\beta_{ij}^{(k)}) = \left(\frac{\Gamma\left(\alpha_{ij}^{(k)}+\beta_{ij}^{(k)}\right)}{\Gamma\left(\alpha_{ij}^{(k)}\right)\Gamma\left(\beta_{ij}^{(k)}\right)}\right)\cdot\left(x^{\alpha_{ij}^{(k)}-1}\left(1-x\right)^{\beta_{ij}^{(k)}-1}\right) \tag{6.1}$$

$$\tau_{ij}^{(k)} = \mathbb{E}\left[X_{ij}^{(k)}\right] = \frac{r_{ij}^{(k)}+1}{r_{ij}^{(k)}+s_{ij}^{(k)}+2} \tag{6.2}$$

Considering the time-varying behavior of the agents, the evolution of r and s parameters needs to be in a way that the more recent information receive more relative importance comparing to the older ones. Therefore, we have:

$$r_{ij}^{(k+1)} = \rho_1 r_{ij}^{(k)} + I_{ij}^{(k+1)} \tag{6.3}$$

$$s_{ij}^{(k+1)} = \rho_2 s_{ij}^{(k)} + 1 - I_{ij}^{(k+1)}, \tag{6.4}$$

where ρ_1 and ρ_2 are positive forgetting factors. In order to remember malicious behaviors longer than benign behaviors, we may choose $0 < \rho_1 < \rho_2$. The binary function $I_{ij}^{(k+1)} \in \{0,1\}$ models the instantaneous perspective of node i on the behavior of node j in $(k+1)^{th}$ round. In the next section, we define this function based on the messages communicated between FL agents.

6.3.3 Global Trust Model

At each instance k, within the local trust model, each node i computes its local trust for all nodes $j \in \mathcal{N}_i$ in the communication graph. In order to make more accurate estimates, node i will need to take into account the opinions of other network nodes who have first-hand evidence on node j's behavior. Following the approach in [108], node i computes in an iterative fashion its global trust estimate for node j, i.e. $t_{ij}^{(k)}$ using the opinions of its neighbors as:

$$t_{ij}^{m} = \begin{cases} 1 & \text{if } i = j \\ \sum_{l \in \mathcal{N}_i, l \neq j} w_{il} t_{lj}^{m-1} & \text{if } i \neq j \end{cases} \tag{6.5}$$

where $w_{il} = \frac{\tau_{il}}{\sum_{l \in \mathcal{N}_i, l \neq j} \tau_{il}}$. In other words, node i pays more attention to the opinions of those of its neighbors who it trusts more. We note again that the global trust computation is an iterative process that is going to be embedded in each iteration of the trust-aware protocol. Therefore, to avoid any confusion we have used the iteration counter m for this process. Here, we have dropped the superscript k as we assume the value of local trust remains constant within the loop of computing the global trust. In the next section, we will show how trust can be used to improve the performance of decentralized FL.

6.3.4 Trust Evaluation Method

In this section we elaborate on the methods that are used in evaluating the trustworthiness of the network agents. These methods are embedded into the trust evaluation scheme used in the trust model in section 6.3.1. Throughout the FL protocols, the updates corresponding to the local models of the agents are communicated within the network as messages play an important role

in determining the trust level of the agents. We enumerate two methods that assign trust values to the agents based on the communicated updates:

- *Clustering-based Method*: In this method, the trustor entity i compares the messages it has received from trustee j, to all the other messages it has received from other parties. Formally, for each neighbor j, party i computes:

$$\mathrm{dev}_{ij}^{(k)} = \sum_{l \in \mathcal{N}_i^+} \frac{\left|\left| w_l^{(k)} - w_j^{(k)} \right|\right|_2^2}{\left|\mathcal{N}_i^+\right|} \tag{6.6}$$

  where $\mathcal{N}_i^+$ is the set of the neighbors of agent i including agent i itself. Then, for each trustee j, it will benchmark the value of $dev_{ij}^{(k)}$ against a multiple of the median of all the deviations:

$$I_{ij}^{C(k)} = \begin{cases} 1 & dev_{ij}^{(k)} \leq th_i \times \text{ median } \left(\left\{ \mathrm{dev}_{ij}^{(k)} \right\} \right) \\ 0 & o.w. \end{cases} \tag{6.7}$$

  This way, by adjusting the value of th_i at iteration k, the trustor party can decide not to trust those parties from which it has received too-far-away messages.

- *Distance-based Method*: In this method, the trustor party i computes the distance between its local model and the model at trustee j, and uses this distance as a measure to assign trust values to its neighbors. Formally, party i computes the distance between the message it has received from party j in the previous and the current iterations; i.e.

$$d_{ij}\left(w_i^{(k-1)}, m_j^{(k-1)}\right) = \left|\left| w_i^{(k-1)} - m_j^{(k-1)} \right|\right|_2 \tag{6.8}$$

and,

$$d_{ij}\left(w_i^{(k-1)}, m_j^{(k)}\right) = \left\| w_i^{(k-1)} - m_j^{(k)} \right\|_2 \tag{6.9}$$

Then, party i computes the difference between the two computed distances and decides on the value of $I_{ij}^{D(k)}$ as follows:

$$I_{ij}^{D(k)} = \mathbb{I}_{\left\{d_{ij}\left(w_i^{(k-1)}, m_j^{(k-1)}\right) - d_{ij}\left(w_i^{(k-1)}, m_j^{(k)}\right) \geq 0\right\}} \tag{6.10}$$

In other words, if node j has a benign behavior, then after one iteration of the protocol, its local model must have shifted towards the local model of party i. If this is not the case, then party j has to be malicious or must be communicating incorrect message to i. We have illustrated further on our trust aggregation framework together with the trust inference terminology in [109] in great details. Such trust evaluation and aggregation method is already used successfully in [110] and [111] for trustworthy network service embedding on trusted infrastructure where trustworthiness of each network component for hosting a specific network function is computed and and taken into account at the trusted decision making process. In the next section, we investigate the trusted decentralized federated learning problem using the methods mentioned earlier.

6.4 Trusted Decentralized FL

In order to address the security issues of the FL training, we propose an attack-tolerant consensus-based FL algorithm by incorporating the trust concept into the consensus step. The

proposed approach is given in Algorithm 4. In particular, let $t_{ij}^{(k)}$ denote the trustworthiness of agent $j \in \mathcal{N}_i$ evaluated at agent i, that is obtained in step 4 based on the model described in section 6.3. It is important to note that we combine the cluster-based and distance-based trust values and build the corresponding trust evaluation model in step 12. In step 7, agent i aggregates the received model updates from its neighbors with the weights of $\frac{D_j t_{ij}^{(k)}}{\sum_{j \in \mathcal{N}_i} D_j t_{ij}^{(k)}}$, i.e. the neighbors of i that have higher trust values contribute more to the aggregated model at i. Then, each agent updates its model parameter vector independently using SGD on its local data and transmits the updated model to its neighbors.

6.5 Numerical Results

In this section we evaluate the performance of our trusted decentralized FL algorithm by means of simulation. For the simulation setup, we adopt the settings of [96]. Similar to [96], in this section we assume that the packet loss is negligible (a realistic assumption for a short-range communication scheme) and leave the study of the problem in the presence of the communication loss to future. Our implementation of the FL process and the validation dataset are both based on [112]. In this experiment, a real-world industrial IoT (IIoT) use case is considered in which a network of $N = 80$ IIoT devices sense their surroundings using frequency-modulated continuous-wave (FMCW) radars. The goal of the IIoT devices is to detect and track the position of human operators sharing a workspace with a robotic manipulator for safety policy purposes. To address the latency constraints of safety policies as well as scalability, bandwidth-efficiency and reliability challenges, a decentralized FL algorithm is deployed. The connectivity topology of the IIoT devices is assumed to be κ-regular (all agents have the same number of

Algorithm 4 *Trusted Decentralized FL*

Initialize $\mathbf{w}_i^{(0)}$
for each round $k = 1, 2, ...$ **do**
    Agent i receives the messages $\{\mathbf{m}_j^{(k)}\}_{j \in \mathcal{N}_i}$
    Agent i updates the trust values for all its neighbors:
$t_{ij}^{(k)} \leftarrow ComputeTrust(i, j, \{\mathbf{m}_j^{(k)}, j \in \mathcal{N}_i\})$
    $\mathbf{\Psi}_{i,k} \leftarrow \mathbf{m}_i^{(k)}$
    for each agents $j \in \mathcal{N}_i$ **do**
        $\mathbf{\Psi}_{i,k} \leftarrow \mathbf{\Psi}_{i,k} + \epsilon_k \frac{D_j t_{ij}^{(k)}}{\sum_{j \in \mathcal{N}_i} D_j t_{ij}^{(k)}} (\mathbf{m}_j^{(k)} - \mathbf{w}_i^{(k)})$
    end for
    Each agent i computes:
$\mathbf{w}_i^{(k+1)} \leftarrow ModelUpdate(\mathbf{\Psi}_{i,k}, \mu_k)$ and sends $\mathbf{m}_i^{(k+1)}$ to all its neighbors
end for
$ComputeTrust(i, j, \{\mathbf{m}_i^{(k)}, i \in \mathcal{N}\})$:

    Agent i computes $I_{ij}^{C(k)}$ and $I_{ij}^{D(k)}$ from (6.7) and (6.10)

    Agent i computes $I_{ij}^{(k)} = I_{ij}^{C(k)} \vee I_{ij}^{D(k)}$

    Agent i computes its local trust for j ($\tau_{ij}^{(k)}$) from (6.2)

    Agent i computes its global trust for j ($t_{ij}^{(k)}$) from (6.5)

    Return $t_{ij}^{(k)}$
$ModelUpdate(\mathbf{w}_k, \mu_k)$:
    Initialize $\mathbf{\Psi}_{i,k} \leftarrow \mathbf{w}^{(k)}$
    $\mathcal{B}$ = mini-batch of size B
    for $b \in \mathcal{B}$ **do**
        $\mathbf{\Psi}_{i,k} \leftarrow \mathbf{\Psi}_{i,k} - \mu_k \nabla f(\mathbf{\Psi}_{i,k})$
    end for
    Return $\mathbf{w}_i^{(k)} \leftarrow \mathbf{\Psi}_{i,k}$

neighbors), with the number of neighbors $|\mathcal{N}_i| = 2$.

6.5.1 Experimental Setup

Dataset: The data sample d characterized by $(\mathbf{x}_d, \mathbf{y}_d)$ has the input of $\mathbf{x}_d$, a 512-point fast Fourier transform (FFT) of the beat signals obtained from the radar echoes and averaged over ten consecutive frames. Each input FFT measurement is labeled by one of the $C = 8$ classes representing the distance between the cooperating robot and human worker. Data distribution is non-IID as the local data samples collected independently by each device may contain only a subset of labels. Each device obtains $D_i = 25$ samples. The size of the validation dataset is $D = 16000$.

ML model: The considered learning model is a 2NN with a first fully connected (FC) layer of 32 hidden nodes of dimension 512×32 followed by a ReLu layer and a second FC layer of dimension $32 \times C$ followed by a softmax layer. We note that although the considered model is simple, it is a realistic use case for IoT devices with limited computation and power capabilities.

Attack model: We implement a model poisoning attack at the presence of 80 devices participating in the FL task where $p = 5\%, 10\%$ and 20% of the nodes are corrupt. The attacker parties will generate the poisoned model by choosing and transmitting the weights randomly in the range $(q * w_{min}, q * w_{max})$, where w_{min} and w_{max} are the minimum and maximum of the weights they receive from their neighbors. We assume that $q = 2$.

In the following, we present the numerical results validating the effectiveness of our proposed solution.

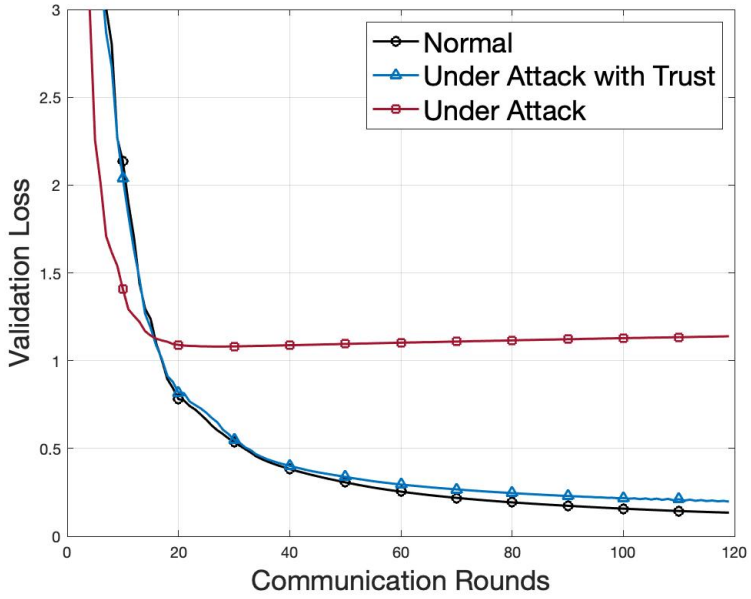


Figure 6.3: Effect of Trust on Resilience Against Attacks

6.5.2 Resilience Against Attacks

In Fig. 6.3, we compare the performance of the proposed trusted FL algorithm in the presence of the attacked agents, with both the normal (no attack) system and under attack system without trust incorporation. The validation loss is shown for 120 communication rounds and we assume that 10% of the agents are under attack. Fig. 6.3 illustrates how incorporating trust into the decentralized federated learning framework can protect the protocol from being invaded by malicious parties. In the absence of the trust mechanism and under corrupt network agents, the validation loss will not converge to the correct value corresponding to when all the agents are operating normally. Therefore, the performance of the trained model on test data degrades significantly, even with a moderate model poisoning attack. However, when the trust mechanism is in place, the trained model will converge to that of the normal implementation. We note that the

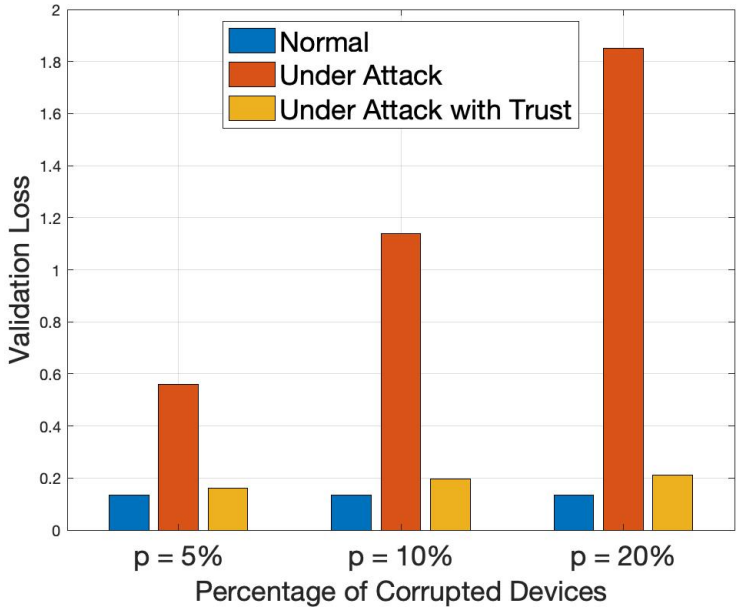


Figure 6.4: Impact of the Percentage of Compromised Agents

validation loss after 120 epoch converges to 0.14, and 0.18 for the normal (without any attacks) and the attacked trust-aware processes, and diverges from 1.89 for the attacked unprotected process. We note that the value of 0.14 validation loss corresponds to a 90% of accuracy on the test data.

6.5.3 Impact of the Percentage of the Compromised Agents

In Fig. 6.4, the validation loss of the normal system, the system under attack with trust, and the system under attack without trust consideration is depicted for three different values of p, corresponding to mild ($p = 5\%$), moderate ($p = 10\%$) and severe ($p = 20\%$) model poisoning attacks. It is observed that while the validation loss of the attacked system is significantly higher than the normal scenario, the proposed trusted decentralized FL algorithm is able to improve the performance of the trained model noticeably for all three values of p. In particular, we note

that while the validation loss increases from 0.56 for a mild attack scenario to 1.89 and 1.85 for the moderate and severe attack cases respectively, incorporating trust in the decentralized FL algorithm results in a maximum loss value of 0.21 for the severe attack scenario.

6.6 Conclusion

Motivated by MEC and the large-scale deployment of edge devices generating unprecedented huge data, it is expected that a large number of intelligent applications will be deployed at the edge of the network in NGMN. While traditional cloud-based learning can support only a limited number of such applications due to its cost, reliability, latency, and privacy issues, FL has been proposed as an effective learning paradigm to realize wide edge learning in NGMN. In this chapter, we proposed trust as a metric to enable reliable FL. We presented a mathematical framework for trust evaluation and propagation within a networked system. We argued that trust inference can be modeled as a consensus problem, while emphasising on the concepts of local and global trust estimates, and argued that trust evaluation can be embedded as a block into the framework of decentralized FL. We showed by means of simulation that our trusted decentralized FL algorithm can tolerate model poisoning attacks in the training of a neural network. Among our future directions is to account for the communications packet losses and also study the communication overhead of the proposed scheme.

Chapter 7: Conclusion and Future Works

In this dissertation, we proposed novel approaches to address four interrelated and challenging resource allocation problems in NGMN. In the first problem, we solved the resource provisioning of mobile network slicing under demand uncertainties. We used stochastic programming to model the E2E resource provisioning problem and proposed a two-timescale approach that solves the resource allocation problem in two stages. In the first one, the proposed stochastic programming model is solved to provision resources for different requested slices across the core network and RAN. In the second stage, the RAN slice provisioning is adjusted as the observed spatial demand varies. Through extensive simulations, we showed that the proposed methodology is able to address the demand uncertainties in a robust manner. In the second part of this dissertation, we studied the problem of application deployment in NGMN. Taking into account the MEC and MCC paradigms, we proposed novel resource allocation frameworks for the deployment of multi-component applications in multi-tiered compute and networking systems. We studied this problem from the two perspectives of (i) InP, (ii) application. From the perspective of the InP, we proposed a collaborative cloud-edge-local computation offloading scheme that is shown to outperform the conventional edge-only and cloud-only computation offloading schemes. From the higher level of application viewpoint, we proposed a joint network and compute reservation solution for microservices-based applications that results in significant resource savings while

maintaining application performance. In a further step, we developed an RL-based resource orchestration solution for dynamic environments. Next, we addressed the resource allocation problem of aerial platforms in NGMN. We proposed a framework for joint UAV placement and route optimization in a multi-hop UAV relaying communications system, taking into account the mobility of the ground nodes, the capacity of A2A and A2G links, UAVs mobility constraints, and UAVs propulsion energy consumption. The numerical simulations provide insights into the effect of the users' mobility and the dynamic relocation of UAVs on the decision-making process and service degradation. Among our future directions are to investigate the computation offloading and service delivery problem together with the resource allocation in UAV-aided systems towards an extension of network slicing to aerial networks. Motivated by MEC and the large-scale deployment of edge devices generating unprecedented huge data, it is expected that a large number of intelligent applications will be deployed at the edge of the network in NGMN. While traditional cloud-based learning can support only a limited number of such applications due to its cost, reliability, and issues, FL has been proposed as an effective learning paradigm to realize wide edge learning in NGMN. In the final chapter of this dissertation, we addressed the problem of reliable edge learning. We proposed a trust-aware FL framework that utilizes trust as a metric to find out reliable agents followed by a revised aggregation model. In the future, we aim to extend the trust-aware method to address the low-quality updates caused by high-speed mobility, low-computation or low-energy power, and communication bottleneck issues.

Bibliography

[1] Senni Perumal, John S Baras, Charles J Graff, and David G Yee. Aerial platform placement algorithms to satisfy connectivity, capacity and survivability constraints in wireless ad-hoc networks. In *MILCOM 2008-2008 IEEE Military Communications Conference*, pages 1–7. IEEE, 2008.

[2] Alcardo Alex Barakabitze, Arslan Ahmad, Rashid Mijumbi, and Andrew Hines. 5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges. *Computer Networks*, 167:106984, 2020.

[3] Chengchao Liang and F. Richard Yu. Wireless network virtualization: A survey, some research issues and challenges. *IEEE Communications Surveys Tutorials*, 17(1):358–380, 2015.

[4] Shunliang Zhang. An overview of network slicing for 5g. *IEEE Wireless Communications*, 26(3):111–117, 2019.

[5] 3GPP. System Architecture for the 5G System (5GS). *Technical Specification TS 23.501, Release 16*, Apr. 2021,.

[6] Yi Shi, Yalin E. Sagduyu, and Tugba Erpek. Reinforcement learning for dynamic resource optimization in 5g radio access network slicing. In *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6, 2020.

[7] Hatim Chergui, Luis Blanco, Luis A. Garrido, Kostas Ramantas, Sławomir Kukliński, Adlen Ksentini, and Christos Verikoukis. Zero-touch ai-driven distributed management for energy-efficient 6g massive network slicing. *IEEE Network*, 35(6):43–49, 2021.

[8] Peter Rost, Christian Mannweiler, Diomidis S Michalopoulos, Cinzia Sartori, Vincenzo Sciancalepore, Nishanth Sastry, Oliver Holland, Shreya Tayade, Bin Han, Dario Bega, et al. Network slicing to enable scalability and flexibility in 5g mobile networks. *IEEE Communications magazine*, 55(5):72–79, 2017.

[9] Mugen Peng, Yaohua Sun, Xuelong Li, Zhendong Mao, and Chonggang Wang. Recent advances in cloud radio access networks: System architectures, key techniques, and open issues. *IEEE Communications Surveys & Tutorials*, 18(3):2282–2308, 2016.

[10] Ruoyu Su, Dengyin Zhang, Ramachandran Venkatesan, Zijun Gong, Cheng Li, Fei Ding, Fan Jiang, and Ziyang Zhu. Resource allocation for network slicing in 5g telecommunication networks: A survey of principles and models. *IEEE Network*, 33(6):172–179, 2019.

[11] Latif U Khan, Ibrar Yaqoob, Nguyen H Tran, Zhu Han, and Choong Seon Hong. Network slicing: Recent advances, taxonomy, requirements, and open research challenges. *IEEE Access*, 8:36009–36028, 2020.

[12] Quang-Trung Luu, Sylvaine Kerboeuf, and Michel Kieffer. Uncertainty-aware resource provisioning for network slicing. *IEEE Transactions on Network and Service Management*, 18(1):79–93, 2021.

[13] Vincenzo Sciancalepore, Konstantinos Samdanis, Xavier Costa-Perez, Dario Bega, Marco Gramaglia, and Albert Banchs. Mobile traffic forecasting for maximizing 5g network slicing resource utilization. In *IEEE INFOCOM 2017-IEEE conference on computer communications*, pages 1–9. IEEE, 2017.

[14] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[15] Johanna Andrea Hurtado Sánchez, Katherine Casilimas, and Oscar Mauricio Caicedo Rendon. Deep reinforcement learning for resource management on network slicing: A survey. *Sensors*, 22(8):3031, 2022.

[16] Adedotun T. Ajibare and Olabisi E. Falowo. Resource allocation and admission control strategy for 5g networks using slices and users priorities. In *2019 IEEE AFRICON*, pages 1–6, 2019.

[17] Davit Harutyunyan, Riccardo Fedrizzi, Nashid Shahriar, Raouf Boutaba, and Roberto Riggio. Orchestrating end-to-end slices in 5g networks. In *2019 15th International Conference on Network and Service Management (CNSM)*, pages 1–9, 2019.

[18] Bin Xiang, Jocelyne Elias, Fabio Martignon, and Elisabetta Di Nitto. Joint network slicing and mobile edge computing in 5g networks. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[19] NGMN Alliance. 5g white paper. *Next generation mobile networks, white paper*, 1(2015), 2015.

[20] Sajjad Gholamipour, Behzad Akbari, Nader Mokari, Mohammad Mahdi Tajiki, and Eduard Axel Jorswieck. Online admission control and resource allocation in network slicing under demand uncertainties. *arXiv preprint arXiv:2108.03710*, 2021.

[21] Andreas Baumgartner, Thomas Bauschert, Fabio D'Andreagiovanni, and Varun S Reddy. Towards robust network slice design under correlated demand uncertainties. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2018.

[22] Arled Papa, Alba Jano, Serkut Ayvaşık, Onur Ayan, H Murat Gürsu, and Wolfgang Kellerer. User-based quality of service aware multi-cell radio access network slicing. *IEEE Transactions on Network and Service Management*, 2021.

[23] Tao Guo and Alberto Suárez. Enabling 5g ran slicing with edf slice scheduling. *IEEE Transactions on Vehicular Technology*, 68(3):2865–2877, 2019.

[24] Weisen Shi, Junling Li, Peng Yang, Qiang Ye, Weihua Zhuang, Xuemin Shen, and Xu Li. Two-level soft ran slicing for customized services in 5g-and-beyond wireless communications. *IEEE Transactions on Industrial Informatics*, 18(6):4169–4179, 2021.

[25] Afonso Oliveira and Teresa Vazao. Mapping network performance to radio resources. In *2022 International Conference on Information Networking (ICOIN)*, pages 298–303. IEEE, 2022.

[26] 3GPP. Study on management and orchestration of network slicing for next generation network. *Technical Report TR 28.801, V15.0.1, Release 15*, Jan. 2018.

[27] Andreas Baumgartner, Thomas Bauschert, Fabio D'Andreagiovanni, and Varun S. Reddy. Towards robust network slice design under correlated demand uncertainties. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7, 2018.

[28] Rongpeng Li, Zhifeng Zhao, Jianchao Zheng, Chengli Mei, Yueming Cai, and Honggang Zhang. The learning and prediction of application-level traffic data in cellular networks. *IEEE Transactions on Wireless Communications*, 16(6):3899–3912, 2017.

[29] Qianqian Zhang, Walid Saad, Mehdi Bennis, Xing Lu, Mérouane Debbah, and Wangda Zuo. Predictive deployment of uav base stations in wireless networks: Machine learning meets contract theory. *IEEE Transactions on Wireless Communications*, 20(1):637–652, 2021.

[30] Chen Yu, Yang Liu, Dezhong Yao, Laurence T. Yang, Hai Jin, Hanhua Chen, and Qiang Ding. Modeling user activity patterns for next-place prediction. *IEEE Systems Journal*, 11(2):1060–1071, 2017.

[31] Vincenzo Sciancalepore, Konstantinos Samdanis, Xavier Costa-Perez, Dario Bega, Marco Gramaglia, and Albert Banchs. Mobile traffic forecasting for maximizing 5g network slicing resource utilization. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, 2017.

[32] Salvatore D'Oro, Francesco Restuccia, Alessandro Talamonti, and Tommaso Melodia. The slice is served: Enforcing radio access network slicing in virtualized 5g systems. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 442–450. IEEE, 2019.

[33] Alexander Shapiro. Simulation-based optimization—convergence analysis and statistical inference. *Stochastic Models*, 12(3):425–454, 1996.

[34] François Baccelli and Bartłomiej Błaszczyszyn. *Stochastic geometry and wireless networks*, volume 1. Now Publishers Inc, 2009.

[35] Pavel Mach and Zdenek Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, 19(3):1628–1656, 2017.

[36] M. D. Hossain, L. N. T. Huynh, T. Sultana, T. D. T. Nguyen, J. Ho Park, C. S. Hong, and E. Huh. Collaborative task offloading for overloaded mobile edge computing in small-cell networks. In *2020 International Conference on Information Networking (ICOIN)*, pages 717–722, 2020.

[37] H. Guo, J. Liu, H. Qin, and H. Zhang. Collaborative computation offloading for mobile-edge computing over fiber-wireless networks. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, 2017.

[38] Kaustabha Ray, Ansuman Banerjee, and Nanjangud C Narendra. Proactive microservice placement and migration for mobile edge computing. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 28–41. IEEE, 2020.

[39] https://nextgalliance.org/white_papers/6g-technologies/.

[40] Anousheh Gholami and John S. Baras. Collaborative cloud-edge-local computation offloading for multi-component applications. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 361–365, 2021.

[41] Sujit Das and Elizabeth Mao. The global energy footprint of information and communication technology electronics in connected internet-of-things devices. *Sustainable Energy, Grids and Networks*, page 100408, 2020.

[42] Congfeng Jiang, Tiantian Fan, Honghao Gao, Weisong Shi, Liangkai Liu, Christophe Cérin, and Jian Wan. Energy aware edge computing: A survey. *Computer Communications*, 151:556–580, 2020.

[43] Yanting Wang, Min Sheng, Xijun Wang, Liang Wang, and Jiandong Li. Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. *IEEE Transactions on Communications*, 64(10):4268–4282, 2016.

[44] Feng Wang, Jie Xu, Xin Wang, and Shuguang Cui. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. *IEEE Transactions on Wireless Communications*, 17(3):1784–1797, 2017.

[45] Xinchen Lyu, Hui Tian, Wei Ni, Yan Zhang, Ping Zhang, and Ren Ping Liu. Energy-efficient admission of delay-sensitive tasks for mobile edge computing. *IEEE Transactions on Communications*, 66:2603–2616, 2018.

[46] Min Sheng, Yanting Wang, Xijun Wang, and Jiandong Li. Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server. *IEEE Transactions on Communications*, 68(3):1524–1537, 2020.

[47] Mingjie Feng, Marwan Krunz, and Wenhan Zhang. Joint task partitioning and user association for latency minimization in mobile edge computing networks. *IEEE Transactions on Vehicular Technology*, 70(8):8108–8121, 2021.

[48] J. Liu and Q. Zhang. Code-partitioning offloading schemes in mobile edge computing for augmented reality. *IEEE Access*, 7:11222–11236, 2019.

[49] Tayebeh Bahreini and Daniel Grosu. Efficient algorithms for multi-component application placement in mobile edge computing. *IEEE Transactions on Cloud Computing*, pages 1–1, 2020.

[50] Mithun Mukherjee, Suman Kumar, Constandinos X. Mavromoustakis, George Mastorakis, Rakesh Matam, Vikas Kumar, and Qi Zhang. Latency-driven parallel task data offloading in fog computing networks for industrial applications. *IEEE Transactions on Industrial Informatics*, 16:6050–6058, 2020.

[51] Yuyi Mao, Jun Zhang, and Khaled B Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590–3605, 2016.

[52] Norman D Curet. A primal-dual simplex method for linear programs. *Operations Research Letters*, 13(4):233–237, 1993.

[53] Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[54] Ernesto QV Martins and Marta MB Pascoal. A new implementation of yen's ranking loopless paths algorithm. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(2):121–133, 2003.

[55] S. Eman Mahmoodi, R. N. Uma, and K. P. Subbalakshmi. Optimal joint scheduling and cloud offloading for mobile applications. *IEEE Transactions on Cloud Computing*, 7(2):301–313, 2019.

[56] Anousheh Gholami, Kunal Rao, Wang-Pin Hsiung, Oliver Po, Murugan Sankaradas, and Srimat Chakradhar. Roma: Resource orchestration for microservices-based 5g applications. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9, 2022.

[57] Qiang Liu and Tao Han. DIRECT: Distributed cross-domain resource orchestration in cellular edge computing. In *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 181–190, 2019.

[58] Salvatore D'Oro, Leonardo Bonati, Francesco Restuccia, Michele Polese, Michele Zorzi, and Tommaso Melodia. Sl-EDGE: Network slicing at the edge. In *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, pages 1–10, 2020.

[59] Cagatay Sonmez, Can Tunca, Atay Ozgovde, and Cem Ersoy. Machine learning-based workload orchestrator for vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, 22(4):2239–2251, 2020.

[60] Mutaz Al-Tarawneh and Saif Alnawayseh. Performance Assessment of Context-aware Online Learning for Task Offloading in Vehicular Edge Computing Systems. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 12:304–320, 05 2021.

[61] João Paulo Esper, Abdallah S. Abdallah, Stuart Clayman, Waldir Moreira, Antonio Oliveira, Sand Luz Correa, and Kleber Vieira Cardoso. eXP-RAN—An emulator for gaining experience with radio access networks, edge computing, and slicing. *IEEE Access*, 8:152975–152989, 2020.

[62] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. EdgeCloudSim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11):e3493, 2018.

[63] Kunal Rao, Giuseppe Coviello, Wang-Pin Hsiung, and Srimat T. Chakradhar. ECO: Edge-Cloud Optimization of 5G applications. In *The 21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid 2021), Melbourne, Victoria, Australia*, pages 649–659, 2021.

[64] Multitech. https://www.multitech.com/brands/multiconnect-ecell.

[65] Celona. https://celona.io/.

[66] Kubernetes. https://kubernetes.io/.

[67] Ffmpeg. http://ffmpeg.org/.

[68] Glpk (gnu linear programming kit). https://www.gnu.org/software/glpk/.

[69] Marco Wiering and Jürgen Schmidhuber. Fast online q (λ). *Machine Learning*, 33(1):105–115, 1998.

[70] Yong Zeng, Rui Zhang, and Teng Joon Lim. Wireless communications with unmanned aerial vehicles: Opportunities and challenges. *IEEE Communications Magazine*, 54(5):36–42, 2016.

[71] Anousheh Gholami, Usman A Fiaz, and John S Baras. Drone-assisted communications for remote areas and disaster relief. *arXiv preprint arXiv:1909.02150*, 2019.

[72] Anousheh Gholami, Nariman Torkzaban, John S. Baras, and Chrysa Papagianni. Joint Mobility-Aware UAV Placement and Routing in Multi-Hop UAV Relaying Systems. In *Ad Hoc Networks*, pages 55–69. Springer International Publishing, 2021.

[73] Shuhang Zhang, Hongliang Zhang, Qichen He, Kaigui Bian, and Lingyang Song. Joint trajectory and power optimization for uav relay networks. *IEEE Communications Letters*, 22(1):161–164, 2017.

[74] Yong Zeng, Rui Zhang, and Teng Joon Lim. Throughput maximization for uav-enabled mobile relaying systems. *IEEE Transactions on Communications*, 64(12):4983–4996, 2016.

[75] Jiangbin Lyu, Yong Zeng, and Rui Zhang. Uav-aided offloading for cellular hotspot. *IEEE Transactions on Wireless Communications*, 17(6):3988–4001, 2018.

[76] Nariman Torkzaban, Anousheh Gholami, John S Baras, and Chrysa Papagianni. Joint satellite gateway placement and routing for integrated satellite-terrestrial networks. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020.

[77] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, Young-Han Nam, and Mérouane Debbah. A tutorial on uavs for wireless networks: Applications, challenges, and open problems. *IEEE communications surveys & tutorials*, 21(3):2334–2360, 2019.

[78] Bin Li, Zesong Fei, and Yan Zhang. Uav communications for 5g and beyond: Recent advances and future trends. *IEEE Internet of Things Journal*, 6(2):2241–2263, 2018.

[79] Konstantinos Poularakis, Qiaofeng Qin, Kelvin M Marcus, Kevin S Chan, Kin K Leung, and Leandros Tassiulas. Hybrid sdn control in mobile ad hoc networks. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 110–114. IEEE, 2019.

[80] Qingqing Wu, Yong Zeng, and Rui Zhang. Joint trajectory and communication design for multi-uav enabled wireless networks. *IEEE Transactions on Wireless Communications*, 17(3):2109–2121, 2018.

[81] Arvind Merwaday and Ismail Guvenc. Uav assisted heterogeneous networks for public safety communications. In *2015 IEEE wireless communications and networking conference workshops (WCNCW)*, pages 329–334. IEEE, 2015.

[82] Boris Galkin, Jacek Kibilda, and Luiz A DaSilva. Deployment of uav-mounted access points according to spatial user locations in two-tier cellular networks. In *2016 Wireless Days (WD)*, pages 1–6. IEEE, 2016.

[83] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, and Mérouane Debbah. Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage. *IEEE Communications Letters*, 20(8):1647–1650, 2016.

[84] Mohamed Alzenad, Amr El-Keyi, and Halim Yanikomeroglu. 3-d placement of an unmanned aerial vehicle base station for maximum coverage of users with different qos requirements. *IEEE Wireless Communications Letters*, 7(1):38–41, 2017.

[85] Pankaj K Sharma and Dong In Kim. Random 3d mobile uav networks: Mobility modeling and coverage probability. *IEEE Transactions on Wireless Communications*, 18(5):2527–2538, 2019.

[86] R Irem Bor-Yaliniz, Amr El-Keyi, and Halim Yanikomeroglu. Efficient 3-d placement of an aerial base station in next generation cellular networks. In *2016 IEEE international conference on communications (ICC)*, pages 1–5. IEEE, 2016.

[87] Elham Kalantari, Halim Yanikomeroglu, and Abbas Yongacoglu. On the number and 3d placement of drone base stations in wireless cellular networks. In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, pages 1–6. IEEE, 2016.

[88] Ursula Challita and Walid Saad. Network formation in the sky: Unmanned aerial vehicles for multi-hop wireless backhauling. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[89] Guangchi Zhang, Haiqiang Yan, Yong Zeng, Miao Cui, and Yijun Liu. Trajectory optimization and power allocation for multi-hop uav relaying communications. *IEEE Access*, 6:48566–48576, 2018.

[90] Zhenyu Kang, Changsheng You, and Rui Zhang. Placement learning for multi-uav relaying: A gibbs sampling approach. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020.

[91] Akram Al-Hourani, Sithamparanathan Kandeepan, and Simon Lardner. Optimal lap altitude for maximum coverage. *IEEE Wireless Communications Letters*, 3(6):569–572, 2014.

[92] Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching-Chuan Chiang. A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 53–60, 1999.

[93] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[94] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[95] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008. PMLR, 2014.

[96] S. Savazzi, M. Nicoli, and V. Rampa. Federated learning with cooperating devices: A consensus approach for massive IoT networks. *IEEE Internet of Things Journal*, 7(5):4641–4654, 2020.

[97] Y. Wang. Trust quantification for networked cyber-physical systems. *IEEE Internet of Things Journal*, 5(3):2055–2070, 2018.

[98] G. Theodorakopoulos and J. S. Baras. On trust models and trust evaluation metrics for ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24(2):318–328, 2006.

[99] Anousheh Gholami, Nariman Torkzaban, and John S Baras. Trusted decentralized federated learning. In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2022.

[100] Gan Sun, Yang Cong, Jiahua Dong, Qiang Wang, Lingjuan Lyu, and Ji Liu. Data poisoning attacks on federated machine learning. *IEEE Internet of Things Journal*, 2021.

[101] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[102] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[103] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, 15:911–926, 2020.

[104] Jiawen Kang, Zehui Xiong, Dusit Niyato, Yuze Zou, Yang Zhang, and Mohsen Guizani. Reliable federated learning for mobile networks. *IEEE Wireless Communications*, 27(2):72–80, 2020.

[105] Jinhyun So, Başak Güler, and A. Salman Avestimehr. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications*, 39(7):2168–2181, 2021.

[106] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1605–1622, 2020.

[107] Audun Jøsang and Roslan Ismail. The beta reputation system. In *15th bled electronic commerce conference*, pages 41–55, 2002.

[108] X. Liu and J. S. Baras. Using trust in distributed consensus with adversaries in sensor and other networks. In *17th International Conference on Information Fusion (FUSION)*, pages 1–7, 2014.

[109] Anousheh Gholami, Nariman Torkzaban, and John S Baras. On the importance of trust in next-generation networked cps systems: An ai perspective. *arXiv preprint arXiv:2104.07853*, 2021.

[110] Nariman Torkzaban, Chrysa Papagianni, and John S. Baras. Trust-aware service chain embedding. In *2019 Sixth International Conference on Software Defined Systems (SDS)*, pages 242–247, 2019.

[111] Nariman Torkzaban and John S. Baras. Trust-aware service function chain embedding: A path-based approach. In *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 31–36, 2020.

[112] Data repository. (2019). federated learning: Example dataset (fmcw 122ghz radars). 2021.