EFFICIENT 5G MOBILE NETWORK MANAGEMENT: NETWORK SLICING AND GLOBAL ROAMING OPTIMIZATION

by

SERGI ALCALÁ MARÍN

in partial fulfillment of the requirements for the degree of Doctor in

Telematic Engineering

Universidad Carlos III de Madrid

Advisor: Marco Fiore & Albert Banchs
Tutor: Albert Banchs

October 2024

Efficient 5G Mobile Network Management: Network Slicing and Global Roaming Optimization

Prepared by:
Sergi Alcalá Marín, IMDEA Networks Institute, Universidad Carlos III de Madrid
contact: sergi.alcala@imdea.org

Under the advice of:
Marco Fiore, IMDEA Networks Institute
Albert Banchs, Universidad Carlos III de Madrid
Telematic Engineering Department, Universidad Carlos III de Madrid

This work has been supported by:

Thank you!

Acknowledgements

I consider myself fortunate. All the people around me helped me become the person I am today.

First, I really want to thank Albert Banchs and Marco Fiore for the outstanding advisors they were (and are) for me. You were so helpful in my life, not only to be a good researcher but also to grow and be a better human. Your advice on all the faces of life was extremely helpful for me. Albert, thank you for allowing me to be part of IMDEA Networks Institute; I will always be so grateful. Marco, you are the reason I want to keep on research. Your commitment to all our Network Data Science group and showing us that the most important is to be a complete researcher. I really appreciate it and I do not have enough words to express my gratitude to both of you.

Second, I have to thank Javier Tejada, my first advisor as a researcher at Univesitat de Barcelona. Javier, you were the motivation I needed to keep doing research. Thanks to you, I realized that I love to see things that I would be the first one to see them in the world.

I want to continue thanking all the first generation of researchers of Network Data Science group. We have seen that this is not a race or a competition. The PhD carreer is something that you have to enjoy (and suffer) in a collaborative way. And we did, we were so helpful between us, and we will achieve our goals, together. I could extend ten pages expressing my gratitude, but I will only mention you, and you know what you are for me.
Thank you: Dr. Michele Gucciardo, Dr. Antonio Bazco-Nogueras, Leonardo Lo Schiavo, Sachit Mishra, Alan Collet, André Zanella, Orlando Martínez-Durive, Aristide Akem.

I also want to thank all IMDEA Networks staff. It was a pleasure to feel at home at work. I want to mention Elvira Conti, who suffered me with all the travels and IDINET issues that we had. You were helpful in my PhD journey and became a fantastic friend.

I want to make a special mention to all the friends I made while doing my research at TU Delft. You made me feel like one more of your friend's group.

I will continue extending my gratitude to my family. Even though they did not know what I was doing, they always supported my decisions and motivated me to do what I thought was best for me. I want to thank Christina for her support through these three years. You were the one celebrating my successes even more than myself. I will never forget all the travels, especially the one we made to IEM Katowice.

Lastly, all this work is dedicated to you, Francisca. You always believed in me, and I know you would be so proud of the man I am now.

Published and Submitted Content

This thesis is based on the following published papers:

[1] Sergi Alcalá-Marín, Aravindh Raman, Weili Wu, Andra Lutu, Marcelo Bagnulo, Ozgu Alay, Fabián Bustamante. Global Mobile Network Aggregators: Taxonomy, Roaming Performance and Optimization. Published in *The 20th ACM International Conference on Mobile Systems, Applications, and Services. (ACM MobiSys 2022)*, 25 June - July 1, 2022, Portland, OR, USA. https://doi.org/10.1145/3498361.3538942

- This work is fully included and its content is reported in Chapter 5.
- The author's role in this work is focused on the design, implementation and experimentation with regarding of the concepts proposed in the paper.
- The material from this source included in this thesis is not singled out with typographic means and references.

[2] Sergi Alcalá-Marín, Antonio Bazco-Nogueras, Albert Banchs, Marco Fiore . kaNSaaS: Combining Deep Learning and Optimizationfor Practical Overbooking of Network Slices. Published in *The 24th International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (ACM MobiHoc 2023)*, October 23-26, 2023, Washington DC, USA. https://doi.org/10.1145/3565287.3610265

- This work is fully included and its content is reported in Chapter 3.
- The author's role in this work is focused on the design, implementation and experimentation with regarding of the concepts proposed in the paper.
- The material from this source included in this thesis is not singled out with typographic means and references.

[3] Sergi Alcalá-Marín, Weili Wu, Aravindh Raman, Marcelo Bagnulo, Ozgu Alay, Fabián E. Bustamante, Marco Fiore, Andra Lutu. A comparative analysis of Global Mobile Network Aggregators. This work has been submitted for publication.

- This work is fully included and its content is reported in Chapter 5.

- The author's role in this work is focused on the design, implementation and experimentation with regarding of the concepts proposed in the paper.

- The material from this source included in this thesis is not singled out with typographic means and references.

[4] Sergi Alcalá-Marín, Dario Bega, Marco Gramaglia, Albert Banchs, Xavier Costa-Perez, Marco Fiore. AZTEC+: Long and Short Term Resource Provisioning for Zero-Touch Network Management. This work has been submitted for publication.

- This work is fully included and its content is reported in Chapter 4.

- The author's role in this work is focused on the design, implementation and experimentation with regarding of the concepts proposed in the paper.

- The material from this source included in this thesis is not singled out with typographic means and references.

Abstract

The rapid transformation of mobile network infrastructures from hardware-based systems to software-defined networks (SDNs) has introduced several novel paradigms that aim to enhance scalability, flexibility, and efficiency. One of the most significant advancements in this shift is the concept of network slicing. In essence, network slicing allows a single physical network to be divided into multiple virtual networks, each tailored to the specific requirements of different use cases. These slices can vary in terms of bandwidth, latency, security, and resource allocation, making them highly adaptable to different industries, such as autonomous vehicles, smart cities, and Internet of Things (IoT) applications. This capability offers unprecedented flexibility for service providers, as they can deploy and manage various services on the same physical infrastructure without the need for expensive and complex hardware upgrades.

However, realizing the full potential of network slicing presents several challenges. One of the key issues is the complex management of multiple network slices that may have conflicting requirements. For example, ensuring ultra-reliable low-latency communication for one slice while maximizing bandwidth for another slice can be difficult, especially when they share the same underlying physical resources. The dynamic nature of network slicing, where slices are created, modified, and terminated on demand, also requires highly efficient orchestration and automation systems. Furthermore, maintaining security and isolation between slices is essential to prevent interference or data breaches. As mobile networks continue to evolve, addressing these challenges will be crucial to fully unlocking the benefits of network slicing and realizing the vision of highly flexible, software-driven networks.

This thesis explores innovative approaches to improve the profitability of Mobile Network Operators (MNO) through advanced network slicing strategies in 5G networks. The research focuses on three key areas: overbooking network slices for maximize financial gains, cost-efficient network slice management, and operational performance of Mobile Network Aggregators (MNA) within Network Slicing as a Service (NSaaS) context under roaming scenarios. These innovations leverage state-of-the-art techniques, including deep learning, classical optimization, and data-driven algorithms, to tackle the complex challenges of resource provisioning, network function lifecycle management, and global service optimization.

This document delves into these advancements, structured across several chapters that explore these key areas in depth. In the first contribution, we explore the NSaaS concept and introduce

slice overbooking as a promising strategy to maximize resource utilization and boost net profit in cloud-native mobile networks. Network slicing enables the creation of virtualized, custom-tailored network slices, but managing these slices efficiently remains a challenge. Overbooking allows network operators to admit more slices than available physical resources by capitalizing on the fact that tenants seldom use their total reserved capacity simultaneously. The chapter presents a complete NSaaS management solution, overbooKing-Aware Network Slicing as a Service (kaNSaaS), which integrates deep learning with classical optimization to address the dual problems of admission control and resource allocation. Through extensive experimentation with large-scale real-world data on tenant demands, the results show that kaNSaaS boosts operator profits potentially multiplying it by four compared to non-overbooking strategies under real-world conditions.

In the second contribution, the focus shifts to zero-touch management systems, which promise autonomous network operation with minimal human intervention. As modern network infrastructures have evolved into systems with numerous virtualized network functions, traditional manual approaches to lifecycle management are increasingly inadequate. In response, this chapter introduces AZTEC+, a data-driven solution for anticipatory resource provisioning in network slicing environments. Using a hybrid and modular deep learning architecture, AZTEC+ forecasts future service demands and determines optimal trade-offs between resource provisioning, instantiation, and reconfiguration costs and performance requirements. Tested on a large-scale network, AZTEC+ outperformed existing state-of-the-art management strategies by up to 5.85 times, proving its effectiveness in reducing network costs and addressing the complexity of virtualized mobile networks. It effectively balances costs associated with resource instantiation and reconfiguration, making it a highly efficient solution for managing dynamic network slices autonomously. This chapter emphasizes how zero-touch management, paired with anticipatory resource provisioning, offers a scalable approach to future network management.

In the third contribution, we explore the added complexity introduced by MNAs, which represent a new frontier in global mobile telecommunications in NSaaS. MNAs, such as Google Fi, Twilio, and Truphone, operate by leveraging multiple MNOs to provide mobile communication services across different regions. Unlike traditional MNOs, which are limited by geographic boundaries, MNAs dynamically connect to the MNO that offers the best performance based on location and time, ensuring optimal service quality for users, especially those frequently crossing borders. However, the dynamic nature of MNAs introduces a new layer of complexity in meeting network slicing's Quality of Service (QoS) guarantees as the isolation and management of slices become more intricate with the involvement of multiple, regionally dispersed MNOs. To address this, we quantify and compare the performance of MNA-driven models against traditional MNOs, offering insights into the challenges and trade-offs in achieving reliable QoS in these advanced operator frameworks. This section presents a detailed performance analysis of the three aforementioned MNAs, comparing their performance for key applications like web browsing and video streaming within NSaaS across diverse geographical regions, namely the USA and Spain. While MNAs may introduce slight delays compared to local MNOs in certain regions, they

significantly outperform the traditional home-routed roaming model in terms of service quality. Moreover, emulation studies using open-source 5G implementations deployed across Amazon Web Services (AWS) locations illustrate the performance gains MNAs can achieve through advanced network function virtualization. This chapter highlights the potential of the MNA model to reshape global mobile services, offering more flexible, efficient, and seamless experiences for end-users.

Overall, this research provides valuable insights into NSaaS overbooking management, zero-touch resource provisioning, and the global reach of MNAs. Together, these innovations represent significant strides toward realizing the next generation of mobile networks, where resource efficiency, automation, and global service quality are paramount. The research highlights the economic benefits of slice overbooking, demonstrating how operators can significantly increase profitability by intelligently managing their resources. Furthermore, by integrating hybrid models of AI and optimization, it lays a strong foundation for future developments in network slicing and cost optimization, offering practical implications for both MNO and application developers. As 5G networks continue to evolve, this work sheds light on the shifting landscape of global network operators and provides a roadmap for addressing the growing complexities of resource allocation and service management in a cloud-native, AI-driven environment. Through a blend of deep learning, anticipatory algorithms, and network virtualization, the telecommunications industry is better positioned to meet the ever-evolving demands of users while optimizing network performance across a wide range of dynamic environments.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AC	Admission Control
AI	Artificial Intelligence
AMF	Access and Mobility Management Function
AR	Augmented Reality
BBU	Baseband Unit
CAPEX	Capital Expenditures
CDN	Content Delivery Network
CN	Core Network
CNN	Convolutional Neural Networks
C-RAN	Cloud Radio Access Network
CU	Centralized Unit
CUPS	Control and User Plane Separation
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
DU	Distributed Unit
E2E	End-to-end
eMBB	enhanced Mobile BroadBand
EPC	Evolved Packet Core

ETSI	European Telecommunications Standards Institute
GTP	GPRS Tunneling Protocol
HMNO	Home Mobile Network Operator
HR	Home-routed Roaming
IHBO	IPX Hub Breakout
IPX	IP Packet Exchange
IPX-P	IPX Hub Breakout Provider
IoT	Internet of Things
KP	Knapsack Problem
LBO	Local Breakout
LSTM	Long Short Term Memory Networks
LTP	Long-Term Predictor
MAE	Mean Absolute Error
MCCMNC	Mobile Country Code (MCC) / Mobile Network Code (MNC)
ML	Machine Learning
MNA	Mobile Network Aggregator
MNO	Mobile Network Operator
mMTC	massive Machine-Type Communication
MSE	Mean Squared Error
MTC	Machine-Type Communications
MVNO	Mobile Virtual Network Operator
NDA	Non-Disclosure Agreement
NSaaS	Network Slicing as a Service
NSSF	Network Slice Selection Function
OPEX	Operating Expenses

PLT	Page Load Time
PoP	Point of Presence
QoE	Quality of Experience
QoS	Quality of Service
RA	Resource Allocation
RBO	Regional Breakout
RL	Reinforcement Learning
RNN	Recurrent Neural Networks
RTT	Round-Trip Time
RU	Remote Unit
SBA	Service-Based Architecture
SDN	Software-Defined Networks
SEPP	Security Edge Protection Proxy Network Function
SLA	Service-Level Agreement
SMF	Session Management Function
SP	Service Provider
STP	Short-Term Predictor
TES	Thresholded Exponential Smoothing
TN	Transport Core Network
TSN	Time-Sensitive Networking
UPF	User Plane Function
uRLLC	ultra-Reliable Low-Latency Communication
VM	Virtual Machine
VNF	Virtual Network Functions
VPN	Virtual Private Network

VR Virtual Reality

ZSM Zero-Touch Network and Service Management

“The secret of getting ahead is getting started.”
Mark Twain (1864 – 1910)

1 Introduction

Network data traffic is experiencing massive changes due to increasing complexity and heterogeneity of devices, applications, and communication protocols. The growing diversity of connected devices, ranging from smartphones and laptops to Internet of Things (IoT) sensors and autonomous machines, introduces a wide range of traffic patterns and requirements. Similarly, the proliferation of various applications, such as video streaming, online gaming, cloud services, and real-time communications, each with its own distinct latency, bandwidth, and Quality of Service (QoS), contributes to the complexity. Historically, network traffic was predominantly comprised of data originating from computers and servers. However, the future of network traffic is set to be a diverse mix, with a variety of applications (Virtual Reality (VR), Augmented Reality (AR)) and IoT devices utilizing the network. This will necessitate a network infrastructure capable of handling a myriad of traffic types, including sensor data, video streaming, Machine-Type Communications (MTC), Software-Defined Networks (SDN), and smart cities, each with its unique QoS requirements. Moreover, the emergence of smart cities will significantly contribute to the surge in network traffic, with the entire urban infrastructure interconnected and exchanging data. Furthermore, the fast adoption of cloud services and network virtualization has added another layer of complexity to resource management.

According to Ericsson reports, the demand for internet traffic is increasing each year [5]. Global mobile network data traffic will exceed 400 EB per month in 2029. Figure 1 describes the expected increase in traffic and the breakdown of data categories. As a key insight, Video traffic data will be the most used globally. Indeed, the traffic will be heterogeneous and have different QoS which will be addressed with network slicing. Moreover, the heightened network complexity and diversity necessitate the implementation of precise resource management.

Nowadays, and in the future, with the diversity of applications and devices that will use the network, ensuring the optimal usage and allocation of the resources for Mobile Network Operators (MNOs) is mandatory. This complexity will continuously grow due to the dynamic nature of modern networks, where the resources must be allocated and reallocated in real time to accommodate the different demands and QoS. Furthermore, the trend of hosting applications and

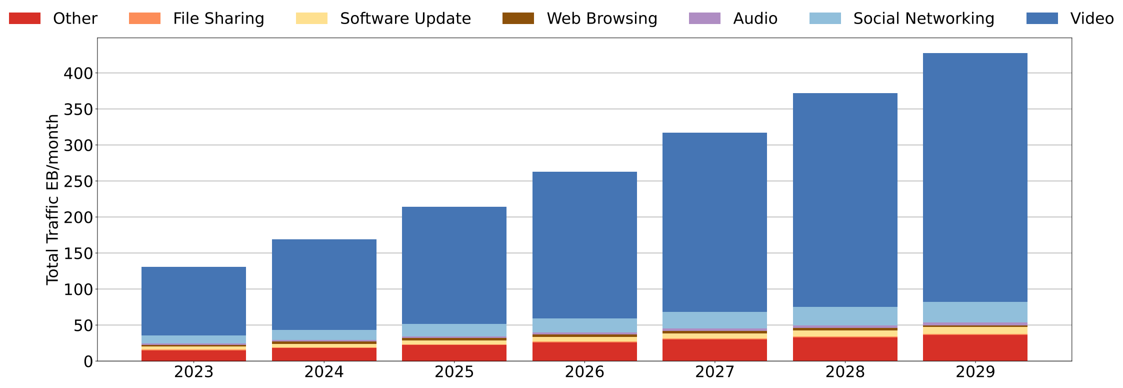


Figure 1.1: Mobile traffic by application category, adapted from [5].

services in cloud environments makes it critical to ensure efficient resource allocation to achieve end-users' Quality of Experience (QoE). MNOs will be challenged to efficiently and optimally manage bandwidth, processing power, and data storage.

To cope with these challenges, MNOs are turning to more configurable and automated network management strategies. Traditional reliance on human interaction for network management is proving insufficient to handle modern network complexities. Historically, the approach was to overprovision resources, ensuring all services meet their Service-Level Agreements (SLAs) by preparing for peak demand scenarios. However, this strategy leads to significant underutilization of resources during periods of lower activity, resulting in increased operational costs for MNOs.

In response to this inefficiency, cloud-native mobile network architectures and automation technologies are becoming crucial components of MNO strategies. By leveraging softwarization, MNOs can deploy more flexible and adaptable networks. The shift towards cloud-native architectures, in collaboration with major cloud service providers, is unlocking new business opportunities by enabling rapid configuration and scaling based on real-time demand. Additionally, automated management, powered by Machine Learning (ML) and Artificial Intelligence (AI), allows for more efficient resource allocation, improving network performance while reducing operational costs.

Within this evolving landscape, network slicing has emerged as one of the most promising innovations in the evolution of modern telecommunications networks, particularly with the advent of 5G and beyond. At its core, network slicing allows a single physical infrastructure to be divided into multiple, independent, and highly customizable logical instances, referred to as "slices" [8].

Fig. 1.2 illustrates a high-level representation of network slicing architecture, showcasing how diverse service requirements can be accommodated by logically isolating network resources. This is achieved by overlaying distinct logical slices over a shared physical infrastructure, which enables a single network to handle varied service types efficiently, each with unique performance and reliability demands. At the core of this structure is the 5G Service-Based Architecture (SBA), which comprises essential functions like the Network Slice Selection Function (NSSF), Access and Mobility Management Function (AMF), and Session Management Function (SMF). These

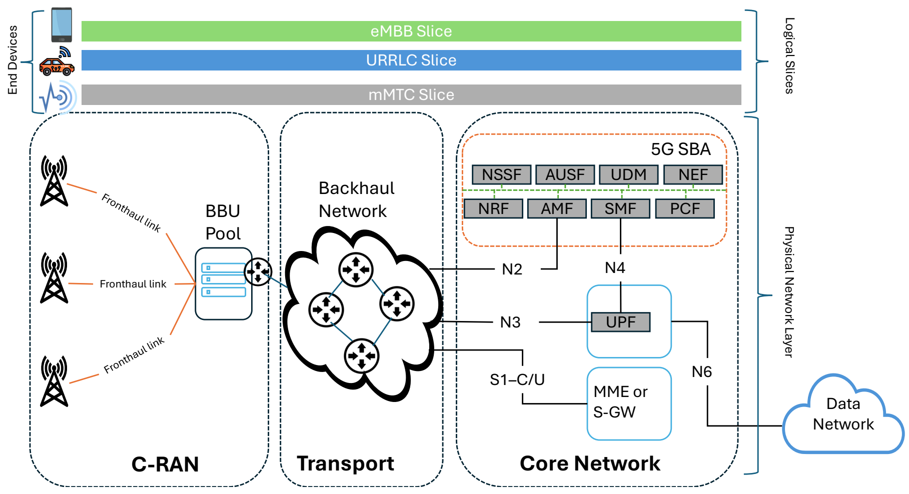


Figure 1.2: High-level representation of network slicing architecture, illustrating logical slices (eMBB, uRLLC, and mMTC) over a shared infrastructure, managed by core 5G functions (e.g., NSSF, SMF) and supported by C-RAN and transport networks to meet varied service needs efficiently. Adapted from [9].

components work together to allocate, monitor, and manage the necessary resources for each slice. The 5G SBA facilitates seamless connectivity between different network segments, ensuring a responsive and efficient experience across diverse applications. Supporting the logical slices is the Cloud Radio Access Network (C-RAN), which includes the Baseband Unit (BBU) Pool and the Transport Core Network (TN). This setup provides a scalable and centralized physical infrastructure that can flexibly adapt to network demands. The C-RAN connects via both fronthaul and backhaul networks to maintain high-speed communication between the access and core networks, providing a robust foundation for network slicing. Three primary logical slices—enhanced Mobile BroadBand (eMBB), ultra-Reliable Low-Latency Communication (uRLLC) and massive Machine-Type Communication (mMTC)—are shown. Each slice is optimized for a specific type of service. The eMBB slice supports high-data-rate applications such as video streaming and augmented reality, the uRLLC slice caters to applications that require ultra-low latency and high reliability, such as autonomous driving and telemedicine, while the mMTC slice supports a massive number of connections for IoT applications like smart cities and industrial automation [10]. The ability to define, allocate, and manage these slices individually not only provides efficiency but also ensures that the SLA for each service is met, guaranteeing performance parameters such as speed, reliability, and security. This capability is particularly significant in an increasingly diverse digital landscape where different types of applications have vastly different networking requirements. Traditional static networking approaches are no longer sufficient to handle the varying demands of real-time gaming, autonomous vehicles, telemedicine, and large-scale IoT networks [11]. Network slicing addresses this challenge by providing flexibility and scalability, enabling network operators to dynamically allocate resources based on the specific needs of each service [12]. It also reduces the complexity and cost associated with deploying and managing multiple physical infrastructures for different services, as the same physical network can simultaneously support several use cases [13].

1.1. Motivation

One of the most significant challenges in network slicing is the effective management of resources. Network slicing involves creating isolated virtual networks on a shared physical infrastructure, which inherently increases the overall capacity requirements. As each slice operates independently, the isolation of resources across these slices adds complexity, making dynamic, preemptive, and efficient resource allocation critical to maintaining performance and efficiency in such networks. According to Márquez et al. [14], the isolation of resources not only impacts capacity but also heightens the demand for more refined resource management techniques, which are essential for the realization of network slicing in practice.

In traditional, non-sliced networks, resource management has typically been handled through reactive, human-driven approaches, where manual intervention is required to adjust resource allocation based on real-time demands. However, these legacy approaches are becoming increasingly inadequate in the context of sliced networks due to several factors. First, the service demands in modern networks fluctuate rapidly. Services that utilize network slices often experience varying levels of traffic, from low-latency applications like autonomous driving to high-bandwidth services such as video streaming. This dynamic nature requires real-time adjustments in resource distribution, which human-driven systems are ill-equipped to handle efficiently [15]. Furthermore, the complexity of the slicing ecosystem —where multiple network slices may be created, modified, or terminated on-demand— further exacerbates the limitations of traditional resource management methods. The static allocation strategies used in the past cannot meet the dynamic requirements of network slicing, leading to inefficiencies and potential service degradation.

Another important aspect of network slicing is its role in the emerging trend where network infrastructures no longer serve a single operator but are shared among multiple operators with different business models. Traditionally, network operators owned and managed their infrastructures independently. However, with the advent of 5G and beyond, the increasing complexity and cost of deploying and maintaining these networks have led to the rise of multi-tenant infrastructures, where different MNO, Mobile Virtual Network Operator (MVNO), and even third-party service providers can share the same physical resources. In this scenario, network slicing becomes crucial as it allows operators to create isolated virtual networks that can be customized for their specific needs while sharing the underlying infrastructure. This shift introduces new challenges, including how to ensure fair and efficient resource allocation among competing operators, how to maintain security and performance isolation across slices, and how to handle diverse SLA that may vary widely between operators. It is, therefore, paramount to accurately characterize these new operator models and address the unique challenges they bring, as failing to do so could result in inefficiencies and conflicts that undermine the benefits of network slicing.

In addition to the shared infrastructure models, there is a rapidly growing demand for connecting highly heterogeneous devices and terminals operating across diverse global environments, each with unique performance requirements. This demand stems not only from the proliferation of IoT devices but also from the increasing number of individuals embracing a digital nomad lifestyle, where constant connectivity is essential for work and personal life. The COVID-19 pandemic has accelerated this shift, creating a large, mobile workforce that requires seamless, ubiquitous connectivity as they move across different regions [16]. Moreover, IoT applications, such as connected vehicles, smart meters, and other critical services, now rely on uninterrupted global connectivity to function effectively [17, 18]. Addressing these demands adds another layer of complexity to network slicing, as operators must ensure robust support for international roaming, diverse SLAs, and highly varied device requirements. This further emphasizes the importance of scalable, flexible network architectures capable of adapting to a wide range of use cases.

In conjunction with network slicing, the integration of AI into network management processes represents a critical advancement in improving the efficiency and performance of modern networks. AI and ML techniques can be leveraged to predict network demand from vertical tenants, such as Service Providers (SPs), who may require different levels of capacity based on the services they offer [19]. By analyzing historical data and real-time network metrics, AI-driven systems can forecast future network traffic patterns and resource needs with high accuracy. This predictive capability enables more efficient resource allocation, ensuring that the appropriate amount of bandwidth, computational power, and other network resources are available when needed, without overprovisioning or underutilization.

As a result, the research community has increasingly focused on finding innovative solutions for resource allocation and admission control in network slicing. Admission control involves deciding whether to accept a new slice request based on available resources, while resource allocation deals with assigning the appropriate amount of resources to each accepted slice. Together, these tasks are central to the success of Network Slicing as a Service (NSaaS), as the decisions made during these processes directly impact the quality of service and efficiency of the network. Much of the potential advantage that NSaaS can offer to MNO depends on making correct decisions on whether to accept a network slice and, if so, determining the appropriate allocation of resources to it [15]. A misallocation could lead to either underutilization of network capacity or SLA violation, both of which can degrade overall network performance.

To address the challenges posed by traditional methods, automated decision-making processes are increasingly being integrated into network orchestrators, the systems responsible for managing network slices. These orchestrators must now make real-time, intelligent decisions on resource allocation, often with minimal or no human intervention. By leveraging AI techniques, these systems can analyze network conditions, predict traffic patterns, and optimize resource distribution across slices more effectively than human operators could [20]. AI-powered network orchestration is seen as a critical enabler for the future of network slicing, as it promises to enhance the scalability, efficiency, and adaptability of sliced networks. For instance, AI can help predict when a particular slice will require additional resources based on historical traffic data and adjust resource allocations accordingly, thereby preventing congestion or service interruptions.

The integration of AI into network management paves the way for the realization of *zero-touch networks*, a concept that envisions fully autonomous communication infrastructures capable of self-managing without human intervention. This vision aligns with the growing trend toward automation in network management, where AI algorithms handle everything from fault detection and recovery to resource provisioning and optimization. The path to zero-touch networks involves automating not only routine tasks but also complex decision-making processes, such as those involved in resource management for network slices [21]. By enabling networks to operate autonomously, zero-touch solutions aim to improve reliability, reduce operational costs, and enable more responsive and efficient service delivery.

The idea of zero-touch networks is not merely theoretical but is actively being pursued by industry bodies like European Telecommunications Standards Institute (ETSI). The ETSI Zero-Touch Network and Service Management (ZSM) group is working on standardizing architectures and frameworks that will allow for fully automated network operations, including in the context of network slicing. The goal is to create a standardized approach to managing slices in a way that minimizes human intervention while maximizing operational efficiency and scalability [22]. This movement toward standardization is essential to ensure that different vendors and operators can implement zero-touch solutions in a consistent and interoperable manner, facilitating broader adoption across the telecommunications industry.

Furthermore, the use of AI for predictive management shifts the paradigm for MNOs from traditional reactive network management to a more proactive approach. In a reactive model, network issues are addressed after they occur, often resulting in degraded service and poor user experiences. In contrast, predictive AI systems can identify potential network bottlenecks or performance issues before they affect end-users, allowing operators to take preemptive action. This shift results in improved network availability, reduced downtime, and better overall performance [23]. By optimizing the distribution of resources in real-time, AI not only enhances the efficiency of network operations but also contributes to cost savings, as network resources are used more effectively and waste is minimized. Consequently, the integration of AI with network slicing offers significant benefits for both operators and end-users, paving the way for a more adaptive, resilient, and high-performing network infrastructure.

1.2. Contributions

In this thesis, we propose two novel frameworks for increasing the profit of MNOs through overbooking the allocated capacity of network slices and to reduce the cost of a MNO of allocation of resources in network slicing. Furthermore, we will explore the global roaming scene from the end-user perspective.

The key contributions of this doctoral thesis have resulted in two publications: one in ACM International Conference on Mobile Systems, Applications, and Services (MobiSys 2022) and another in ACM International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc 2023), both recognized as tier-1 conferences according to CORE2023[1] and QUALIS2012[2] rankings. In addition, one paper has been submitted to IEEE Transactions on Network and Service Management (TNSM) and another to the IEEE Transactions on Network and Service Management (TNSM) special issue on Research Advances Towards Effective and Sustainable Next Generation Networks.
Detailed contributions include:

A. **Overbooking in NSaaS: Forecasting and Optimization for Enhanced Profitability**:

   In our first contribution, we design and implement an innovative, comprehensive solution for overbooking-aware NSaaS. This approach addresses the joint challenges of slice admission control and proactive resource allocation by integrating: (i) deep learning-based slice demand forecasting and (ii) optimization-driven decision-making. The flexible, modular framework is adaptable to any SLA specifications and OPEX cost models. We examine the benefits for MNO across multiple aspects, including the flexibility of resource orchestration, the costs associated with resource allocation to slices, and the operator's over-provisioning strategies. This work presents the first realistic evaluation of overbooking in NSaaS, revealing the true advantages of this technology in practical applications. Our findings demonstrate that our solution improves MNO profitability by 300% compared to traditional NSaaS management methods and by over 20% compared to the latest slice overbooking strategies.

B. **Anticipatory Capacity Allocation for Cost-Efficient Network Slice Management**

   Our second contribution presents a groundbreaking model for anticipatory capacity allocation to network slices, which meticulously considers all operational costs associated with the process. These costs encompass (i) unnecessary resource overprovisioning, which can lead to inefficient use of resources and increased expenses, (ii) unmet service demands that result in lost opportunities and customer dissatisfaction, (iii) the costs related to the instantiation of resources, which include the setup and initialization expenses, and (iv) the expenses incurred during resource reconfiguration, which may arise due to changing network requirements and demand variability. To address these challenges, we introduce an approach based on multi-timescale orchestration, enabling dynamic resource management across different timescales. This model allows us to explore the balance between various cost and performance trade-offs, providing a comprehensive solution for cost-efficient slice resource orchestration. By integrating AI-based forecasting with traditional optimization methods, our approach achieves precise and adaptable resource allocation, ensuring efficient use while minimizing costs.

[1] https://portal.core.edu.au/conf-ranks/
[2] http://www.conferenceranks.com/

C. **Global Mobile Network Aggregators: Taxonomy, Network Analysis, Performance and Optimization**

   In this final contribution, we explore the importance of the roaming feature within globally operating Mobile Network Aggregators (MNAs), explaining how it is applied across various MNA models, allowing a better understanding of how MNAs impact on network slicing. We investigate the end-user performance associated with various network operator models, specifically, Mobile Network Operators (MNOs), Mobile Virtual Network Operators (MVNOs) and Mobile Network Aggregators (MNAs) under roaming conditions. This analysis focuses on the challenges MNAs encounter in managing network complexity to maintain QoE for services such as web service and video streaming NSaaS, examining key factors indicators (*e.g.*, DNS resolution delay and traceroute) and discuss the implications of various solutions for implementing international roaming. In the final part, we leverage the 5G Control and User Plane Separation (CUPS) concept to implement a realistic approach for global cellular operations, addressing some limitations of current roaming solutions. CUPS is crucial in 5G networks, allowing the separation of the packet core into a control plane that remains centralized (e.g., in the "home" country) while the user plane moves closer to the supported application (e.g., in the visited country of the end-user). We demonstrate this through a pilot implementation of a Regional Breakout (RBO) solution using open-source software, deployed on AWS infrastructure. Our results show the performance benefits of this roaming solution, highlighting its potential to enhance global cellular operations.

1.3. Outline of the Thesis

The remainder of this thesis is structured as follows: Chapter 2 provides the background and foundational concepts relevant to the research, establishing the context for the proposed solutions. Chapters 3 through 5 focus on the technical aspects and introduce the actual contributions made. Chapter 6 concludes the thesis and offers perspectives for future work. An overview of each chapter is presented below.

Chapter 2 establishes the foundation of the thesis by presenting the essential background and context necessary for understanding the subsequent research and proposed solutions. It begins by examining the integration of AI and ML in mobile network management, focusing on optimizing resource allocation and enhancing service efficiency for the increasing complexity of 5G and IoT demands. This section covers the AI/ML lifecycle, detailing stages such as data preparation, feature extraction, and model development with an emphasis on AI-driven learning for network slice traffic forecasting, comparing AI methods like Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short Term Memory Networks (LSTM) for their ability to handle the spatiotemporal dependencies inherent in mobile traffic. Additionally, the section provides a taxonomy of mobile network operators (MNOs, MVNOs, MNAs) and analyzes roaming performance within global network aggregators. Key research gaps are identified, including overbooking mechanisms, real-world tenant demands, cost-oriented orchestration, and the end-user experience in QoE evaluations, especially in international scenarios.

Chapter 3 presents a practical overbooking-aware Network Slicing as a Service solution, kaNSaaS, which demonstrates the feasibility of implementing overbooking for network slices in a real-world scenario by leveraging AI models and optimization techniques. This approach results in a fourfold increase in the net profit of an MNO. The chapter begins with an introduction to the problem, highlighting the advantages of network slicing, cloud-native NSaaS management, and the concept of network slice overbooking itself. The proposed kaNSaaS framework is then introduced as a two-timescale decision-making model consisting of long-term admission control and short-term resource allocation for network slices. The chapter concludes with a comprehensive experimental evaluation of the framework and a detailed presentation of the results.

Chapter 4 presents AZTEC+, a multiscale time orchestration framework designed to minimize the overall operational costs associated with network slicing. The chapter begins by discussing the significance of zero-touch resource allocation, as well as the importance of resource instantiation and reconfiguration. It then introduces the orchestration model and its inherent trade-offs, followed by a detailed explanation of the AZTEC+ framework. AZTEC+ is capable of identifying the optimal interval for resource allocation that minimizes the total cost of SLA violations, overprovisioning, resource allocation, and resource instantiation by utilizing deep neural networks and optimization algorithms. It achieves this by dividing the total network capacity into two segments: dedicated capacity for each slice and shared capacity among all slices. Through multiscale time orchestration, AZTEC+ efficiently allocates the necessary resources for each slice, thereby minimizing operational costs. The chapter concludes with a comprehensive evaluation of the results, comparing AZTEC+ against various benchmarks.

Chapter 5 explores how new operator models influence the performance of various services supported by distinct network slices. Specifically, it delves into the performance and user experience impact of different MNA as they operate under roaming conditions in Spain and the USA. The chapter begins by introducing the landscape and evolution of Network Operators. It then provides a comprehensive taxonomy of MNO, MVNO, and MNA, along with an overview of roaming fundamentals. Following this, the chapter details the experimental setup designed to conduct a series of tests aimed at understanding the roaming operations of MNA and characterizing their performance. To achieve this, we subscribed to several MNA in both the US and Spain and analyzed key NSaaS such as web and video services, characterizing key factors such as traceroute and DNS resolution delay. It follows by comparing the performance of MNA with that of local MNO in both Spain and the USA. In the final part, we apply the 5G Control and User Plane Separation (CUPS) concept to create a practical solution for global cellular operations, overcoming some limitations of current roaming systems. We showcase this through a pilot implementation of a RBO solution using open-source software on AWS infrastructure. The results demonstrate the performance improvements of this approach, indicating its potential to enhance global cellular operations.

Chapter 6 summarizes the key findings and conclusions of the thesis, providing a comprehensive overview of the research outcomes. It then outlines potential directions for immediate future research, followed by a broader vision of the research field over the coming decade.

2 Background

In the rapidly advancing mobile network landscape, MNO are increasingly relying on AI and ML to optimize network management and resource allocation, especially as network demands multiply with the integration of 5G and IoT. To manage this complexity, AI and ML have become indispensable tools in network orchestration and resource management, playing a pivotal role in optimizing network slicing and ensuring efficient service delivery.

This background section explores the lifecycle of AI and ML in network management, the role of AI in network slicing and examine the challenges faced by MNO and MNA in ensuring seamless connectivity and quality of service for a diverse range of applications. By identifying the gaps and limitations in current solutions, this section lays the foundation for the contributions and innovative frameworks proposed in the subsequent chapters of this thesis.

2.1. The Lifecycle of AI/ML in Network Management

Integrating AI and ML into network management represents a transformative shift in how modern networks are monitored, optimized, and secured. As illustrated in 2.1, The lifecycle of AI/MLL in network management encompasses a series of stages that include data preparation, development, and operation phases [6]. This section overviews each phase, highlighting key processes, tools, and best practices that facilitate successful AI/ML integration in network operations, ensuring robust, adaptive, and future-ready network infrastructure.

2.1.1. Data Preparation

Data quality sets the upper limits for the performance of any AI/ML-based product, driving the recent shift toward data-centric AI approaches [24]. However, high-quality datasets are often scarce in real-world network environments due to their inherent complexities. On average, ensuring data quality consumes around 60% of the time in AI/ML projects [25]. Therefore, it's essential to focus on data preparation, particularly on data acquisition and feature extraction, to provide ML algorithms with reliable, high-quality input.

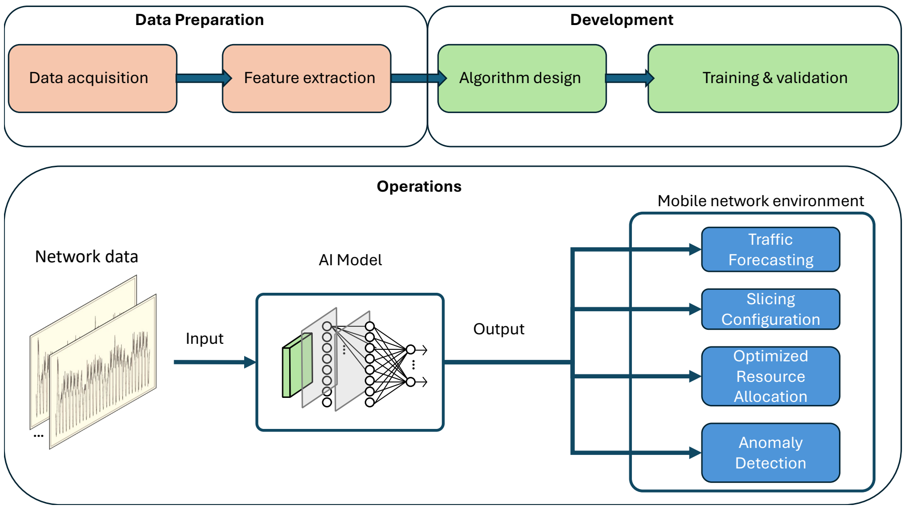


Figure 2.1: ML Lifecycle in production settings. Adapted from [6].

Data acquisition. Typically, data sources fall into three categories: (i) live networks, (ii) controlled environments, (iii) curated public datasets [6] and synthetic data (iv). When collecting data directly from live networks, various methods can be employed, though they often entail significant operational costs, requiring careful trade-offs [26]. For instance, in high-speed networks, sampling is usually preferred over per-packet collection to minimize disruption to the data path. Additionally, collecting data in real-world conditions can introduce challenges like packet drops, sampling biases, or schema changes, which may result in data irregularities or outliers. Labeling data is a particularly labor-intensive process, requiring extensive human input that doesn't scale well with large data volumes [27].

While advanced techniques like weak supervision, semi-supervision, transfer learning, and active learning aim to address data scarcity, they still rely on initial labeled datasets or human intervention, limiting scalability and effectiveness in managing complex datasets. In controlled environments and public datasets (cases ii and iii), the data often originates outside the target networks, which can cause statistical misalignments with the deployment environment, leading to issues like data drift. Rigorous testing is therefore essential to identify potential biases or anomalies and ensure that the model performs reliably when deployed.

Synthetic data, as an emerging approach, adds another powerful tool for addressing data limitations. Unlike traditional methods, synthetic data is generated programmatically, simulating conditions within a target environment to create realistic data samples. This approach enables large-scale data creation without the need for continuous access to live network conditions or extensive labeling efforts, significantly reducing costs and privacy concerns. Synthetic data also allows researchers to engineer specific scenarios or edge cases that may be rare or hard to capture in live environments, improving model robustness and generalizability. However, the fidelity of synthetic data must be carefully evaluated; any statistical misalignments between synthetic and real-world data could lead to issues such as bias or overfitting if the synthetic data does not sufficiently represent deployment conditions. Consequently, synthetic data serves as a complement to real-world and controlled data, providing scalable and flexible data sources while underscoring the need for careful validation to ensure effective model performance.

Feature extraction. To make raw network data usable for AI/ML algorithms, it must first be transformed into relevant features. This process, however, presents several challenges. Different feature sets impact both system costs and model performance, warranting careful selection and refinement. Many ML-based network solutions rely on empirically defined, custom features, which can be difficult to scale or replicate in real-world deployments. Additionally, feature selection methods may require frequent adjustments as networks evolve. Network conditions and traffic patterns continually shift in live environments, potentially rendering existing features outdated and necessitating the development of new ones [28]. Thus, feature engineering must be dynamic, adapting to ongoing changes to maintain model accuracy and efficiency over time.

2.1.2. Development

Model development in network management is an iterative process, where each cycle involves assessing the current model's performance against previous versions to determine its readiness for live deployment [29]. This process includes two essential steps: algorithm design and model training and validation. Both are critical in shaping a solution's ability to meet the specific requirements of the target network. Algorithm design focuses on selecting or creating a model architecture that can effectively address the network's unique challenges. Model training and validation then fine-tune this design, ensuring that the model performs well on historical data and is robust enough for dynamic, real-world network conditions.

ML in network management serves three main objectives: (i) utilizing existing knowledge effectively, (ii) uncovering structured insights from unknown data, and (iii) achieving specific goals through learned behaviors. These objectives correspond to three primary ML approaches—Supervised Learning, Unsupervised Learning, and Reinforcement Learning (RL)—with hybrid methods, such as semi-supervised or self-supervised learning, bridging gaps between them. In this section, we will focus solely on Supervised Learning, as it is the approach we will use for traffic forecasting within network slices—a task best addressed for regression taks. Supervised Learning leverages labeled data to train models that can make accurate predictions based on historical trends. In network slice traffic forecasting, regression models allow us to predict traffic demand across different slices by analyzing historical usage patterns and identifying trends specific to each slice. This enables proactive resource management, optimizing slice performance and reliability based on anticipated traffic patterns across the network. In the context of model training and validation, considerations such as inference efficiency, generalizability, and safety are just as important as accuracy [6].

2.1.3. Operations

Once our AI model is trained and validated, it is ready for operational deployment. In this phase, the model receives input in the form of network data, encompassing both real-time and historical metrics such as traffic patterns, latency measurements, and signal strength. Operating in inference mode, the model applies its learned insights to optimize specific aspects of mobile network management, including traffic forecasting, slice configuration, resource allocation, and anomaly detection.

Our primary focus will be on traffic forecasting within network slices. By predicting traffic demand across different network slices, the model enables proactive adjustments, helping operators manage capacity more effectively and maintain consistent QoS across diverse network environments. This targeted approach to traffic forecasting of network slices supports efficient, adaptable, and resilient mobile network operations.

2.2. Traffic Forecasting of Network Slices

Knowing the required resources for each slice in advance over a specified time interval is advantageous, as it allows for pre-allocation of resources and helps avoid SLA violations [30]. Traditionally, non-ML approaches like ARIMA [31] and Holt-Winters [32] have been popular for temporal forecasting; however, they lack the capability to capture complex spatiotemporal patterns in mobile traffic influenced by user mobility [33]. Consequently, various ML-based, particularly Deep Learning (DL)-based, forecasting techniques have gained traction in recent years, as they effectively capture spatiotemporal dependencies in mobile traffic and enable automated, intelligent resource provisioning for network slicing [34]. Empirical studies further highlight the superiority of DL algorithms, such as LSTM, over traditional models like ARIMA and Holt-Winters [35, 36].

Phyu *et al.* [9] provide a comprehensive survey on how deep learning has been employed to enhance the forecasting of network slicing. They divide the contributions on traffic forecasting in network slicing in two categories: (i) CNN-based forecasting and (ii) RNN-based forecasting:

CNN-based forecasting. CNN and 2DCNN have notable limitations in capturing temporal features, as they are primarily designed for image-processing tasks [37]. To address this, the authors in [38] proposed Deepcog, a cost-aware network capacity forecasting framework based on 3DCNN. In their approach, historical antenna-level data traffic from each base station serves as input to train the model. The cost to the operator is linked to both resource over-provisioning and SLA violations. Deepcog forecasts network demand by analyzing spatiotemporal features, which enables more efficient resource pre-allocation for individual mobile services. A unique cost-aware loss function was also designed to optimize the ML framework, reducing the operator's overall expenses. When tested, Deepcog outperformed baseline methods, including traditional non-ML solutions, LSTM, and autoencoder models, particularly in forecasting at a five-minute interval with a time window extending up to eight hours. The results show that Deepcog reduces resource overprovisioning, SLA violations, and overall monetary costs. Notably, a longer forecasting time window tends to increase SLA violations and resource overprovisioning. For example, a five-minute prediction window results in approximately 3% SLA violations and 15% resource overprovisioning for a given slice, whereas an eight-hour prediction window raises these values to 10% and 30%, respectively. Similarly, the study in [39] 3DCNN for forecasting to reduce resource overprovisioning and proactively allocate resources to meet anticipated slice demand. Their findings further validate that ML-based forecasting methods can lower monetary costs by over 50% across all tested scenarios compared to traditional non-ML forecasting approaches.

RNN-based forecasting. Many studies employ LSTM or its variants, such as ConvLSTM, as forecasting techniques for resource management and automated network slicing. Despite its popularity and effectiveness in capturing spatiotemporal and long-term dependencies, LSTM is computationally intensive. To address this, some research leverages lower-complexity methods like GRU or simple two- or three-layer ANN architectures as alternative solutions for network slicing. LSTM's strength in learning complex dependencies makes it a promising tool for forecasting in network slicing applications. For instance, [40] uses LSTM within resource management processes to predict slice bandwidth demand over a time window of 200 seconds, with 20-second intervals. Experimental evaluations demonstrate that this LSTM-based algorithm enables more users to access the network efficiently.

In [41], LSTM, CNN, and DNN models are applied to traffic forecasting for resource management within a vehicular-specific slice in an SDN-enabled 5G network. Simulation results indicate that LSTM achieves the highest average forecasting accuracy at 99.36%, compared to DNN and CNN, which reach 92.58% and 95%, respectively. Similarly, [42] demonstrates the effectiveness of LSTM for End-to-end (E2E) slice traffic forecasting with a five-second time interval and a 300-second time window, achieving accuracy three times higher than linear regression. Using LSTM for resource reservation, [43] proposes an efficient strategy from the SP perspective, where MVNO data such as traffic loads and base station capacity is not disclosed to the SP. This LSTM-based model outperforms the baseline ARIMA model in terms of Mean Squared Error (MSE), as well as in reducing over-reservations and under-reservations, with a forecasting time interval of 10 minutes and a time window of 744 hours.

Beyond slice traffic forecasting, LSTM has been applied to other tasks within the sliced cellular architecture. For instance, [44] leverages LSTM to predict transmission channel conditions with a 24-hour time window and one-hour intervals, enabling efficient slice creation by service providers. Here, the LSTM output feeds into a DNN to assess the network's capacity to accommodate new slice requests, demonstrating performance improvements over traditional analytical approaches.

Drawing from this extensive literature, it is clear that LSTM and its variants are widely applied for forecasting network slice traffic and resource utilization. However, LSTM's computational cost is relatively high compared to other methods like GRU, which can achieve similar forecasting results being less computianotally expensive [45]. the authors of [46] propose a streamlined GRU model for forecasting resource usage per network slice. Their model simplifies the GRU structure by removing the reset gate and replacing the standard tanh activation function in the update gate with a softplus function, which enhances the speed of forecasting hourly resource usage over a 120-hour time window. In this approach, the authors also integrate SLA constraints and minimize the MSE loss function within their resource management framework for network slicing. Their experimental results demonstrate that this lightweight GRU model achieves significantly faster computation times than LSTM, benefiting from its simplified architecture without sacrificing forecasting performance.

While DL-based traffic forecasting offers significant benefits, it also has limitations. One major challenge is the reliance on high-quality, real-time data. Traffic forecasting models require large amounts of data to train effectively, and the accuracy of their predictions depends on the availability of up-to-date traffic information. In cases where data is missing or inaccurate, the performance of these models can degrade, leading to suboptimal resource allocation.

To address the challenges associated with resource limitations, *overbooking* has emerged as a practical approach in network slicing. Overbooking allows service providers to allocate resources to multiple network slices based on statistical demand patterns rather than peak demand, capitalizing on the likelihood that not all slices will require their maximum resources simultaneously. This approach improves overall resource utilization by accommodating occasional traffic spikes without dedicating excessive resources to each slice individually. When paired with traffic forecasting, overbooking can enable a more agile and cost-effective resource management strategy, balancing the risk of SLA violations against the gains in efficiency.

The authors of [47] explore the concept of overbooking network slices end-to-end, demonstrating how overbooking can improve resource utilization in 5G networks. Their approach leverages dynamic resource allocation to ensure that resources are distributed across slices based on real-time demand, allowing operators to overbook resources without risking congestion. By overbooking resources, operators can maximize efficiency and reduce operational costs while maintaining the QoS for each slice. In [48], the authors build on this concept by proposing a yield-driven orchestration approach for overbooking network slices. Their approach focuses on dynamically reallocating resources based on real-time demand, allowing operators to adjust the resources allocated to each slice based on current traffic conditions. Operators can overbook resources while minimizing the risk of service degradation, ensuring critical services receive necessary resources during peak demand periods.

However, while overbooking offers significant benefits, it also carries risks. The primary risk is the potential for congestion and service degradation if demand exceeds available resources. In scenarios where traffic surges unexpectedly, overbooked resources may not be sufficient to meet demand, leading to degraded performance for some services. To mitigate this risk, operators must carefully manage the balance between overbooking and resource availability, ensuring that critical services receive priority during periods of high demand.

Additionally, overbooking requires sophisticated monitoring and control capabilities to ensure that resources are allocated efficiently and minimize congestion risk. This adds another layer of complexity to network management, particularly in large-scale 5G networks with diverse services and traffic patterns. Another layer of complexity is added by new operator models, such as MNA, introducing specific challenges to network slicing by further complicating resource management and SLA enforcement, as described in the following section.

2.3. Mobile Network Aggregators and Roaming

Mobile network aggregators combine resources from multiple network operators to offer seamless, flexible services to end-users across different networks and regions. This model increases the demand for sophisticated slice orchestration, as it requires managing resources and performance across diverse networks with potentially conflicting SLAs and varying quality standards while preserving QoE for users across various services, such as web browsing and video streaming delivered via NSaaS. Additionally, MNAs must address interoperability issues, as each operator may utilize different technologies, configurations, and management frameworks. This diversity heightens the complexity of coordinating slices, ensuring uniform service quality, and preventing resource contention. Moreover, with multiple operators involved, data privacy, security, and compliance become critical concerns, as sensitive user data flows across network boundaries. In this section, we will describe a comprehensive taxonomy of the state-of-the-art mobile operators and the contributions related with roaming performance.

2.3.1. Taxonomy: MNO, MVNO and MNA

There are several types of mobile operators with different operation models available in the market today; we capture these configurations in Figure 2.2.

An MNO[1] is an entity that owns (or has the exploitation rights) of a cellular network (i.e., base stations, network core, spectrum, etc). This was the initial operation model deployed to provide mobile communication services. Examples of MNOs include Vodafone, Orange, O2, Movistar, AT&T, NTT to name a few. Later on, the MVNO operation model emerged [49]. Specifically, the MVNO is an entity that offers mobile network services to end-users, but does not own nor operate a full cellular network. The MVNO is defined by its lack of ownership of radio spectrum resources.

In order to operate, an MVNO needs to have agreements in place to access the network of a base MNO. The implementation of the MVNO varies, and thus there are many different types on MVNOs. The type of MVNO is determined by how “thick” or “thin” a technological layer the MVNO adds over its access to its base MNO’s network [50–52].

[1]This terminology distinguishes mobile operators as a general concept and Mobile Network Operators (MNOs) that is a specific type of mobile operators (i.e., mobile operators = set(MNOs, MVNOs, MNAs)).

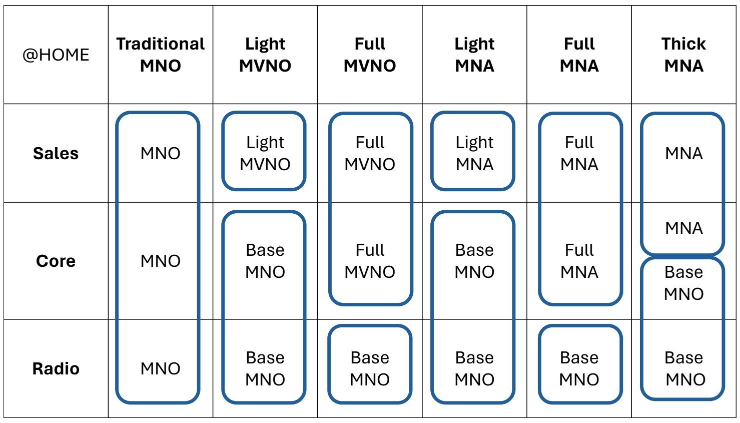


Figure 2.2: Types of MNOs.

A *branded reseller* (also known as a "skinny" MVNO) completely relies on the base MNO facilities to operate. They do not own any network elements, but they may operate their own commercial department.

A *light MVNO* is a service provider that does not run its own core network, though it has its own customer support, marketing, sales and distribution operations, and may have the ability to set its tariffs independently from the retail prices of the base MNO. One such example is giffgaff in the UK, which uses O2 UK as a base MNO.

A *"thick" MVNO* partially manages its own core network deployment with its own infrastructure, which allows the MVNO more control; however, it still depends on a base MNO for some core network functions. These MVNOs have a heavier focus on branding, customer-ownership, and differentiation through added services like data and SIM applications.

A *full MVNO* has a core network implementation operating essentially the same technology as an MNO, only missing their own radio network. They thus run their own core network, and rely on a base MNO who can offer access to radio resources. One example is Sky Mobile, which operates as a full MVNO in the UK, using O2 UK as a base operator.

Much more recently, we have witnessed the emergence of a new type of MVNO, namely the MNA [53]. While “traditional” MVNOs have agreements with a single base MNO, an MNA is an MVNO that exploits more than one base MNO, either in one single economy, or across different economies. Examples of MNAs include Google Fi, Truphone (now re-branded to 1Global), Twilio or Lycamobile. Aggregating multiple base MNOs allows the MNA’s customers to dynamically change the base MNO to which they attach. This change of base MNOs depends on different factors, including policy, coverage or performance.

In this thesis, we extend the currently used MVNO-specific taxonomy [50–52] to include the MNAs. We further classify them into *full/thick/light* MNAs, depending on whether they operate their own complete/partial core network or not. We also differentiate the MNAs based on the geographic coverage of the multiple base MNOs they aggregate. In the general case, the base MNOs aggregated can cover the same or different geographic regions. A particular case is when the different base MNOs aggregated provide coverage in different geographic regions that do not overlap (notably, different economies). If this is the case, we call this specific type of MNA a multi-country MVNO. These multi-country MVNOs usually have commercial offers in each of the different economies where they operate.

We acknowledge that, as in most taxonomies, there are corner cases that we cannot neatly classify into one of the categories. In our case, there is the case where a full MVNO has a commercial agreements with one or several IPX Hub Breakout Provider (IPX-P) [54], and does not depend on a specific base MNO (e.g., the MVNO might use global IMSI ranges). In this case, with a single agreement, the MVNO has “direct” access to several base MNOs located in different economies, depending on the footprint of the IPX-P. This configuration lies somewhere between the full MNA and the full MVNO, since it has a single agreement but connects to multiple base MNOs. However, we classify this in the full MVNO category, since it is closer to the case where the MVNO has an agreement with a single entity and leverages its roaming agreements.

We highlight the lack of knowledge in our community around how these different models of MNAs satisfy the need for global coverage for their end-users. We further give background on roaming, which is one of the fundamental functions mobile operators ensure to their end-users, and is specifically relevant for these operators that aim for global uninterrupted service.

Network performance in roaming scenarios has become increasingly critical as mobile networks evolve to support diverse applications, from traditional mobile communications to IoT, robotics, and high-speed vehicular networks that will need services globally. Recent studies highlight key advancements in network architectures, signaling protocols, and optimization techniques essential for enabling seamless connectivity in high-mobility and cross-border contexts

2.3.2. Network Performance in Roaming

The transition from 3G to 4G architectures has brought significant improvements in throughput and latency, directly impacting roaming performance. The authors of [55] extensively evaluated these performance metrics and highlighted the performance gains from 3G to 4G, demonstrating the potential for enhanced roaming services with the continued evolution of mobile architectures. Their findings indicate that 4G technology’s increased data rates and reduced latency provide a strong foundation for roaming advancements as networks prepare for 5G and beyond. Complementing these findings, in [56], the authors explored the potential of regional breakouts within the IPX framework for global roaming, showing that routing data through regional breakouts can reduce latency by minimizing the data’s travel distance across networks. These improvements in roaming latency provide substantial benefits for user experience in international scenarios and serve as essential features as networks continue to expand globally.

The unique signaling demands of IoT devices add another layer of complexity to roaming performance, as these devices often require frequent, small data transmissions that place heavy loads on both home and visited networks. A study in [57] investigated the effects of signaling traffic on roaming networks, particularly focusing on IoT applications, and emphasized the need for optimized signaling protocols to prevent congestion caused by frequent IoT signaling. The challenges of supporting IoT devices in roaming contexts are further highlighted by [17], who explore the differences between traditional mobile and IoT devices. They note that IoT devices, unlike conventional mobile devices, generate smaller but more frequent data packets, necessitating specialized roaming strategies to manage this distinct traffic pattern effectively. Together, these studies indicate that while traditional roaming protocols efficiently support mobile phones, IoT applications require tailored protocols and infrastructure to achieve similar levels of performance.

QoE in roaming contexts has been a focal point of research, especially in regions like Europe, where cross-border connectivity has become more prevalent with initiatives like "Roam Like Home." In [58], the authors conducted a detailed analysis of European roaming configurations, revealing that the commonly used home-routed roaming approach can add over 60 milliseconds of latency, significantly impacting QoE. They find that this additional latency degrades user experience, particularly in intercontinental cases where latency-sensitive applications, such as browsing and streaming, suffer from increased load times. The authors of [50] build on this work by comparing the roaming performance of MNO and MVNO. Their study finds notable differences in Round-Trip Time (RTT) and routing efficiency, with MNOs generally providing more direct paths and reduced latency compared to MVNOs. These distinctions are crucial for QoE, as they highlight the influence of infrastructure and operator type on roaming performance.

In high-mobility and industrial applications, where even brief interruptions can disrupt operations, roaming demands new approaches to maintain consistent performance. In [59], the authors explored Time-Sensitive Networking (TSN) redundancy within Wi-Fi 6 and 5G networks to ensure zero-delay connectivity for mobile robots. Their approach leverages IEEE 802.1CB TSN redundancy to route traffic across multiple channels, achieving seamless, delay-free handovers even during rapid mobility. Such innovations are particularly valuable in factory environments and industrial applications, where continuous connectivity is essential to prevent operational disruptions. This focus on high mobility is further reflected in the realm of 5G with Mobility Robustness Optimization (MRO) algorithms. The study in [60] examined various MRO algorithms to support the frequent handovers required in dense urban 5G networks. They found that the Distance algorithm was the most effective in reducing handover failures and optimizing signal quality. These findings confirm MRO's role as a Self-Optimizing Network (SON) function in 5G, facilitating uninterrupted connectivity in high-density, high-mobility environments.

2.4. Research Gaps and Limitations

The landscape of network slicing research, despite its rapid evolution, is still marked by significant gaps that hinder the practical applicability of proposed solutions. These gaps, which we specifically address in this thesis, stem from the following limitations observed in prior studies: reliance on synthetic data, inadequate treatment of overbooking mechanisms, limited evaluation of real-world tenant demands, and a narrow focus on aggregate traffic rather than service-level performance. Additionally, the orchestration of network resources, a core challenge in NSaaS, has often been treated in a simplified manner, without considering the full economic impact of reconfiguration and overprovisioning decisions. Finally, the end-user perspective, particularly in terms of QoE across different MNAs, has been largely overlooked in the literature, especially when evaluating performance under international roaming conditions.

2.4.1. Challenges in Hybrid AI and Classical Optimization Models

In the field of network slicing, integrating AI and classical optimization techniques, (e.g. Mixed-Integer Linear Programming (MILP), Golden-Section Search algorithm, BOBYQA, among others), presents a promising yet underexplored approach for enhancing forecasting, admission control, and resource allocation. Despite a few studies combining AI and classical optimization techniques, a unified framework that comprehensively addresses all three aspects remains largely absent in current literature, revealing a significant research gap. Typically, the studies that try to use hybrid models solely focus on isolated aspects of network slicing. For instance, [61] developed a framework for admission control in 5G networks using a Deep Reinforcement Learning (DRL) model alongside ILP-based resource allocation algorithms. This study demonstrated enhanced admission control and network resource management capabilities but did not include mechanisms for forecasting slice demand, leaving an open area for research. Additionally, [62] proposed a hybrid solution combining DRL with MILP-based heuristics for network slice placement, enhancing resource utilization. However, while effective for slice placement, this approach did not include forecasting capabilities, which are critical for proactive resource allocation in dynamic networks.

While some studies have explored the combination of AI and classical optimization models for specific tasks like admission control and resource allocation, there remains a significant gap in developing a holistic model that simultaneously addresses forecasting, admission control, and resource allocation for network slicing. This research addresses this challenge in Chapters 3 and 4, where we propose hybrid models incorporating deep learning for forecasting alongside classical optimization techniques for admission control and resource allocation.

2.4.2. Comprehensive Cost-Oriented Orchestration Strategies

Another key gap we address relates to the orchestration of resources in NSaaS. Many existing solutions in the literature are designed to minimize overprovisioning or prevent unmet service demands, but they often fail to account for the economic costs associated with continuous resource instantiation and reconfiguration. While cost-aware orchestration solutions do exist, they tend to focus narrowly on a single dimension, such as limiting overprovisioning costs or reducing resource waste. For instance, state-of-the-art capacity forecasting models, like the one proposed by Bega et al. [63], dynamically adjust resources at every re-orchestration opportunity. While this approach minimizes overprovisioning and unmet demands, it also incurs significant costs due to the constant reconfiguration of resources. We tackle this in Chapter 4 proposing AZTEC+, based on multi-timescale orchestration, which addresses this gap by offering a holistic, cost-oriented approach that balances competing economic considerations. We introduce a novel orchestration strategy that operates at two timescales. The long-timescale orchestrator allocates a fixed dedicated capacity to each network slice, which remains constant over an extended period. This reduces the frequency of resource instantiation and, in turn, lowers associated costs. Additionally, we allocate shared capacity that can be dynamically reallocated at shorter timescales, allowing for flexibility in response to short-term demand fluctuations. This dual approach mitigates the economic burden of both overprovisioning and resource reconfiguration, resulting in a more cost-effective orchestration solution.

Through this multi-timescale orchestration model, we explore trade-offs between overprovisioning and reconfiguration costs that have been largely neglected in prior work. For the first time, we empower network operators with a comprehensive solution for managing NSaaS resources that considers the full spectrum of operating costs, including the hidden costs of excessive reconfiguration.

2.4.3. End-User QoE Across Global MNAs and Roaming Scenarios

Finally, our work also addresses a gap in the literature concerning the evaluation of QoE from the end-user perspective, particularly in the context of MNAs operating globally. Most previous studies have focused on network-level performance metrics such as throughput, latency, and resource utilization, without adequately considering the impact on user-perceived service quality. Furthermore, evaluations of MNA performance, especially under international roaming conditions, are extremely limited. Measuring performance in these contexts requires cooperation across multiple network operators and the ability to gather data from diverse international vantage points, making it a logistically complex endeavor.

In Chapter 5, we address this gap by providing a comprehensive analysis of MNA performance across multiple international contexts, including both Europe and the USA. This broader scope allows us to capture performance differences across MNAs operating under both roaming and domestic conditions, offering the first global view of MNA QoE. Our findings reveal important variations in performance that were previously unknown, particularly in how MNAs handle international traffic compared to domestic traffic. By focusing on the end-user perspective, we provide insights that are directly relevant to service providers seeking to improve user satisfaction in increasingly globalized mobile networks.

2.4.4. Dependence on Synthetic Data and Limited Real-World Evaluation

A significant portion of the network slicing literature relies on synthetic data to evaluate slicing algorithms and NSaaS management strategies. While synthetic data can be useful in controlled experimental settings, it fails to capture the complexity and variability of real-world tenant demands. This undermines the reliability of the conclusions drawn from such studies. For instance, in existing works like Hurtado et al. [64], the data used to validate slicing approaches is artificially generated, which limits the generalizability of the findings to real-world networks.

Our work, described in Chapter 3-5, departs from this approach by using large-scale, real-world measurements of tenant demands. These measurements are derived from actual mobile traffic, offering a much more representative view of network performance under realistic conditions. The studies that do incorporate real-world data tend to focus on aggregate traffic across services, failing to capture the granular behavior of individual tenants and their specific demands [65–67]. Such aggregated traffic may not fully reflect the unique service-level requirements that are critical to accurate overbooking decisions.

In summary, this work addresses several critical gaps in the literature on network slicing and NSaaS. We offer a unique contribution by evaluating overbooking strategies using real-world traffic measurements, providing a more dependable basis for future research and practical application. Our multi-timescale orchestration model introduces a comprehensive, cost-aware solution that balances overprovisioning and reconfiguration costs in a novel way. Finally, we contribute to the underexplored domain of end-user QoE by evaluating global MNA performance within NSaaS context across both domestic and roaming scenarios. These advancements not only fill key gaps in the current body of research but also provide actionable insights for network operators seeking to optimize both the efficiency and the user experience of their network slicing solutions.

3 Overbooking in NSaaS: Forecasting and Optimization for Enhanced Profitability

As discussed in Chapter 1, Softwarization has marked the evolution of mobile network infrastructures over the past decade, and MNO are today experimenting with proofs-of-concept and early deployments of cloud-native network technologies, supported by major cloud service providers [68, 69]. The dramatic increase in flexibility granted by production-grade cloud-native mobile network architectures will finally open new and long-envisioned business opportunities for MNOs. One of the most promising is network slicing, which abstracts a single physical infrastructure into multiple logical instances, or slices [70]. Each network slice is dedicated to specific traffic flows (*e.g.*, the video streaming demand generated by mobile clients of a given platform) and is configured so as to provide strong guarantees that the SLA for such traffic is met (*e.g.*, in terms of latency, throughput, or jitter).

Cloud-native NSaaS management. Cloud-native network architectures offer a natural support to network slicing operations [71]: they allow assigning dedicated resources (*e.g.*, spectrum, transport capacity, compute or memory resources, depending on the target network domain) to each slice [72], configuring dynamically the Virtual Network Functions (VNF) according to the SLA of each slice [73], and monitoring the fulfillment of such SLA [74]. As a result, the cloudification of networks implicitly paves the way to the realization of NSaaS models. Here, MNOs deliver slices to vertical customers, *i.e.*, Service Providers (SPs) who are able to configure their assigned slices up so as to best run their applications [75]. The NSaaS model ultimately creates a new marketplace that allows operators to maximize their revenue through an appropriate slice brokering [76].

Network slicing has drawn significant attention from the research community in the past years, and studies have tackled many challenges in the practical implementation of this paradigm. Among those, admission control and resource allocation are central tasks: a great portion of the potential advantage that NSaaS can bring to MNOs depends on correct choices on whether to accept a slice, and, if so, with what dedicated resources. As we discussed in detail in Chapter 2, prior works have addressed both these problems, possibly in a joint fashion. Yet, the vast majority of the studies in the literature overlooks an important degree of freedom for the operator, *i.e.*, its flexibility in allocating resources that are not necessarily those specified by the SLA, as explained next.

Overbooking network slices. Cloud-native technologies allow the MNO to orchestrate resources and VNFs at much faster timescales than those of NSaaS brokering. Thus, while the network slice tenant requests resources for its peak consumption over long reservation periods, the actual allocation and re-configuration of slice-dedicated resources can be performed at a finer time granularity. In addition, the operator has in-depth visibility of the actual infrastructure utilization, and can hence allocate resources based on the real resource occupancy generated by the service demands, beyond the capacity requests issued by the vertical tenants. These technical advantages, illustrated in Fig. 3.1a, open the door to large slice multiplexing gains, letting the MNO make a more efficient use of its resources [30] and ultimately increasing its profit.

Specifically, reducing the amount of capacity needed to serve each slice can free up space for accommodating more requests, as exemplified in Fig. 3.1b. In other words, the MNO can sell more capacity than it has deployed, considering that vertical customers will not use all the capacity they requested all the time. The strategy maps to *overbooking*, a well-known revenue management approach used to maximize profit in scenarios where limited resources must be reserved based on stochastic requests [77]. In the case of overbooking for NSaaS, errors in admitting excess slices come at the cost of monetary fees for violating the SLA with one or multiple tenants during some fraction of time, as also shown in Fig. 3.1b. Overbooking has been recently considered as a way to increase NSaaS revenues for the MNO, with promising results [48, 78, 79]. Yet, as detailed in Chapter. 2, prior studies are few, have technical limitations, and none has demonstrated practical solutions with real-world traffic demands of vertical customers collected in actual operational networks. The latter aspect is especially critical now that cloud-native networks are bringing slicing closer to deployment, and there is a clear need to understand how overbooking would perform in production systems. While we cannot disclose the latter due to confidentiality agreements, we release[1] the synthetic traffic together with our implementation of kaNSaaS, so as to foster the reproducibility of our study and support further investigations.

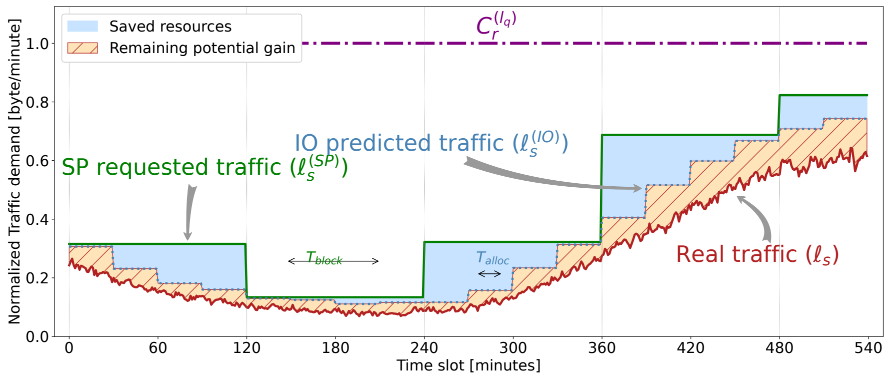
Overall, our work contributes to advance the state of the art in NSaaS management, and sheds light on the actual gains that overbooking can bring to the MNO in production settings.

3.1. System Model

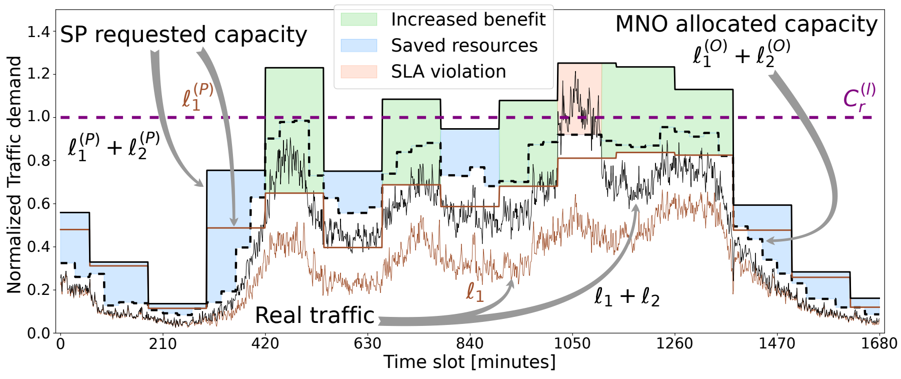
We consider a dynamic resource allocation scenario where an MNO running the mobile network infrastructure aims at maximizing the profit obtained from NSaaS. To this end, the MNO needs to take decisions on admission control of slice requests and allocation of resources[2] to active slices. The problem can be instantiated at any target network location where slicing is implemented, *e.g.*, from individual Remote Units (RUs) where spectrum can be sliced, all the

[1] Code and data are available at https://github.com/nds-group/kansaas.

[2] We use the terms *resources* and *capacity* interchangeably in the following, as the amount of dedicated resources directly determines the capacity that can be provisioned.



(a)



(b)

Figure 3.1: Overbooking in NSaaS, with notation. (a) Real traffic generated by one mobile service, requested slice capacity by the associated service provider (SP) at every slice brokering interval T_{SLA}, and actual capacity allocated by the Mobile Network Operator (MNO) thanks to a fast orchestration at periodicity T_{RA} and a fine-tuning of allocated resources closer to the actual service demand. We highlight the resource savings with respect to a blind allocation of the exact capacity requested by the SP, along with the remaining gain margin with respect to a perfect allocation matching the actual traffic. (b) Example of how overbooking improves the MNO NSaaS operation in a simple case with two slices. We show the real traffic and requested resources of the first slice (solid brown lines) and those of both slices (solid black lines). As per plot (a), the MNO can perform a faster and more accurate anticipatory allocation of resources to both slices (dotted line). This leads to resource savings (blue areas) with respect to allocating all the SP requests when those are below the capacity limit $C_r^{(l)}$. More importantly, it allows accepting both slices even if their aggregated requests exceed the MNO available capacity (green areas). Overbooking errors may however lead to SLA violations, when the actual demand of the accepted slices cannot be served (red area).

way to Core Network (CN) datacenters where compute and memory resources are reserved to run slice-tailored VNFs. Let *layer l* denote the layer whose nodes serve, on average, the aggregated traffic of l RUs; then for each node at layer l, we model the system as follows.

3.1.1. NSaaS Operation

The MNO serves a set of N Service Providers (SPs), which we denote by $\mathcal{S} \triangleq \{s_n\}_{n\in\mathcal{N}}$, $|\mathcal{S}| = N$, where we define $\mathcal{N} \triangleq \{1, \dots, N\}$ for any natural number N. The MNO monitors the demand generated by each SP within a short interval (*e.g.*, per minute in Fig. 3.1a). We denote the traffic generated by SP s at the monitoring interval k as $\ell_s[k]$ (see Fig. 3.1a). At any time, a service provider can request a network slice associated to an SLA with the following parameters.

- T_{slice}: time during which the slice must be active and the related SLA satisfied, *e.g.*, the whole span of Fig. 3.1a.
- T_{SLA}: duration of an *SLA block* of an SP request, *i.e.*, time interval during which a constant capacity is requested by the SP.
- $\overline{\ell}_s^{(\text{P})}(t)$: Requested capacity by an SP s for the t-th SLA block, *e.g.*, the ordinates of the 4 constant segments requested in Fig. 3.1a.
- $M_s\overline{\ell}_s^{(\text{P})}(t)$: Price that the SP s is offering to pay for the t-th SLA block. We model prices as linearly proportional to the capacity by a factor M_s (in \$/bps), but other definitions are possible.

The MNO decides whether to accept the slice requests,[3] and what resources to allocate to them if accepted. These decision are based on the request attributes above, as well as on the next parameters.

- T_{hor}: time horizon for the overall system optimization, *e.g.*, the multiple repetitions of the span of Fig. 3.1a.
- T_{dec}: time interval between admission decisions. The operation is batched, such that the MNO considers all requests arrived over the last T_{dec}, decides which slices are accepted, continued or dismissed, and estimates the resources that shall be dynamically reserved to each slice for its duration.
- T_{RA}: duration of one *Resource Allocation (RA) block*, *i.e.*, the interval during which the capacity allocated by the MNO to a slice remains constant, which is typically bounded by the technology available at the target network domain. We also define as $n_{\text{RA}} \triangleq \frac{T_{\text{SLA}}}{T_{\text{RA}}}$ the number of RA blocks per SLA block, *i.e.*, the amount of re-allocation opportunities for the MNO while the requested capacity $\overline{\ell}_s^{(\text{P})}$ stays fixed. As an example, in Fig. 3.1a we have $T_{\text{RA}} = 30$ and $T_{\text{SLA}} = 120$ minutes, yielding $n_{\text{RA}} = 4$.

[3]We refer to the slice that serves the traffic of SP s as slice s, $\forall s \in \mathcal{S}$.

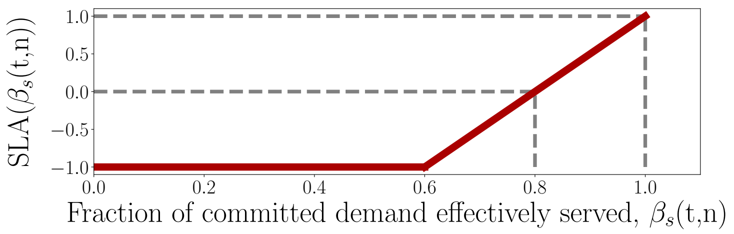


Figure 3.2: SLA function adopted for our experiments.

- $C_r^{(l)}$: capacity available at the target node r of layer l, which sets the boundary to the total traffic demand that can be served at any time instant, as shown in Fig. 3.1b.
- $\ell_s^{(\mathrm{O})}(t,n)$: Capacity actually reserved by the MNO for slice s.
- c_{OPEX}: MNO's OPEX (in \$/bps) of reserving one unit of capacity to a slice during a whole RA block, due, *e.g.*, to the energy or monetary cost of running dedicated VNF containers or CPU cores.
- ρ_{op}: operational profit ratio between the revenue from the SP and the OPEX of reserved resources, *i.e.*, $M_s/(n_{\mathrm{RA}}c_{\mathrm{OPEX}})$.[4]

Based on the above, we define a hierarchical time indexing. The index t refers to the SLA block index, and the notation (t,n) refers to the n-th RA block of the t-th SLA block; *e.g.*, in Fig. 3.1a, T_{RA} is represented for the $(3,2)$ RA block. Moreover, we consider $\ell_s(t,n) \triangleq \max_{k\in[T_{\mathrm{RA}}]} \ell_s[k]$ as the real demand in the (t,n) RA block. To avoid cluttering notation, we focus hereinafter on a given node r in layer l, and omit the dependence on r and l. Also, we let $T_{\mathrm{slice}} = T_{\mathrm{SLA}} = T_{\mathrm{dec}}$, *i.e.*, a slice has the same duration (T_{slice}) of an AC time slot of the MNO (T_{dec}), and the capacity requested by SP s is constant for the whole duration of the slice (T_{SLA}), typically in the order of hours. In this way, a certain scenario can be succinctly referred to as $\mathcal{E} = \{T_{\mathrm{hor}}, T_{\mathrm{SLA}}, T_{\mathrm{RA}}, C\}$.

3.1.2. SLA Function

The SLA defines the monetary compensation or penalty associated to an accepted slice. It is modeled as a function that depends on the requested capacity (for which the SP pays a fee as set out in Sec. 3.1.1) and the actual traffic served by the MNO. Specifically, at the n-th RA block of the t-th SLA block, the MNO commits to serve slice s with a capacity $\ell_s^{\mathrm{c}}(t,n) \triangleq \min(\ell_s(t,n),\ \overline{\ell}_s^{(\mathrm{P})}(t))$: if the slice traffic is below the level requested in the SLA, the operator only needs to serve such traffic and not the requested capacity in the SLA.
The actual served traffic is $\ell_s^{\mathrm{eff}}(t,n) = \min(\ell_s(t,n),\ \ell_s^{(\mathrm{O})}(t,n))$, *i.e.*, an over-allocation of resources does not bring any benefit to the MNO.

The full monetary compensation set out by the SLA as per Sec. 3.1.1 is paid by the SP s to the MNO when $\ell_s^{\mathrm{eff}}(t,n) \geq \ell_s^{\mathrm{c}}(t,n)$. If instead the served demand is below that the MNO committed

[4]$n_{\mathrm{RA}} = T_{\mathrm{SLA}}/T_{\mathrm{RA}}$ transforms the cost per RA block into cost per SLA block.

to accommodate, the SLA defines a reduction of the MNO's revenues. We model the compensation as a generic function $\mathrm{SLA}_s(\beta_s(t,n))$, where

$$\beta_s(t,n) \triangleq \min\left(1,\ \frac{\ell_s^{\mathrm{eff}}(t,n)}{\ell_s^{c}(t,n)}\right) \tag{3.1}$$

is the fraction of committed demand that is effectively served by the MNO, capped at 1 in the case where MNO unnecessarily serves more traffic than committed.

While our definition of SLA above is general, and our solution can accommodate other expressions, for the experiments carried out in this paper we leverage the SLA portrayed in Fig. 3.2. The rationale is that the agreed compensation drops linearly as the MNO fails to deliver the required capacity down to 80% of what it committed to. Below such a threshold, the MNO must pay a monetary fee (*i.e.*, a negative gain in the plot) to the SP, which grows up to the original compensation when only 60% of the slice traffic is served.

3.1.3. MNO Profit

The MNO's objective is to maximize the total net profit over the operating horizon T_{hor}. The profit is the result of the overall revenue obtained from the slice brokering minus the costs incurred by the operator, *i.e.*, SLA violations that may induce a penalty as discussed in Sec. 3.1.2 and operating expenses derived from allocating the network resources. Formally:

- **Revenues** correspond to the compensations from meeting SLAs with SPs, *i.e.*, $M_s\bar{\ell}_s^{(\mathrm{P})}(t)$ for t-th SLA block of duration T_{SLA}, possibly decreased according to $\mathrm{SLA}_s(\beta_s(t,n))$.

- **SLA violation costs** are incurred when accepted slices are poorly served. This cost is embedded in the SLA definition when it takes values < 0, hence a single expression $M_s\bar{\ell}_s^{(\mathrm{P})}(t)\cdot \mathrm{SLA}_s(\beta_s(t,n))$ captures both revenues and SLA violation costs.

- **OPEX costs** are proportional to the capacity $\ell_s^{(\mathrm{O})}(t,n)$ actually reserved by the MNO, by the c_{OPEX} factor.

- **Profits**, denoted by $p_{\mathcal{E}}$, combine the previous as follows

$$p_{\mathcal{E}} = \sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} x_s(t)\sum_{n=1}^{n_{\mathrm{RA}}}\left(\frac{M_s\bar{\ell}_s^{(\mathrm{P})}(t)}{n_{\mathrm{RA}}}\,\mathrm{SLA}_s(\beta_s(t,n)) - c_{\mathrm{OPEX}}\ell_s^{(\mathrm{O})}(t,n)\right) \tag{3.2}$$

In (3.2), the binary variable $x_s(t)\in\{0,1\}$ is set to 1 if slice s is accepted in SLA block t. In case the slice is admitted, the profit is the difference between the revenue (or SLA violation cost) and the OPEX cost across all n_{RA} RA blocks composing the SLA block. The total profit is then computed over all slice requests $\mathcal{S}$ and through the whole temporal set of SLA blocks $\mathcal{T} \triangleq \{t \mid t\in[T_{\mathrm{hor}}/T_{\mathrm{SLA}}]\}$.

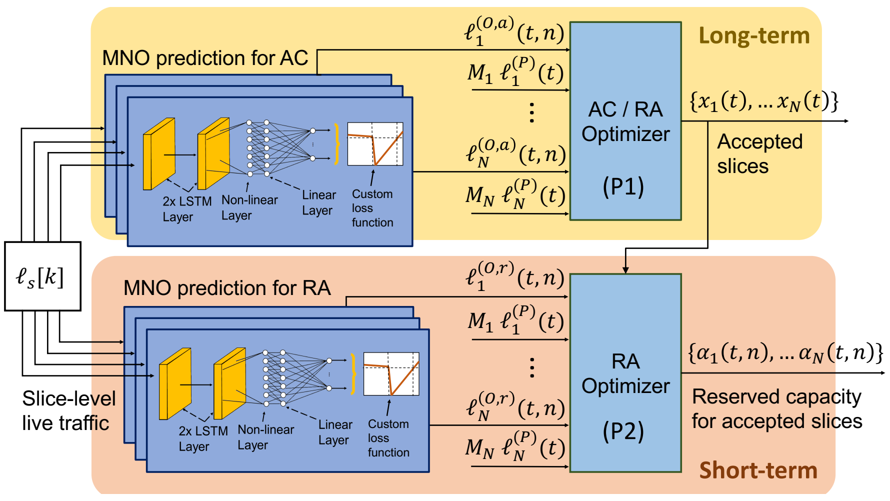


Figure 3.3: Overall architecture of kaNSaaS, with long-term and short-term prediction-enabled AC and RA components.

3.2. The kaNSaaS Framework

In order to maximize the MNO profit in (3.2), we propose a novel solution for overboo<u>k</u>ing-<u>a</u>ware <u>NSaaS</u>, or kaNSaaS. Our approach addresses the joint problem of (i) *Admission Control (AC)*, *i.e.*, deciding which slice requests to accept, and (ii) *Resource Allocation (RA)*, *i.e.*, determining how many resources to allocate to each of the accepted slices. It is important to note that both parts of the problem are inherently *anticipatory* in nature, but operate *at different timescales*. In AC, the MNO must admit slices at the start of each SLA block so as to ensure that their demand is accommodated during the future T_{SLA} time interval. In RA, the MNO has to reserve resources at the beginning of each RA block in a way to best serve the traffic through the following T_{RA} interval.

The overall architecture of kaNSaaS is illustrated in Fig. 3.3. Our design abides by the observations above, and hinges upon tailored forecasting models that support decision-making at two different timescales, *i.e.*, long-term SLA blocks and short-term RA blocks, as highlighted in the figure and detailed next.

- First, at every T_{SLA} time slots, the MNO computes slice demand predictions that inform a tentative resource allocation for the next T_{SLA} time interval. In turn, this provisional allocation is used to guide AC and decide which slices are accepted. Formally, the long-timescale component sets the binary variables $x_s(t) \in \{0, 1\}$ that indicate whether a service s is accepted at SLA block t.

- Then, at each T_{RA} time slot n within the SLA block t, the operator performs the actual allocation of resources for the admitted slices for the next RA block, leveraging a more accurate forecast that is limited to the shorter T_{RA} interval. Formally, the short-timescale component defines $\alpha_s(t, n)$, *i.e.*, the percentage of the (predicted) traffic capacity that is to be reserved by the MNO to slice s.

Clearly, AC actions at the longer timescale constrain and drive RA decisions at the shorter timescale, as shown by the arrow connecting the long- and short-term operations in Fig. 3.3.

The design above lets kaNSaaS take advantage of traffic forecasting to spot gaps between the capacity $\bar{\ell}_s^{(\mathrm{P})}(t)$ requested by the SPs at each SLA block t and the future demand generated by the slices at every RA block (t, n). Then, it uses suitable optimizers to maximize the MNO profit in (3.2) from the identified gaps, via overbooking.

From a technical viewpoint, kaNSaaS combines apt machine learning and optimization tools to implement each component. Specifically, we adopt data-driven approaches to implement the prediction components at both short and long timescales, since DL models have been largely proven to outperform statistical models in time series forecasting tasks [80]. The predicted demands are then fed to dedicated optimizers that take rapid, effective decisions on AC and RA based on explainable logic.

Next, we detail the structure and operation of the two components, focusing on the long-term first and on the short-term after.

3.2.1. Long-term Admission Control

The AC operation takes place at the beginning of each SLA block, and aims at selecting slices so as to maximize the MNO profit, through overbooking based on long-timescale forecasts. The traffic prediction and decision-making parts are implemented as follows.

Traffic prediction. The MNO forecasts the expected traffic volume for each one of the RA blocks belonging to the next SLA block. Consequently, this Long-Term Predictor (LTP) outputs $T_{\mathrm{SLA}}/T_{\mathrm{RA}} = n_{\mathrm{RA}}$ traffic values for the next T_{SLA} time slots. Our implementation uses a separate LTP for each requested slice s, which (i) best adapts to the diverse temporal dynamics of the heterogeneous mobile services associated with each slice, and (ii) allows for a modular design where predictors for SPs entering or leaving the slice brokering process can be dynamically added or removed.

This long-term prediction, denoted as $\bar{\ell}^{(\mathrm{O,a})}(t, n)$, makes use of the state-of-the-art Thresholded Exponential Smoothing (TES)-RNN model for traffic forecasting [81]. This model combines statistical modeling and machine-learning tools, via a Recurrent Neural Networks (RNN) whose inputs are first passed through a TES [82] block. The key aspect is that the TES coefficients are simultaneously optimized with the RNN weights through a unified gradient descent [81]. The original TES-RNN is limited to output a single-value forecast, which is not suitable for our problem. We thus extend TES-RNN to support a multidimensional output (*i.e.*, a set of n_{RA} values), as well as to handle different time scales of input (*i.e.*, the monitoring samples $\ell_s[k]$) and output (*i.e.*, the RA block interval). Our TES-RNN implementation takes as input traffic samples for the last 8 hours for each service, and hinges on a deep neural network architecture with 2 LSTM hidden layers with dilations (1,3) and (6,12), both having a state size of 50, followed by a nonlinear layer with 50$\times$50 state size and a linear adapter to the output size.

We train our enhanced TES-RNN model offline, with the α-OMC loss parametrized with $\gamma = 0.75$ [38]. This asymmetric loss avoids underestimations that may lead to SLA violations, while trying to minimize overprovisioning that increases Operating Expenses (OPEX) costs.

Admission control. The predicted values $\bar{\ell}^{(\mathrm{O,a})}(t,n)$ are fed together with the compensation for each slice s to an admission control optimizer. Let $\mathcal{V}_\alpha \triangleq \{\alpha_s(t,n) | n \in \{1,\ldots,n_{\mathrm{RA}}\}, s \in \mathcal{S},\, t \in \mathcal{T}\}$ and $\mathcal{V}_x \triangleq \{x_s(t) | s \in \mathcal{S},\, t \in \mathcal{T}\}$. The optimization problem is

$$\max_{\mathcal{V}_\alpha,\mathcal{V}_x} \sum_{t\in\mathcal{T}} \sum_{s\in\mathcal{S}} x_s(t) \sum_{n=1}^{n_{\mathrm{RA}}} \left(\frac{M_s \bar{\ell}_s^{(\mathrm{P})}(t)}{n_{\mathrm{RA}}} \, \mathrm{SLA}_s(\alpha_s(t,n)) - c_{\mathrm{OPEX}} \alpha_s(t,n) \ell^{(\mathrm{O,a})}(t,n) \right) \quad \text{(P1)}$$

$$\text{s.t.} \sum_{s\in\mathcal{S}} x_s(t) \alpha_s(t,n) \bar{\ell}_s^{(\mathrm{O,a})}(t,n) \leq C \quad \forall n, t \quad (3.3)$$

$$0 \leq \alpha_s(t,n) \leq 1 \qquad \forall \alpha_s(t,n) \in \mathcal{V}_\alpha \quad (3.4)$$

$$x_s(t) \in \{0,1\} \qquad \forall x_s(t) \in \mathcal{V}_x. \quad (3.5)$$

Problem (P1) maximizes the operational profit of the MNO defined in (3.2), *i.e.*, revenue minus costs of SLA violations and OPEX, subject to the available network capacity C and on the basis of the predicted demands $\bar{\ell}_s^{(\mathrm{O,a})}(t,n)$ of each slice s through the next SLA block. It does so by identifying the admitted slices for which $x_s(t) = 1$, via a tentative allocation of resources $\alpha_s(t,n)$ in each RA block n of the future SLA block t. We remark that the MNO does not directly apply the resource allocation $\alpha_s(t,n)$ obtained from (P1); instead, it triggers the short-term stage to fine tune the resource allocation, as described in Sec. 3.2.2 hereafter. Problem (P1) is NP-hard and can be modeled as a knapsack problem, as we formally prove below. However, the number of variables (*i.e.*, slices) that would be handled in NSaaS do not grow exponentially, and efficient solvers exist for knapsack problems [83].

The proof of Problem P1 follows by reduction from the Knapsack Problem (KP) [84], which can be written as follows: considering a knapsack with capacity W and a set of I items to store where each item i has positive reward r_i and positive weight w_i, $i \in \{1,\ldots,I\}$. The objective expression is $\max \sum_{i=1}^{I} r_i x_i$, where x_i is a binary variable indicating whether item i is allocated to the knapsack, and where the objective expression has to be maximized under the constraint that $\sum_{i=1}^{I} w_i x_i \leq W$. Let us consider a particular case of our problem, in which $n_{\mathrm{RA}} = 1$ and the SLA function is given by $\mathrm{SLA}_s(\alpha_s(t,n)) = 1$ if $\alpha_s(t,n) \geq 1$ and $\mathrm{SLA}_s(\alpha_s(t,n)) = -1$ if $\alpha_s(t) < 1$, i.e., any underprovisioning incurs a penalty equivalent to the possible revenue. In this particular case, we have that, for any solution in which some slice s is accepted ($x_s(t) = 1$) with α_s different than 1 for that same slice, we can find another solution improving the objective value just by setting $x_s = 0$. Consequently, our problem is equivalent to maximize $\sum_{s\in\mathcal{S},\, t\in\mathcal{T}, n\in n_{\mathrm{RA}}} p_s(t) x_s(t)$ over the set $\{x_s(t)\}_{s\in\mathcal{S}}$ subject to $x_s(t) \in \{0,1\}$ and $\sum_{s\in\mathcal{S}} \bar{\ell}_s^{(\mathrm{O,a})}(t,n) x_s(t) \leq C_r^{(l_q)}$ for each $t \in \mathcal{T}$ and $n \in n_{\mathrm{RA}}$.

Hence, our problem is equivalent to a KP with weights $\ell_s^{(\mathrm{O,a})}(t,n)$, rewards $p_s(t)$, and capacity C_{tot}. This means that the KP is a particular case of our problem, and thus our problem is at least as complex as the KP, which is NP-hard. Since this reduction can be built in polynomial time, it follows that our problem is NP-hard.

3.2.2. Short-term Resource Allocation

Once the admission control for the next n_{RA} RA blocks is decided in (P1), the MNO performs the actual resource reservation at the start of each RA block. The rationale is that this requires a forecast over a shorter future horizon, which is inherently more accurate and allows higher savings on OPEX for the MNO. The implementation of such short-term operation, depicted in the bottom half of Figure 3.3, has a structure similar to the long-term decision component.

Traffic prediction. As anticipated, the MNO only forecasts the expected traffic volume for the next RA block. To this end, we use one Short-Term Predictor (STP) for each admitted slice s. The STP is based on the exact same hybrid model as the LTP, although the STP only outputs a single value. The predicted values $\bar{\ell}^{(\mathrm{O,r})}(t,n)$ serve as input for the RA shot-term optimization presented next.

Resource allocation. At every T_{RA}, *i.e.*, n_{RA} times per SLA block, kaNSaaS solves an optimization problem to determine the exact resources $\alpha_s(t,n)$ reserved for each slice for the following RA block. Formally, for a given RA block (t,n), the optimization is

$$\max_{\mathcal{S}^{(a)}} \sum_{s\in\mathcal{S}^{(a)}} \Big(\tfrac{M_s \bar{\ell}_s^{(\mathrm{P})}(t)}{n_{\mathrm{RA}}} \, \mathrm{SLA}_s(\alpha_s(t,n)) - c_{\mathrm{OPEX}} \alpha_s(t,n) \bar{\ell}_s^{(\mathrm{O,r})}(t,n) \Big) \tag{P2}$$

$$\text{s.t. } \textstyle\sum_{s\in\mathcal{S}^{(a)}} \alpha_s(t,n) \bar{\ell}_s^{(\mathrm{O,r})}(t,n) \le C \tag{3.6}$$

$$0 \le \alpha_s(t,n) \le 1 \qquad \forall s \in \mathcal{S}^{(a)}. \tag{3.7}$$

Problem (P2) aims at maximizing the contribute of the current RA block to the profit of the MNO, hence has a similar expression to that of (P1). The main differences are that (i) there is no admission control decision, (ii) it only considers the services in the set accepted in the last (P1) problem, denoted as $\mathcal{S}^{(a)} \subseteq \mathcal{S}$, and (iii) it leverages the STP forecast $\bar{\ell}_s^{(\mathrm{O,r})}(t,n)$. This problem is considerably simpler than the MIP in (P1), as it does not contains discrete variables, and its complexity depends solely on the expression of the SLA function. Thus, (P2) is a linear programming (LP) problem with the SLA in Sec. 3.1.2, yet it could turn non-linear under a different SLA function.

3.2.3. MNO Profit from AC/RA Decisions

The AC/RA decisions taken by the components described in Sec. 3.2.1 and Sec. 3.2.2 determine the MNO profit, as follows. The values $\alpha_s(t,n)$ denote the percentages of the *predicted* future traffic that is reserved for slice s; thus, the actual served traffic at RA block (n,t) is:

$\ell_s^{\mathrm{eff}}(t,n) = \min(\ell_s(t,n),\ \alpha_s(t,n)\overline{\ell}_s^{(\mathrm{O,r})}(t,n))$. From this expression, we obtain the value of $\beta_s(t,n)$ in (3.1), *i.e.*, the percentage of agreed traffic volume that is effectively served by the MNO. The final profit uses the $\beta_s(t,n)$ above and is given by

$$p_{\mathcal{E}} = \sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} x_s(t) \sum_{n=1}^{n_{\mathrm{RA}}} \Big(\frac{M_s \overline{\ell}_s^{(\mathrm{P})}(t)}{n_{\mathrm{RA}}} \,\mathrm{SLA}_s(\beta_s(t,n)) - c_{\mathrm{OPEX}}\alpha_s(t,n)\ell_s^{(\mathrm{O,r})}(t,n)\Big). \tag{3.8}$$

Discrepancies between (3.8) and the objective functions of (P1) and (P2) are possible due to slice traffic prediction errors, and are part of the complexity of an overbooking-based NSaaS.

3.3. Performance Evaluation

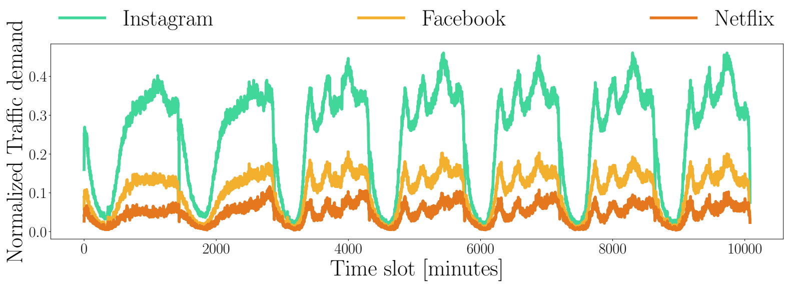
We evaluate kaNSaaS using both real and synthetic datasets of slice demands, presented in Sec. 3.3.1, and we compare its performance to benchmark NSaaS management approaches, described in Sec. 3.3.2. The results of our evaluation are presented in Sec. 3.3.3.

3.3.1. Datasets

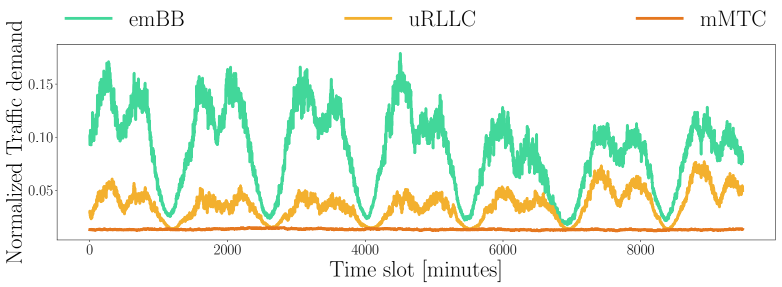
A contribution of our study is a first assessment of the potential gain of overbooking in NSaaS in presence of real-world demands generated by a variety of service providers, based on measurements of the traffic of individual mobile applications in a metropolitan-scale network of a major European MNO. As this dataset is protected by a Non-Disclosure Agreement (NDA) with the MNO, we also generate a synthetic dataset that mimics the main properties of the real-world service-level demands, and leads to comparable results in our evaluations. We disclose this second dataset to allow reproducibility of our results and foster further research in NSaaS.

Measurement dataset. The real-world mobile demand dataset captures all traffic generated by several millions of users in a large metropolitan area during 8 consecutive weeks. The data consist of the traffic loads served by each of the hundreds of base station covering the target geographical region, at a time granularity of 1 minute. The traffic is reported separately for 20 different mobile services, which include the most popular smartphone applications, such as YouTube, Instagram, Twitter or various Google services. Such data was collected and aggregated by the MNO in its production infrastructure, using passive measurement probes deployed in the Evolved Packet Core (EPC). The probes run commercial and proprietary traffic classifiers to identify the service associated to each IP flow. The measurement dataset allows defining realistic SP demands, by assigning one slice to each service. As shown in Fig. 3.4a-b, the resulting slices have heterogeneous traffic volumes and time dynamics, which opens opportunities for multiplexing.

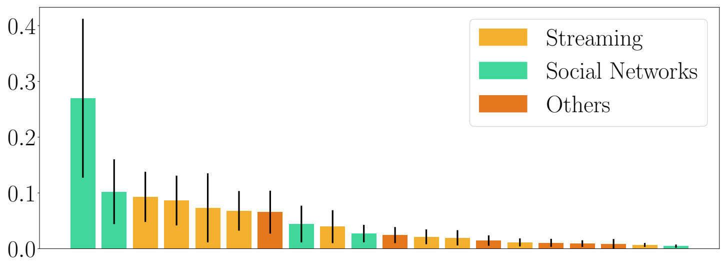
Synthetic dataset. We emulate demands for the three main network slicing categories for 5G, namely enhanced Mobile BroadBand (eMBB), ultra-Reliable Low-Latency Communication (uRLLC) and massive Machine-Type Communication (mMTC). For the eMBB and uRLLC classes, we created a weekly pattern of 5 work days and 2 weekend days, with each day containing



(a) Sample measurement slice traffic



(b) Sample synthetic slice traffic



(c) Ranked SP traffic

Figure 3.4: (a)-(b) Temporal demands of 3 slices in the measurement and synthetic datasets; (c) Sorted normalized traffic per SP.

a sinusoidal dynamic that mimics the well-known circadian rhythm of mobile traffic. Based on our measurement data, we also model two daily traffic peaks, higher in the morning and lower in the afternoon. Instead, mMTC slices are characterized by a steady demand over time, to reflect to deterministic behavior of many applications in that class.

Individual slices are told apart by their generated traffic volumes and weekly patterns. We use peak traffic values of 60 Mbps for eMBB, 7 Mbps for mMTC, and 35 Mbps for uRLLC, and introduce diversity across slices of the same category, by scaling all values of a slice by a random factor between 0.7 and 1.4. Also, we reflect the temporal variability observed in the measurement data by randomly shifting each slice demand by up to 1.5 hours. We then imitate the inherent randomness of mobile device behaviors, by adding a coloured noise with a standard deviation equal to 35% of the mean throughput. This power-law noise has a power spectral density per unit of bandwidth proportional to $\frac{1}{f^{\phi}}$ [85], where $\phi = 0$ implies white noise, $\phi = 1$ represents pink noise, and Brownian noise corresponds to $\phi = 2$. We select $\phi = 1.08$ to strike a balance between trend and noise. Ultimately, our synthetic dataset consists of $90,720$ data points at 1 minute granularity for each emulated slice, which is consistent with the real-world data, as exemplified in Fig. 3.4b.

Slice requests. We generate the SP requests from the datasets of service demands above, by considering that each SP forecasts the expected future traffic generated by its service, and asks for sufficient slice resources to process it. For fairness, we consider that the SPs make use of the state-of-the-art model from [81], which is suitably configured to predict a single value corresponding to the maximum traffic demand for the next SLA block of duration T_{SLA}.

We also assume that SPs rely on more conservative predictions than the MNO, since they are committed to ensure proper quality of experience to their users. To this end, we parameterize the α-OMC loss with a higher $\gamma = 1.5$ parameter that induces a higher safety margin against underprovisioning in SP forecasts [38].

3.3.2. Benchmarks

We consider two baselines that allow contextualizing the performance of kaNSaaS, and one state-of-the-art benchmark, as follows.

Legacy NSaaS. This is a traditional network slicing management strategy where the MNO allocates exactly what the service provider requests, and does not perform any slice overbooking. This implies that $\bar{\ell}_s^{(\mathrm{O})}(t,n) = \bar{\ell}_s^{(\mathrm{P})}(t)$. We denote by $\mathcal{E}^{(\mathrm{P})}$ the Legacy NSaaS counterpart of a given overbooking scenario $\mathcal{E}$ (as defined at the end of Sec. 3.1.1), and compute the net profit gain of overbooking as

$$\bar{G}_p(\mathcal{E}) \triangleq \frac{p_{\mathcal{E}} - p_{\mathcal{E}^{(\mathrm{P})}}}{p_{\mathcal{E}^{(\mathrm{P})}}}. \tag{3.9}$$

Oracle NSaaS. This is an unfeasible but optimal slice AC/RA management where $\bar{\ell}_s^{(\mathrm{O})}(t,n) = \ell_s(t,n)$ and $T_{\text{RA}} = 1$. We denote by $\mathcal{E}^{\star}$ the ORACLE counterpart of a given scenario $\mathcal{E}$, and calculate the similarity of a practical solution with the optimal as the ratio

$$\bar{D}_p(\mathcal{E}) \triangleq \frac{p_{\mathcal{E}}}{p_{\mathcal{E}^{\star}}}. \tag{3.10}$$

CoNEXT. This is the state-of-the-art solution for NSaaS overbooking, originally introduced by Salvat*et al.* [48] and denoted by CoNEXT in the following. As partially anticipated in Chapter. 2, CoNEXT relies on SP demand forecasts returned by the multiplicative version of the three-smoothing Holt-winters (HW) algorithm with seasonality. Admission decisions of slices are taken by solving a stochastic yield management optimization problem, which is however based only on a short-term forecast over the future T_{RA} interval. As slice AC is enforced through a longer interval of T_{SLA} time slots, CoNEXT then updates the forecast and compute the resource re-allocation at every subsequent RA block of duration T_{RA}. In other words, CoNEXT lacks the two-timescale operation of kaNSaaS, which forces it to take long-term SLA AC decisions based on short-term demand forecasts. In addition to such a fundamental design gap, the solution differs from kaNSaaS in the implementation of both the prediction and decision modules.

3.3.3. Evaluation

We assess the performance of kaNSaaS, Legacy NSaaS and CoNEXT in presence of both measurement and synthetic demands. All results are expressed in terms of the profit similarity from the performance of the ORACLE approach $\bar{D}_p$, as defined in (3.10). Unless stated otherwise, we use the following default settings throughout all experiments: $T_{\text{RA}} = 30$ min, $T_{\text{SLA}} = 120$ min, $c_{\text{OPEX}} = 0.9\frac{M_s}{n_{\text{RA}}}$ (*i.e.*, a profit margin $\rho_{op} = 11\%$), and the network capacity is set to be equal to the maximum aggregated traffic over services over the whole dataset, or $C_r^{(l)} = \max_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}} \ell_s(t)$.

Overall overbooking gain with kaNSaaS. The main results of our performance evaluation are summarized in Fig. 3.5. The figure reports the profit of each tested solution (as different curves), under real-world (left column) and synthetic (right column) SP demand datasets. As anticipated, all values are indicated as similarity to the optimum ORACLE performance, so as to favor interpretability.

The vertical greyed region in each plot highlights the default settings, for which the following key observations are in order.

- The gain of kaNSaaS with respect to Legacy NSaaS is very large at around 300%. This implies that *overbooking can grow fourfold the economic profit for the MNO* with respect to a case where the operator just abides by the requests of the SPs in a typical case.

- The gain of kaNSaaS with respect to CoNEXT under the default settings is of 20% with real demands, and of 40% with synthetic traffic. In other words, thanks to a more complete two-timescale design and more effective implementations of forecasting and AC/RA decision models, *kaNSaaS significantly grows the net profit for the MNO over the best available solution for NSaaS overbooking.*

- The similarity to the performance of the ORACLE is at 0.5 denoting that the default settings scenario does not allow kaNSaaS to take full advantage of the potential of NSaaS overbooking.

These considerations shed light on the actual monetary advantage that overbooking can bring to an MNO in a dependable slice demand scenario, and unveil in particular how such an advantage can be surprisingly large. We next investigate how the system settings affect the overbooking performance.

Impact of profit margin. A chief parameter of interest for the MNO is the profit margin at which it can operate the service. This element directly determines the feasibility and interest of NSaaS use cases. We analyze the impact of the profit margin by varying the ratio $\rho_{op} = \frac{M_s}{c_{\text{OPEX}} n_{\text{RA}}}$. In particular, we vary the coefficient c_{OPEX} from 0 (*i.e.*, neglecting OPEX) to $0.99\frac{M_s}{n_{\text{RA}}}$ (beyond

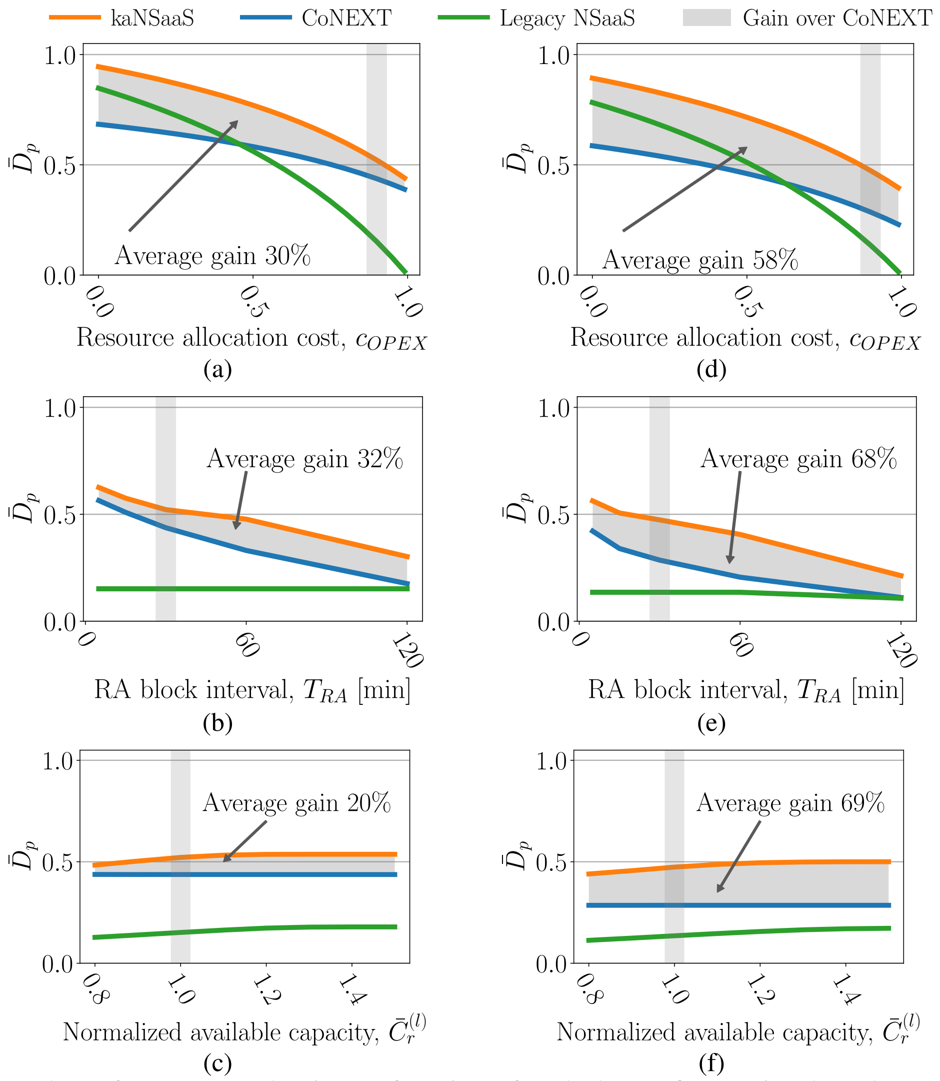


Figure 3.5: Performance evaluation as function of (a,d) the profit margin, (b,e) the orchestration interval T_{RA}, and (c,f) network capacity. Results refer to (left) real-world demands and (right) synthetic traffic. We compare kaNSaaS, CoNEXT [48], and Legacy NSaaS. Performance is expressed as a similarity $\bar{\boldsymbol{D}}_{\boldsymbol{p}}$ with the unfeasible ORACLE. Gray vertical lines represent the default settings listed in Sec. 3.3.3.

which NSaaS is not profitable for the MNO, as $\rho_{op} \to 1$). Fig. 3.5a shows how varying c_{OPEX} (as a coefficient of $\frac{M_s}{n_{\text{RA}}}$) impacts the profit in real-world demands, and Fig. 3.5d shows the same for synthetic traffic. The two plots highlight the very significant role of resource operation costs in controlling the gain of overbooking strategies.

- On the one hand, the gain of kaNSaaS over Legacy NSaaS is maximum at profit margins around 10-20%, *i.e.*, for c_{OPEX} in the $[0.8, 0.9]$ range ($\times\frac{M_s}{n_{\text{RA}}}$). Here, overbooking grants dramatic manyfold boosts in the economic profit.

- On the other hand, the performance of kaNSaaS tends to get closer to the optimal ORACLE as c_{OPEX} decreases, with a similarity $\bar{D}_p$ closer to 1 as c_{OPEX} tends to 0. Indeed, as we will later show in Sec. 3.3.3, all the cost induced by our solution is imputable to overdimensioning, and reducing c_{OPEX} shrinks that cost.

- The state-of-the-art CoNEXT solution performs well under low profit margins, yet the quality of its AC/RA decisions tends to rapidly deteriorate as c_{OPEX} is reduced, up to the point where the profits it grants become lower than those of a no-overbooking Legacy NSaaS strategy. The reason is that, as shown in Sec. 3.3.3, CoNEXT aggressively overbooks resources, serving more traffic than all other approaches but also incurring in many SLA violations. The cost of such violations dominates in absence of significant OPEX costs, thus penalizing this solution. Our proposed kaNSaaS does not suffer from this problem, and stays a better choice than Legacy NSaaS and CoNEXT across all c_{OPEX}.

Impact of resource orchestration interval. We vary T_{RA} from 5 to 120 minutes, which is the same as the duration of the T_{SLA} requested by each SP and thus an upper bound to the RA block duration. Fig. 3.5b and Fig. 3.5e show how kaNSaaS increases its profit gain over CoNEXT as T_{RA} increases, *i.e.*, as the resource re-allocation decisions are spaced apart: the gain grows from 20% at $T_{\text{RA}} = 30$ minutes to 72% at $T_{\text{RA}} = 120$ minutes. Also, the profit gain of CoNEXT over Legacy NSaaS is only of 16% at $T_{\text{RA}} = 120$ minutes, while kaNSaaS still doubles the profit over the baseline NSaaS without overbooking. These gains demonstrate the importance of DL-based forecasting to predict traffic over longer time horizons.

Impact of network capacity. Let us denote the maximum sum traffic over the whole dataset by $L \triangleq \max_{t\in\mathcal{T}} \sum_{s\in\mathcal{S}} \ell_s(t)$: we then define the normalized available capacity as $\bar{C}_r^{(l)} = C_r^{(l)}/L$, and vary $\bar{C}_r^{(l)} \in [0.8, 1.5]$. The results are in Fig. 3.5c and Fig. 3.5f. For kaNSaaS, the performance with respect to the ORACLE approach is not strongly affected by the network capacity, as it is similar (at 48% and 54%) in the extremes case where the network is underdimensioned ($\bar{C}_r^{(l)} = 0.8$) or overdimensioned ($\bar{C}_r^{(l)} = 1.5$). The performance of CoNEXT is similarly not affected by the capacity of the network, with a steadily lower 43% performance with respect to ORACLE. The result of Legacy NSaaS is very far from ORACLE, with low similarity that varies from 17.4% for $\bar{C}_r^{(l)} = 0.8$ to 22% at $\bar{C}_r^{(l)} = 1.5$. Ultimately, these results prove how the overbooking gains are only marginally affected by the available capacity.

Profit deconstruction. In order to better understand how kaNSaaS outperforms the benchmarks, we break apart the economic gain and costs incurred by each NSaaS scheme, so as to reveal how the final net revenue is obtained. Fig. 3.6 shows the total revenues from accepted slices, the OPEX from resource allocation, the fees due to SLA violations, and the profit resulting from subtracting the latter two costs from the initial revenues. We first observe how Legacy NSaaS does not incur in any cost in terms of SLA violations, owing to a conservative policy of abiding by the requests of the SPs, which are in turn designed to avoid any service disruption. However, this result comes at a very high OPEX cost, since SP requests tend to be overdimensioned. Also, Legacy NSaaS yields the lowest total revenues, as its conservative approach leads to accepting a lower number of slices. On the other end of the spectrum, CoNEXT is the most aggressive approach,

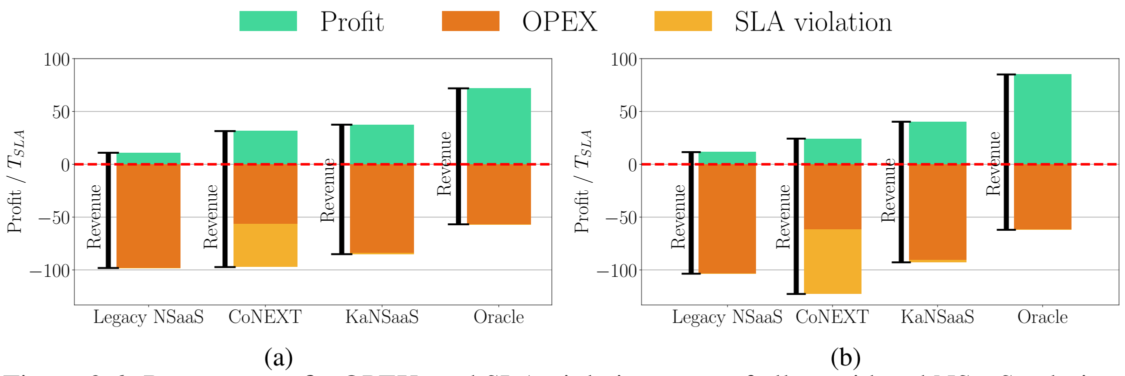


Figure 3.6: Revenue, profit, OPEX, and SLA violation costs of all considered NSaaS solutions under (a) measurement and (b) synthetic demands, and with default system settings.

which accepts the highest number of slices and thus attained the highest total revenues. To do so, this NSaaS solution heavily employs overbooking, but it also pays a substantial penalty from SLA violations. Ultimately, SLA violation costs curb the MNO net profit. kaNSaaS achieves a better trade-off between slice overbooking (accepting more slices than Legacy NSaaS, thus increasing the total revenue) and SLA violation avoidance (paying a negligible cost for those, especially when compared with CoNEXT). By striking this balance, and even though the total revenues are lower and the OPEX is higher than those of CoNEXT, kaNSaaS creates a significantly higher total profit for the MNO.

We also observe that our solution is the one that resembles the most the optimal ORACLE approach: indeed, the two solutions accepts roughly the same SP requests, which results in a very similar total revenue. The difference is then ascribed to the fact that ORACLE relies on a perfect prediction over instantaneous RA block intervals, which is not feasible in practice.

3.4. Exhaustive System Analysis

In this section, we carry out a complete analysis of the NSaaS overbooking performance across the additional system dimensions that we did not explore in Sec. 3.3. Specifically, we analyze how sensitive the performance is to the network layer at which the NSaaS management is performed, as well as how different levels of accuracy in the MNO traffic prediction affect the final net profit. In order to control the second aspect, we replace the actual LTPs and STPs models with a parametrizable overprovisioning factor a that multiplies the actual peak throughput in the following RA block. In other words, we employ an abstract predictor that achieves a tunable accuracy instead of practical forecasting solutions. Specifically, the default values for overprovisioning are 20% for SPs and 5% for the MNO, such that we have that $\overline{\ell}_s^{(\mathrm{P})}(t) = 1.2\max_{n\in[n_{\mathrm{RA}}]}\ell_s(t,n)$ and $\overline{\ell}_s^{(\mathrm{O,a})}(t,n) = 1.05\ell_s(t,n)$. In the following, we only report results obtained with the real-world measurement demands, since those returned with synthetic data did not show significant differences. Also, all results are presented in terms of the gain of kaNSaaS over Legacy NSaaS, as per (3.9).

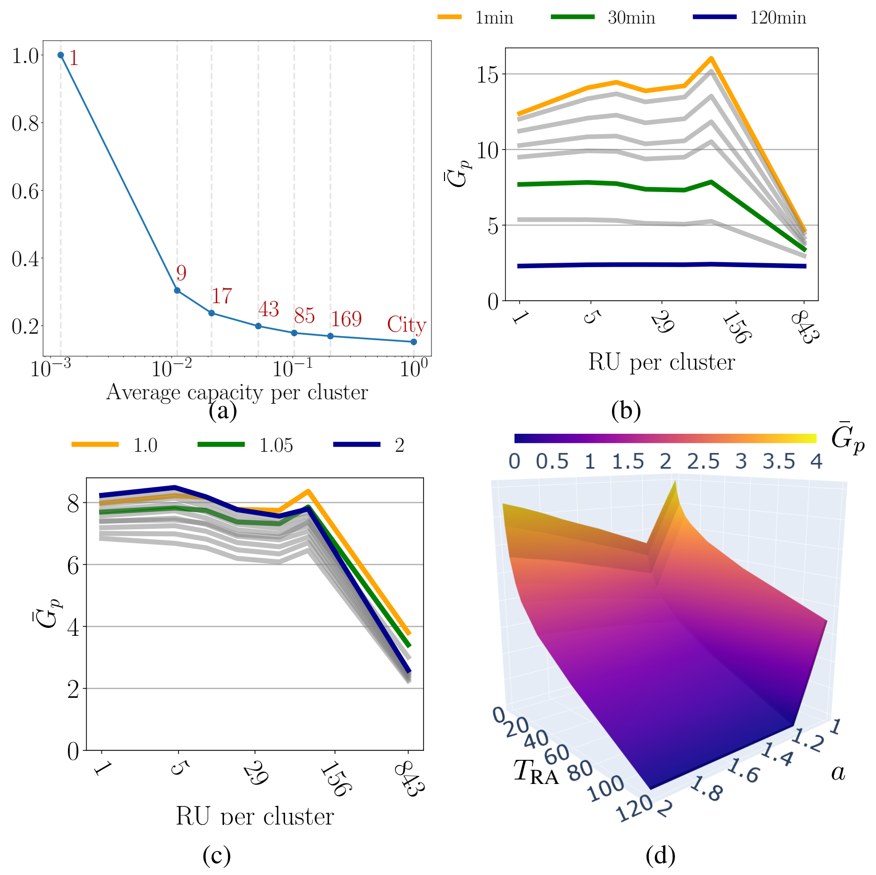


Figure 3.7: (a) Total capacity required to serve all demands at different network layers. (b,c,d) System performance as function of: (b) the network layer where NSaaS occurs, for different $\boldsymbol{T}_{\text{RA}}$ from 1 to 120 minutes; (c) the network layer where where NSaaS occurs, for different prediction accuracy $\boldsymbol{a}$ from 1 to 2; and, (d) the RA block interval $\boldsymbol{T}_{\text{RA}}$ and prediction accuracy a, at CN layer.

3.4.1. Network Layer

The NSaaS management problem tackled by kaNSaaS can be instantiated at different locations of the network infrastructure. For instance, slicing could occur for spectrum at the level of individual Remote Units (RUs), for radio scheduling at the level of Distributed Units (DUs), for compute resources at the level of Centralized Unit (CU), or for transport or Cloud resources in the Core Network (CN). We model such different network layers as nodes that aggregate an increasing volume of traffic, or equivalently serve a growing number of clustered RUs, as we move from the radio access to the network core. Fig. 3.7a shows the total network capacity required to serve the whole demand in our measurement dataset at different network layers. The x-axis represents the average traffic demand per minute per cluster, normalized, in a logarithmic scale. As we move towards the RU layer (left-most value), the required capacity grows exponentially. Hence, although handling NSaaS at RUs provides the MNO with higher gains and flexibility, the required management and resources escalate at an unaffordable rate. We represent the gain over Legacy NSaaS in Fig. 3.7b and Fig. 3.7c, where the x-axis is the number of RU per cluster, and where $x = 843$ represents the CN layer. Fig. 3.7b shows the different performance when the orchestration interval T_{RA} varies from 1 to 120 min, whereas Fig. 3.7c shows the variation with respect to the level of overprovisioning, modeled by a coefficient a, such that $\overline{\ell}_s^{(\mathrm{O})}(t,n) = a \cdot \ell_s(t,n)$, and where a varies from 1 to 2. We only highlight the extreme and the default values for the sake of clarity.

The profit gain $\bar{G}_r$ exceeds 600%, *i.e.*, the operator can multiply its profit by seven times with overbooking. Importantly, the profit gain increases as we approach the RU layer, where traffic is more dynamic and there are more opportunities to multiplex demands.

3.4.2. Traffic Prediction Accuracy

Finally, we jointly analyze the impact of jointly varying T_{RA} (30 min by default) and the prediction accuracy level ($a = 1.05$ by default). The gain over Legacy NSaaS ($\bar{G}_p$) is show in the 3D plot of Fig. 3.7d.

We observe a non-monotonic behavior in the accuracy axis. This is due to the fact that, for $a \leq 1.2$, a increases for the MNO but not for the SPs, which keeps the default value $a = 1.2$. Hence, reducing a improves the performance because we are assuming perfect predictions with no uncertainty. For $a > 1.2$, however, both MNO and SP share the same a (because SP will never be more aggressive provisioning than the MNO); then, increasing a is beneficial for the MNO due to the fastest resource allocation decision. We also observe that the peak gain is achieved at $T_{\mathrm{RA}} = a = 1$, because we do not have prediction errors (and thus SLA violations) in this controlled evaluation. Even when $T_{\mathrm{RA}} = 60$ and $a = 1.2$, overbooking increases the MNO profit by 75 %.

3.5. Summary

The chapter introduces kaNSaaS, an overbooking-aware solution for NSaaS that leverages AI models and optimization techniques. The framework addresses both long-term admission control and short-term resource allocation. Through experimental evaluation, the solution shows a significant fourfold increase in MNO profitability. The chapter also highlights the benefits of cloud-native NSaaS management and overbooking strategies, presenting the feasibility of implementing overbooking in real-world network slicing scenarios. Finally, it demonstrates the framework's flexibility and efficiency in resource management and net profit maximization for MNOs.

4 Anticipatory Capacity Allocation for Cost-Efficient Network Slice Management

"The following section is based on the methodology described in [7]. Pages 45 to 59 of this thesis are mainly reused from [7], with the inclusion of a new framework (Fig.4.3) and an explanation of the newly introduced Block 0. The new framework employs a different deep neural network (DNN) model, and all results presented in this thesis are distinct from those in the original work. © 2020 IEEE. Permission has been obtained from the senior author to incorporate this material into this thesis."

Leveraging on the lessons learned from the 5G design, implementation, and deployment, one of the most important characteristics that shall be considered when envisioning the next generation of mobile network architecture is the overall sustainability of the system. Energy and resource efficiency play a crucial role in the aspects that impact this area. In particular, always operating the network with the correct amount of resources, including cloud resources and spectrum resources, can reduce Capital Expenditures (CAPEX) and Operating Expenses (OPEX), yielding an increase in the overall energy efficiency of the system.

Zero-touch resource allocation. In this context, a prominent difficulty is resource management. Isolation of resources across different network slices inherently increases network capacity requirements [14], and a dynamic, preemptive, and efficient allocation of resources to slices is key to keeping efficiency under control in sliced networks [15]. The rapid fluctuations in service demands, as well as the size and complexity of the slicing ecosystem, make legacy reactive, human-driven approaches to resource management inadequate in the emerging context. In sliced networks, automated decisions on resource assignment shall be taken by network orchestrators. By running dedicated AI and ML solutions [20], network orchestrators are expected to enable the vision of *zero-touch networks* [21], *i.e.*, fully self-operating communication infrastructures whose standardization is already ongoing [22].

Current state-of-the-art solutions for capacity forecasting in network slicing take into account the costs due to (i) the allocation of unnecessary resources that go unused, and (ii) the insufficient provisioning of resources that cannot accommodate the demand and lead to violations of the Service-Level Agreement (SLA) with the slice tenant. Hence, they aim to minimize overprovisioning while avoiding SLA violations.

Resource instantiation and reconfiguration. Limiting the problem to the simple trade-off above implicitly assumes that resource instantiation and reconfiguration occur at no cost. While this may hold for some types of resource (*e.g.*, CPU time within the same bare metal machine), it is not generally valid for slice resource management scenarios. Instantiation and reconfiguration costs are capital in NFV technologies that enable the *cloudification* of the access and core networks by entrusting many network functions to Virtual Machines (VMs) running in data centers. Examples include baseband processing in Cloud Radio Access Networks (C-RAN) [86], interconnection functionalities towards the external packet networks through the User Plane Function (UPF) [87], or central office operations [88].

In all the above cases, resource instantiation does not take place for free: Virtual Machine (VM) boot times in prominent public cloud services like Amazon AWS or Google Cloud Platform (GCP) consistently exceed 50 seconds for *cold startup* and 25 seconds for *warm startup*, topping at 120 seconds in worst-case scenarios [89]. Cold startup time refers to the VM's startup time when a user creates a new VM, and warm startup time indicates the startup time measured when a user re(initiates) an existing (and stopped) VM.

Even in recent tests, booting a lightweight VM containing an Alpine Linux takes around 30 seconds in a local deployment [90]. Reconfiguring already allocated resources has also a non-negligible cost: modern software architectures such as Kubernetes need several seconds to execute new pods, *e.g.*, on VMs that are already running [90]. In addition, re-orchestration often implies recomputing paths on the transport networks and implementing them via, *e.g.*, SSDN architectures: the latency is in the order of hundreds of milliseconds in a small five-switch topology and with precomputed routing [91], and this figure has to be scaled to thousands of switches with on-the-fly path re-calculation.

All unavoidable delays above entail monetary fees for the operator, in terms of both violations of the SLA with the tenants (*e.g.*, due to infringement of guarantees on end-to-end latency), and user dissatisfaction (with ensuing high churn rates). By neglecting these sources of cost, present capacity forecast solutions risk introducing uncontrolled data flow latency once deployed in operational networks, ultimately causing economic losses to the operator.

To this end, we adopt a novel approach based on the concept of *multi-timescale orchestration* illustrated in Figure 4.1.

On the left, a state-of-the-art solution for capacity forecasting [63] tries to accommodate the demand and to limit overprovisioning by reconfiguring resources at every re-orchestration opportunity (top); by doing so, it minimizes costs (i) and (ii) above (second plot). However, it also ceaselessly instantiates or de-commissions capacity, and reallocates available resources in a sustained way. This incurs in substantial instantiation and reconfiguration fees (third plot) that ultimately lead to a high overall economic cost (bottom).

On the right, our model performs the orchestration at two timescales and by telling apart

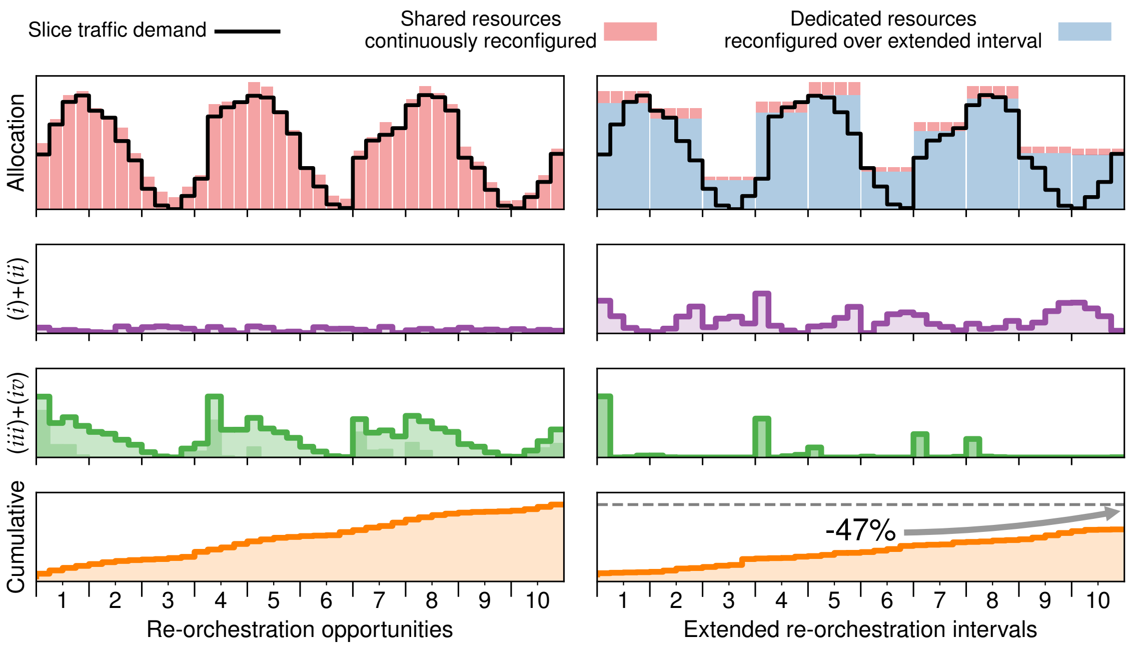


Figure 4.1: Toy example illustrating the costs of resource allocation in network slicing. Top: traffic demand generated by a representative slice (black solid line), along with the capacity allocated based on predictions, and reconfigured at all available re-orchestration opportunities (*i.e.*, shared, in red) or only periodically over extended intervals (*i.e.*, dedicated, in blue). Second row: Monetary costs of (i) overprovisioning and (ii) non-serviced slice traffic. Third row: Monetary costs of resource (iii) instantiation and (iv) reconfiguration. The costs are obtained with a system configuration $\kappa_o = \kappa_s = \kappa_i = 1$ and $\kappa_r = 0.5$, whose meaning is explained in Section 4.1. Bottom: cumulative overall cost over time, for components (i)-(iv). Left: a legacy capacity forecasting model [63] updates the prediction at the fastest rate possible, closely following the demand fluctuations but forcing continuous reconfigurations. Right: our proposed multi-timescale capacity forecasting model trades slightly increased overprovisioning for much-reduced instantiation costs (which are only incurred once per extended interval) and reconfiguration fees (which are completely avoided for dedicated resources). Adapted from [7]. © 2020 IEEE.

two classes of resources. A *long-timescale orchestrator* operates over extended intervals that span multiple re-orchestration opportunities; it allocates a *dedicated capacity* to each slice and also reserves an additional *shared capacity* accessible by any slice. Both capacities remain constant across the extended interval, limiting the frequency of instantiation and thus cost (iii). Only the shared capacity is then reallocated at every re-orchestration opportunity by a *short-timescale orchestrator*, while the configuration of the dedicated capacity is preserved throughout the extended interval, thus reducing cost (iv). Both long- and short-timescale orchestrators decide on the amount of (dedicated and shared) resources to be allocated to each slice to also minimize the usual costs (i) and (ii). This comprehensive strategy results in a 47% reduction of the total cost in the example in Figure 4.1.

Interestingly, the sample case study also clarifies that reducing instantiation and reconfiguration costs has a price in terms of increased overprovisioning. A multi-timescale orchestration model

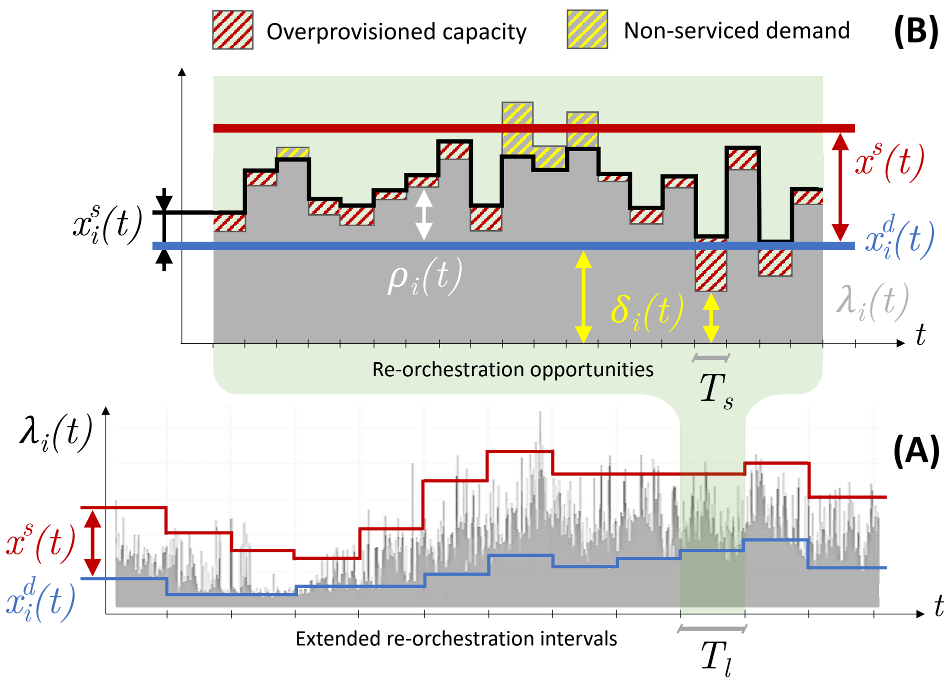


Figure 4.2: Orchestration model. (A) Long-timescale orchestration. The background time series represents the traffic demand generated by slice i (grey). The curves portray the time evolution of the dedicated capacity $x^d_i(t)$ allocated to slice i (blue), and of the shared capacity $x^s(t)$ (red) over extended intervals of duration T_l. Note that $x^s(t)$ is added to the dedicated resources to determine the total available capacity, and, unlike $x^d_i(t)$, is not reserved for slice i but available to all slices. (B) Short-timescale orchestration during one extended interval. At every $T_s < T_l$, a portion $x^s_i(t)$ (black solid curve) of the (fixed) shared capacity $x^s(t)$ is allocated to slice i, based on the residual demand $\rho_i(t)$ not satisfied by the (fixed) dedicated resources $x^d_i(t)$. The plot also highlights the volume of overprovisioned capacity and non-serviced demand (pattern regions), and the slice traffic below dedicated capacity $\delta_i(t)$. Adapted from [7]]. © 2020 IEEE.

allows exploring this and more trade-offs for the first time, empowering an unprecedently comprehensive solution for cost-driven orchestration of slice resources [48] via a mixture of AI and traditional optimization.

4.1. Orchestration Model and Trade-offs

Our orchestration model is outlined in Figure 4.2, which also serves the purpose of illustrating the notation used in the remainder of the paper. Let us denote by $\lambda_i(t)$ the traffic demand generated by services running in slice $i \in \mathbb{S}$ at time t. The long-timescale orchestrator operates on extended intervals of duration T_l. At the beginning of each such interval, it takes decisions on the dedicated capacity $x^d_i(t)$ allotted to slice i, $\forall i \in \mathbb{S}$, and on the additional shared capacity $x^s(t)$ available to all slices; all capacities are conserved throughout the following T_l. The bottom plot (A) in Figure 4.2 depicts an example of allocation resulting from a long-timescale orchestration.

Within an extended interval, the short-timescale orchestrator assigns resources to each slice i at all re-orchestration opportunities, occurring at every T_s. Decisions are based on the (estimated) future residual demand $\rho_i(t) = \max\{0, \lambda_i(t) - x_i^d(t)\}$, and lead to the allocation of an additional capacity $x_i^s(t)$, for each slice i. The resources $x_i^s(t)$ may be re-configured at every T_s, and are provisioned on top of the dedicated $x_i^d(t)$. The top plot (B) in Figure 4.2 illustrates these definitions for a sample short-timescale orchestration during one extended interval.

4.1.1. Sources of Monetary Cost

Building on the notation above, we can formally introduce the different costs associated to the management of resources in sliced networks. As anticipated in the introduction of the chapter, there are four sources of economic penalty for the operator, as follows.
(*i*) Unnecessary resource provisioning: the operator incurs a monetary cost in terms of both CAPEX and OPEX that is directly proportional to the amount of unused resources it allocates to a slice. Such capacity it is instantiated and configured to no purpose and could be allotted, *e.g.*, to other slices to increase the global system efficiency. This cost at time t is

$$\sum_{i\in\mathbb{S}} f_1\left(\max\{0, x_i^d(t) - \delta_i(t)\}\right) + \sum_{i\in\mathbb{S}} f_1\left(\max\{0, x_i^s(t) - \rho_i(t)\}\right) + f_1(x^s(t) - \sum_{i\in\mathbb{S}} x_i^s(t)), \quad (4.1)$$

where $\delta_i(t) = \min\{\lambda_i(t), x_i^d(t)\}$ denotes the portion of the demand of slice i served by the dedicated capacity at time t, as shown in plot (B) of Figure 4.2. The first two terms in (4.1) denote the cost of overprovisioning at slice i and time t, due to the unneeded allocation of dedicated and shared capacity, respectively; again, we refer the reader to plot (B) of Figure 4.2 for an exemplification. The third term captures instead the overprovisioned shared capacity that is not allocated to any slice by the short-term orchestrator. The function $f_1(\cdot)$ describes the scaling of cost with capacity overprovisioning. As in [63], in our evaluation we assume a linear increase of the penalty, *i.e.*, $f_1(x) = \kappa_o x$, where κ_o is the monetary cost of one unit of capacity and is expressed in \$/Mbps. However, our model can easily accommodate different definitions of the scaling law, which may apply to specific network functions.
(*ii*) Non-serviced demand: every time the operator does not allocate sufficient resources to serve the traffic demand of a slice, it violates the SLA with the tenant, which triggers a monetary compensation. The associated cost at time t is

$$\sum_{i\in\mathbb{S}} \kappa_s \cdot \mathbb{1}_{<\rho_i(t)}\left(x_i^s(t)\right), \quad (4.2)$$

where $\mathbb{1}_A(x)$ is an indicator function that takes a value 1 if the argument satisfies condition A, and 0 otherwise. Thus, $\mathbb{1}_{<\rho_i(t)}\left(x_i^s(t)\right)$ activates when the portion of shared capacity assigned to i does not suffice to meet the service demand; this corresponds to an underprovisioning situation, as depicted in Figure 4.2. In these cases, the operator has to indemnify the tenant for a value κ_s, in \$, per SLA violation. This definition is also in line with those used in the literature [63].

(*iii*) Resource instantiation: in presence of substantial variations of the total traffic demand, the operator needs to instantiate new resources to serve the demand of the slice. In these cases, as discussed at the start of Chapter 4, there exists a cost associated with enabling such new resources. As an example, if additional VMs have to be bootstrapped or migrated to serve the slice, the operator has increased expenses in terms of power consumption and CPU cycles. In addition, there may be an indirect penalty in terms of perceived QoS, as this operation may take minutes [90] and disrupt the end-user experience. The cost, triggered at every T_l in our multi-timescale model, can be modeled as

$$\sum_{i \in \mathbb{S}} f_2\left(\delta_i(t)\right) \cdot \mathbb{1}_{>x_i^d(t-1)}\left(x_i^d(t)\right) + f_2\left(\min\{\rho_i(t), x_i^s(t)\}\right) \cdot \mathbb{1}_{>x^s(t-1)}\left(x^s(t)\right). \tag{4.3}$$

The first term in (4.3) represents the penalty incurred when new dedicated resources must be instantiated, which occurs when $x_i^d(t) > x_i^d(t-1)$. The second term is equivalent to the first one, but refers to the shared capacity instantiated to all slices in $\mathbb{S}$. Note that the costs induced by both terms are functions $f_2(\cdot)$ of the affected traffic that may experience disruption, *i.e.*, $\delta_i(t)$ and $\min\{\rho_i(t), x_i^s(t)\}$, respectively.[1] In our performance evaluation, we consider the cost to be directly proportional to the affected traffic, *i.e.*, $f_2(x) = \kappa_i x$, where the parameter κ_i captures the estimated fee for delaying one unit of capacity due to resource instantiation, expressed in \$/Mbps.

(*iv*) Resource reconfiguration: while resources are only instantiated at every T_l, a short-timescale orchestration of the shared capacity within each extended interval allows accommodating faster fluctuations of the slice demand. Every time the operator reconfigures the shared capacity, it incurs a cost; as mentioned in the introduction of the chapter, this is the case with the reconfiguration of the SDN transport networks, the setup of load balancers, or the creation of new instances of a VNF on a VM previously used by another slice. All these operations have a price in terms of management delay [90], expressed as

$$\sum_{i \in \mathbb{S}} f_3\left(\min\{\rho_i(t), x_i^s(t)\}\right) \cdot \mathbb{1}_{\neq x_i^s(t-1)}\left(x_i^s(t)\right). \tag{4.4}$$

The above cost is present whenever the shared resources must be reconfigured for a slice i, *i.e.*, $x_i^s(t) \neq x_i^s(t-1)$. In such situations, the cost is dependent on the amount of traffic affected by the reconfiguration process, *i.e.*, $\rho_i(t)$ bounded by $x_i^s(t)$. In our study, we assume that the economic penalty is the same for any bit of traffic using reconfigured resources, hence $f_3(x) = \kappa_r x$, where κ_r is in \$/Mbps. Also, in this case, other functions can be easily embedded in the overall framework to represent distinctive cost models identified by the operator.

[1] Instantiation and reconfiguration costs are proportional to the full amount of impacted demand (rather than, *e.g.*, to the increment of demand with respect to the previous re-orchestration opportunity) since finding a stable and optimal allocation of resources typically requires reconsidering their organization within a large portion of the network, or even across all slices [92].

4.1.2. Trade-offs in Capacity Allocation

The basic trade-off in anticipatory resource assignment is that between overprovisioning and non-serviced demands.

- **Trade-off A.** Increasing the amount of resources makes overprovisioning more likely, but reduces the probability that the allocated capacity is not sufficient to serve the future demand. This results in opposing costs (i) and (ii).

Current capacity forecasting models aim at identifying the optimal compromise that minimizes the joint penalty of the costs in trade-off A above [63]. However, these models do not offer any control over instantiation and reconfiguration. By adopting a multi-timescale approach, we are instead capable of factoring such variables in. Specifically, the model presented in Section 4.1.1 tells apart the capacity allocated to each slice into a dedicated capacity, re-orchestrated over long timescales with period T_l, and a shared capacity, re-orchestrated over short timescales with period T_s. This unlocks additional degrees of freedom: the orchestrator can decide not only how many resources to assign to a slice, but also which portion of those shall be of each type, and for how long they stay unaltered.

Our model still allows addressing trade-off A above, by controlling the total allocated capacity during each extended interval T_l, *i.e.*, $x^s(t) + \sum_{i \in \mathbb{S}} x_i^d(t)$, and modulate the costs of overprovisioning and non-serviced demands. Yet, its flexibility enables the exploration of the following additional trade-offs.

- **Trade-off B.** By increasing the dedicated capacity $x_i^d(t)$ allocated to slice i during an extended interval, the orchestrator can serve a larger fraction of the slice traffic with resources that do not need reconfiguration. However, such resources cannot be reused by other slices during T_l whenever they are not needed by slice i. For instance, in plot (B) of Figure 4.2, increasing $x_i^d(t)$ would reduce the residual demand $\rho_i(t)$ that is served with reconfiguration-heavy shared capacity; but it would also generate additional overprovisioning, *e.g.*, in the fourth and second to last re-orchestration opportunities. This leads to a trade-off between costs (i) and (iv).

- **Trade-off C.** Allocating a larger shared capacity $x_i^s(t)$ to slice i during an interval T_s reduces the risk that the resources will not be sufficient to serve the future slice demand. Nevertheless, it also causes the reconfiguration of more resources. As an example, in plot (B) of Figure 4.2, increasing $x_i^s(t)$ in the third T_s slot could remove the non-serviced demand, but would also grow the reconfiguration penalty. A trade-off exists between costs (ii) and (iv).

- **Trade-off D.** Increasing the duration T_l of the extended interval reduces the cost of resource instantiation, which only occurs once per extended interval. However, a higher T_l also forces the dedicated capacities $x_i^d(t)$ and the total shared capacity $x^s(t)$ to remain

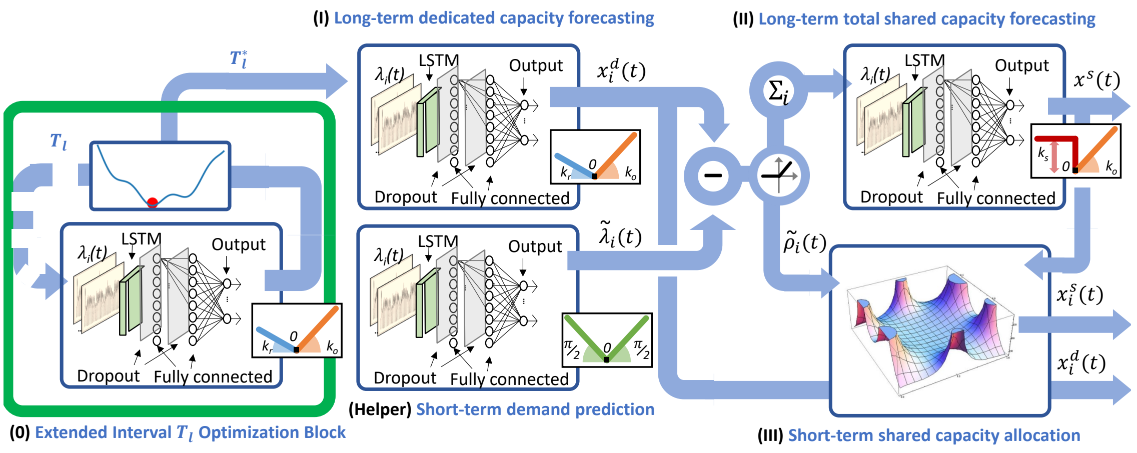

(I) Long-term dedicated capacity forecasting
(II) Long-term total shared capacity forecasting
T_l^*
T_l
LSTM
Output
Dropout
Fully connected
$\lambda_i(t)$
$x_i^d(t)$
$\tilde{\lambda}_i(t)$
Σ_i
$\tilde{\rho}_i(t)$
$x^s(t)$
$x_i^s(t)$
$x_i^d(t)$
k_r
k_o
k_s
0
$\pi/2$
(0) Extended Interval T_l Optimization Block
(Helper) Short-term demand prediction
(III) Short-term shared capacity allocation

Figure 4.3: Overview of the AZTEC+ framework. The learning flow proceeds from left to right. The input mobile data is processed by deep neural networks that, jointly with a Golden-Section Search algorithm, find the optimal Extended interval T_l (Block 0), which is the addition for AZTEC+, in the green edges. Then, for each slice $i \in S$, a deep neural network returns the long-term dedicated capacity $x_d^i(t)$, having as input not only the mobile data, but also the optimal extended interval T_l^*, and the short-term estimated demand $\lambda_i(t)$, respectively. These values are combined to obtain the estimated residual demands $\hat{\rho}_i(t)$. The aggregate residual demand over all slices is input to a further deep neural network to determine the long-term shared capacity $x_s(t)$. Such capacity is then fed, along with per-slice residuals, to an optimization module that allocates the shared resources $x_s^i(t)$.

.

constant for a longer time. With a reduced capability to tailor the network resources to the fluctuations of the slice traffic demands, the orchestrator may incur in increased overprovisioning or underprovisioning. For instance, extending the timespan of plot (B) in Figure 4.2 to cover a prolonged demand $\lambda_i(t)$ may create additional situations where $x_i^d(t) > \lambda_i(t)$, *i.e.*, dedicated resources go wasted, as in the second to last T_s interval; it can also generate new cases where $x_i^s(t) < \rho_i(t)$, and traffic peaks are not serviced, as in the central part of the example. This results in a trade-off between cost (iii) and joint costs (i) and (ii).

Next, we present a framework that takes automated, anticipatory decisions on capacity allocation by addressing all trade-offs outlined above, thanks to the multi-timescale model.

4.2. The AZTEC+ framework

Our framework, named AZTEC+ (*i.e.*, capacity Allocation for Zero-Touch nEtwork sliCing), automatically solves trade-offs A, B, and C above by finding an effective compromise among the opposing goals of reducing the operator's costs in terms of (i) overprovisioning, (ii) non-serviced slice demands, and (iv) resource reconfiguration. In addition, AZTEC+ dynamically and automatically selects the extended interval T_l^* that minimizes the penalty associated with (iii) capacity instantiation, to address trade-off D. Next, we first provide an overview of the framework, and then discuss the implementation of its different components for long- and short-timescale orchestration and for extended interval selection.

Block (0) selects the Extended Interval T_l^* that minimizes the instantiation and reconfiguration costs through the Golden-Section Search Algorithm [93]. To do so, it performs the forecasting of the long-term dedicated capacity for each network slice $x_d^i(t)$, using as input information, the actual traffic generated by each slice during the preceding N re-orchestration opportunities and a candidate extended interval T_l. During one cycle of the Golden-Section Search Algorithm, the DNN is trained using the candidate T_l value, and its validation loss is evaluated. The Golden-Section Search Algorithm, progressively adjusting T_l, converges to the T_l^* that returns the minimum validation loss. The validation loss is computed as the sum of the reconfiguration and instantiation costs. The cycle is repeated 10 times, and the Extended Interval T_l, which achieves the lowest validation loss, is selected.

Block (I) performs the forecasting of the long-term dedicated capacity for each network slice $x_i^d(t)$, using as input information about the actual traffic generated by each slice during the preceding N re-orchestration opportunities. As discussed for trade-off B in Section 4.1.2, the capacity $x_i^d(t)$ modulates the impact of (i) provisioning unnecessary dedicated resources versus (iv) re-configuring the shared resources needed to serve the residual demand beyond $x_i^d(t)$. Hence, block (I) identifies $x_i^d(t)$ minimizing costs (i) and (iv).

Block (II) determines the long-term shared capacity $x^s(t)$ available to any slice during the subsequent time interval T_l. The shared capacity is used to serve the residual demands of all slices; hence $x^s(t)$ shall be dimensioned to the aggregate residual traffic $\sum_{i\in\mathbb{S}} \rho_i(t)$. Thus, block (II) receives as input an estimate of such aggregate during the previous extended interval, *i.e.*, $\sum_{i\in\mathbb{S}} \tilde{\rho}_i(t)$, $t \in [t-(T_l/T_s), t-1]$, and predicts the next $x^s(t)$ such that (i) overprovisioning of shared resources is reduced as much as possible, and (ii) all residual demands can be accommodated within $x^s(t)$. In this way, block (II) addresses trade-off A, jointly minimizing costs (i) and (ii). Each approximate $\tilde{\rho}_i(t) = \max\{0, \tilde{\lambda}_i(t) - x_i^d(t)\}$ is computed from a forecast $\tilde{\lambda}_i(t)$ returned by a *helper* legacy traffic predictor that forecasts per-slice demands over the next re-orchestration opportunity t, as per Figure 4.3. Note that, at this stage, the actual residuals $\rho_i(t)$ could be directly determined from the (known) real traffic demand $\lambda_i(t)$ observed during the previous interval T_l. However, we opt to feed block (II) with an estimate based on $\tilde{\lambda}_i(t)$ to ensure overall system consistency. Indeed, the short-timescale orchestrator –implemented by block (III) and presented next– necessarily operates on the predicted residuals $\tilde{\rho}_i(t)$ over the future T_s interval. Considering the same estimates in the long-timescale module allows allocating the shared capacity $x^s(t)$ in a way that is conscious of the inaccuracy of the information available during the following short-term resource assignment phase of the framework.

Once the long-term capacities $x_i^d(t)$, $\forall i \in \mathbb{S}$, and $x^s(t)$ are set, the short-timescale orchestrator assigns portions $x_i^s(t)$ of the total shared resources to each slice. This allocation occurs at every re-orchestration opportunity, spaced by T_s, and is carried out by block (III) of the AZTEC+ framework as follows.

Block (III) receives as input the long-term shared capacity $x^s(t)$, and the future residual demand $\tilde{\rho}_i(t)$ expected for each slice i during the following T_s. The available total capacity is allotted to each slice in a way to solve the trade-off C described in Section 4.1.2. Therefore, block (III) computes $x_i^s(t)$ for each $i \in \mathbb{S}$, by minimizing the combination of costs (ii) and (iv), corresponding to insufficient allocated capacity and additional shared resource reconfiguration, respectively.

Overall, blocks (I)-(III) return a forecast of all capacities $x_i^d(t)$, $x^s(t)$ and $x_i^s(t)$ that the operator shall allocate over both long and short intervals of duration T_l^* set by block (0), and T_s, respectively. The resulting anticipatory allotment reduces the network management costs associated with the provisioning of exceeding or inadequate resources and their reconfiguration over time.

We remark that differently from AZTEC, AZTEC+ takes automated decisions on the value of the extended re-orchestration interval, T_l in order to tackle the penalty of network resource instantiation. As explained above, control on instantiation costs is achieved by block (0) that automatically and dynamically sets T_l to cope with trade-off D. Thanks to the addition of block (0), the system is not only more reliable due to the automatic selection of T_l (avoiding the human in the loop), but also more adaptive to other scenarios.

Having clarified the role of each block, we detail in the following their implementation, which leverages a combination of deep learning and numerical optimization methods.

4.3. Long-timescale Resource Assignment

The long-timescale orchestration is carried out by blocks (0), (I) and (II) of the AZTEC+ framework, as follows:

4.3.1. Forecasting for Extended Interval and Dedicated Capacity

The DNN architecture of Block (0) and Block (I) is illustrated in the respective components of Figure 4.3. It consists of an LSTM layer with 128 cell units and two Fully Connected (FC) layers of 64 and $||\mathbb{S}||$ neurons, respectively, where the operator $||\cdot||$ returns the cardinality of the argument set. The layers are interleaved by two dropout layers [94].

All layers employ ReLU as the neuron activation function, except for a linear function in the last FC layer. The loss function that drives the DNN training is a custom expression designed to account for the actual management costs incurred by the operator in case of errors in the orchestration of the dedicated capacity. If the operator were able to allocate to a slice i a constant capacity $x_i^d(t)$ that perfectly matched the actual demand $\lambda_i(t)$ over the next T_l, the error and cost would be nil: this is the ideal scenario where all the demand is serviced, without any overprovisioning or re-configuration. However, in practical cases, it is impossible to perfectly predict $\lambda_i(t)$, which is also very unlikely to be constant over the whole T_l. In this case, positive errors $x_i^d(t) - \lambda_i(t)$ lead to overprovisioning, with a cost set by the first term of (4.1) in Section 4.1.1, and negative errors imply that the demand in excess of $x_i^d(t)$ needs to be served by the shared capacity, with (4.4) setting the re-configuration penalty.[2]

Positive errors yield $\delta_i(t) = \lambda_i(t)$, while $\rho_i(t) = \lambda_i(t) - x_i^d(t)$ for negative errors. Then, the loss function for $x_i^d(t)$ allocated at t is $\sum_{t \in \mathbb{T}} \ell_i^{(\mathrm{I})}(t)$, where $\mathbb{T}$ is the set of concerned re-orchestration opportunities $\{t, \ldots, t + (T_l/T_s) - 1\}$, and

$$\ell_i^{(\mathrm{I})}(t) = \begin{cases} f_3\left(\lambda_i(t) - x_i^d(t)\right) & \text{if } x_i^d(t) \leq \lambda_i(t) \\ f_1\left(x_i^d(t) - \lambda_i(t)\right) & \text{otherwise.} \end{cases} \tag{4.5}$$

Importantly, (4.5) has partial derivatives that form a piece-wise constant function, which guarantees robust and fast convergence under popular first-order optimizers like Adam [95].

In order to further improve the quality of the allocation of dedicated resources, we leverage a recent result in neural network design, which allows estimating the uncertainty of the learning outcome. Specifically, adding dropout layers during model testing is mathematically equivalent to generating an approximation of the probabilistic deep Gaussian process [96]. This observation, which holds for DNNs with arbitrary depth and non-linearities, provides a way to return a distribution instead of a scalar output value. We thus activate the dropout layers during testing, and adopt a Monte Carlo strategy; namely, we perform the forward pass L times for each test input, obtaining outputs $\{x_i^{d,1}(t), \ldots, x_i^{d,L}(t)\}$ for, slice i at time t. We then compute the mean $\mu_i^d(t) = 1/L \cdot \sum_{l=1}^{L} x_i^{d,l}(t)$ as well as the variance $\sigma_i^d(t) = 1/L \cdot \sum_{l=1}^{L} (\mu_i^d(t) - x_i^{d,l}(t))^2$, and approximate the model uncertainty as $\mathcal{N}(\mu_i^d(t), \sigma_i^d(t))$.

Knowledge of the model uncertainty allows adding a safety margin to the standard DNN outcome. Since the whole support of $\mathcal{N}(\mu_i^d(t), \sigma_i^d(t))$ represents potentially correct values of the dedicated capacity, block (I) returns the 99th percentile of the distribution; this makes it very unlikely to output a $x_i^d(t)$ that is lower than the best one, minimizing the risk of SLA violations. Thus, when the DNN is confident about the quality of the result, it returns a value close to the mean; otherwise, it adds a substantial safety margin. An example is in Section 4.5.1.

4.3.2. Forecasting for Long-term Shared Capacity

Block (II) is implemented using a dedicated DNN, although with a simpler structure due to the reduced richness of the input. The DNN is fed with a single time series of the total residual demand in the past extended interval, *i.e.*, $\sum_{i \in \mathbb{S}} \tilde{\rho}_i(t)$, $t \in [t - (T_l/T_s), t - 1]$. The input is processed by three FC layers with 128, 64 and 1 neurons; the first two use ReLU activation functions, while the last uses a linear function. A dropout layer is between the first and second FC layers.

The DNN is trained with a different custom loss function that accounts for the correct sources

[2] As the long-term orchestrator is agnostic of the short-timescale operation, it cannot factor $x_i^s(t)$ in the cost of negative errors. So, block (I) assumes perfect management of the shared capacity that always accommodates a continuously varying residual demand, reducing (4.4) to $\sum_{i \in \mathbb{S}} f_3\left(\rho_i(t)\right)$.

of monetary penalty in case of errors. An ideal case where a fixed long-term shared capacity $x^s(t)$ perfectly matches a constant aggregate residual demand is unrealistic, and errors are unavoidable. Positive errors yield overprovisioning of $x^s(t)$, whereas negative ones have a cost in terms of denied traffic. The former corresponds to the second and third terms[3] in (4.1), while the latter maps to the monetary fee[4] for SLA violations in (4.2). The loss function jointly capturing these costs for the extended interval starting at t is $\sum_{t\in\mathbb{T}} \ell_i^{(\mathrm{II})}(t)$, where

$$\ell_i^{(\mathrm{II})}(t) = \begin{cases} \kappa_s & \text{if } x^s(t) < \sum_{i\in\mathbb{S}} \tilde{\rho}_i(t) \\ f_1\left(x^s(t) - \sum_{i\in\mathbb{S}} \tilde{\rho}_i(t)\right) & \text{otherwise.} \end{cases} \tag{4.6}$$

The expression in (4.6) needs to be slightly modified by adding minimum slopes that make the function differentiable over $\mathbb{R}$. With this, the loss function has the same desirable properties as the ones mentioned for (4.5). Finally, we take advantage of the dropout layer also in this case: we thus approximate the model uncertainty, and return the 99th percentile of the distribution as a safety margin on the correct value of $x^s(t)$.

4.4. Short-timescale Resource Assignment

The short-timescale orchestration consists of block (III) supported by a *helper* short-term demand predictor, as outlined in Figure 4.3. The predictor uses a DNN architecture that is very similar to that adopted by block (I); indeed, the two DNNs operate on the same input, and produce per-slice forecasts. The main differences between them are in the frequency of operation and, most notably, in the loss function. The helper predictor outputs a prediction at every T_s instead of at every T_l. Furthermore, it uses a traditional) instead of the cost-aware loss Mean Absolute Error (MAE) function in (4.5): MAE considers identical contributions by the positive and negative errors, thus producing an output $\tilde{\lambda}_i(t)$ that tries to follow as closely as possible the upcoming slice traffic demands.

Given the total shared capacity $x^s(t)$, and the estimated residual demands $\tilde{\rho}_i(t)$, AZTEC has to decide how to distribute $x^s(t)$ across the requesting slices at every T_s. This is implemented in block (III) using numerical optimization.

The primary objective of the shared resource assignment performed by block (III) is avoiding SLA violations due to insufficient available capacity, which would induce a cost modeled in (4.2). At the same time, issuing non-essential resources has a penalty in terms of unnecessary reconfiguration cost, as captured by the expression in (4.4).

[3]Also in this case, the shared capacity allotted to individual slices $x_i^s(t)$ used in (4.1) has not yet been determined at this stage. The safest option is hence to assume that the whole residual demands will be correctly assigned by the short-term orchestrator. This leads to approximating the allocated shared resources $x_i^s(t)$ by $\tilde{\rho}_i(t)$. Under this hypothesis, the second and third terms in (4.1) reduce to 0 and $f_1\left(x^s(t) - \sum_{i\in\mathbb{S}} \tilde{\rho}_i(t)\right)$, respectively.

[4]By assuming $x_i^s(t) = \tilde{\rho}_i(t)$, unserviced demands are possible only when $x^s(t)$ is insufficient. Then, (4.2) translates into $\kappa_s \cdot \mathbb{1}_{<\sum_{i\in\mathbb{S}} \tilde{\rho}_i(t)}\left(x^s(t)\right)$.

This corresponds to trade-off C in Section 4.1.2, and can be formulated as:

$$\begin{aligned} \min_{x_i^s(t)} \quad & \sum_{i \in \mathbb{S}} \kappa_s P\left(\tilde{\rho}_i(t) > x_i^s(t)\right) \\ \text{subject to} \quad & \sum_{i \in \mathbb{S}} x_i^s(t) \leq x^s(t), \end{aligned} \tag{4.7}$$

The above minimizes the expected value of the expenditure for non-serviced slice demands in (4.2). Note that, by squeezing as many slices as possible within the total capacity, the solution to (4.7) implicitly addresses the challenge of minimizing reconfiguration costs. Also, the formulation in (4.7) deals with probabilities: this is consistent with the probabilistic nature of $\tilde{\rho}_i(t)$ granted by uncertainty approximation via dropout layers.

Due to the empirical nature of the probability distribution $\tilde{\rho}_i(t)$, (4.7) has no closed form and thus we cannot employ classical optimization methods. Furthermore, as we do not have a differentiable objective function, we cannot apply approaches that depend on gradients. Hence, we resort to numerical methods that search for the optimal solution within the feasible set of $[x_i^s(t)]$. The following change of variable

$$p_i(t) = \begin{cases} x_i^s(t) / \left(x_i^s(t) + x_{i+1}^s(t)\right) & \text{if } i \in [1, \|\mathbb{S}\| - 1] \\ \sum_i x_i^s(t) / x^s(t) & \text{if } i = \mathbb{S} \end{cases} \tag{4.8}$$

yields $p_i(t) \in [0,1]$, $\forall i \in \mathbb{S}$, which simplifies the search to the N-dimensional variable space with fixed bounds $[0,1]$ on all variables. Note that the first $\|\mathbb{S}\| - 1$ variables $p_i(t)$ represent the relative amount of shared capacity assigned to slice i with respect to the slice $i+1$, while the last $p_{\|\mathbb{S}\|}(t)$ represents the overall amount of shared capacity assigned to the services. Therefore, the variable change in (4.8) serves the following purposes: first, the constraint on the sum of the variables is now enforced through a constraint on each variable; second, we avoid ties between variables, allowing a safe exploration of the solution space where we can focus one variable, and changing its $p_i(t)$ value within the entire range without impacting any of the other variables $p_j(t)$ for $j \neq i$.

Algorithm 1 details the numerical solution for shared resource assignment adopted by AZTEC. A main function takes as input the total available shared capacity $x^s(t)$ and the empirical distribution of the capacity needed by each service in the next time interval $\tilde{\rho}_i(t)$. Two helper functions, `COST` and `TRANSFORM`, compute the cost expected value for a given assignment $p_1(t), \ldots, p_{\|\mathbb{S}\|}(t)$, and transform $p_i(t)$ back to $x_i^s(t)$, respectively. Thanks to the variable change in (4.8), we can use a gradient-free algorithm that works with constrained input variables for the minimization of (4.7). In particular, we used the BOBYQA algorithm [97], a gradient-free optimizer that supports constrained variables.

All ratios of shared capacity are initialized to 0.5, *i.e.*, all slice start with the same amount of resources. However, given the possibly high number of slices that can be included in the system, there may be different local minima and there exists the chance of getting stuck in a local

Algorithm 1: Shared resource assignment algorithm

```
Function TRANSFORM($p_1, \dots, p_{\|\mathbb{S}\|}$):
$x^s_{\|\mathbb{S}\|} = \frac{p_{\|\mathbb{S}\|} x^s}{\left(\sum_{i=1}^{\|\mathbb{S}\|-1} \prod_{j=i}^{\|\mathbb{S}\|-1} \frac{p_j}{1-p_j}\right)+1}$
$x^s_i = \left(\prod_{j=i}^{\|\mathbb{S}\|-1} \frac{p_j}{1-p_j}\right) x^s_{\|\mathbb{S}\|}$, $i \in [1, \|\mathbb{S}\| - 1]$
return $x^s_1, \dots, x^s_{\|\mathbb{S}\|}$
Function COST($p_1, \dots, p_{\|\mathbb{S}\|}$):
    $x^s_1, \dots, x^s_{\|\mathbb{S}\|} \leftarrow$ TRANSFORM($p_1, \dots, p_{\|\mathbb{S}\|}$)
    $X \leftarrow \sum_i \kappa_s P\left(\tilde{\rho}_i > x^s_i\right)$
    return X
Function MAIN($x^s, \tilde{\rho}_i$):
    $c \leftarrow$ BOBYQA(COST($p_1 = 0.5, p_2 = 0.5, \dots, p_{\|\mathbb{S}\|-1} = 0.5, p_{\|\mathbb{S}\|}$))
    $p^0_1, \dots, p^0_{\|\mathbb{S}\|} \leftarrow$ GOLDEN ($c(p_{\|\mathbb{S}\|})$)
    $p_1, \dots, p_{\|\mathbb{S}\|} \leftarrow$ BOBYQA ($p^0_1, \dots, p^0_{\|\mathbb{S}\|}$)
    $x^s_1, \dots, x^s_{\|\mathbb{S}\|} \leftarrow$ TRANSFORM($p_1, \dots, p_{\|\mathbb{S}\|}$)
    return $x^s_1, \dots, x^s_{\|\mathbb{S}\|}$
```

minima that does not deliver a good performance. To reduce the probability that this happens, we perform a preliminary search for the best starting $p_{\|\mathbb{S}\|}(t)$, via the Golden Section method [98] and considering $p_{\|\mathbb{S}\|}(t)$ as the only variable for the cost.

4.5. Evaluation Results

As a preliminary step in our evaluation of AZTEC+, we investigate the impact of including the knowledge of the estimated uncertainty in the forecast produced by the different DNNs that are part of the framework.

4.5.1. Harnessing the Forecast Uncertainty

As presented in Section 4.2, we leverage a recent result on the approximation of uncertainty in DNN to include a safety margin in the model forecast. Figure 4.4 visually shows the benefit of this design choice, by juxtaposing the performance of AZTEC+ with and without uncertainty; in the second case, dropout layers are deactivated during test, and all DNNs produce a single-value output. In each plot, we report the anticipatory allocation of capacities $x^d_i(t)$ and $x^s_i(t)$ to the target slice i, as well as that of the total shared capacity $x^s(t)$; the actual demand is on the background.

The plots illustrate well how the time-varying resource allocation achieved by AZTEC+ nicely follows the fluctuations of the slice traffic. More interestingly, AZTEC+ (in the top plot) achieves a more reliable assignment of slice resources by accounting for the level of uncertainty of the predictions. The total capacity $x^d_i(t) + x^s_i(t)$ is smoother and avoids situations where the demand cannot be serviced. Conversely, the framework not accounting for uncertainties (in the bottom plot) yields a capacity allocation that is noisy and incurs in substantial SLA violations due to unsatisfied demands.

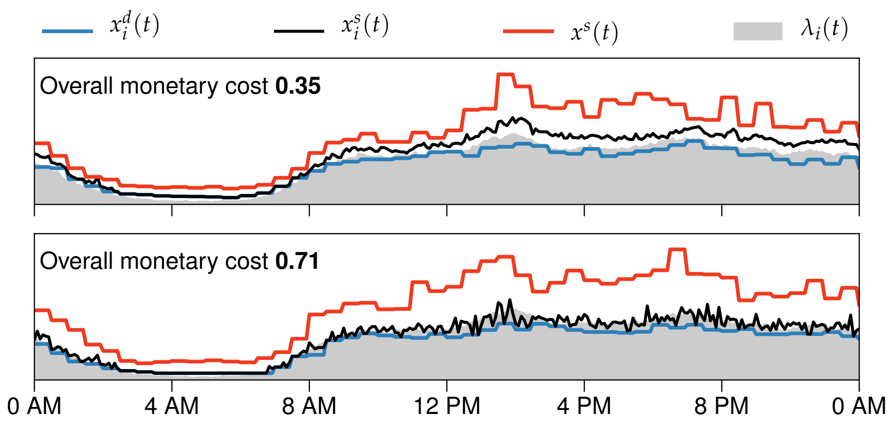


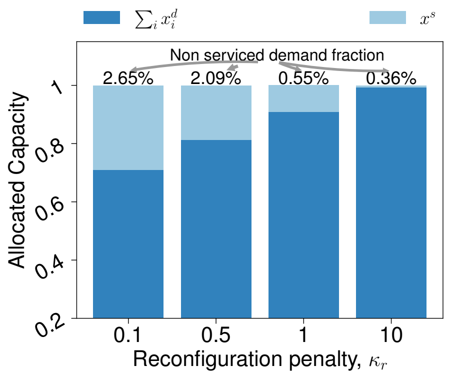
Figure 4.4: Time series of sample resource allocations. Top: AZTEC+ framework. Bottom: equivalent framework where no uncertainty estimates are used. Adapted from [7]. © 2020 IEEE.

In addition, AZTEC+ achieves such a result while saving on the amount of allocated resources (note the lower $x^s(t)$ curve), which ultimately results in an overall monetary cost that is half of that incurred by the framework without uncertainties. While this is an excerpt from a specific test, we recorded similar gains for all settings explored in our analysis.

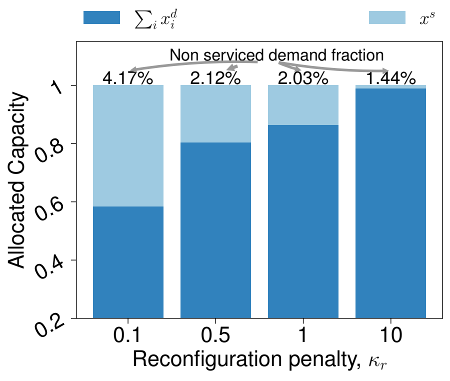
4.5.2. Capacity Breakdown

Fig. 4.5 illustrates the capacity breakdown for different instantiation penalties (i.e. $k_i = 1$ (Fig.4.5a), $k_i = 10$ (Fig.4.5b), $k_i = 50$ (Fig.4.5c), $k_i = 100$ (Fig.4.5d)) as well as reconfiguration penalties (i.e. $k_r = 0.1$, $k_r = 0.5$, $k_r = 1$, $k_r = 10$)). Across all results, a consistent trend is observed: a greater proportion of traffic is assigned to the more flexible shared capacity when the reconfiguration fee is low, while redirecting traffic to the dedicated capacity becomes more cost-effective as k_r increases. Indeed, over-provisioning dedicated resources becomes more practical than reconfiguring shared resources as k_r grows. Additionally, we can also notice that the usage of shared capacity is higher as k_i increases.

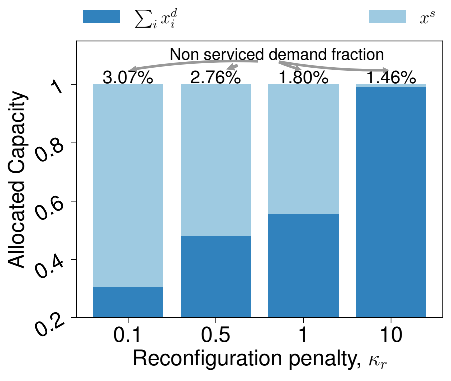
AZTEC+ tends to have higher SLA violation when using more resources from the shared capacity (low reconfiguration penalty k_r) than when using more resources from the dedicated capacity. This is a direct consequence of the selection of longer T_l^*. We observe a high variation in the usage and distribution of shared/dedicated capacity depending on the instantiation penalty cost, ranging from 70% of dedicated capacity when reconfiguration penalty $k_r = 1$ and instantiation penalty $k_i = 1$ to a 30% when $k_r = 0.1$ and $k_i = 100$. Indeed, when the reconfiguration cost is higher, the system tends to allocate the maximum dedicated capacity for each slice. Finally, there is a clear trade-off between k_r and k_i and their impact on resource allocation. For higher instantiation penalties and lower reconfiguration penalties, more resources are assigned to shared capacity x_s. Conversely, for higher reconfiguration penalties, resources are allocated to dedicated capacity x_d.



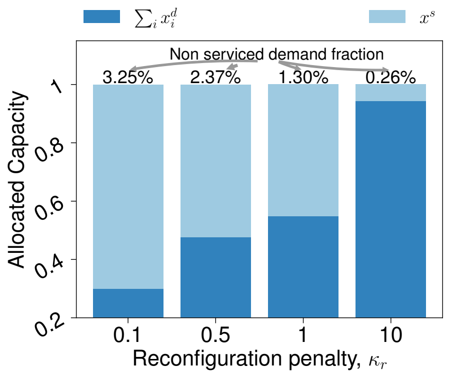
(a) $k_i = 1$



(b) $k_i = 10$



(c) $k_i = 50$



(d) $k_i = 100$

Figure 4.5: Total dedicated capacity $\sum_{i \in \mathbb{S}} x_i^d(t)$ and shared capacity $x^s(t)$ allocated by AZTEC+ versus κ_r. Numbers denote the fraction of re-orchestration opportunities with insufficient allocated resources.

4.5.3. Block III Internals

In Figure 4.6, the detailed operativity of Block (III) is illustrated. The vertical black line represents the actual value of the dedicated capacity $x_d^i(t)$. The grey line represents the probability that the real load is below the dedicated capacity ($\delta_i(t)$) and the red line represents the probability that the real load exceeds the dedicated capacity ($\rho_i(t)$), potentially leading to an SLA violation. The probability distribution is obtained from the Helper output. The blue dot indicates the actual real traffic value $\lambda_i(t)$, which Block (III) is unaware of. Block (III), by solving the optimization problem, assigns shared capacity to slice i, aiming at minimizing the red area. The green dot represents the assigned shared capacity for slice i, $x_s^i(t)$, derived from the Block (III) output. There are four possible scenarios to consider, as follows.

(i) **No allocation.** If given the dedicated capacity allocated to slice i by Block (I), the probability of an SLA violation is 0, then Block (III) does not allocate resources from the shared capacity (see Fig. 4.6a). Here, Block (III) observes the probability distribution derived from the helper output and the value of static capacity $x_d^i(t)$ allocated by Block (I) and decides not to allocate resources from $x_s(t)$.

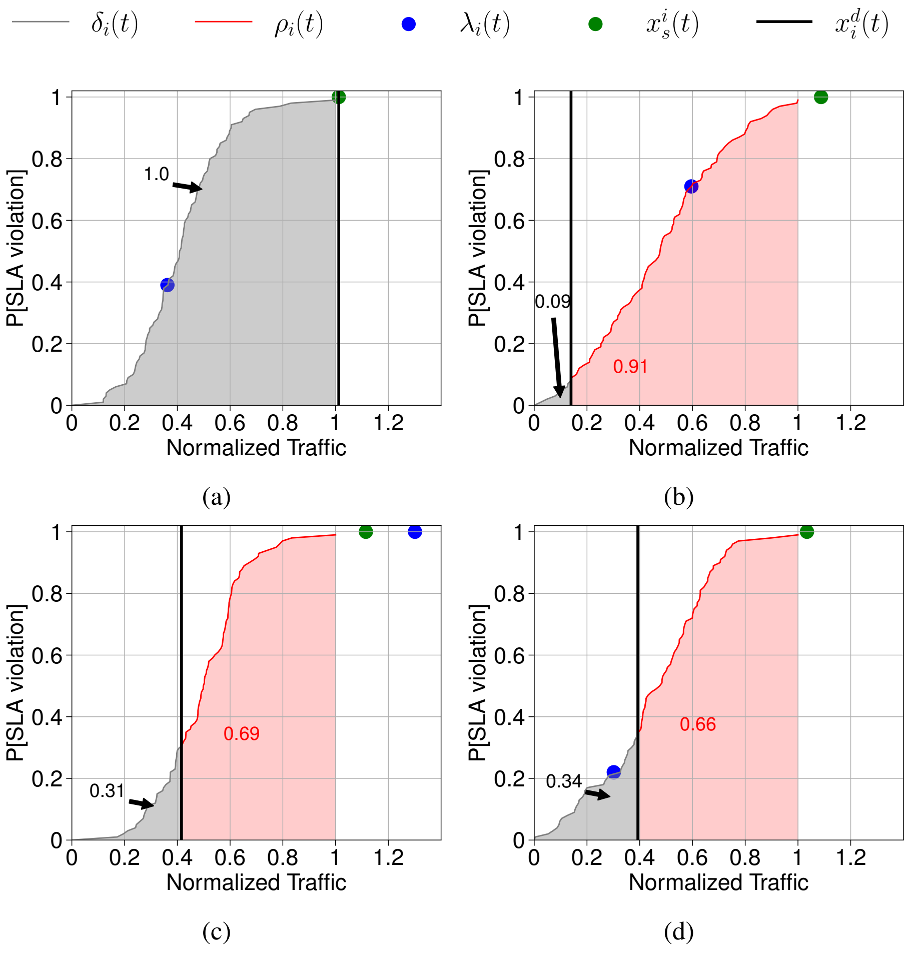
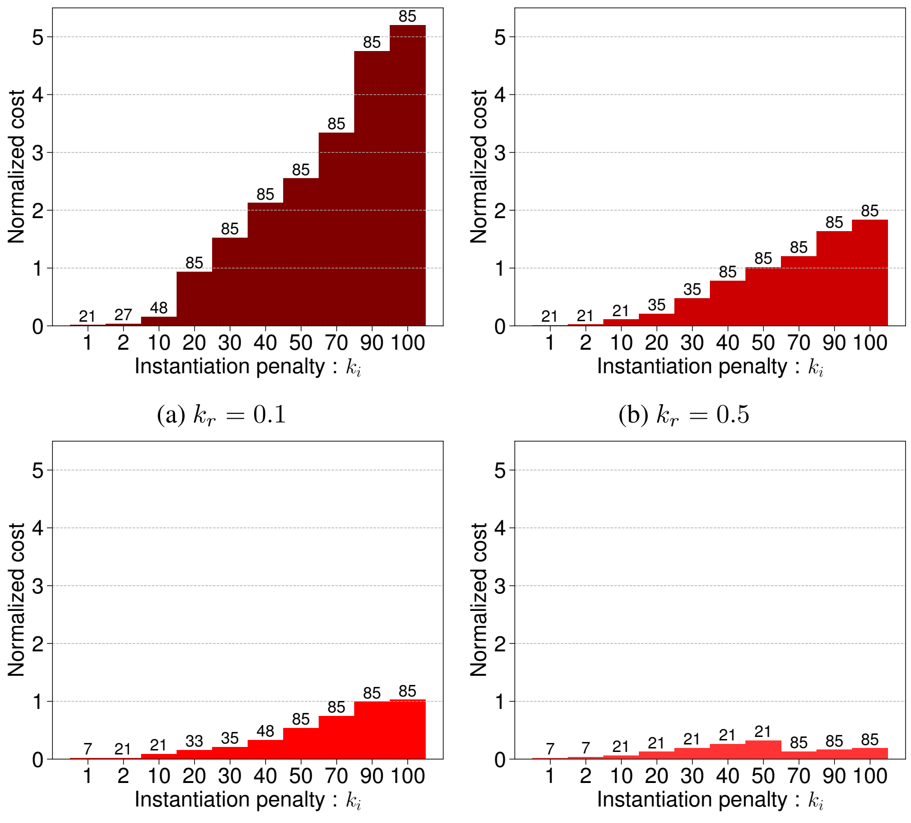


Figure 4.6: Different scenarios of cumulative probability of SLA violation. (a) No allocation of shared resources. (b) SLA minimization. (c) Maximum allocation of shared resources. (d) Overprovisioned allocation.

(ii) **SLA minimization.** If given a certain dedicated capacity $x_d^i(t)$, the probability of an SLA violation is high, then Block (III) allocates resources from the shared capacity (see Fig. 4.6b). A high probability of incurring an SLA violation is measured, and then enough resources from $x_s(t)$ are allocated to slice i to minimize the probability of incurring SLA violations.

(iii) **Max. allocation.** The optimization problem solved by Block (III) allocates shared capacity to slice i to minimize the probability of SLA violations. In a certain scenario, the allocated shared capacity is enough to minimize the probability of SLA violations based on the information obtained from Helper, but the actual real traffic value ($\lambda_i(t)$) exceeds the allocated resources $x_i^s(t)$. This situation could occur for two reasons: either the forecast provided by Helper for that timeslot was not accurate enough, or there were not enough shared capacity resources available. In the latter case, Block (III) aims to allocate as many resources as possible to minimize the SLA violation penalty.

(iv) **Overprovisioned allocation.** A last scenario may happen when Block III allocates resources from the shared capacity $x_s(t)$ that are not currently needed. In this scenario, the information provided by the Helper suggests that there is a high probability of violating the SLA by only



(a) $k_r = 0.1$ (b) $k_r = 0.5$

(c) $k_r = 1$ (d) $k_r = 10$

Figure 4.7: Instantiation normalized cost for reconfiguration cost scaling factors $k_r = 0.1$ (a), $k_r = 0.5$ (b), $k_r = 1$ (c), $k_r = 10$ (d).

allocating the dedicated capacity $x_i^d(t)$. So, Block (III) allocates shared resources to slice i with the intent of preventing SLA violations. However, upon computing the actual value $\lambda_i(t)$, the shared resources allocated were not necessarily needed as the real traffic was already below the dedicated capacity allocated by Block (I). As a result, there is an unnecessary overprovisioning of resources (see Fig. 4.6d). It is important to note that Block (III) operates only leveraging information provided by the output of Helper, Block (I), and Block (II), and not using the actual real value $\lambda_i(t)$ that is not known at the time of allocating resources.

4.5.4. Instantiation Cost Analysis

Fig. 4.7 shows the contribution of only the instantiation to the overall cost for all the different configurations (k_r 0.1- 10, k_i 1-100). We observe that for low values of reconfiguration cost, when the instantiation cost k_i increases, Block (0) selects larger extended intervals trying to minimize the instantiation cost derived by the dedicated capacity. Nevertheless, since for low reconfiguration cost, the system tends to allocate a higher amount of total shared capacity, AZTEC+ overall performance is similar to AZTEC as can be seen in Fig. 4.8a. When increasing the reconfiguration cost, AZTEC+ allocates a lower amount of total shared capacity resulting in

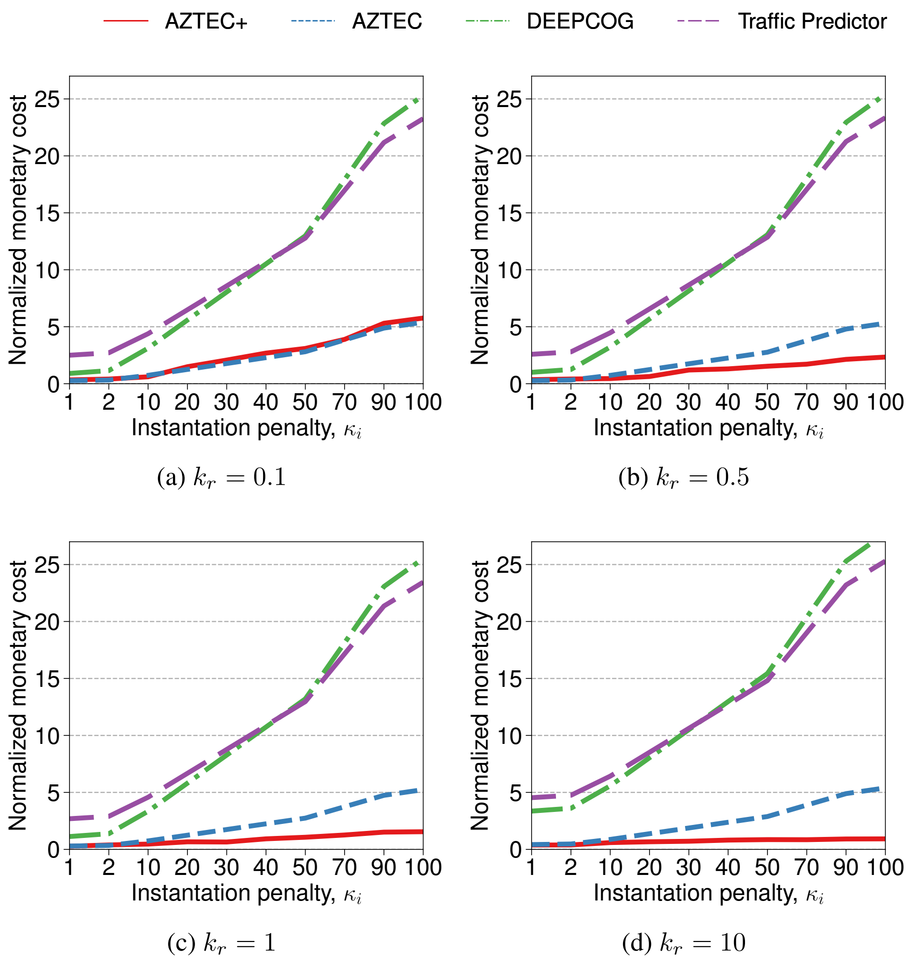
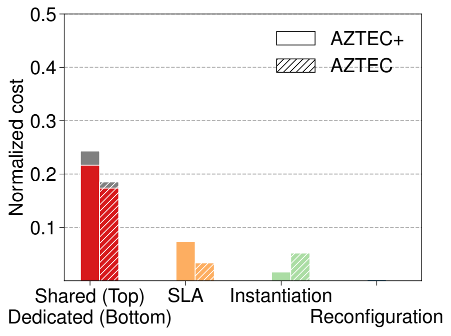


Figure 4.8: Normalized monetary cost of AZTEC+ and benchmarks versus the instantiation cost scaling factor k_i for different reconfiguration cost scaling factors.

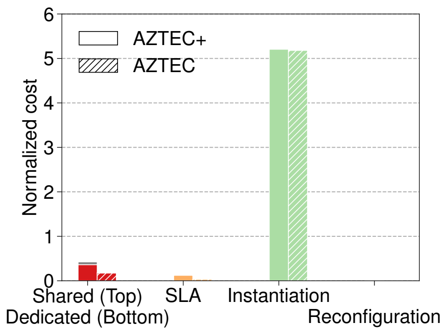
lower normalized instantiation cost mainly due to the minimization of the dedicated capacity instantiation cost. For $k_r = 10$, there is a decreasing step between $k_i = 50$ and $k_i = 70$; allocating low amount of total shared capacity, AZTEC+ is more effective in reducing the normalized instantiation cost by dynamically selecting the extended interval T_l^* that minimizes the dedicated capacity instantiation cost.

4.5.5. Monetary Cost

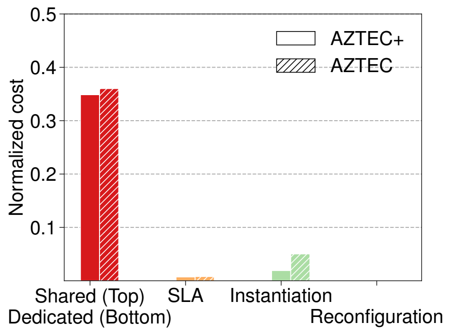
We next assess the performance of AZTEC+ against three benchmarks: Traffic Predictor, a state-of-the-art mobile net-work traffic predictor [99]; Deepcog, capacity forecasting model for network resource allocation [38] and AZTEC, as the extended and improved work [7]; all solutions are based on custom-built DNNs. Traffic Predictor is a traditional demand predictor that is agnostic of all resource management costs, whereas Deepcog makes anticipatory decisions on capacity allocation that aim exclusively at minimizing the trade-off A of overprovisioning and non-serviced demands.



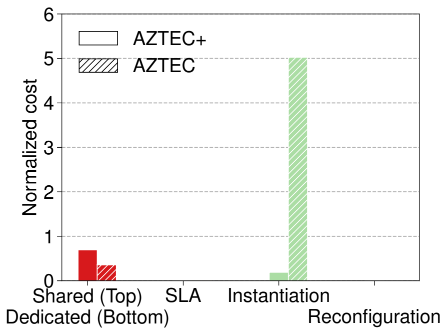
(a) $k_r = 0.1$, $k_i = 1$



(b) $k_r = 0.1$, $k_i = 100$



(c) $k_r = 10$, $k_i = 1$



(d) $k_r = 10$, $k_i = 100$

Figure 4.9: Cost breakdown for AZTEC+ (solid) and AZTEC (hatch) for the corner cases depicted in Section 4.5.4. The cost of shared capacity is shown in gray. The cost of dedicated capacity is shown in red at the bottom. The cost of SLA is depicted in orange, the cost of Instantiation is shown in green, and the cost of Reconfiguration is depicted in blue.

Fig.4.8 provides the result of the comparative evaluation among AZTEC+ and the aforementioned benchmarks, showing the overall normalized monetary cost of the anticipatory resource management against instantiation cost k_i for different reconfiguration cost k_r values. The gain of AZTEC+ compared with Deepcog and Traffic Predictor is clear as AZTEC+ outperforms both benchmarks in any configuration. Deepcog has an increment on the cost for the operator by a factor that ranges from $2.69\times$ to $4.39\times$ when the reconfiguration cost k_r is $k_r = 0.1$ (Fig. 4.8a) and ranges from $8.91\times$ to beyond $30\times$ for high reconfiguration costs (Fig. 4.8d). As expected, the benchmark solutions suffer more when reconfiguration costs grow, which they neglect. However, even in a situation favorable to reconfiguration-agnostic approaches where such costs are small (e.g., for $k_r = 0.1$), AZTEC still yields a lower economic fee. Then, we can discriminate two scenarios: Deepcog vs. Traffic Predictor and AZTEC+ vs. AZTEC. The cluster of Deepcog and Traffic Predictor remains constant regardless of reconfiguration. The capacity predictor of Deepcog avoids SLA violations by overprovisioning. At the same time, the Traffic Predictor, using MSE as a loss function, tends to forecast the traffic without discriminating if the predicted value is higher (overprovisioning) or lower (SLA violation) than the truth value. Depending on the value of k_i, the instantiation cost may become higher than the SLA cost, meaning that using Deepcog can be more expensive than using a standard loss function for forecasting.

For low reconfiguration cost (k_r = 0.1), AZTEC+ monetary cost is almost the same as AZTEC, being slightly worse (Fig. 4.8a). When the reconfiguration cost (k_r = 0.5) is increased, the normalized cost of AZTEC+ decreases substantially, using bigger extended intervals and less shared capacity. This is thanks to the dynamic selection of T_l^*, which varies depending on the instantiation cost, while for AZTEC it is set manually. From k_r = 0.5 (Fig. 4.8b), the usage of an optimization block for selecting T_l^* in order to minimize the instantiation cost, leads to outperforming AZTEC from k_i = 10 onwards, in a range of $1.62\times$ to $2.27\times$. For k_r = 1, (Fig. 4.8c) AZTEC+ outperforms AZTEC in the whole range of k_i, from $1.08\times$ to $3.38\times$. In the extreme case of k_r = 10 (Fig. 4.8d), AZTEC+ outperforms AZTEC from $1.11\times$ to $5.85\times$. The conclusion is that depending on the reconfiguration cost, we achieve a benefit in terms of monetary cost by automatically selecting the best-extended interval. AZTEC extended interval manual selection leads to sub-optimal performance, that can be improved by leveraging the novel introduced framework, that automatically and dynamically selects the window that minimizes instantiation cost.

4.5.6. Monetary Cost Breakdown

This section analyzes how each cost impacts the extreme cases (k_r=0.1, k_i=1; k_r=10, k_i=1; k_r=0.1, k_i=100; k_r=10, k_i=100). We will compare our performance with the primary benchmark, AZTEC, to depict the intrinsic contribution of the different costs. Regarding the scenario with k_r=0.1 and k_i=1 (Fig. 4.9a), it is important to note that we will encounter more overprovisioning costs and more SLA violation, but less instantiation cost. The overall cost is higher than AZTEC. This is mainly due to a larger extended interval (21 vs 6), which means AZTEC+ experiences more overprovisioning and SLA. Despite the overall cost, having a low reconfiguration cost k_r often leads to a more extensive span of shared capacity. However, the larger window also results in more overprovisioning costs. We also achieve our primary goal, minimizing the instantiation cost. When the reconfiguration cost is high (k_r=10, k_i=1, Fig. 4.9c), AZTEC+ tends to have negligible shared capacity. In this case, we have less overprovisioning cost, less SLA cost, and less instantiation cost, having a slightly bigger window size (7 vs 6). When k_r=0.1 and k_i=100 (Fig. 4.9b), we observe a considerable instantiation cost. Even with a huge window, we can observe how significant the instantiation penalty is. This is due to the cost of instantiating the total shared capacity. Since k_r is low, the system allocates a large total shared capacity, which is costly to instantiate while minimizing the cost derived from the allocation of the dedicated capacity. Still, AZTEC+ provides a similar overall monetary cost. Lastly, for k_r=10, $k_i = 100$, shown in Fig. 4.9d, AZTEC+ tends to have a lower amount of total shared capacity allocated due to a high reconfiguration cost. Thus, in this case, AZTEC+ minimizing the dedicated capacity instantiation cost results in minimal overall instantiation cost.

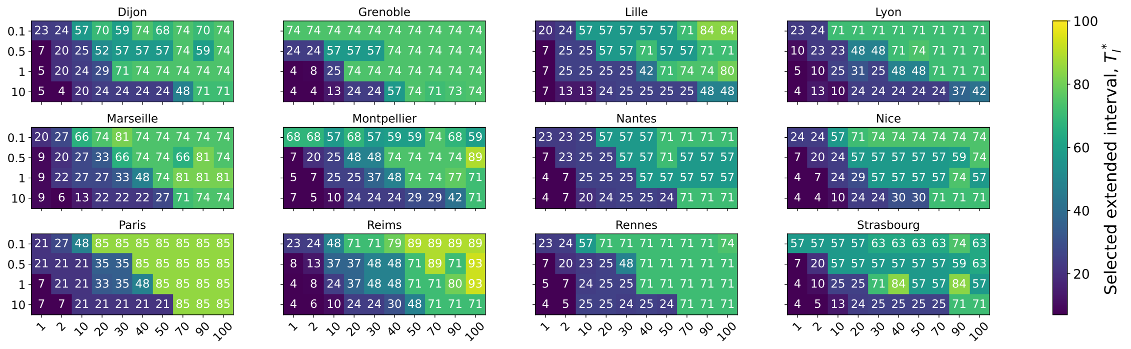


Figure 4.10: Extended Interval T_l^* across 12 cities of France for different k_r and k_i.

4.5.7. Comprehensive City Analysis

After demonstrating that AZTEC+ cost minimization and extended interval optimization effectively outperformed the existing benchmarks, we extend our investigation to a broader urban context. The goal is to validate the scalability and adaptability of our enhanced model across different cities in France. Specifically, we examine the potential variances in the Extended Interval T_l^* of our model in the following cities: Dijon, Grenoble, Lille, Lyon, Marseille, Montpellier, Nantes, Nice, Paris, Reims, Rennes, and Strasbourg. For each city, we deploy the AZTEC+ framework aiming to identify the values of T_l^* for the different k_i and k_r that minimize the overall cost.

Fig. 4.10 illustrates the selected extended interval T_l^* for various configurations of k_r and k_i for the 12 cities mentioned previously. The primary finding is that as the instantiation cost k_i increases, the model selects a longer extended interval T_l*. This is because instantiate resources becomes more expensive. As a result, the system tends to select a larger extended interval. Additionally, for any instantiation cost k_i, an increase in k_r leads to a lower selected extended interval T_l^*. In essence, we observe a trade-off between the impact of k_r and k_i. When both values are low, k_i has a greater impact on the selection of T_l^* (resulting in a larger value). If k_i remains low but k_r increases, the window T_l^* decreases. When both increase, the window continues to grow. Given the complexity of the framework and the heterogeneity of the different services' time series for each city, it is evident that some outliers deviate from the overall trend. However, it is important to note that also for these outliers selected values of T_l^* effectively minimizes the overall instantiation cost.

4.6. Summary

This chapter presented AZTEC+, a multiscale time orchestration framework that optimizes operational costs for network slicing. It focuses on minimizing costs from overprovisioning, SLA violations, resource allocation, and resource instantiation by leveraging AI-based forecasting and optimization algorithms. AZTEC+ divides network capacity into dedicated and shared resources for each slice, ensuring dynamic and efficient resource management. The chapter highlights the trade-offs involved in balancing cost and performance, emphasizing zero-touch resource orchestration. It concludes with an in-depth evaluation of AZTEC+ and its performance, demonstrating its effectiveness compared to existing benchmarks.

5 Mobile Networks: Taxonomy, Roaming, Performance and Optimization

In this chapter, we will conduct an in-depth exploration of end-user performance across various MNAs as they deliver services like web browsing and video streaming in the context of NSaaS while users are roaming. Roaming is a crucial service that facilitates the operations of MNAs in multiple countries, eliminating the necessity of seeking a local communication provider in each country where their end-users are active. MNAs have the possibility to either connect to directly connect to a local MNO or to leverage on the extensive global network infrastructure established by international carriers, such as incumbent tier-one operators like T-Mobile, Telefónica or Tata.

Our focus will be on understanding how effectively different MNAs support high-demand applications delivered as NSaaS during roaming, using performance metrics that directly impact user experience, such as DNS resolution times and traceroute paths. These metrics are vital for assessing routing efficiency and potential latency issues, both of which are critical for seamless web browsing and video streaming. By analyzing these performance aspects within the roaming context, we aim to provide insights into optimizing NSaaS offerings, particularly regarding how different network connection choices impact end-user experience when crossing borders.

5.1. Roaming for Virtual Operators

MNOs have customers, which are the end-users of mobile devices that normally attach to the MNO's network to access mobile communications services. The MNO represents the Home Mobile Network Operator (HMNO) for these end-users. The customers of an MNO can also attach to other radio networks owned by different MNOs. This happens when the HMNO does not offer radio coverage in the geographical region where the end-user wants to connect and access mobile communication services. A typical example for this is when the end-user travels abroad, in a foreign country. In this case, the end-user is *roaming*, and thus can attach to a "visited" cellular network, which the *visited MNO* operates in the foreign country. Mobile roaming is a fundamental characteristic of mobile service, which the cellular ecosystem enables through a tightly interconnected network of carriers and MNOs [54].

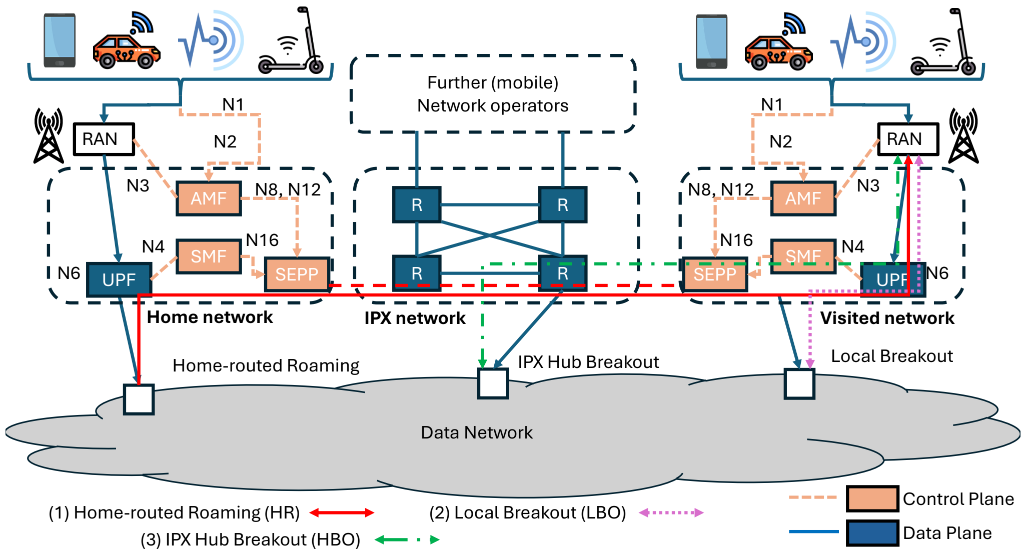


Figure 5.1: Internet access options when roaming home-routed roaming, local breakout, and IPX hub breakout roaming configurations.

5.1.1. Roaming Background

MNOs commonly connect with each other through an IP Packet Exchange (IPX) network. An IPX [54] is a hub service, to which MNOs connect over a private IP backbone network. Usually, telco carriers operate as IPX-Ps, offering the IPX service, and interconnecting in a full mesh with all the other IPX-Ps to for the IPX Network. An IPX-P has connections to multiple MNOs, and thus enables each MNO to connect to other operators via a single point of contact. The interconnections between MNOs are accompanied by roaming agreements that enable the operators to apply policies, control network access for roaming subscribers, and manage their roaming services. Figure 5.1 illustrates the three main schemes that MNOs employ for providing data roaming services, which we further describe below.

With **Home-routed Roaming (HR)** [100], the IP address of the roaming user is provided by the home network. All traffic to and from the mobile user is routed through the home network, for which a GPRS Tunneling Protocol (GTP) tunnel is set up between the User Plane Function (UPF) of the visited network and the UPF of the home network (solid red path in the figure) for the Data plane connection and Security Edge Protection Proxy Network Function (SEPP) of the visited network and the home network for the Control plane connection. With the IP end point in the home network, all services will be available in the same way as in the home network.

When **Local Breakout (LBO)** [100] is used, the IP address of the roaming user is provided by the visited network. The GTP tunnel is terminated at the UPF of the visited network and IP-based services can be accessed directly from there (purple path in the figure). This does not add latency and reduces network resource usage, but may restrict access to private services in the user's home network. Service control and charging also become more complex using LBO.

IPX Hub Breakout (IHBO) [54], or regional breakout, provides an alternative to overcome the limitations of home-routed roaming and local breakout. Here, the IP address of the roaming user is provided by the IPX network. The GTP tunnel from the UPF in the visited network terminates at a UPF in the IPX network (green path in the figure). There may be multiple UPFs so that latency and resource usage can be reduced by selecting one geographically close to the visited network. As the IPX network maintains a trusted relationship with the home network, it may assign an IP address recognized by the home network to the roaming user, thereby allowing the user access also to private services in the home network. IHBO can also simplify setup and management as a single GTP tunnel, terminated in the IPX network, can be used for roaming users from different home networks.

These configurations might have an impact on the communication performance [101]. For instance, when the node accesses services inside the *visited network*, the performance is likely to be worse in the HR case, because all packets travel twice between the visited and the home country; less so when the communication peer is in a third country and minimally when accessing services in the home country. This may also have implications in the selection of Content Delivery Network (CDN) when roaming abroad, because the mobile user will access a server in the home network rather than one close to their location.

5.1.2. Mix-and-match: MVNO and MNA Roaming

In this section, we discuss and analyze the global operations for the different types of (virtual) operators we introduced in Section 2.3.1 (see Figure 5.2). Previous work [101] shows that currently, the vast majority (if not all) MNOs use HR for roaming, assuming the associated performance penalties that it implies. As we show in the first column of Figure 5.2, this means that the MNO relies for radio access on the visited MNO, while using its own core network functions.

In the case of the MVNO, when end-users of MVNOs roam internationally, there are several options for managing their connectivity. Given that, by definition, a light MVNO relies on a single base MNO, it then follows that they also rely on the roaming agreements that the base MNO has with visited networks in the roaming location. The difference between the light MVNO and the full MVNO is that, in the latter case, since the MVNO operates their own core network, they also handle roaming independently from the base MNO (e.g., they rely on a roaming hub service from an IPX-P). Thus, the full MVNO use their own core network for the roaming solution, while the light MVNO relies on the base MNO's core network.

Regarding the MNAs, since they rely on multiple base MNOs (accross different economies), they can obtain connectivity while roaming through a local MNOs with which they have a direct agreement (i.e., the local MNO in the visited country acts as a base MNO for the MNA) or they can connect to a visited MNO that has a roaming agreement with one of the MNA's base MNO elsewhere. Depending on whether it is a full or a light MNA, it will use its own core network, or it will rely on the core network of one of the base MNOs. Notably, multi-country MNA break out in the same country or in the same region where the device is roaming.

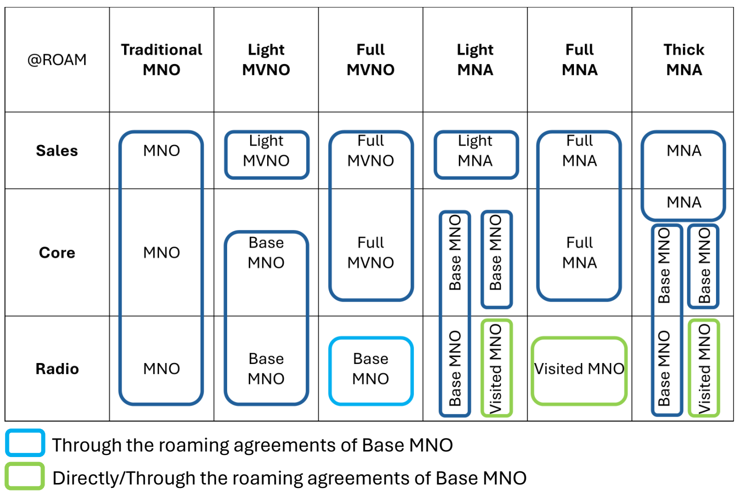


Figure 5.2: Roaming operations for the different types of mobile operators.

Thick MNAs, however, bring more complexity to the landscape: they only operate specific network functions (e.g., they operate only the pGw/UPF over third-party infrastructure) and they are able to leverage the eSIM technology as well. Providers like Airalo or HolaFly function as thick MNAs, and they partner with few base MNOs to sponsor the eSIMs that travelers can use world-wide, either in roaming or native configuration.

5.2. Experimental Setup

We perform a number of experiments to understand roaming operations for MNAs and characterize their performance. To this end, we subscribe to several MNAs in the U.S., and we perform the experiments while roaming in different countries in Europe (Spain) and the US. For Europe, we use three different MNAs, namely Google Fi, Truphone and Twilio. For the US, we also characterize the performance of Google Fi and Truphone. Also, we will run the experiments using the local MNOs that have partnership with the aforementioned MNAs. Later in the document, we will provide a detailed explanation for our choice of also using KPN (Netherlands), elucidating the reasons behind this selection within the context of our study. Fi is Google's MNA service. Fi is only available for U.S. customers. This means that we must activate any new Fi account in the U.S., and, only after that, we can use it internationally.[1] Fi only works with a limited set of mobile phones, and some features (e.g., the capability of dynamically switching between underlying MNOs) are supported only in a subset of the devices that are "designed for Fi",[2] which in most cases only include the U.S. version of the devices. Fi automatically connects to the Virtual Private Network (VPN) provided by Google. This not only provides security for the connection, but it also allows to preserve the IP address used by the mobile device when communicating (even if the

[1]Activate Google Fi service: https://support.google.com/fi/answer/6078618?co=GENIE.Platform%3DiOS&hl=en&oco=0

[2]Fi supported phones: https://fi.google.com/about/phones/.

MNO used to attach to the network varies). It is possible to disconnect the VPN service manually.

In the US, Fi uses T-mobile, Sprint and U.S Cellular as base MNOs[3]. Fi also connects to a large number (millions, as claimed by Fi) of pre-selected WiFi hotspots. While roaming, Fi claims that it has coverage is over 200 countries, but provides very little information about how they achieve this. In particular, Fi announced an agreement with MNO Three (owned by Hutchison Telecommunications) to improve performance for end-users in roaming [102]. While the selection of the base MNOs is automatic, it is possible to force the the MNOs used by Fi using dialer codes [103]. These codes allow the selection of the base MNOs; however, they do not allow the explicit selection of the visited MNOs while roaming. Changing the base MNOs may affect the visited MNOs (which depends on the roaming agreements of the base MNOs with the visited MNOs available in the visited countries. Based on the publicly available information, we classify Fi as a light MNA, according to our proposed taxonomy.

Truphone is an MNA with headquarters in the UK, and with (MVNOs) separate agreements with base MNOs in 8 countries (Australia, Germany, Hong Kong, Poland, Spain, the Netherlands, United Kingdom and the United States) and "bi-lateral roaming agreements in place with a wide range of operators around the world" (from https://en.wikipedia.org/wiki/Truphone). Truphone is a light MNA, as per the taxonomy we propose. The company has built a mobile network with core network technology distributed across four continents. Truphone uses these local Point of Presence (PoP) to reduce the distance that mobile traffic has to travel, which comes with a promise to reduce latency, and improve the end-user experience.

Twilio's Super SIM is an MNA targeting IoT devices [104]. Twilio's Super SIM can connect to 343 networks in 174 countries [105] and it uses its own mobile core that runs in the AWS cloud. As such, Twilio's Super SIM is a full MNA according to our taxonomy. However, local breakout outside the U.S. is still under development at the time of this writing. We also tested Twilio's Wireless SIM, which is an MVNOs operating on top of T-Mobile (US) as base MNOs (similar to Google Fi) [106].

To perform the measurements in Europe, we subscribed to the three aforementioned MNAs services in the US, and ran experiments while roaming in Europe (Spain). For the US measurements, we also subscribe the two most impactful MNAs while roaming in USA. For end-user equipment, we use Pixel 4A (U.S. version) mobile phones in all experiments. In addition to the MNA services, we also subscribe to a local MNOs, namely, Orange (Spain), T-Mobile (US) and AT&T (US) , in order to be able to compare the MNA roaming solution with a local breakout roaming configuration. Moreover, we also subscribed to Three (UK/AT based), in order to also consider the case of a regional breakout in Europe (based on the configuration of Fi). Finally, we subscribed to KPN (Netherlands), since our subsequent experiments in the US reveal that Truphone's e-SIM configurations are based in the Netherlands, and they have chosen KPN as base MNOs. In the case of Fi, we can use the dialer codes we described earlier to select the base MNOs, so we toggle between the different possibilities available during the measurements. Also, in the case of

[3]See the answer to "What is unique about Google Fi's network?" in the FAQ: https://fi.google.com/about/faq/.

Fi, even though the default behavior is to connect the VPN, we performed measurements with and without VPN. In Table 5.1 we include all configurations that we used for our experiments while measuring in Spain.

Table 5.1: Experimental configurations: MNAs, base MNOs and visited MNOs we used for measurements in Spain. the Breakout column includes the breakout point identified through our experiments.

Name	**MNA**	**Base MNO**	**Visited MNO**	**Comments**	**Breakout**
Fi/TM/VPN	Fi	T-Mobile (US)	Vodafone (ES)	VPN enabled	US
Fi/TM/noVPN	Fi	T-Mobile (US)	Vodafone (ES)	VPN disabled	US
Fi/3/VPN	Fi	Three	Movistar / Orange/ Vodafone/ Yoigo	VPN enabled	US
Fi/3/noVPN	Fi	Three	Movistar / Orange/ Vodafone/ Yoigo	VPN disabled	UK
Truphone	Truphone	Orange (ES)	N/A		Europe
Twilio_WS	N/A	T-Mobile (US)	Movistar	MVNO	US
Twilio_SS	Twilio	N/A	Movistar	MNA	US
Orange	N/A	Orange (ES)	N/A	Baseline (Spain)	Spain
3NET	N/A	Three	Movistar / Orange/ Vodafone/ Yoigo		UK

Table 5.2: Experimental configurations: MNAs, base MNOs and visited MNOs we used for measurements in the US. the Breakout column includes the breakout point identified through our experiments.

Name	**MNA**	**Base MNO**	**Visited MNO**	**Comments**	**Breakout**
Fi/TM/VPN	Fi	T-Mobile (US)	N/A	VPN enabled	US
Fi/TM/noVPN	Fi	T-Mobile (US)	N/A	VPN disabled	US
Truphone	Truphone	KPN (NL)	AT&T (US)		Europe
KPN	N/A	KPN(NL)	AT&T (US)		Europe
T-Mobile	N/A	T-Mobile (US)	N/A	Baseline (US)	US
AT&T	N/A	AT&T (US)	N/A	Baseline (US)	US

For each of these configurations, we run the following set of experiments:

- Traceroute: We performed a number of traceroutes to discover and characterize the paths from the end-user to reach different destinations. We selected targets in the US (i.e. home), in Spain (visited country), in Belgium (visited region) as well as content served by a CDN.

- DNS measurements: We measured the resolution time for both cached and non cached domain names.

- Web performance: We measured the Page Load Time (PLT) for web pages hosted by a server located in the US (home), Spain (visited country), Belgium (visited region) as well a served by a CDN.

- Video performance: We measured the average quality, the number of rebuffering events and the bandwidth obtained while streaming.

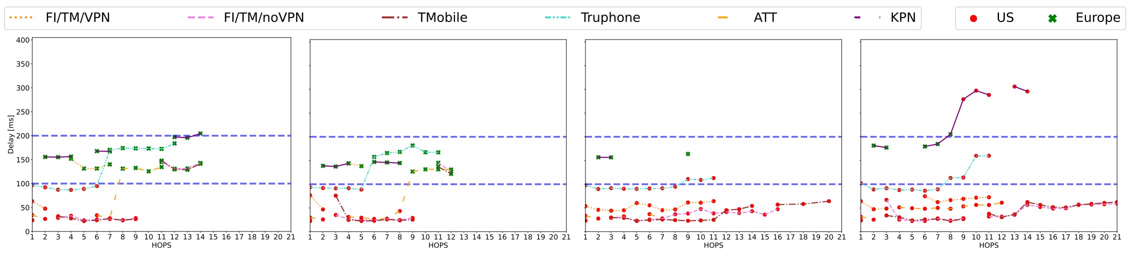
5.3. MNA Roaming Configuration

In order to learn about the MNAs' underlying infrastructure and roaming configuration, we run traceroute from end-users (roaming) in Spain towards four different targets in Europe and in the US. We repeat the measurements for each experimental end-user SIM configuration that we include in Table 5.1. We select as targets simple web pages that are not served by CDNs, and operate from servers located in different countries (namely, ucla.edu (US), uclouvain.be (Belgium) and url.edu (Spain)). In addition, we also target a web page served by a CDN (i.e., mit.edu). We performed 20 traceroute measurements for each target, and we only keep the minimum RTT value observed for each hop, as we are interested in measuring the fixed components of delay at this stage. We also run additional tests to further assess performance in Section 5.4.

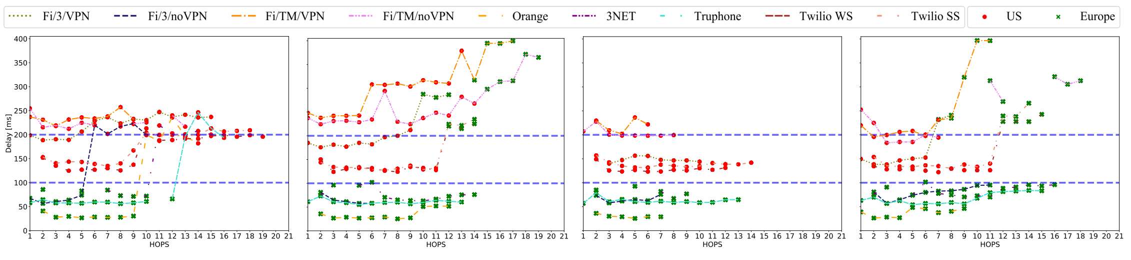
We analyze the data paths, and infer the roaming configuration these operators deploy (e.g., HR, LBO or IHBO). We infer the geolocation of each hop along the data paths that traceroute uncovers. For this, we use reverse DNS information, WHOIS information and the Maxmind geolocation database. We acknowledge the limitations of all these approaches, and mention that given the country-level geo-location we aim to achieve, the approach we use is accurate enough [107].

By extending the scope of our analysis to include traceroutes from the US, we aim to not only corroborate our previous findings but also to deepen our understanding of the infrastructure and roaming configurations specific to the US context. To ensure consistency and comparability with the previous study, we employ the same set of target destinations consisting of web pages hosted on servers in Europe and the US. This set includes umich.edu[4] as a representative target in the US, along with uclouvain.be (Belgium) and url.edu (Spain) as targets in Europe. Additionally, we consider a web page, mit.edu served by a CDN. We repeat the measurements for each experimental end-user SIM configuration described in Table 5.2. It is worth noting that the difference in Google Fi between both scenarios (Europe and US) is that in Europe we were able to connect to a regional breakout (e.g. 3 Network) and the partner T-Mobile whereas in US we only connect to T-Mobile. Similar to the previous study, we perform 20 traceroute measurements for each target from the US end-users, compute all available routing paths and record the minimum RTT values observed for each hop. The geolocation of each hop is inferred using reverse DNS information, WHOIS and Maxmind geolocation database, following the methodology established in the previous study. The limitation of these approaches will be discussed later. By conducting this comparative analysis, we aim to gain insights into the potential variations in the infrastructure and performance characteristics of MNAs when serving domestic users in the US compared to international roaming users in Europe. In the subsequent subsections, we will conduct a detailed analysis and comparison of parallel experiments. These include tracing routes from Europe to the United States and vice versa (Sec.5.3.1), within Europe, and within the U.S. (Sec.5.3.2), as well as to CDNs (Sec.5.3.3).

[4]In our previous paper, we used ucla.edu as an US domain, but at the day of performing experimentation in the US, ucla.edu runs in a CDN. Thus, we found another website with the same features that ucla.edu had in 2022



(a) US → Spain (b) US → Belgium (c) US → CDN (d) US → US



(e) Spain → US (f) Spain → Belgium (g) Spain → CDN (h) Spain → Spain

Figure 5.3: Hops encountered with the latency towards targets in US, Belgium, Spain and within a CDN.

5.3.1. Traceroute EU-US and US-EU

Figure 5.3e shows the delays for each hop replying to the traceroute probes from a mobile device (roaming) in Spain towards a server (ucla.edu) located in the US (the home country). We can observe that the overall delay to the target varies between 200 ms and 240 ms (20% variation) across the different SIM configurations (Table 5.1).

One major difference that we observe is the relative location of the transatlantic link in the path. When we measure Fi/T-Mobile (with and without VPN) or Fi/3/VPN, we find that the transatlantic link is before the first hop. The large delay value we measure in the first hop (≈200ms), and the inferred geo-location of the IP address of the first hop in the US both corroborate our deduction.

However, when using the Orange Spain subscription (i.e., the visited MNO that Fi attaches to in Spain when using Three as base MNO), we find that the first hop geolocates in Spain (the delay is ≈40 ms, and the geolocation of the IP of the first hop maps to Spain). We can easily identify the transatlantic link later in the path due to the large increase in the delay (≈200ms), and due to the fact that the IP geolocation shifts from Spain to the US.

In the case of Fi/3/noVPN we find that the first hop is within Europe (but not in Spain), and that the transatlantic jump also happens later along the path. Similarly, for Truphone, we observe that the first hop is within Europe (but not in Spain) and it behaves similarly to a local MNO (namely, similar to Orange Spain, as we observe in the Fig. 5.3e). However, Truphone has slightly higher delay values in the first hop compared to Orange Spain, since it breaks out in the Netherlands.

When using Twilio, we observe two large increases in the delay, one in the first hop (≈150ms), and another one later on (an extra of ≈100ms).

For Google Fi, we further explain the different delay values we observe as effects of two specific design factors that the MNA includes in their roaming implementation, namely the HR roaming configuration and the location of the VPN endpoint. When the VPN is enabled, the first hop in the data path is the other VPN tunnel endpoint. From the experiments, we conclude that this is located within Google's infrastructure in the US. So when the VPN is enabled, the traffic is routed to Google in the US, and then towards the Internet through Google.

When we disable the VPN, we observe the effect of HR roaming. When using Fi/T-Mobile, we note that the traffic is first routed to T-Mobile in the US, and it then exits to the Internet through T-Mobile's US network. When using Fi/Three or directly Three, we note that the traffic is first routed to Three network in Europe and then to the Internet through Three's network.

In the case of Twilio, we can explain the behavior we capture because Twilio breaks out in the East Coast (VA) [104], while our target is located on the West Coast.

For Truphone, since Truphone breaks out in Europe, the behavior we observe is very similar to the local MNO (i.e., Orange Spain).

Figure 5.3a and 5.3b show the delays for each hop replying to the traceroute probes from a mobile device (roaming) in the US towards a server (url.edu) located in Spain and (uclouvain.be) in Belgium.

We can observe that the overall delay to the target varies between 150 ms and 200 ms (20% variation) across the different SIM configurations (Table 5.2) for the Spanish server and between 120ms and 180 for the Belgium Server. We observe again the transatlantic link, in this case for all SIM configurations but not KPN. This is because KPN already has the first hop in europe, inducing a high penalty already in the first hop. The large delay value we measure in the first hop (150ms) , and the inferred geo-location of the IP address of the first hop in the Europe both corroborate our deduction. For the server located in Spain, KPN has a delay on the last hop of $\approx 200ms$ and for the belgian server, 150ms , this is due to having the server located very close to the breakout point. However, when using the T-Mobile or AT&T subscription (i.e., the visited MNOs that Fi and Truphone attaches to in the U.S,respectively), we find that the first hop geolocates in the US (the delay is $\approx$ 30 ms for T-Mobile and $\approx$ 40ms for ATT, and the geolocation of the IP of the first hop maps to US). We can easily identify the transatlantic link later in the path due to the large increase in the delay ($\approx$100ms), and due to the fact that the IP geolocation shifts from the US to Spain. When using GF/TM/noVPN, we observe that it has the same hops and almost same latencies for both servers than TMobile, indicating that has the same routing for the packets than TMobile. In the case of GF/TM/VPN, the VPN tunneling induces to a complete diferent path of hops for the packets, having a higher first hop delay (aprox 70ms) but having an approximate delay in the last hop, but having way less hops compared to GFI/TM/noVPN and T-Mobile (7 vs 14 hops for spain and 5 vs 12 hops for belgium). For Truphone, we observe that the eSIM configuration belongs to its base MNO, KPN, but it has the breakout point in the US, since it first hop is in the US. It is worth nothing that its first hop, even being in the US, is the one with higher latency compared with GFIs and the benchmarks ($\approx$ 100ms). We can also identify the transatlantic link later in the path due to the increase in the delay ($\approx$150ms).

5.3.2. Traceroute EU-EU and US-US

We next analyze the traceroute results we collect from measuring towards a server located in Belgium (uclouvain.be), which we show in Fig. 5.3f. In the previous section, the measurements target located in the US. (ucla.edu) forced all the data paths to traverse from Europe to the US. Consequently, the home routed roaming configuration of the MNAs did not translate into a significant impact in the overall experienced delay, even when using a local MNO from Spain, such as Orange. In our current setup, we do observe significant difference in the overall delays as a consequence of HR roaming, because we keep both the target and all the end-user devices within Europe.

We observe that for Orange, Truphone and Fi/3/noVPN we measure a significantly lower delay (between 50 ms and 75 ms) than the remaining SIM configurations. For both Twilio setups we measure 220 ms of total delay (200% to 300% increase compared to the above-mentioned operators). Even more, both Fi setups with VPN enabled as well as Fi/T-Mobile/noVPN have a delay ranging between 300 ms and 400 ms, which represent 500% to 700% increase compared to the values we measured for the above-mentioned group of operators. We conclude that these latter SIM configurations are impacted by circuitous routes from Spain to Belgium through the US. This hairpinning effect is either because of the VPN (for Google Fi) or because of the HR setup for roaming traffic flowing from Spain to Belgium. We observe smaller delay values for Orange, Truphone, Fi/3/noVPN because the corresponding traffic never leaves Europe, taking a much shorter path. We find that Twilio (in both configurations) also uses HR roaming, but the breakout geo-locates on the US East Coast. Thus, the penalty in terms of latency is lower than we measure for Fi/T-Mobile, which breaks out in the US West Coast.

To further analyze the impact of the content location within Europe, we further run traceroute experiments between a mobile device in Spain and a server also located in Spain. We show our result in Figure 5.3h. If we look at the total delay we captured, we find that Orange Spain gives the smallest overall delay. We observe the second-smallest overall delay when using Fi/3/noVPN and Truphone, while the other cases (Fi/T-mobile with and without VPN and Fi/3/VPN) suffer an exceedingly larger delay (with Fi/T-Mobile/VPN being the largest).

Similar to the previous cases, we can explain these results as side-effect of a combination of HR and VPN tunnelling. However, in this case, the situation is more extreme because both the mobile device and the server are located in the same visited country, so the circuitous routing through the US or through the UK injects the extra delay towards the foreign network twice (to go and to come back to Spain). We validate this by the two large leaps in delay in Figure 5.3h (one for the first hop and another one later in the path). Nevertheless, the breakout in Europe (Three, Fi/Three, Truphone) introduces a significantly lower delay than the breakout in the US (for all other Fi configurations and Twilio). In the following, we study the traceroute results we collect from measuring towards a server located in the US, which we show in Fig. 5.3d. We observe that for GFI/TM/VPN , GFI/TM/noVPN, TMobile and ATT the delay is significantly lower (between

50 ms and 75 ms) than for Truphone and KPN. It is the same behaviour that we observed in the EU-EU Traceroute, where Orange, Truphonem and Fi/3/noVPN, having the breakout in Europe, had the lowest latency compared with the other SIM configurations. For Truphone, we observe that it already starts with 100ms of delay in the first hop, and it increases until aprox 180ms. we conclude that this is happening because Truphone's breakout is in the south west coast and the measurements were taken in middle-east of US and we are acessing to umich.edu (Michigan). Finally, for KPN, we observe the same behaviour as FI/TM/noVPN, FI/TM/VPN, and Twilio setups, having a notably big first hop delay (aprox 180ms) and achieving 300ms. Again, this is the behaviour observed due to HR setup for roaming traffic.

5.3.3. Traceroute to a CDN

In this section, we discuss the traceroute results we collected when measuring towards a CDN-hosted web service (i.e., mit.edu). In this case, the CDN replica located close to the breakout point (where the end-user traffic is injected to the public Internet) provides the web content, and thus represents the target for our measurements. We illustrate our results in Figure 5.3g. As expected, we observe that the end-user experienced the shortest delay when using Orange Spain, followed by Fi/Three without VPN, Truphone and Three – all Europe-based operators. Both Twilio solutions, Fi/3/VPN, and Fi/T-mobile (with and without VPN) resulted in much higher delays. This implies that when the MNA deploys the VPN and/or the HR roaming to the USA, the end-user is accessing a content replica in the US. When using Fi/3/noVPN or Truphone, the end-user accesses a content replica in Europe (but not in Spain), resulting in a shorter overall delay. We measure the smallest overall delay when using Orange, which means that the end-user retrieves the content from a replica in Spain.

In the Measurements done in the US, we observe that ATT is the one with the shortest delay (30ms), followed by GF/TM/noVPN, Tmobile and GF/TM/VPN, accessing all of them so the closest CDN from where the measurements were taken. Next, we find Truphone that has a smiliar delay in the first and in the last hop (90-110ms), since it is accessing to the closest CDN in the west coast. Regarding KPN, we also observe the same performance than FI/TM/VPN FI/TM/noVPN and Fi/3/VPN in the prior measurements, the end-user is accessing to a replica in the Netherlands, resulting in a high overall delay.

5.3.4. Takeaways

MNOs usually deploy the HR configuration for international roaming (Section 5.1). This configuration results in an added latency penalty for the end-user, especially when the other end of the communication is located topologically close to the visited location of the roaming device [101, 108]. We observe a similar behavior for the three MNAs we measured. The difference, however, comes from the capability of the MNA to change their base MNO (nationally and internationally), thus implicitly also changing their "home" operator.

For the delay measurements, we used as a baseline the delay we observe when using Orange Spain subscription within Spain. This is the delay the end-user would experience when directly attaching to a local network in the visited country.

Google Fi: When T-Mobile (without VPN) is the base MNO, Fi home-routes the traffic back to the US, resulting in large penalties for end-users roaming in Spain. This is especially noticeable when accessing content available locally in Spain/Europe. Even in the case when we target content served by a CDN, we still observe a large delay penalty. The only case when no significant delay is added is when the end-user targets content only available in the US. However, using T-Mobile as base MNO is *not* the default policy Fi has for end-users roaming in Europe. When in Europe, Fi switches to using Three as a base MNO. When using Fi/Three without VPN, the latency penalty (albeit still existent in some cases) drops. This is because, even with HR roaming, the distance between the visited location and the new "home" location (i.e., U.K.) is smaller compared to relying on T-Mobile. This approach is a middle-ground between LBO and HR, and significantly reduces latency with a small overhead in terms of roaming agreements (only one extra agreement for a whole region). We model this configuration as a "regional breakout" model, similar to the IPX breakout model.

However, the benefits the "regional breakout" bring are lost when the Fi end-user enables the VPN service (which is the default behavior for Fi), because the other endpoint of the tunnel is located in the US.

When measuring from the US, we observe a very similar performance with its base MNO, namely, T-Mobile, in both configurations (i.e. with and without VPN). As it is a service that you can only subscribe within the US, it makes sense that they have a better performance in the US in all configurations than in Europe if the packets should be routed to the HR. We observe that having the aforementioned Regional Breakout, the first hop has 4 times longer delay than doing HR (20 ms $GFI/TM/noVPN_{US}$ vs 80 ms delay $GFI/Three/noVPN_{Europe}$) for all the accessed domains. On the contrary, accessing to a domain within Europe, the last hop using regional breakout decreases dramatically the delay (127 ms $GFI/TM/noVPN_{US}$ vs 58 ms delay $GFI/Three/noVPN_{Europe}$) demonstrating the huge impact that and advantages that Regional Breakout has for users in roaming.

Truphone: Overall, we observe that Truphone provides a delay experience slightly higher than that of the local MNO in the visited country (i.e., Orange Spain) as it breaks out in Netherlands. From the measurements performed in the US, we find that the breakout point is in the west-coast of the US, having a worse performance than GFI but still improving its base MNO, namely, KPN, that is using HR. Also, Truphone was worse performance than ATT, its visited MNO. We thus confirm that Truphone uses their POPs to implement the Regional Breakout approach to reduce the distance that mobile traffic has to travel. We observe that this indeed reduces the delay.

Twilio: In the case of Twilio, both configurations (i.e., Super SIM and Wireless SIM) rely on HR roaming back to the USA with the corresponding penalties in terms of delay.[5] However,

[5]Some resources within the Twilio (i.e., the Super SIM API and Internet breakouts outside of the United States,

because Twilio breaks out in the US East Coast, the penalties are reduced in our case that the device is roaming in Europe. Devices roaming in the Asian Pacific region for instance, should observe the opposite effect.

5.4. MNA Performance

Our traceroute results confirmed that MNOs implement "regional breakout" to reduce the distance that mobile traffic has to travel compared to the HR roaming configuration. However, it is unclear how this approach further improves the end-user experience for popular applications and services. In this section, we evaluate the impact of the MNA roaming configuration on DNS resolution delay, web browsing performance and video streaming quality.

5.4.1. DNS Resolution Delay

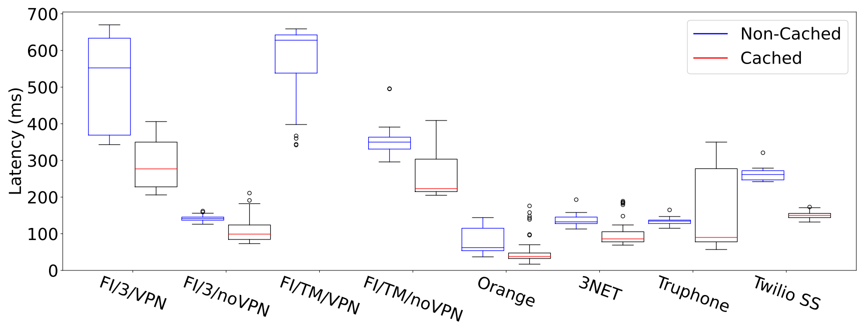
We measure the DNS resolution delay using the different experimental configurations in Table 5.1 for Europe and Table 5.2 for the US to compare their performance. We first measure the resolution delay for a non-cached name. In order to do that, we set our own authoritative domain, and we configure it with a wildcard DNS record, so it responds to all names under that domain. We then perform queries for unique names under our domain, ensuring that the name queried is never repeated, ensuring that caching is not involved in the resolution. One of the authoritative servers is located in Spain, serving the worst case for roaming devices as we inferred from the traceroute experiments (see Figure 5.3). For the USA measurements, we also configured a wildcard DNS resolver using Amazon Web Services (AWS), specifically on the Amazon S3 platform located in the US. We did 20 queries for each experimental configuration for the end-user SIM.

In Figure 5.4a, we show the delays we measure while querying a non-cached and cached domain names. We note that two clusters of operators emerge while querying non-cached domain names. First one includes both VPN-enabled Fi configurations (namely, Fi/TM/VPN and Fi/3/VPN), which exhibit the largest DNS resolution delay, followed by Fi over T-Mobile without VPN (Fi/TM/noVPN), and the two Twilios SIMs.[6] We observe that a second cluster of operators includes Three, Fi over Three (without VPN), Truphone and, finally, the (native) Orange with the smallest delay. We obtained similar results to Orange Spain with other local MNOs (e.g., Vodafone Spain), which, again, we do not include in Figure 5.4 to improve readability. These operators show much smaller resolution delay, comparable to the one a use connecting via a local operator in the visited country might experience.

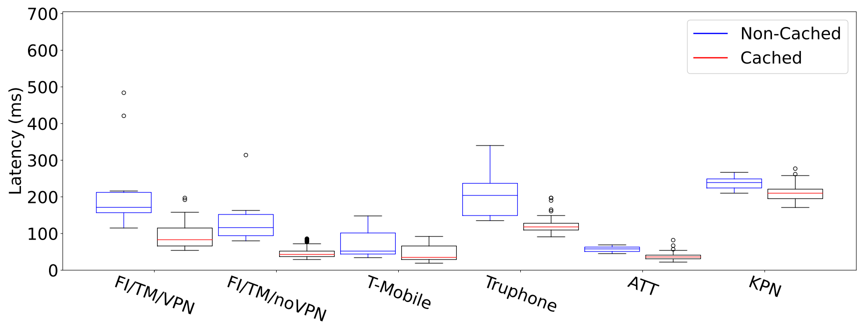
Overall, we find that both VPN-enabled Fi configurations experience a surprisingly long delay (i.e., the mean delay is over 550 ms) and also a high variance. Fi over T-Mobile without VPN

such as Frankfurt, Germany) are still in Pilot or Beta at the time we performed our measurements in July-August 2021. Thus, our results are consistent with the Twilio configuration for Super SIM, where traffic breaks out to the Internet via the Twilio Mobile Core in Ashburn, US.

[6]Figure 5.4 depicts only one Twilio configuration (Twilio_WS) to avoid cluttering the plot; however, we found very similar results for both Twilio solutions for connectivity.



(a)



(b)

Figure 5.4: DNS resolution delays we measured for a non-cached and a cached DNS name in (a) Spain and (b) US, using different experimental configurations for the end-user SIM (see Table 5.1) and Table 5.2.

follows, with a mean delay of over 350ms. Then, we find Twilio with a mean delay close to 250ms.

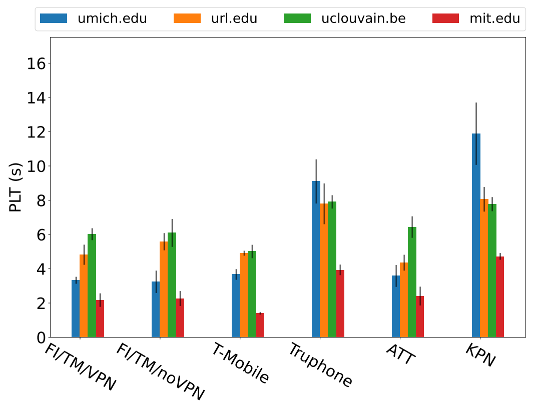
The two remaining configurations (Fi/3/noVPN and Truphone) have a mean delay close to 150 ms, while for the native Orange Spain configuration we obtain a mean close to 50 ms. Based on this, we conclude that the HR Roaming configuration which routes the traffic back to the USA imposes a penalty of 200-500ms compared to LBO roaming configuration. This accounts for a penalty that can vary between 300% and 1,000%. In the same time, we find that the regional breakout MNA leverage (e.g., here in the case of Fi using Three as base MNO and no VPN) only bring an extra 100 ms of delay, which accounts for 200% penalty. In particular, Truphone delivers best on the promise of offering optimal (i.e., close to using a local operator in the visited country) global experience to the end-user. Even when measuring with an US Truphone subscription in Spain, we note that the Mobile Country Code (MCC) / Mobile Network Code (MNC) (MCCMNC) changes to the one of the local operator Truphone registered in Spain.

In Fig.5.4b we show the delays for the US measurements, for the resolution delay for a non-cached name, we observe that ATT and T-Mobile, as expected, have the mean lowest resolution delay, i.e. 50 ms. Then, we find that FI/TM/VPN and FI/TM/noVPN have a similar behaviour, with some variance, but with a mean delay of 170 ms and 120 ms, respectively. As observed in Figure 5.3, GFI/TM/VPN has some delay incurred by the VPN tunneling, producing higher resolution delays. Next, we find Truphone, that has a higher resolution delay mean with some variance, i.e. 200 ms +- 50ms. This is due to have the breakout point in the west coast as commented in (see Sec. 5.3). Finally, the remaining configuration, namely, KPN, has a mean value of 240 ms. We see again how the HR roaming imposes a penalty of 100-200 ms compared to LBO roaming configuration. For this exoeriment, we can compare the performance of KPN (EU MNO accessing to US domain from the US) with FI/3/VPN, FI/TM/VPN, FI/TM/noVPN and Twilio's (US services accessing to Spanish domain from the US), and we observe that KPN has a dramatically smaller DNS resolution delay for the Non-cached domains.

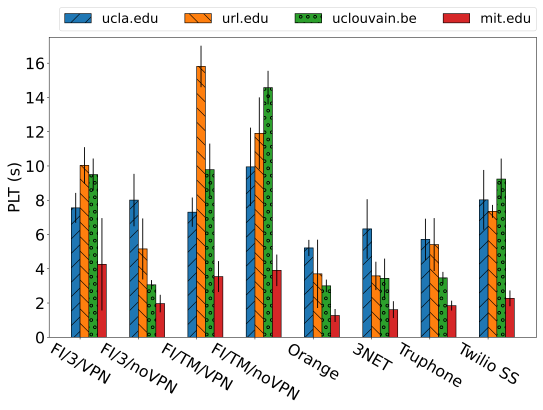
We next measure the delay for the resolution for a query that is present in the resolver's cache. To ensure this, we make a first query for a domain name in order to populate the resolver's cache. We next clear the DNS client cache (to force the client to query the resolver again), and query for the same domain name to measure the DNS resolution time. We repeat this 20 times for three popular domains (namely, www.amazon.com, www.facebook.com and www.youtube.com).

In Figure 5.4a, we compare the DNS resolution delay for a cached name with that of non-cached ones for Spain measurements. Overall, we find similar relative performance across the different configurations to that we observed in the case of querying a non-cached domain. The absolute values, however, as well as the absolute differences are smaller than in the case of the DNS queries for the non-cached domains. Indeed, for both configurations of Fi over T-Mobile and for Fi/Three with VPN, the delays are in the order of 250 ms. This is followed by Twilio with 150ms mean delay, while for Fi/Three without VPN and Truphone the delays are in order of 100ms. We observe that for Orange the DNS resolution delay drops to 50ms in median. This means that the penalty for using Home Routing back to the US is 300% - 400%, while for the regional breakout the penalty is 100% compared to the potential local breakout..

In Figure 5.4b, we compare the DNS resolution delay for a cached name with that of non-cached ones for Spain measurements. Again, we follow the same procedure as in the measurements taken in Spain, doing a first query for a domain name to populate the resolver's cache, cleaning the DNS client cache and query for the same domain name to measure the DNS resolution time. Overall, the performance we observe is relatively similar to the both (cached- non cached measurements in Spain) and non-cached in the US, that in absolute values are smaller. For T-Mobile and ATT, the mean resolution delays are in the order of 40 ms. These are followed by the both FI Configurations (noVPN / VPN) with around 60 ms and 80 ms, respectively. For Truphone has a delay of 110 ms and KPN of 200ms. Again, in the measurements in the US the overall resolution delay was lower, because the Regional Breakout was in the same country. What we observe is that, for Truphone, the geolocation of the POP plays a role in the latency of the packets and resolution delay, even being in different countries in the case of Europe, it has a lower resolution delay compared to the one in the US, having twice the distance to the breakout point. (ES-NL, 1400km. US west cpast, US middle-west, 3000 km).



(a)



(b)

Figure 5.5: PLT measured towards web pages in US (ucla.edu), Belgium (uclouvain.be), Spain (url.edu) and within a CDN (mit.edu) from a mobile device roaming in (a) Spain and (b) US with different SIMs (see Table 5.1 and Table 5.2. For the web pages located in US, in (a), we use ucla.edu and in (b), we use umich.edu.

5.4.2. Web Performance

While DNS resolution plays a crucial part in the overall performance of the service MNAs offer to their end-users, it might not necessarily translate into a significant impact on the end-user QoE. In order to further capture how the different MNA solutions actually deliver on end-user experience, we measure the impact of the delay introduced by the different experimental configurations on web browsing. We use the PLT metric to characterize web performance, which captures the time it takes for a webpage to load. We extract the PLT from the Navigation Timing API [109], available in native android web browser (Google Chrome). We calculate the PLT from initiation (the `LoadEventStart`) to completion (the `NavigationStart`), when the page is fully loaded in the browser. Essentially, this is the time it takes for the last object in the page to download. It occurs when all the HTML files and any sub-resources (images, fonts, css, videos, etc.) are loaded. Note that not all these elements are actually required to complete the rendering of the visible portion of the page. Though many other metrics focus on different aspects of webpage performance, recent studies [110] showed that PLT is good enough to capture the experience of the users in various radio contexts, showing strong correlation to other QoE metrics such as First Paint or Speed Index. Thus, in this paper we focus on the PLT, which we use as a proxy for end-user QoE.

We measure the PLT using the different configurations for the end-user SIM (see Table 5.1 and Table 5.2), while targeting different web pages. We selected web pages of roughly the same size (with a 5% range) in different geographic locations For the measurements done in Spain, we select www.ucla.edu in the US, www.uclouvain.be in Belgium, www.url.edu in Spain and www.mit.edu which is served by a CDN. For the measurements done in the US, we use the same web pages, but changing the one in the US because at the day of the extension, www.ucla.edu has moved to a

CDN. Thus, we select www.umich.edu as web page in the US for the experiments in the US. For each end-user SIM configuration and for each web page, we collect 20 different measurements. We show our results in Figure 5.5.

We observe similar trends to the ones we obtained in the case of the DNS resolution delay (Figure 5.4) and traceroute (Figure 5.3). As before, the penalties are much larger when the web server is in Spain, followed (closely) by the case of the web server hosted in Belgium , and, finally, the server in the US, where the overall penalty is significantly smaller. Regarding the content served by the CDN, the overall PLT is smaller in all the cases, but the relative differences remain similar to the previous cases. In all cases, the smallest PLT is achieved with the native service offered by Orange in Spain. We then observe the cluster of MNAs which delivers the closest experience to that of the native operator, namely Fi over Three without VPN (Fi/3/noVPN) and Truphone. Next, we have Twilio and, finally, with the highest PLT, we see both VPN-enabled configuration for Fi (Fi/3/VPN, Fi/TM/VPN), as well as Fi over T-Mobile with no VPN (Fi/TM/noVPN). We would like to highlight that serving content through a CDN – while it reduces the delay for all configurations – does not eliminates the differences in the PLT for the different configuration. This is so because, even when a CDN is used, the content is retrieved from the replica that is closest to Internet access point associated to the end-user (i.e., the breakout point). When HR roaming or VPN are used, the mobile accesses to a replica in the U.S.A. while when using Three, and Fi/Three, the mobile connects through U.K. and access to a replica in Europe while connective native, the mobile connects to a replica that is likely to be in Spain.

For the US measurements, we observe analogeous trends from the ones obtained in Spain. In the context of this experiment, the US functioned as both the home and visited country for Google Fi and the local MNOs. However Truphone had a different home and visited country, which might have contributed to its comparatively larger latency. In this case, the penalties when accessing to a web server in Belgium are slightly large than the ones for Spain and followed by the server in the US, where the global penalty observed is substantially lower. Again, for the CDN web server, the overall PLT is the smallest in all cases, and the configuration with the least one is T-Mobile. We observe a similar experience when using GFI wtih and without VPN to T-Mobile, even having a marginally better experience when accessing to the web server located in the US (i.e. 0.5s faster). This is something that will not change the QoE of an end-user. When we use Truphone, we observe the largest PLT of the MNA configurations, having an even larger PLT for accessing the web server located in the US than the ones located in Europe. When roaming in the US (KPN configuration) we observe the clear effect of HR roaming: When accessing to the US server, the penalty obtained is much larger than the one in Europe, having a slightly difference between Belgium and Spain (being Belgium smaller because it is closer to the Breakout point).

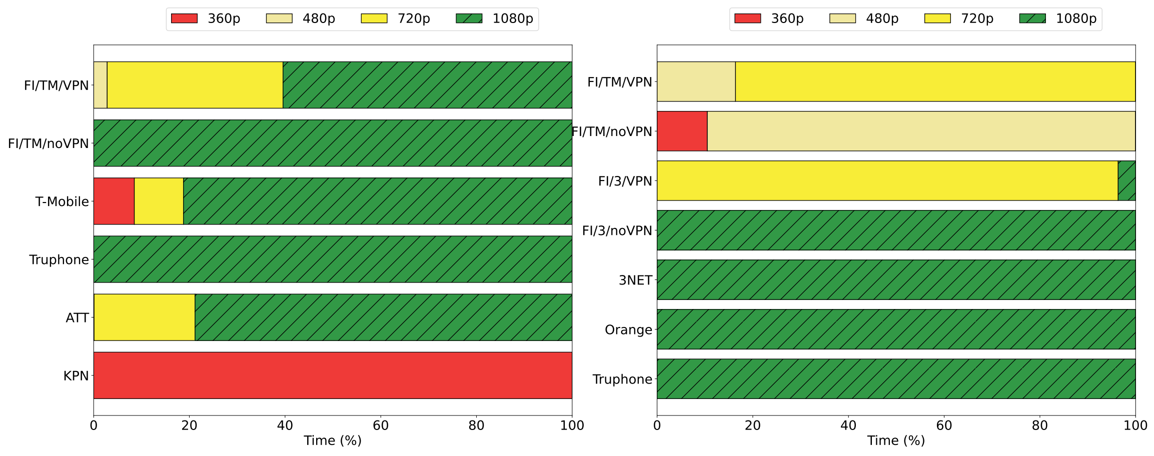
5.4.3. Video Performance

Though web browsing represents a critical service in the mobile Internet ecosystem, video traffic accounts for the largest proportion. In this section, we thus investigate how the different

MNA configurations in roaming impact the video delivery performance. We run active end-user measurements using the YoMoApp [111]. YoMoApp is developed by Wurzburg University that allows to rate the stream quality of YouTube videos, as well as obtain different metrics from the network. Once the measurement completes, it uploads to yomoapp.de/dashboard/, from where it is possible to retrieve the results. The tool allows us to measure the video quality, download throughput, stalling events, and also capture the radio access technology and buffer level.

We measure the video performance using the different configurations in Table 5.1, except for Twilio. We omitted Twilio from these tests because their products are oriented to IoT. Though they do offer video streaming APIs, the amount of traffic we needed to perform the video tests proved to be prohibitive.

We played short videos (2-3 min) from YouTube. We used three different videos, a sports video[7], a music video[8] and a movie trailer[9], to be representative of the different types of content available. We measured each video/configuration combination 20 times. We next present and analyze the results of the measurements obtained when the MNA end-user is roaming in Spain. In Figure 5.6b we show the time spent in each quality for the different configurations for the music video. We mention that we obtained similar distribution for the other types of videos.



(a) (b)

Figure 5.6: Video performance for each configuration in (a) USA and (b) Spain.

We can observe that for Three UK, Fi with Three (without VPN), Truphone and Orange, the videos render in the highest resolution (1080p) – while in Fi over Three with VPN and for Fi over T-Mobile with VPN, the most common quality is 720p. Furthermore, for Fi with T-Mobile without VPN the most common quality is 480p, dropping to 360p 10% of the times.

Moreover, we look into the initial delay and the buffer levels in the video traces (these results are not presented here due to space limitations). We observe that Fi over Three with VPN and Fi

[7]https://www.youtube.com/watch?v=JEoubjE2PBQ
[8]https://www.youtube.com/watch?v=l6N-Yq9Fw4U
[9]https://www.youtube.com/watch?v=0WVDKZJkGlY

over T-Mobile with and without VPN have introduced significant initial delay (highest initial delay is experienced by Fi over T-Mobile without VPN) compared to the other setups. We also observe that these three setups experience much lower buffer levels compared to the other setups, indicating inefficient use of buffering mechanism. We conclude that the long link delays experienced with these three setups, impact the ABR mechanisms in a negative way, leading to a much lower video quality experience compared to the other setups.

We again confirmed the result of the previous experiment. We observe that for Truphone, and Google Fi without VPN, the videos are mostly rendered in the highest resolution, and for Google Fi with VPN, the most common resolution is 720p. The good performance of Truhpone counter the previous results from DNS and web experiments, where Truphone was the worse configuration. It proves that depending on the use case, the QoE can change. TMobile and ATT are outperformed by Truphone and Google Fi without VPN. These findings contradict the expectation in the sense that MNOs should have better performance than MNAs who leverage their infrastructures to provide services, as is confirmed in the web experiments. We attribute this anomaly to ISP-level throttling. Previous research has uncovered instances of US ISPs throttling multimedia streaming services. To investigate this, we utilized WeHe [112], a VPN-based traffic differentiation detection tool. Our test results revealed that T-Mobile and AT&T were indeed throttling YouTube video playback, while Truphone and Google Fi showed no signs of such behavior. In conclusion, the exceptional performance of Truphone and Google Fi suggests that US MNOs can indeed provide superior services to subscribers. Given that Google Fi mainly relies on T-Mobile and Truphone primarily depends on AT&T in the area where we conducted our experiments, the performance discrepancy between these MNAs and their partner MNOs indicates that the MNO's does not classify the traffic generated by all the MNAs in our study.

5.4.4. Takeaways

We find that the HR roaming configuration all MNAs deploy impacts in a similar way critical services for the end-user, namely DNS resolution, web browsing and video streaming. However, by switching the "home" (i.e., the base MNO) closer to the end-user visited country, some MNAs such as Google Fi and Truphone succeed in reducing the penalty that home routed roaming introduces.

Google Fi: The use of regional breakout in Europe on top of Three's network helps Fi to reduce significantly the delay their users experience in Europe. For DNS resolution, we find that using Three as base MNO reduced the delay penalty to 200% from 300%-1000% compared to using a local native operator. In terms of further impact on application performance, we find that Fi/3/noVPN allowed the end-user to stream videos in the higher resolution (1080p), and also provided the closest web browsing performance to that a local MNO would provide. When operating in the US, we observe a very similar performance to its MNO, namely T-Mobile, having the big difference when using video application, where Google Fi was able to outperform T-Mobile.

Twilio: With the SuperSIM solution still in a very early roll-out phase at the time of our measurements campaign (July-August 2021), we find that the Twilio end-user is still impacted significantly by the HR roaming setup. Given that their breakout point is closer to the visited location (Spain) than the ones for Fi over T-Mobile or Fi with VPN active, we find a slightly better performance for this MNA.

Truphone: Leveraging their mature setup with different PoPs, we find that Truphone is able to deliver the performance closest to the one provided by a local MNO in a visited country in Europe. This is true for all the different services and applications we tested. From the US perspective, we observed that Truphone was not able to achieve the same performance than in Spain. This is due to the PoP in the US, located in the middle-west and the domain analyzed located in the west coast. Again, for the Video measurements, it outperformed its benchmark MNO, namely AT&T, due to ISP-level Throttling on the MNO side.

As a conclustion, our experiments yielded insightful results concerning the performance of the analysed MNAs and MNOs. This demonstrates the potential of MNAs to offer robust and efficient services that are on par with or even surpass those of established MNOs in certain aspects. In terms of web performance, our study revealed some interesting trends. The local MNOs often outperformed the MNAs, with the latter exhibiting slightly longer page load times (PLT). However, it is noteworthy that the differences in PLT were not so substantial as to impact the end-user experience severely. This suggests that MNAs can deliver satisfactory web performance despite operating on top of existing MNO infrastructures. The video performance tests provided some unexpected results. Despite the MNAs leveraging the infrastructure of MNOs, Truphone and Google Fi without VPN demonstrated superior video performance in certain scenarios. This could potentially be attributed to ISP-level throttling practices by certain MNOs. These findings underscore the importance of examining network performance under diverse real-world conditions, as well as the need for more transparent network management policies. While MNOs often exhibited superior performance metrics in our tests, MNAs proved to be robust alternatives that can offer competitive services. This reveals the potential of MNAs to disrupt the traditional mobile network market by offering flexible and efficient services.

5.5. Regional Breakout in 5G

The split of user/data plane from the control plane at both the radio front [113, 114] and the core [115] is one of the major upgrades in 5G. This paves the way not just for better management of data and control packets, but also to enable truly global operations of an MNO. We argue that, with this approach, any MNO can potentially convert into a global provider, avoiding HR roaming configuration, and enabling the end-user to achieve a good experience potentially world-wide through regional breakout. We envision a scenario where the MNO provides the local breakout solution to each end-user. In this section, we provide a proof-of-concept implementation and evaluate how the separation at the mobile core can support global operations.

Table 5.3: Locations of deployed UPF and DNN names as identified by the UEs to breakout at a visited location (namely, US or Germany).

	UK		US				EU (Germany)	
BO type	Home (London)	Edge (Vodafone)	Edge (Verizon)	Local (Las Vegas)	Regional (Oregon)	National (Ohio)	Edge (Vodafone)	Regional (Frankfurt)
DNN	edge.london	home	edge.sfo	local.las	regional.or	national.oh	edge.ber	regional.fra

Table 5.4: Ping latency (in ms) to the instances deployed across various zones in the AWS from an instance located at same zone as Edge.

Edge	Local	Regional	National	Home
0.22 ± 0.03	26.2 ± 0.08	49.8 ± 0.1	78.6 ± 1.5	157.3 ± 0.2

5.5.1. Experimental setup

We utilise the edge (wavelength), local and regional deployments of Amazon global infrastructure [116] to deploy control and user plane functions of open-source 5G implementations (namely, Open5GS [117] and UERANSIM [118]).

Infrastructure. The setup we build aims to emulate different roaming configurations (see Section 5.1). For this, we rely on the global infrastructure including both storage and compute services that AWS offers. This includes:

- a *regional* infrastructure with data centers present in a region (e.g., US East/Ohio region). Within this deployment, a cluster of isolated and physically separate data centers are found in a geographical area.
- a *local* infrastructure hosted as an extension of regional infrastructure to run latency sensitive and high bandwidth applications. For example, Netflix uses AWS local zone deployments for their content creation process.
- an *edge* infrastructure hosted within telecommunications providers' data centers and connected to the operator's 5G network. We consider this as first point an user can breakout to the Internet from MNOs network.

Connectivity. For our pilot deployment, we assume an end-user with 5G connectivity who has their network home location in the UK. To emulate the user roaming behavior, we test two different scenarios, where the user roams in two locations: (*i*) in the US (San Francisco) and (*ii*) within Europe (Berlin, Germany). With current 4G/LTE technologies, the default roaming configuration would be HR roaming, where the traffic is routed back to the UK. We argue that, by using 5G Control and User Plane Seperation at the core, we can keep the control plane functions in the trusted, centralized home network location, while dynamically moving the user plane function with the roaming user. We handle this connectivity by deploying control and user planes built using Open5GS [117]. We deploy the control plane, which includes SMF and AMF, at the regional infrastructure in the user's home location (London, UK). The user plane enables breakout to the internet, and hence we deploy it across multiple locations in the US and EU (as

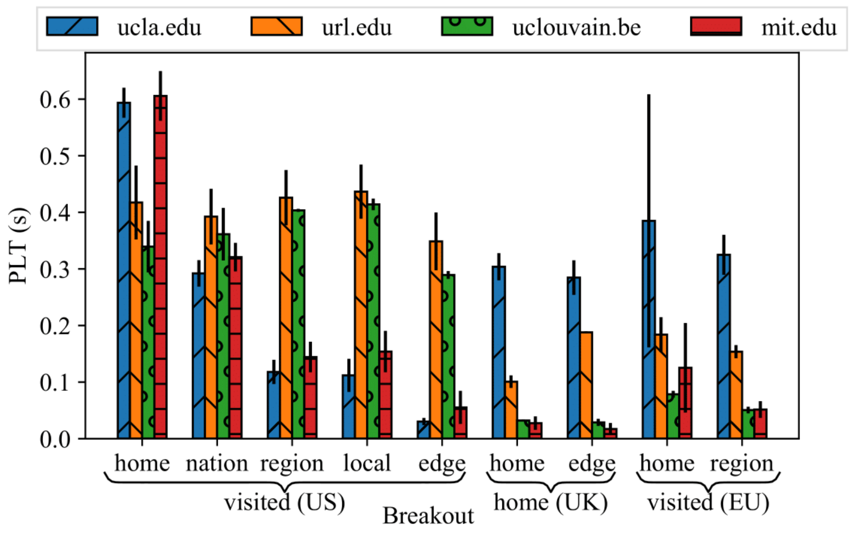


Figure 5.7: Page load time to different websites in function of the breakout point considered for the roaming configuration.

per Table 5.3). The selection of UPF to breakout is chosen by the Deep Neural Network (DNN) setting in the 5G UE. We use UERANSIM [118] to deploy a simulated environment of 5G RAN and 5G UE in the edge infrastructure.

We leverage the different existent AWS architecture to place the UPF at various locations in relation to the roaming end-user. Namely, we emulate the case of *edge breakout* by placing the function in the AWS wavelength deployment within the carrier infrastructure. We use one such existing Wavelength Zone in the US (Verizon central office in San Franciso), corresponding to breaking out the visited MNO, very closely to the location of the end-user. We then migrate the UPF further from the visited location (but still within the visited country) in each of the roaming scenarios, as we show in Table 5.3.

The UE and the migrated UPF instance across the Amazon EC2 zones in the US, UK and EU are connected via a private backbone network [119] and through virtual private cloud peering connections [120]. We show ping latency within the instances in the US and to UK (home) in Table 5.4.

5.5.2. Pilot Results

In order to capture the end-user performance under the scenarios we include in Table 5.3, we measure web performance. We focus on the PLT as the representative metrics, similar to our approach in Section 5.4. We use as target the same four website as previously in Section 5.4.2 (namely, www.ucla.edu in the US, www.uclouvain.be in Belgium, www.url.edu in Spain and www.mit.edu which is served by a CDN). We show our results in Figure 5.7. We use as a baseline the measurements for the non-roaming scenario (marked “home (UK)” in Figure 5.7). The goal of our measurements analysis is to establish which roaming configuration can offer the same performance that the end-use enjoys while at home. We find than when content is served by a CDN, the regional breakout configuration offers comparable performance to the no-roaming scenario, regardless of the location where the end-user travels (e.g., US or Germany, in our case). This is a direct consequence of the close location of the content replica to the end user, which is dictated by the location of the breakout point. From the case of end-user roaming in the US, we see that the local breakout configuration yields similar performance to regional breakout, likely as a results of the small distance between the locations of the infrastructures used in this scenario.

When a CDN is not serving the web content, the distance between the location of the end-user breakout point and the content location impacts the web performance. For example, if edge breakout in San Francisco offers the optimal performance for accessing content hosted in California (ucla.edu), we see this is no longer the case when accessing content hosted in Europe (uclouvain.be, url.edu). The PLT we measure in this latter case is, in fact, similar to the one we measure under the HR roaming configuration. The same is true for accessing US-based content from Germany, under the regional breakout configuration. Surprisingly, however, we find that the PLT for Europe-hosted web content is slightly smaller in the US (San Francisco) edge breakout scenario than all of the other configurations (local/regional/national/home breakout). We conjecture that this is a side-effect of relying on the carrier's infrastructure (i.e., Verizon), while for the other configurations the AWS private backbone impacts the delays between the various instances (see Table 5.4).

The results we present here invite more broader discussions on the role of cloud service providers in supporting cellular connectivity providers. Given the freedom that the 5G allows in terms of dynamically locating the UPF for a roaming user, we envision this as a first step towards an adaptive approach for managing the end-user cellular connectivity.

5.6. Applicability

There are a growing number of use cases that heavily rely on roaming, enabled by both the surge in demand for global deployment of IoT devices (e.g., smart meter, connected cars) with pre-provisioned connectivity [17], and in the demand for global seamless connectivity from digital nomads. A new breed of global operators aims to directly respond to the needs of such users (e.g., Twilio for IoT and Fi & Truphone for digital nomads) through specialized applications such as web browsing and video streaming NSaaS, offering tailored solutions that cater to the unique requirements of each application.

Our work shows that light MNAs such as Fi still have to mitigate the impact of latency, and consider regional breakout solutions. Furthermore, MNAs that deploy VPN features should also consider deploying them in the same regional breakout approach, to enable the privacy and security benefits for their end-users without penalizing performance.

Full MNAs (such as Twilio) should carefully consider the underlying infrastructure hosting their core network, as this will impact the performance. Our work highlights that one option might be to explore using CDNs with a wide geographical footprint as underlying infrastructure, and benefit from dynamically migrating different core functions to the optimal location.

For app developers, our work gives tangible insights into what they might expect in terms of performance from this new breed of global operators. Thus, an app developer or app provider (i.e., the entity providing services through the internet) who is aware of the performance limitations of a global connectivity model can accommodate and potentially mitigate them (e.g., for multimedia streaming apps, the developer might consider using larger buffers for video). This is especially relevant for the services aimed at IoT devices that IoT vendors distribute worldwide, and that are permanently roaming.

5.7. Summary

This chapter presented a comprehensive study on the performance of MNA under roaming conditions, focusing on end-user experiences in Spain and the USA. It begins with an overview of the evolution and classification of MNOs, MVNOs, and MNAs under roaming conditions. The chapter explores the challenges that MNAs face in managing network complexity to sustain QoE for services like web browsing and video streaming NSaaS. It examines critical performance indicators, such as DNS resolution delay and traceroute metrics, and discusses the impact of different solutions on implementing international roaming effectively. One of the key insights revealed in this research is the significant throttling applied by MNOs in the United States specifically to video streaming services—a practice that is notably absent among MNAs It then discusses the implementation of the 5G Control and User Plane Separation (CUPS) concept, which enhances roaming efficiency by separating the control and user planes. The chapter concludes by comparing MNA and local MNO performance, showcasing the benefits of the CUPS-enabled Regional Breakout (RBO) solution in improving global cellular operations.

6 Conclusions and Perspectives

Network slicing will be a crucial technology of 5G to enhance the profit of the MNO. In this thesis, we investigated two ways of doing that. Firstly, we developed a system for doing overbooking of the network. Secondly, we showed how to minimize the prominent costs of the network.

We also give a shed light on the network performance and analysis from an end-user perspective in roaming, developing a comprehensive taxonomy of MNO, MVNO and MNA. We have determined the main research problems and by using AI and Optimization Algorithms,we designed novel algorithms and solutions to enhance MNO profit and explored the current 5G roaming environment.

6.1. Main Takeaways

In the first Chapter of the thesis, We proposed a novel solution for NSaaS overbooking that builds on joint DL and optimization at different timescales in order to maximize the net profit of the MNO.

Our model, kaNSaaS, was compared against legacy and state-of-the-art overbooking-based methods for NSaaS management in presence of slice demands of unprecedented realism. We have proved that, under real-world service-level demands collected in a large-scale production network and with realistic operating cost margins, the net profit of the MNO from a practical overbooking-based NSaaS solution can be multiplied by a factor four. We also presented kaNSaaS, a practical solution that achieves the gains above over legacy NSaaS models, and largely outperforms the state-of-the-art scheme for NSaaS overbooking. This suggests that the potential of overbooking is boosted for typical real uses cases where the margins of mobile operators are rather limited. We have also seen that the orchestration layer plays an important role in terms of scalability and amount resources required for the MNO Indeed, at RU level, the temporal series of traffic demand is more bursty and noisy, with traffic peaks considerably exceeding the average traffic demand, whereas at the city level the traffic is smoother and with less variability w.r.t. the average at each given day time. This induces that as the capacity required to serve the peak throughput exponentially increases as we approach the RU layer. is much higher that the capacity required at CN, following the mathematical truth that the sum of maxima is always greater or equal than the maximum of the sum. Furthermore, the benefit of overbooking does not depend on the network dimensionality, and the overbooked allocation can give us more than a 200% extra profit over the standard approach. Ultimately, our work shed new light on the performance and clear advantages of overbooking under real-world slice demand scenarios. These results and the fact that we intend to openly release our code and the synthetic data used for part of the evaluation pave the road to further research towards production-grade deployments of overbooking in future sliced networks.

In Section 4.2, we introduced AZTEC+, a framework able to minimize the overall cost of the network. In the context of ever-increasing complex networks and the need of zero-touch management AZTEC+ demonstrates the viability of automatically optimizing the different cost trade-offs that happen in a virtualized mobile network, to minimize resource provisioning. In detail, AZTEC+ automatically selects resource orchestration intervals based on past service-level demands. We evaluate AZTEC+ with data coming from a large-scale production network over different cities in France. Our experiment shows that, depending on the different cost factors contributing to the overall network, AZTEC+ outperforms the different benchmarks by a factor up to $5.85\times$.

In Chapter 5, we introduced a way to analyze the operations of MNAs and consider their potential development. We expanded the existing taxonomy of MVNOs to include the concept of MNA. Our research involved measuring the roaming operations of three MNAs with different operational models: Twilio, Truphone, and Fi from Europe and the US. We evaluated their performance and measured the impact of their operational approach on various applications such as DNS, web browsing, and video streaming. We discovered that, unlike MNOs and MVNOs which rely on HR roaming, MNAs are gradually adopting limited forms of LBO roaming. While current MNA operations may not enable LBO in the visited country, some implement regional breakout to keep traffic within the same continent/region and avoid long transoceanic links. This difference is clearly seen in the performance of the various applications we tested. Our goal was to understand the operational models of MNAs and their impact on application performance, rather than to compare the different MNOs and MNAs tested. We also observed how the location of the breakout point significantly affects the end-user's QoE and how MNAs can outperform MNOs depending on their network configuration for certain applications. Furthermore, we investigated the potential evolution of the MNA model and explored the performance gains that could result from fully utilizing LBO. Depending on the application, regional breakout can provide significant benefits, and there are limited additional advantages from implementing full LBO.

The research conducted in this thesis sheds light on the potential of integrating AI in network slicing in 5G to enhance the profit of the MNO and analyzes the current roaming environment. We can disseminate three main takeaways from this work and the perspective for future work:

1. We have demonstrated that using different multi-time orchestration approaches, combining AI and optimization, the MNO can obtain more profit doing (I) overbooking of network slices and (ii) reducing the overall cost of the network leveraging zero-touch management. Future extensions could leverage the proposed frameworks and tailor them to required QoS

2. We showed the clear benefit that gives having RBO for the end-user performance, obtaining similar results than the home MNO. The perspective that we deliver for the MNOs will help to the future network deployments to help the ever-increasing diverse categories of devices that are currently using internet, offering seamless connectivity around the world with the required QoS. Also, our work gives tangible insights into what they might expect in terms of performance from this new breed of global operators. Thus, an app developer or app provider (i.e., the entity providing services through the internet) who is aware of the performance limitations of a global connectivity model can accommodate and potentially mitigate them (e.g., for multimedia streaming apps, the developer might consider using larger buffers for video). This is especially relevant for the services aimed at IoT devices that IoT vendors distribute worldwide, and that are permanently roaming.

6.2. Perspectives

6.2.1. Short-to-medium-term Perspectives

The presented research lays the groundwork for immediate and practical applications in optimizing network slicing and roaming within 5G networks in the short to medium term. One of the most promising developments is the adoption of the overbooking framework by MNOs. This system offers a low-cost approach to significantly enhance profitability without requiring major infrastructure changes. In the next two to three years, MNOs will likely implement pilot projects, testing the overbooking system in localized environments. These early trials will allow operators to measure its real-world performance and adaptability to fluctuating network demands. As network slicing becomes more prominent, these systems will play a key role in dynamically adjusting resources while maintaining the required QoS. AI integration will refine this process, allowing for better prediction of traffic surges or drops, thus fine-tuning resource allocation to maximize both profit and performance.

In parallel, AZTEC+ will find applications in resource optimization across larger, more complex urban networks. This AI-driven system allows operators to automatically manage the trade-offs between cost and performance in resource provisioning. As 5G networks expand and more traffic is handled in real time, automated systems like AZTEC+ will become essential to reducing operational costs while maintaining network efficiency. Early implementations will likely focus on refining machine learning algorithms to predict traffic patterns based on user behavior, enabling more precise orchestration of resources. As this technology matures, it could extend beyond telecommunications, integrating with other smart systems in cities, such as transportation networks and energy grids, to improve overall resource management.

Roaming will also see significant enhancements, particularly through the advancement of RBO mechanisms used by MNAs. These systems, which reduce latency by breaking out traffic within the same region, are becoming more relevant as MNAs expand their operations. Within the next few years, MNAs will improve RBO capabilities, offering better connectivity for services like video streaming and IoT-based applications. These enhancements will be crucial as IoT devices increasingly rely on global connectivity, and applications demand low-latency performance. By optimizing RBO, MNAs will provide more seamless connectivity across regions, helping ensure that performance remains consistent, even when devices are permanently roaming.

6.2.2. Long-term Perspectives

In the long term, the evolution of 5G technologies will lay the groundwork for more advanced systems, including 6G networks, where the integration of AI, machine learning, and network virtualization will become even more critical. The developed overbooking framework could become an integral part of fully autonomous networks, where slices dynamically adjust resources based on real-time demands and AI-powered predictions of future traffic patterns. As AI systems become more sophisticated, the role of human oversight in network management will continue to diminish. Network slicing will evolve to cater to more complex applications, such as fully autonomous vehicles, remote surgeries, and ultra-reliable low-latency communications. AI-driven resource allocation will not only manage the telecommunications resources more effectively but also integrate with other verticals, such as energy management or smart transportation, optimizing resources across sectors.

Beyond 5G, as 6G networks begin to take shape, research into AI-driven cost optimization could play a pivotal role in creating intelligent, self-sustaining networks. These networks will be characterized by their ability to autonomously negotiate resource allocations between different services, ensuring optimal performance without operator intervention. AI systems will likely take on a broader role, managing real-time economic models for resource pricing, where operators can dynamically adjust prices based on real-time demand and network availability. Additionally, the proposed frameworks could promote fairness among Mobile Network Operators (MNOs) by ensuring equitable resource distribution and pricing transparency. For instance, AI systems could balance resource allocations between competing MNOs to prevent monopolization or underutilization of resources, fostering a more competitive and collaborative telecommunications ecosystem. This evolution could be further supported by blockchain or distributed ledger technologies, ensuring secure and transparent management of resources and pricing models across multiple operators.

As we look further into the future, zero-touch management systems will evolve into holistic, fully autonomous networks. These systems will operate with minimal human intervention, capable of self-diagnosis, self-healing, and self-optimization, ensuring network stability and efficiency despite unexpected demand surges or disruptions. The AI systems managing these networks will be context-aware, meaning they will not just respond to traffic patterns but also predict and prepare for future needs, considering factors like seasonal variations, user behavior, and even non-telecommunications-related data like energy consumption or traffic congestion. Fairness considerations could also extend to these AI-driven systems, ensuring that resources are equitably distributed across stakeholders, thus balancing economic incentives with service quality and user experience.

On the roaming side, MNAs will likely blur the lines between traditional MNO models and new global connectivity approaches. As LBO becomes fully integrated into roaming systems, MNAs will provide seamless global connectivity that is indistinguishable from local service, significantly improving user experience for both individual users and IoT devices. Roaming systems in 6G and beyond will offer ultra-reliable, low-latency connections across borders and regions, supporting a wide range of applications, from autonomous vehicles to global industrial IoT systems. The research into MNAs' performance under various configurations will provide a critical foundation for these developments, particularly as roaming agreements evolve to support more complex use cases and services. In addition to enhancing user experience and global connectivity, future systems could explore mechanisms to ensure fairness in roaming cost models and service prioritization among operators, especially in high-demand scenarios.

In summary, the long-term future of telecommunications will be characterized by the growing autonomy of network management systems and the increasing integration of AI-driven solutions in both overbooking and cost optimization frameworks. These advancements will drive the telecommunications industry toward more flexible, efficient, and scalable networks, capable of supporting the complex and diverse applications that will define the next generation of connectivity. Furthermore, the integration of fairness mechanisms into these frameworks will play a crucial role in promoting balanced competition and equitable access to network resources, ensuring that advancements benefit all stakeholders in the ecosystem.

References

[1] S. Alcalá-Marín, A. Raman, W. Wu, A. Lutu, M. Bagnulo, O. Alay, and F. Bustamante, “Global mobile network aggregators: taxonomy, roaming performance and optimization,” in *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, 2022, pp. 183–195.

[2] S. Alcalá-Marín, A. Bazco-Nogueras, A. Banchs, and M. Fiore, “kansaas: Combining deep learning and optimization for practical overbooking of network slices,” in *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, ser. MobiHoc ’23. New York, NY, USA: Association for Computing Machinery, 2023, p. 51–60.

[3] S. A. A-Marín, W. Wu, A. Raman, M. Bagnulo, O. Alay, E. Bustamante, M. Fiore, and A. Lutu, “A comparative analysis of global mobile network aggregators,” 2024.

[4] S. Alcalá-Marín, D. Bega, M. Gramaglia, A. Banchs, X. Costa-Perez, and M. Fiore, “Aztec+: Long and short term resource provisioning for zero-touch network management,” 2024.

[5] S. Davis, R. Batra, G. Blennerud, F. Burstedt, A. Carlsson, S. Elmgren, J. Jelic, D. Kobescak, I. Komljenovic, F. Kronestedt, C. Kuhlins, P. Lindberg, Y. Liu, N. Lövehagen, G. Macharia, J. Malmodin, J. Manssour, S. Onay, R. S. Pandey, T. Tolic, J. Travers, and B. Trollsås, “Ericsson mobility report letter from the publisher 5g standalone brings new opportunities,” 2023.

[6] Q. Liu, T. Zhang, M. Hemmatpour, H. Qiu, D. Zhang, C. S. Chen, M. Mellia, and A. Aghasaryan, “Operationalizing ai/ml in future networks: A bird’s eye view from the system perspective,” *IEEE Communications Magazine*, pp. 1–7, 2024.

[7] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, “Aztec: Anticipatory capacity allocation for zero-touch network slicing,” in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 794–803.

[8] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, “Network slicing in 5g: Survey and challenges,” *Comm. Mag.*, vol. 55, no. 5, p. 94–100, May 2017. [Online]. Available: https://doi.org/10.1109/MCOM.2017.1600951

[9] H. P. Phyu, D. Naboulsi, and R. Stanica, “Machine learning in network slicing - a survey,” *IEEE Access*, vol. 11, pp. 39 123–39 153, 2023.

[10] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, “Network slicing for 5g: Challenges and opportunities,” *IEEE Internet Computing*, vol. 21, no. 5, pp. 20–27, 2017.

[11] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, “Network slicing and softwarization: A survey on principles, enabling technologies, and solutions,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[12] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, “Machine learning in wireless sensor networks: Algorithms, strategies, and applications,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1996–2018, 2014.

[13] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, “Network function virtualization: State-of-the-art and research challenges,” *Commun. Surveys Tuts.*, vol. 18, no. 1, p. 236–262, Jan. 2016. [Online]. Available: https://doi.org/10.1109/COMST.2015.2477041

[14] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez, “How Should I Slice My Network?: A Multi-Service Empirical Evaluation of Resource Sharing Efficiency,” in *Proc. of the 24th Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*, New Delhi, India, Oct. 2018, pp. 191–206.

[15] V. Sciancalepore, C. Mannweiler, F. Z. Yousaf, P. Serrano, M. Gramaglia, J. Bradford, and I. L. Pavón, “A Future-Proof Architecture for Management and Orchestration of Multi-Domain NextGen Networks,” *IEEE Access*, vol. 7, pp. 79 216–79 232, Jun. 2019.

[16] A. Lutu, D. Perino, M. Bagnulo, E. Frias-Martinez, and J. Khangosstar, “A Characterization of the COVID-19 Pandemic Impact on a Mobile Network Operator Traffic,” in *A Characterization of the COVID-19 Pandemic Impact on a Mobile Network Operator Traffic*, ser. IMC’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 19 – 33. [Online]. Available: https://doi.org/10.1145/3419394.3423655

[17] A. Lutu, B. Jun, A. Finamore, F. E. Bustamante, and D. Perino, “Where Things Roam: Uncovering Cellular IoT/M2M Connectivity,” in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 147 – 161.

[18] E. Obiodu, A. Raman, A. Abubakar, S. Mangiante, N. Sastry, and H. Aghvami, “Dsm-moc as baseline: Reliability assurance via redundant cellular connectivity in connected cars,” *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.

[19] H. Zhang, N. Liu, X. Chu, K. Long, A.-H. Aghvami, and V. C. M. Leung, “Network slicing based 5g and future mobile networks: Mobility, resource management, and challenges,” *IEEE Communications Magazine*, vol. 55, no. 8, pp. 138–145, 2017.

[20] C. Zhang, P. Patras, and H. Haddadi, “Deep learning in mobile and wireless networking: A survey,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2224–2287, Mar. 2019.

[21] B. Koley, “The zero touch network,” in *Proc. of the 12th International Conference on Network and Service Management (IEEE CNSM)*, Montreal, Quebec, Canada, Oct. 2016.

[22] European Telecommunications Standards Institute (ETSI), “ZSM Scenarios and key requirements,” ETSI ISG ZSM 001, Oct. 2018.

[23] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, “On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration,” *Commun. Surveys Tuts.*, vol. 19, no. 3, p. 1657–1681, Jul. 2017. [Online]. Available: https://doi.org/10.1109/COMST.2017.2705720

[24] J. B. et al., “The big book of mlops,” https://www.databricks.com/p/ebook/the-big-book-of-mlops, 2023, available at: https://www.databricks.com/p/ebook/the-big-book-of-mlops.

[25] Nokia Networks, “Ava ai and analytics,” https://nokia.com/networks/ai-and-analytics, 2022, available at: https://nokia.com/networks/ai-and-analytics.

[26] Z. Y. et al., “Aquarius—enable fast, scalable, data-driven service management in the cloud,” *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4028–4044, 2022.

[27] L. Y. et al., “Quality monitoring and assessment of deployed deep learning models for network aiops,” *IEEE Network*, vol. 35, no. 6, pp. 84–90, 2021.

[28] J. H. et al., “New directions in automated traffic analysis,” in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2021, pp. 3366–3383.

[29] C. Huyen, *Designing machine learning systems: An Iterative Process for Production-ready Applications*. O’Reilly Media, 2022.

[30] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez, “Resource sharing efficiency in network slicing,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 909–923, 2019.

[31] N. Bui and J. Widmer, “Data-driven evaluation of anticipatory networking in lte networks,” *IEEE Transactions on Mobile Computing*, vol. 17, no. 10, pp. 2252–2265, October 2018.

[32] V. Sciancalepore, K. Samdanis, X. Costa-Perez, M. G. D. Bega, and A. Banchs, "Mobile traffic forecasting for maximizing 5g network slicing resource utilization," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, May 2017, pp. 1–4.

[33] J. Mei, X. Wang, , and K. Zheng, "An intelligent self-sustained ran slicing framework for diverse service provisioning in 5g-beyond and 6g networks," *Intelligent Converged Networks*, vol. 1, pp. 281–294, December 2020.

[34] F. Tonini, M. F. C. Natalino, C. Raffaelli, and P. Monti, "Network slicing automation: Challenges and benefits," in *Proceedings of the International Conference on Optical Network Design and Modeling (ONDM)*, May 2020, pp. 1–6.

[35] T. Jirsik, S. Trcka, and P. Celeda, "Quality of service forecasting with lstm neural network," in *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, April 2019, pp. 251–260.

[36] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," in *Proceedings of the 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, December 2018, pp. 1394–1401.

[37] C.-W. Huang, C.-T. Chiang, , and Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *Proceedings of the IEEE 28th Annual International Symposium on Personal, Indoor, Mobile Radio Communications (PIMRC)*, October 2017, pp. 1–6.

[38] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 280–288.

[39] D. Bega, M. Gramaglia, A. Garcia-Saavedra, M. Fiore, A. Banchs, and X. Costa-Perez, "Network slicing meets artificial intelligence: An ai-based framework for slice management," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 32–38, June 2020.

[40] T. V. K. Buyakar, H. Agarwal, B. R. Tamma, and A. A. Franklin, "Resource allocation with admission control for gbr and delay qos in 5g network slices," in *Proceedings of the International Conference on Communication Systems and Networks (COMSNETS)*, 2020, pp. 213–220.

[41] S. K. Tayyaba, H. A. Khattak, A. Almogren, M. A. Shah, I. U. Din, I. Alkhalifa, , and M. Guizani, "5g vehicular network resource management for improving radio access through machine learning," *IEEE Access*, vol. 8, pp. 6792–6800, 2020.

[42] E. H. Bouzidi, A. Outtagarts, A. Hebbar, R. Langar, and R. Boutaba, "Online based learning for predictive end-to-end network slicing in 5g networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, June 2020, pp. 1–7.

[43] J.-B. Monteil, J. Hribar, P. Barnard, Y. Li, and L. A. DaSilva, “Resource reservation within sliced 5g networks: A cost-reduction strategy for service providers,” in *Proceedings of the IEEE International Conference on Communications Workshops (ICC Workshops)*, June 2020, pp. 1–6.

[44] M. Toscano, F. Grunwald, M. Richart, J. Baliosian, E. Grampín, and A. Castro, “Machine learning aided network slicing,” in *Proceedings of the International Conference on Transparent Optical Networks (ICTON)*, July 2019, pp. 1–4.

[45] S. Yang, X. Yu, , and Y. Zhou, “Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example,” in *Proceedings of the International Workshop on Electronics, Communications, and Artificial Intelligence (IWECAI)*, June 2020, pp. 98–101.

[46] H. Chergui and C. Verikoukis, “Big data for 5g intelligent network,” *IEEE Communications Letters*, 2020.

[47] L. Zanzi, J. X. Salvat, V. Sciancalepore, A. Garcia-Saavedra, and X. Costa-Perez, “Overbooking network slices end-to-end: Implementation and demonstration,” in *Proc. of ACM SIGCOMM*, 2018, p. 144–146.

[48] J. X. Salvat, L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore, and X. Costa-Perez, “Overbooking network slices through yield-driven end-to-end orchestration,” in *Proc. Int. Conf. Emerging Networking EXperiments and Technologies (CoNEXT)*, 2018, p. 353–365.

[49] F. Zarinni, A. Chakraborty, V. Sekar, S. R. Das, and P. Gill, “A first look at performance in mobile virtual network operators,” in *Proceedings of the 2014 conference on internet measurement conference*, ser. IMC ’14, 2014, pp. 165–172.

[50] P. Schmitt, M. Vigil, and E. Belding, “A study of mvno data paths and performance,” in *International Conference on Passive and Active Network Measurement*. Springer, 2016, pp. 83–94.

[51] A. Xiao, Y. Liu, Y. Li, F. Qian, Z. Li, S. Bai, Y. Liu, T. Xu, and X. Xin, “An in-depth study of commercial mvno: Measurement and optimization,” in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’19, 2019, pp. 457–468.

[52] N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, N. Weaver, and V. Paxson, “Beyond the radio: Illuminating the higher layers of mobile networks,” in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’15, 2015, pp. 375–387.

[53] Z. Yuan, Q. Li, Y. Li, S. Lu, C. Peng, and G. Varghese, “Resolving policy conflicts in multi-carrier cellular access,” in *Proceedings of the 24th Annual International*

Conference on Mobile Computing and Networking, ser. MobiCom '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 147–162. [Online]. Available: https://doi.org/10.1145/3241539.3241558

[54] A. Lutu, B. Jun, F. E. Bustamante, D. Perino, M. Bagnulo, and C. G. Bontje, "A first look at the ip exchange ecosystem," *SIGCOMM Comput. Commun. Rev.*, vol. 50, no. 4, p. 25?34, Oct. 2020. [Online]. Available: https://doi.org/10.1145/3431832.3431836

[55] E. Caushaj, I. Ivanov, H. Fu, I. Sethi, and Y. Zhu, "Evaluating throughput and delay in 3g and 4g mobile architectures," *Journal of Computer and Communications*, vol. 02, pp. 1–8, 01 2014.

[56] V. Vomhoff, M. Sichermann, S. Geißler, A. Lutu, M. Giess, and T. Hoßfeld, "A shortcut through the ipx: Measuring latencies in global mobile roaming with regional breakouts," in *2024 8th Network Traffic Measurement and Analysis Conference (TMA)*, 2024, pp. 1–10.

[57] S. Geissler, F. Wamser, W. Bauer, M. Królikowski, S. Gebert, and T. Hossfeld, "Signaling traffic in internet-of-things mobile networks," *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 452–458, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235420498

[58] A. M. Mandalari, A. Lutu, A. Custura, A. S. Khatouni, O. Alay, M. Bagnulo, V. Bajpai, A. Brunstrom, J. Ott, M. Trevisan, M. Mellia, and G. Fairhurst, "Measuring roaming in europe: Infrastructure and implications on users' qoe," *IEEE Transactions on Mobile Computing*, vol. 21, no. 10, pp. 3687–3699, 2022.

[59] S. Sudhakaran, I. Ali, M. Eisen, J. Perez-Ramirez, C. Cazan, V. Frascolla, and D. Cavalcanti, "Zero-delay roaming for mobile robots enabled by wireless tsn redundancy," in *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*, 2023, pp. 1–8.

[60] W. K. Saad, I. Shayea, B. J. Hamza, A. Azizan, M. Ergen, and A. Alhammadi, "Performance evaluation of mobility robustness optimization (mro) in 5g network with various mobility speed scenarios," *IEEE Access*, vol. 10, pp. 60 955–60 971, 2022.

[61] T. Sebastian, V. A. F. Rodriguez, Z. L. M. Moreira, and M. Guido, "Admission control and virtual network embedding in 5g networks: A deep reinforcement-learning approach," *IEEE Access*, vol. 10, pp. 15 860–15 875, 2022.

[62] J. J. Alves Esteves, A. Boubendir, F. Guillemin, and P. Sens, "A heuristically assisted deep reinforcement learning approach for network slice placement," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4794–4806, 2022.

[63] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Deepcog: Cognitive network management in sliced 5g networks with deep learning," in *Proc. of IEEE*

International Conference on Computer Communications (IEEE INFOCOM), Paris, France, Apr. 2019, pp. 280–288.

[64] J. A. Hurtado Sánchez, K. Casilimas, and O. M. Caicedo Rendon, “Deep reinforcement learning for resource management on network slicing: A survey,” *Sensors*, vol. 22, no. 8, 2022.

[65] S. Troia, R. Alvizu, and G. Maier, “Reinforcement learning for service function chain reconfiguration in nfv-sdn metro-core optical networks,” *IEEE Access*, vol. 7, pp. 167 944–167 957, 2019.

[66] J. Koo, V. B. Mendiratta, M. R. Rahman, and A. Walid, “Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics,” in *Int. Conf. on Network and Service Management (CNSM)*, 2019, pp. 1–5.

[67] Q. Liu, T. Han, and E. Moges, “Edgeslice: Slicing wireless edge computing network with decentralized deep reinforcement learning,” in *IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, 2020, pp. 234–244.

[68] The Linux Foundation, “The Linux Foundation and Google Cloud Launch Nephio to Enable and Simplify Cloud Native Automation of Telecom Network Functions,” Apr. 2022, consulted on March 10th 2023.

[69] M. Burman and M. Gall, “Ericsson and Red Hat empower service providers to build multi-vendor networks,” Jun. 2022.

[70] J. Rachid and J. Erfanian, “NGMN 5G initiative white paper,” Feb 2015.

[71] S. D. A. Shah, M. A. Gregory, and S. Li, “Cloud-native network slicing using software defined networking based multi-access edge computing: A survey,” *IEEE Access*, vol. 9, pp. 10 903–10 924, 2021.

[72] S. Arora and A. Ksentini, “Dynamic resource allocation and placement of cloud native network services,” in *IEEE Int. Conf. Commun. (ICC)*, 2021, pp. 1–6.

[73] A. Pino, P. Khodashenas, X. Hesselbach, E. Coronado, and S. Siddiqui, “Validation and benchmarking of cnfs in osm for pure cloud native applications in 5g and beyond,” in *2021 Int. Conf. on Computer Communications and Networks (ICCCN)*, 2021, pp. 1–9.

[74] D. Giannopoulos, P. Papaioannou, C. Tranoris, and S. Denazis, “Monitoring as a service over a 5G network slice,” in *Joint European Conf. on Networks and Commun. & 6G Summit (EuCNC/6G Summit)*, 2021, pp. 329–334.

[75] X. Zhou, R. Li, T. Chen, and H. Zhang, “Network slicing as a service: enabling enterprises’ own software-defined cellular networks,” *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, 2016.

[76] D. Bega, M. Gramaglia, A. B., V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5g infrastructure markets: The business of network slicing," in *Proc. of IEEE INFOCOM*, 2017, pp. 1–9.

[77] K. T. Talluri and G. Van Ryzin, *The theory and practice of revenue management*. Springer, 2004, vol. 1.

[78] C. Sexton, N. Marchetti, and L. A. DaSilva, "On provisioning slices and overbooking resources in service tailored networks of the future," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2106–2119, 2020.

[79] S. Saxena and K. M. Sivalingam, "Slice admission control using overbooking for enhancing provider revenue in 5g networks," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–7.

[80] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," *Int. Journal of Forecasting*, vol. 36, no. 1, pp. 54 – 74, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207019301128

[81] L. L. Schiavo, M. Fiore, M. Gramaglia, A. Banchs, and X. Costa-Perez, "Forecasting for network management with joint statistical modelling and machine learning," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2022, pp. 60–69.

[82] D. Tikunov and T. Nishimura, "Traffic prediction for mobile network using holt-winter's exponential smoothing," in *Int. Conf. on Software, Telecommunications and Computer Networks*. IEEE, 2007, pp. 1–5.

[83] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, ser. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1990. [Online]. Available: https://books.google.es/books?id=0dhQAAAAMAAJ

[84] S. Martello, "Knapsack Problems: Algorithms and Computer Implementations," *Wiley-Interscience series in discrete mathematics and optimiza tion*, 1990.

[85] F. Patzelt, "Colored noise," https://github.com/felixpatzelt/colorednoise, 2022.

[86] A. Garcia-Saavedra, X. Costa-Pérez, D. J. Leith, and G. Iosifidis, "FluidRAN: Optimized vRAN/MEC Orchestration," in *Proc. of IEEE International Conference on Computer Communications (IEEE INFOCOM)*, Honolulu, HI, USA, Apr. 2018, pp. 2366–2374.

[87] J. Kim, D. Kim, and S. Choi, "3GPP SA2 architecture and functions for 5G mobile communication system," *ICT Express*, vol. 3, no. 1, pp. 1–8, Mar. 2017.

[88] L. Peterson, A. Al-Shabibi, T. Anshutz, S. Baker, A. Bavier, S. Das, J. Hart, G. Palukar, and W. Snow, “Central office re-architected as a datacenter,” *IEEE Communication Magazine*, vol. 54, no. 10, pp. 96–101, Oct. 2016.

[89] J. Hao, T. Jiang, W. Wang, and I. K. Kim, “An empirical analysis of vm startup times in public iaas clouds,” in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, 2021, pp. 398–403.

[90] 5G-CORAL, “Refined design of 5G-CORAL orchestration and control system and future directions,” Public Deliverable, D3.2, May 2019.

[91] S. González, A. De la Oliva, C. J. Bernardos, and L. M. Contreras, “Towards a Resilient Openflow Channel Through MPTCP,” in *Proc. of IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (IEEE BSMB)*, Valencia, Spain, Jun. 2018, pp. 1–5.

[92] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, and X. Costa-Pérez, “OVNES: Demonstrating 5G network slicing overbooking on real deployments,” in *Proc. of IEEE International Conference on Computer Communications Workshops (IEEE INFOCOM WKSHPS)*, Honolulu, HI, USA, Apr. 2018, pp. 1–2.

[93] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. USA: Cambridge University Press, 2007.

[94] G. E. Dahl, T. N. Sainath, and G. E. Hinton, “Improving deep neural networks for LVCSR using rectified linear units and dropout,” in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE ICASSP)*, Vancouver, BC, Canada, May 2013, pp. 8609–8613.

[95] K. Janocha and W. M. Czarnecki, “On Loss Functions for Deep Neural Networks in Classification,” *Schedae Informaticae*, vol. 25, pp. 49–59, Mar. 2017.

[96] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,” in *Proc. of The 33rd International Conference on Machine Learning (ICML)*, New York, NY, USA, Jun. 2016, pp. 1050–1059.

[97] M. J. Powell, “The BOBYQA algorithm for bound constrained optimization without derivatives,” *Cambridge Technical Report NA2009/06, University of Cambridge*, pp. 26–46, Aug. 2009.

[98] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in FORTRAN 77: The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 1993, vol. 2.

[99] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, “Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach,” in *IEEE INFOCOM 2017-IEEE conference on computer communications*. IEEE, 2017, pp. 1–9.

[100] G. Association, “LTE and EPC Roaming Guidelines,” https://www.gsma.com/newsroom/wp-content/uploads/IR.88-v15.0.pdf, Nov. 2016, [Online; accessed 06-March-2018].

[101] A. M. Mandalari, A. Lutu, A. Custura, A. Safari Khatouni, O. Alay, M. Bagnulo, V. Bajpai, A. Brunstrom, J. Ott, M. Mellia, and G. Fairhurst, “Experience: Implications of roaming in europe,” in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 179–189. [Online]. Available: https://doi.org/10.1145/3241539.3241577

[102] T. Kugler, “Google Fi: Stay connected abroad with high speed data from Project Fi,” https://www.blog.google/products/project-fi/stay-connected-abroad-with-high-speed/, accessed: 2021-12-10.

[103] “Google Fi dialer codes,” https://arkienet.com/2018/01/google-fi-dialer-codes/, accessed: 2021-12-10.

[104] “Twilio super sim,” https://www.twilio.com/docs/iot/supersim, accessed: 2021-12-10.

[105] “Twilio super sim: Available networks,” https://www.twilio.com/docs/iot/supersim/available-networks, accessed: 2021-12-10.

[106] “Programmable wireless: Gain global cellular iot connectivity,” https://www.twilio.com/iot/wireless, accessed: 2021-12-10.

[107] M. Gharaibeh, A. Shah, B. Huffaker, H. Zhang, R. Ensafi, and C. Papadopoulos, “A look at router geolocation in public and commercial databases,” in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 463–469. [Online]. Available: https://doi.org/10.1145/3131365.3131380

[108] A. Lutu, M. Trevisan, A. Safari Khatouni, A. M. Mandalari, A. Custura, M. Mellia, O. Alay, M. Bagnulo, V. Bajpai, A. Brunstrom, J. Ott, and G. Fairhurst, “Measuring Roaming in Europe: Infrastructure and Implications on Users QoE,” *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[109] M. contributors, “Navigation timing api,” in *MDN Web Docs*, 2020, last accessed in 23/6/2021.

[110] M. Rajiullah, A. Lutu, A. S. Khatouni, M.-R. Fida, M. Mellia, A. Brunstrom, O. Alay, S. Alfredsson, and V. Mancuso, “Web experience in mobile networks: Lessons from two million page visits,” in *The World Wide Web Conference*, 2019, pp. 1532–1543.

[111] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, “Yomoapp: A tool for analyzing qoe of youtube http adaptive streaming in mobile networks,” in *2015 European Conference on Networks and Communications (EuCNC)*. IEEE, 2015, pp. 239–243.

[112] F. Li, A. A. Niaki, D. Choffnes, P. Gill, and A. Mislove, “A large-scale analysis of deployed traffic differentiation practices,” in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 130–144.

[113] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, “Flexran: A flexible and programmable platform for software-defined radio access networks,” in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 427?441. [Online]. Available: https://doi.org/10.1145/2999572.2999599

[114] X. Foukas, M. K. Marina, and K. Kontovasilis, “Orion: Ran slicing for a flexible and cost-effective multi-service mobile network architecture,” in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 127?140. [Online]. Available: https://doi.org/10.1145/3117811.3117831

[115] I. Quintana-Ramirez, A. Tsiopoulos, M. A. Lema, F. Sardis, L. Sequeira, J. Arias, A. Raman, A. Azam, and M. Dohler, “The making of 5g: Building an end-to-end 5g-enabled system,” *IEEE Communications Standards Magazine*, vol. 2, no. 4, pp. 88–96, 2018.

[116] “AWS global infrastructure,” https://aws.amazon.com/about-aws/global-infrastructure/, accessed: 2021-12-10.

[117] “Open5gs v2.3.6,” https://github.com/open5gs/open5gs, accessed: 2021-12-10.

[118] “Ueransim v3.2.4,” https://github.com/aligungr/UERANSIM, accessed: 2021-12-10.

[119] L. Corneo, M. Eder, N. Mohan, A. Zavodovski, S. Bayhan, W. Wong, P. Gunningberg, J. Kangasharju, and J. Ott, *Surrounded by the Clouds: A Comprehensive Cloud Reachability Study*. New York, NY, USA: Association for Computing Machinery, 2021, p. 295–304. [Online]. Available: https://doi.org/10.1145/3442381.3449854

[120] “Amazon virtual private cloud,” https://docs.aws.amazon.com/vpc/latest/peering/what-is-vpc-peering.html, accessed: 2021-12-10.