TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Machine Learning-based Orchestration Solutions for Future Slicing-Enabled Mobile Networks

VOM FACHBEREICH ELEKTROTECHNIK UND INFORMATIONSTECHNIK
DER TECHNISCHEN UNIVERSITÄT KAISERSLAUTERN ZUR VERLEIHUNG
DES AKADEMISCHEN GRADES DOKTOR DER
INGENIEURWISSENSCHAFTEN (DR.-ING.) GENEHMIGTE
DISSERTATION

SUPERVISOR:
Prof. Dr.-Ing Hans Dieter Schotten
Technische Universität Kaiserslautern

WRITTEN BY:
Lanfranco Zanzi
NEC Laboratories Europe GmbH

CO-ADVISOR:
Prof. Xavier Costa-Pérez
NEC Laboratories Europe GmbH
i2CAT Foundation
ICREA - Catalan Institution for Research and Advanced Studies

Tag der mündlichen Prüfung: 29^{th} September, 2022
DE386

CURRICULUM VITAE

PERSONAL INFORMATION

First name, Surname	**Lanfranco Zanzi**
Nationality	Italian

WORK EXPERIENCE

Dates (from - to)	May 2017 – September 2022 (date of submission)
Name and type of organization	NEC Laboratories Europe GmbH, Kurfürsten-Anlage 36, 69115 Heidelberg, Germany
Type of business or sector	R&D
Occupation or position held	Senior Researcher

EDUCATION AND TRAINING

Dates (from - to)	November 2017 – September 2022
Name and address of the organization	Technische Universität Kaiserslautern, Erwin-Schrödinger-Straße 1, 67663 Kaiserslautern
Title of qualification awarded	**PhD**
Dates (from - to)	October 2014 – April 2017
Name and address of the organization	Politecnico di Milano (Technical University of Milan), piazza Leonardo da Vinci 32, Milano, Italy
Type of business or sector	Education
Title of qualification awarded	**M. Sc** in Telecommunication Engineering
Dates (from - to)	September 2011 - September 2014
Title of qualification awarded	**Bachelor** in Telecommunication Engineering

LIST OF PATENTS

F. Giust, V. Sciancalepore, **L. Zanzi**, System and method to support network slicing in an MEC system providing automatic conflict resolution arising from multiple tenancy in the MEC environment, US 2019-0104030 A1, Apr. 4, 2019

V. Sciancalepore, **L. Zanzi**, A. Albanese, X. Costa-Perez, Multi-Resource and Autonomous Hierarchical Brokering Platform to Enable Slice Resource Exchange Among Heterogeneous Network Tenants, US Patent App. 16/819,218, Sept. 2021

To my family.

Main Achievements and Published Contents

This thesis is submitted in partial fulfillment of the requirements for the degree of *Philosophiae Doctor* at the University of Kaiserslautern. The research leading to these results was mainly supported by two european research projects: $i)$ H2020 ITN project 5G AURA (grant no. 675806), $ii)$ H2020 project MonB5G (grant no. 871780). The research presented hereafter was conducted at NEC Laboratories Europe GmbH under the supervision of Professor Xavier Costa-Pérez, and at Technische Unviversität Kaiserlautern under the supervision of Professor Hans D. Schotten.

In the following we summarize the main outcomes and achievements in terms of publications, patents and awards obtained during the doctoral degree program. In particular, the results disclosed in this thesis work are based on conclusions and findings contained in the several research papers, grouped according to the different publication category and listed in chronological order.

Journal publications

[1] **L. Zanzi**, F. Cirillo, V. Sciancalepore, F. Giust, X. Costa-Pérez, S. Mangiante, G. Klas, Evolving Multi-Access Edge Computing to support enhanced IoT deployments, In *IEEE Communications Standards Magazine*, Vol 3(2), 26-34, 2019, doi: 10.1109/MCOMSTD.2019.1800009.

This work is not included in this thesis.

[2] **L. Zanzi**, V. Sciancalepore, A. Garcia-Saavedra, H. D. Schotten, X. Costa-Pérez, LACO: A Latency-Driven Network Slicing Orchestration in Beyond-5G Networks, *IEEE Transactions on Wireless Communications*, 2020, doi: 10.1109/TWC.2020.3027963.

This work is fully included and its contents are reported in Chapter 3.
The author's role in this work is focused on the design and implementation of the reinforcement learning algorithm and model, as well as on its validation by means of hardware experimentation of the concepts proposed in the paper.

[3] **L. Zanzi**, V. Sciancalepore, A. Garcia-Saavedra, X. Costa-Pérez, G. Agapiou, H. D. Schotten, ARENA: A Data-driven Radio Access Networks Analysis of Football Events,

IEEE Transaction on Network Service and Management, 2020, doi: 10.1109/TNSM.2020.3032829.

This work is fully included and its contents are reported in Chapter 4.
The author's role in this work is the design and evaluation of machine learning models for RAN monitoring analysis and prediction in crowded public events.

Full papers in conference proceedings

[4] **L. Zanzi**, F. Giust, V. Sciancalepore, M^{2}EC: A multi-tenant resource orchestration in multi-access edge computing systems, In *IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Spain, pp. 1-6, 2018, doi: 10.1109/WCNC.2018.8377292.

This work is fully included in Chapter 3.
The author's role in this work focused on the definition of the mathematical optimization model and its evaluation through numerical simulations.

[5] J. X. Salvat, **L. Zanzi**, A. Garcia-Saavedra, V. Sciancalepore, X. Costa-Pérez, Overbooking Network Slices through Yield-driven End-to-End Orchestration, In *Proceedings of the 14th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, Creete, Greece, pp. 353–365, 2018, doi: 10.1145/3281411.3281435.

This work is fully included in Chapter 2.
The author's role in this work is focused on contributing to the design of the overall solution and its performance evaluation through testbed implementation.

[6] **L. Zanzi**, A. Albanese, V. Sciancalepore, X. Costa-Pérez, NSBchain: A Secure Blockchain Framework for Network Slicing Brokerage, In *IEEE International Conference on Communications (ICC)*, Dublin, Ireland, pp. 1-7, 2020 ,doi: 10.1109/ICC40277.2020.9149414.

This work is fully included in Chapter 4
The author's role in this work is focused on the design and definition of the blockchain architectural model applied at the network slicing context, contributing to the design and implementation of a testbed as well as its corresponding performance evaluation.

[7] A. Okic, **L. Zanzi**, V. Sciancalepore, A. Redondi, X. Costa-Pérez, π-ROAD: a Learn-as-You-Go Framework for On-Demand Emergency Slices in V2X Scenarios, In *IEEE International Conference on Computer Communications (INFOCOM)*, Vancouver, Canada, pp. 1-10, 2021, doi: 10.1109/INFOCOM42981.2021.9488677.

This work is fully included in Chapter 4.
The author's role in this work is focused on the design of the Machine Learning framework for data analysis and classification, actively participating in the discussion about the development of the concepts proposed in the paper.

[8] **L. Zanzi**, V. Sciancalepore, On Guaranteeing End-to-End Network Slice Latency Constraints in 5G Networks, In *International Symposium on Wireless Communication Systems (ISWCS)*, Lisbon, Portugal, pp. 1-6, 2018, doi: 10.1109/ISWCS.2018.8491249.

This work is not included in this thesis.

Book Chapter

[9] X. Costa-Pérez, V. Sciancalepore, **L. Zanzi**, A. Albanese, Blockchain for Mobile Networks, In Anwer Al-Dulaimi, Octavia Dobre, Chih-Lin I, *Blockchains: Empowering Technologies and Industrial Applications*, to appear in 2021. IEEE/Wiley.

This work is partially included in Chapter 2. The author's role in this work focused on the investigation of the applicability of blockchain solutions into the mobile network scenario, and the corresponding conceptual description into the document.

Poster and Conference Demonstrations

[10] **L. Zanzi**, J. X. Salvat, A. Garcia-Saavedra, V. Sciancalepore, X. Costa-Pérez, Overbooking Network Slices End-to-End: Implementation and Demonstration, In *Proceedings of the ACM SIGCOMM 2018 Conference - Posters and Demos (SIGCOMM WKSHPS)*, Budapest, Hungary, pp. 353–365, 2018, doi: 10.1145/3234200.3234230.

[11] **L. Zanzi**, V. Sciancalepore, A. Garcia-Saavedra, X. Costa-Pérez, OVNES: Demonstrating 5G Network Slicing Overbooking on Real Deployments, In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Honolulu, HI, USA, pp. 1–2, 2018, doi: 10.1109/INFCOMW.2018.8406867.

[12] **L. Zanzi**, J. X. Salvat, V. Sciancalepore, A. Garcia-Saavedra, X. Costa-Pérez, Latency-driven Network Slices Orchestration, In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Paris, France, pp. 965-966, 2019, doi: 10.1109/INFCOMW.2019.8845216.

These works are partially included in Chapter 2.
The author's role in these works focused on the design of the overall solutions and on the setup of the testbed platforms showcased in the corresponding demo and workshop sessions.

Patent Applications

[13] F. Giust, V. Sciancalepore, **L. Zanzi**, System and method to support network slicing in an MEC system providing automatic conflict resolution arising from multiple tenancy in the MEC environment, *US Patent App. 17/102,515, Mar. 2021*.

[14] V. Sciancalepore, **L. Zanzi**, A. Albanese, X. Costa-Pérez Multi-Resource and Autonomous Hierarchical Brokering Platform to Enable Slice Resource Exchange Among Heterogeneous Network Tenants, *US Patent App. 16/819,218, Sept. 2021*.

Awards

L. Zanzi, J. X. Salvat, V. Sciancalepore, A. Garcia-Saavedra, X. Costa-Pérez, Overbooking 5G Network Slices, 2019, **NEC Corporation Award**.

L. Zanzi, J. X. Salvat, V. Sciancalepore, A. Garcia-Saavedra, X. Costa-Pérez, Latency-driven Network Slices Orchestration, In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops*, Paris, France, pp. 965-966, 2019, doi: 10.1109/INFCOMW.2019.8845216 **Best Demo Award (Runner Up)**.

Abstract

The fifth generation mobile networks (5G) will incorporate novel technologies such as *network programmability* and *virtualization* enabled by Software-Defined Networking (SDN) and Network Function Virtualization (NFV) paradigms, which have recently attracted major interest from both academic and industrial stakeholders.

Building on these concepts, *Network Slicing* raised as the main driver of a novel business model where mobile operators may open, i.e., “slice”, their infrastructure to new business players and offer independent, isolated and self-contained sets of network functions and physical/virtual resources tailored to specific services requirements.

While *Network Slicing* has the potential to increase the revenue sources of service providers, it involves a number of technical challenges that must be carefully addressed. End-to-end (E2E) network slices encompass time and spectrum resources in the radio access network (RAN), transport resources on the fronthauling/backhauling links, and computing and storage resources at core and edge data centers. Additionally, the vertical service requirements’ heterogeneity (e.g., high throughput, low latency, high reliability) exacerbates the need for novel orchestration solutions able to manage end-to-end network slice resources across different domains, while satisfying stringent service level agreements and specific traffic requirements.

An end-to-end network slicing orchestration solution shall i) admit network slice requests such that the overall system revenues are maximized, ii) provide the required resources across different network domains to fulfill the Service Level Agreements (SLAs) iii) dynamically adapt the resource allocation based on the real-time traffic load, end-users’ mobility and instantaneous wireless channel statistics.

Certainly, a mobile network represents a fast-changing scenario characterized by complex spatio-temporal relationship connecting end-users’ traffic demand with social activities and economy. Legacy models that aim at providing dynamic resource allocation based on traditional traffic demand forecasting techniques fail to capture these important aspects. To close this gap, machine learning-aided solutions are quickly arising as promising technologies to sustain, in a scalable manner, the set of operations required by the network slicing context. How to implement such resource allocation schemes among slices, while trying to make the most efficient use of the networking resources composing the mobile infrastructure, are key problems underlying the network slicing paradigm, which will be addressed in this thesis.

Zusammenfassung

Die fünfte Generation der Mobilfunknetze (5G) wird neuartige Technologien wie *Netzprogrammierbarkeit* und *Virtualisierung* umfassen, die durch die Paradigmen Software-Defined Networking (SDN) und Network Function Virtualization (NFV) ermöglicht werden und in letzter Zeit sowohl bei akademischen als auch bei industriellen Akteuren auf großes Interesse gestoßen sind.

Aufbauend auf diesen Konzepten hat sich *Network Slicing* als Haupttreiber eines neuartigen Geschäftsmodells herauskristallisiert, bei dem Mobilfunkbetreiber ihre Infrastruktur für neue Geschäftsakteure öffnen, und unabhängige, isolierte und in sich geschlossene Sätze von Netzfunktionen und physischen/virtuellen Ressourcen anbieten können, die auf spezifische Dienstanforderungen zugeschnitten sind, d.h. ‘network slices’.

Obwohl *Network Slicing* das Potenzial hat, die Einnahmequellen von Dienstanbietern zu erweitern, bringt es eine Reihe von technischen Herausforderungen mit sich, die sorgfältig gelöst werden müssen. End-to-End-Netzscheiben (E2E) umfassen Zeit- und Spektrumsressourcen im Funkzugangsnetz, Transportressourcen auf den Fronthauling-/Backhauling-Verbindungen sowie Rechen- und Speicherressourcen in Kern- und Randdatenzentren. Die Heterogenität der vertikalen Serviceanforderungen (z. B. hoher Durchsatz, niedrige Latenz, hohe Zuverlässigkeit) verschärft den Bedarf an neuartigen Orchestrierungslösungen, die in der Lage sind, Network-Slice-Ressourcen über verschiedene Domänen hinweg zu verwalten und dabei strenge Service Level Agreements zu erfüllen.

Eine End-to-End-Orchestrierungslösung muss i) Netzwerk-Slice-Anfragen so zulassen, dass die Gesamteinnahmen des Systems maximiert werden, ii) die erforderlichen Ressourcen über verschiedene Netzwerkdomänen hinweg bereitstellen, um die Service Level Agreements (SLAs) zu erfüllen, iii) die Ressourcenzuweisung dynamisch auf der Grundlage der Echtzeit-Verkehrslast, der Mobilität der Endnutzer und der aktuellen drahtlosen Kanalstatistiken anpassen.

Ein Mobilfunknetz stellt ein sich schnell veränderndes Szenario dar, das durch eine komplexe räumlich-zeitliche Beziehung zwischen der Verkehrsnachfrage der Endnutzer und den sozialen und wirtschaftlichen Aktivitäten gekennzeichnet ist. Herkömmliche Modelle, die eine dynamische Ressourcenzuweisung auf der Grundlage traditioneller Verfahren zur Vorhersage der Verkehrsnachfrage anstreben, können diese wichtigen Aspekte nicht erfassen. Um diese Lücke zu schließen, werden maschinenlerngestützte Lösungen schnell zu vielversprechenden Technologien, um die für das Network Slicing erforder-

lichen Operationen auf skalierbare Weise zu unterstützen.

Wie man solche Ressourcenzuweisungsschemata zwischen Slices implementiert und gleichzeitig versucht, die Netzwerkressourcen, aus denen sich die mobile Infrastruktur zusammensetzt, möglichst effizient zu nutzen, sind Schlüsselprobleme, die dem Paradigma des Network Slicing zugrunde liegen und die in dieser Arbeit behandelt werden.

Contents

Main Achievements and Published Contents iii

Abstract ix

Zusammenfassung xi

Table of Contents xvi

List of Tables xvii

List of Figures xxi

List Of Acronyms xxiv

1 Introduction 1
- 1.1 5G Network Slicing . . . 3
  - 1.1.1 Admission Control . . . 3
  - 1.1.2 Network Orchestration . . . 4
  - 1.1.3 Network Slice Broker . . . 5
  - 1.1.4 Privacy and security . . . 6
- 1.2 Contributions . . . 7
- 1.3 Thesis Outline . . . 8

2 Network Slicing Admission Control and Resource Brokering 11
- 2.1 Overbooking Network Slices End-to-End . . . 11
  - 2.1.1 System Design and Model . . . 12
  - 2.1.2 Service model . . . 12
  - 2.1.3 Resources . . . 13
    - Middleboxes . . . 13
  - 2.1.4 Control Plane . . . 14
  - 2.1.5 Slice Manager . . . 14
  - 2.1.6 E2E Orchestrator . . . 15
  - 2.1.7 Controllers . . . 17
  - 2.1.8 Admission Control & Resource Reservation (AC-RR) Problem . . . 17

2.1.9 Design of the objective function 18
2.1.10 Constraints 19
2.1.11 AC-RR Problem 20
2.1.12 Practical Considerations 21
2.1.13 Algorithms 22
2.1.14 Benders Method 22
2.1.15 Heuristic Algorithm 25
2.1.16 Simulation Results 26
2.1.17 Infrastructure 28
2.1.18 Scenarios 28
2.1.19 Homogeneous cases 29
2.1.20 Heterogeneous cases 31
2.1.21 Experimental Proof-of-Concept 33
2.2 Blockchain for Mobile network resource Brokering 36
2.2.1 The Network Slicing Brokerage Process 37
2.2.2 Related Work 37
2.2.3 The Hierarchical Architecture to evolve the Network Slicing Market . 38
Blockchain 40
2.2.4 The NSBChain framework 41
Smart Contracts 42
2.2.5 Proof-of-Concept Evaluation 45
2.3 Conclusions 48

3 Network Slicing Resource Orchestration 49
3.1 Network Slicing at the Edge 49
3.1.1 Background 50
Related Work 52
3.1.2 A Broker to Support Slicing in MEC 52
3.1.3 Multi-tenant Resource Orchestration 54
Scenario Characterization and System Model 54
MEC Slicing Problem 55
3.1.4 Solution Validation 57
3.2 RAN Slicing with Latency Control 60
3.2.1 Related Work 61
3.2.2 LACO: The framework overview 63
Business scenario 63
Notation 63
Problem Definition 64
Working flow 66
3.2.3 DTMC Model 66
3.2.4 DTMC Monitoring and Prediction 68
3.2.5 Markov Decision Process 70

Reward Definition 70
Complexity analysis 71
Multi-armed Bandit problem 71
Regret analysis 73
3.2.6 Performance Evaluation 75
Simulations setup 75
Multi-armed bandit problem behavior 77
Slice SLA violation analysis 78
Convergence time 79
3.2.7 Experimental Proof Of Concept 81
Implementation 81
Experimental results 83
3.3 Conclusions 86

4 Data-driven Network Slicing Solutions 87
4.1 ARENA: A Data-driven Radio Access Networks Analysis of Football Events . 87
4.1.1 Related Work 89
4.1.2 Preliminary considerations 90
Standardization procedures 91
4.1.3 Data Analysis 92
Traffic Volume Patterns 93
Temporal Distribution of Mobile Users 94
Service Degradation 96
Inter-Feature Time Correlation 96
4.1.4 Model Design 98
Problem definition 99
Bandit convex optimization 100
4.1.5 ARENA: Design and Performance 102
Active users 102
Resource utilization 105
Performance evaluation 108
4.2 A Learn-as-You-Go Framework for On-Demand Emergency Slices in V2X Scenarios 110
4.2.1 Related Work 111
4.2.2 Data Analysis 111
Spatial consideration 112
Temporal patterns consideration 112
Service-based QoS consideration 113
Road events consideration 113
Event propagation effect 114
4.2.3 Design and Model Formulation 116
Input 117

LSTM-based anomaly detection 117
Road Event Localization 118
Road Event Classification 119
Performance Comparison and Practical Considerations 120
4.2.4 ENS orchestration 121
Problem design 122
V2X traffic scheduling 124
4.2.5 Performance Evaluation 124
Localization of the road event 124
Resource Orchestration 125
4.3 Conclusions 128

5 Conclusions and Future Work **129**

Bibliography **133**

List of Tables

2.1 End-to-end network slice template 28
2.2 Detailed HW components in our testbed 33

3.1 Tenant Categories 58
3.2 Notation table 64

4.1 Error characterization over different settings and models. 104
4.2 Event characterization and ENS Requirements. 120
4.3 V2X Slice requirements (c.f. [15, 16, 17]) 122

List of Figures

1.1 3GPP Network Slicing Architecture 5

2.1 Data plane 13
2.2 Control plane 15
2.3 Resource dynamics and resource (under-)provisioning. 16
2.4 (a)-(c): Networks from 3 European operators: red dots indicate the BSs' locations, black dots the routers/switches, and the green dot an edge CU (placed at the most central position). (d)-(e) Path capacity and delay distribution for the 3 networks. . 27
2.5 Relative revenue (percentage) of our approaches over `no-overbooking` in homogeneous scenarios. Variable mean load $\bar{\lambda}$. 30
2.6 Revenue of our approaches (colors) and `no-overbooking` (black) in heterogeneous scenarios. Mean load is $\bar{\lambda} = 0.2\Lambda$. 32
2.7 Testbed 33
2.8 Net revenue over time (a); and resource reservation and actual utilization across BSs (b), two transport links (c) and both CUs (d), respectively, for 9 heterogeneous slice requests arriving at different times. 34
2.9 Network slice brokerage overview 38
2.10 A distributed hierarchical architecture for network slicing. 39
2.11 Example of transaction message exchange within NSBChain. 42
2.12 Private blockchain architecture supporting network slicing. 43
2.13 Performance evaluation for different consensus algorithms and consortium size a) Slice request throughput and Blockchain size growth b) Validation latency. 46
2.14 Transaction acceptance and error rates for different scenarios. 47

3.1 Simplified Multi-access Edge Computing (MEC) architecture enhanced with the MEC broker. The diagram shows how different tenants can access the system via the legacy mechanism (Tenant 1) and the one proposed by the present solution (Tenant 2). 51
3.2 M^{2}EC Slicing Architecture 56
3.3 Host capacity utilization for K=40 and W=4 58
3.4 Simulation results. 59
3.5 Illustration of the network slicing concept. 61

3.6 Workflow illustration. 65
3.7 Radio channel variations as Markov chain. 67
3.8 Impact of different resource allocation chunk sizes. 75
3.9 Cumulative dropped traffic due to latency constraints violations. 76
3.10 (a) Effects of different slice requirements; (b) CDF of latency experienced by served traffic; (c) Empirical cumulative regret for a variable number of slices. 77
3.11 Sensitivity analysis of bandwidth availability and SNR variability on the convergence time to the optimal slice resource allocation. 78
3.12 Experimental setup (a); Architecture overview (b); Arm selection over time (c). . . 80
3.13 Comparison of system dynamics during a) discovery phase and b) MAB convergence. 82
3.14 Evaluation of different performance metrics for different scenarios. 85

4.1 Overview of the stadium area. 90
4.2 Normalized data volume distribution as a function of the number of active users ($\texttt{RRC}_{\texttt{CONNECTED}}$) for downlink (DL) and uplink (UL) and different periods of the day (morning, afternoon, evening, night). 92
4.3 Temporal evolution of the aggregated downlink (DL) and uplink (UL) traffic volume in the stadium area during a regular day (dashed line) and during the day of a large event (straight line). 93
4.4 Downlink PRB utilization as a function of the number of active users during the day of the event and during a regular day across three frequency bands managed by the eNB and different time periods (morning, afternoon, evening, night). 94
4.5 Evolution of users activity. Peak and average number of active users ($\texttt{RRC}_{\texttt{CONNECTED}}$) (outer plot) and peak-to-average ratio (inner plot) during the day of the event and during a regular day. 95
4.6 Evolution of the average downlink (DL) capacity per user (our QoS metric) during the day of the event (outer plot) and a regular day (inner plot). Highlight areas indicate the two periods of time when the sport event is occurring. 95
4.7 Overview of different metrics collected by two eNBs in the stadium area during an event day and weekday (Normalized values). 97
4.8 Temporal correlation of features in different working conditions. 98
4.9 Comparison of different LSTM-based architectures. 103
4.10 Comparison of different LSTM-based forecasting models working on real data from a base station in the stadium area (Normalized values). 105
4.11 Deep neural network architecture used in ARENA. 106
4.12 Impact of different hyperparameter settings on ARENA forecasting performances. . 107
4.13 Forecast of the number of active users or $\texttt{RRC}_{\texttt{CONNECTED}}$ (U_b) during a mass event (blue line) and actual evolution (dashed black line) for three sectors chosen for illustration purposes. Forecast on the radio resource usage (blue straight line), actual usage evolution (dashed black line) and estimation of extra radio resource allocation (green line) for the same sectors. 109
4.14 Overview of the Italian highway (A4). 112

4.15 Aggregated spatial distribution of DL/UL traffic volumes along the highway (top), with the focus on working day and weekend signatures for specific highway areas (bottom). 113
4.16 Temporal distribution of different traffic types volumes. 114
4.17 Spatio-Temporal distribution of road events. 115
4.18 Event propagation on nearby base stations in both directions for selected metrics. . 116
4.19 π-ROAD model architecture 116
4.20 ROC and PR diagrams with indicated AuC and AuPRC scores for different anomaly detection and classification approaches. 120
4.21 Example of road event detection and localization using π-ROAD model along the highway within one day, compared against the event ground-truth information. . . 125
4.22 Performance evaluation of a 4-slices scenario when an ENS is installed. 127

List Of Acronyms

AI Artificial Intelligence 5

AR Augmented Reality 110

BBU Baseband Unit 4

BS Base Station xix, 12–14, 17, 19–21, 27, 28, 33–35

BSS Business Support System 50

CNN Convolutional Neural Network 118–121

CPU Central Processing Unit 4, 13, 17, 28, 33–35

CU Computing Unit xix, 12, 13, 15, 17, 19–21, 27, 28, 33–35

eMBB Enhanced Mobile Broadband 1, 28–32, 35

EPC Evolved Packet Core 4

IaaS Infrastructure as a Service 49, 51

IB Intermediate Broker 36, 39, 41–45, 47

InP Infrastructure Provider 4, 36, 37, 39, 42, 44

IoT Internet of Things 1

KPI Key Performance Indicator 111, 112

LSTM Long Short Time Memory 116, 118

MEC Mobile Edge Computing 49–59

ML Machine Learning 5, 6, 87

MLP Multi-Layer Perceptron 107

mMTC massive Machine Type Communication 1, 28–32, 35

MVNO Mobile Virtual Network Operator 3, 36, 37

NaaS Network-as-a-Service 3

NFV Network Function Virtualization 2, 11, 12, 15, 16

NGMN Next Generation Mobile Networks 11

NSB Network Slice Broker 36, 37

OSS Operations Support System 50, 52, 53

OTT Over-the-Top 4, 37

PBFT Practical Byzantine Fault Tolerance 45

PRB Physical Resource Block 17, 33, 96, 97, 99–101, 107–109, 115, 121–124, 128

QCI QoS Class Identifier 113, 118, 126

QoE Quality of Experience 47, 52

QoS Quality of Service xx, 36, 88, 93, 95, 96, 98–100, 105, 108, 109, 131

RAN Radio Access Network 4, 8, 9, 12–14, 17, 33, 37, 38, 43, 44, 52, 88, 90

RRC Radio Resource Protocol 91

SDN Software Defined Networking 11, 13, 17

SLA Service Level Agreement 13, 14, 18, 29–32, 35–37, 49–52, 54, 59, 110, 122, 126, 127

SR Slice Request 45–48

TCP Transmission Control Protocol 14

TTI Time Transmission Interval 126

UE User Equipment 52, 54, 91, 95, 97

URLLC Ultra Reliable Low Latency Communication 1, 28–31, 35, 110, 126

VNF Virtual Network Function 2, 4, 12, 13, 15

VR Virtual Reality 110

VS Vertical Service 12, 14–16, 19, 34

Chapter 1

Introduction

The ever-increasing mobile traffic demand is pushing network operators to look for novel advanced solutions toward the deployment of future mobile networks. The widespread diffusion of smart devices supporting multimedia application and the high resolution streaming of online contents nowadays deeply impacts on the overall system utilization. Moreover, the number of devices demanding for mobile connection is envisioned to further increase according to the growth of the Internet of Things (IoT) market segment, which requires a synergy with mobile network facilities to efficiently deliver advanced services. For instance, the automotive sector is introducing significant innovations such as vehicular-to-anything communications that will dramatically augment the number of devices connected to the mobile network. This novel ”Internet of Everything” vision implies also ubiquitous, highly reliable, and ultra-low latency connections, that cannot be addressed with the current standard architectural features. The fifth generation (5G) mobile network architecture is expected to drive all these innovations, simultaneously supporting a wider set of market segments with respect to past generations networks. The service diversity in the 5G era could be resumed into three general categories: enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (URLLC), and massive Machine Type Communications (mMTC). eMBB focuses on services characterized by high data rates, such as high definition videos, virtual reality, and augmented reality. URLLC focuses on latency-sensitive services, such as self-driving, remote surgery, or drone control. mMTC focuses on services that have high requirements for connection density, such as those typical for smart city use cases. While such services might share commonalities, they pose heterogeneous requirements from the network standpoint. On the one hand, this service evolution brings several advantages paving the road towards new business models. On the other hand, new challenges need to be addressed. Amongst the others, the most critical one is the reduction of the time-to-the-market, referred as the vertical industries’ need of a faster service delivery. In conventional deployment process, a simple service update may take months. With such a long-term time window, it is difficult to envision a fast service provisioning demand for future networks. Nevertheless, programmability and multi-tenant capabilities ensure a fast deployment of new services. This includes the ability to create, sell and provision composite services in multi-domain environments. To support this flexibility and to provide the dynamic service mixture, fully automatic network management techniques, such as self-optimization and self-installation, are pillars to achieve efficient network operations.

In order to meet the expected capacity improvement and massive device connectivity, 5G centers its design objectives around efficiency, scalability, and versatility. To sustain its commercial viability, 5G networks must be significantly efficient in terms of energy, resource management, and cost per bit. Connecting a massive number of terminals requires the development of scalable and versatile network functions that cope with a wider range of service requirements including: low power, low data rate communication, high data rate multimedia, and delay-sensitive applications, among many other services. Also, 5G must coexist with legacy technologies like 2G, 3G, and 4G. This requirement alone increases cost and complexity. These challenges can be addressed by implementing the 5G network functions as software components using the network functions virtualization (NFV) paradigm. In the past, a given service was provisioned to the end user and managed in a static or semi-automatic way, mainly by operators. From few basic services, we ended in a multitude of very different services that are nowadays provided through the same network. This change was possible thanks to the embracement of the cloud computing paradigm in the mobile ecosystem, which started the renewal process of mobile architecture from monolithic to modular. The concept originated from service providers who were looking to accelerate the deployment of new network services to support their revenue and growth objectives. The constraints of hardware-based appliances led them to applying standard virtualization technologies to their networks, which allows computing power and storage to be shared and offers far greater flexibility. Traditionally, mobile network functions are readily grouped into network entities, each responsible for a predefined set of functions, and interfaces connecting these entities. Using a flexible "network of functions" allows adaptation to diverse services, and optimization using different software rather than using different parameterizations. Each block may be replaceable and could be individually instantiated for each logical network running on the same infrastructure. In NFV, operators implement network functions in software components called virtual network functions (VNFs). VNFs are deployed on high-volume servers or cloud infrastructure instead of specialized hardware. The decomposition of the mobile network functionality would imply a stronger decoupling of logical and physical architecture that is, physical network functions may be executed on a non-virtualized local hardware, while virtual network functions (VNFs) may be executed on local or remote data centers. The main benefit of the described architecture is the opportunity to exploit centralization gains where possible and to optimize the network operation depending on the actual network topology.

The dynamic network slicing concept leverages NFV to create many dedicated end-to-end virtual networks. All end-to-end network slices are created and operated over a common physical infrastructure. A slice is self-contained. It has all the functions and capabilities, appropriately chained together to best meet all the needs of the corresponding services and use-cases. An end to end network slice orchestrator is envisioned to manage all aspects of network slicing. Such an orchestrator will help to manage the life-cycle of the slices, from their creation and setup to the shut-down. This would be applied at different levels and at different times. Network resources would be allocated to a slice based on a trade-off between guaranteeing resources to an individual slice and the advantage of pooling resources from all slices. In this way the fifth generation mobile network could support a multitude of new services and applications with very diverse requirements, mainly related to higher traffic volume, lower latency, due to the huge forecast number of devices that will populate the system. This new Network-as-a-Service (NaaS) paradigm provides new business opportunities by enabling mobile operators to open their network infrastructure to multiple tenants: With network slicing, very heterogeneous services can be provided by the same infrastructure as different network slice instances, where each of these instances consists of a set of virtual network functions that run on the same infrastructure, orchestrated and configured according to specific requirements.

1.1 5G Network Slicing

Flexibility, reactivity and low-complexity have been identified as key-features for ensuring enhanced performance figures. In this context, network virtualization and softwarization concepts represent a turning point in the cellular network design. Starting from physical network components, multiple virtualized functions can be built on top of a common network infrastructure. Functions can be easily combined and placed dynamically to tailor very heterogeneous service requirements as different network slice instances. The result is a novel paradigm, namely Network Slicing, where the overarching pool of network functions and resources is available to be chained into a network slice that can be used, for instance, to improve the figure of merit of specific traffic classes with respect to others. Network operators may create virtualized on-demand isolated and efficient end-to-end networks fully (or partially) dedicated to their customers. The business model could be further enriched introducing new players into the market, such as Mobile Virtual Network Operators (MVNOs), third-party applications, automotive industry and vertical market segments, which may lean a fixed amount of network resources for a limited time interval based on their needs. Additionally, since network slice tenants dynamically share the same infrastructure paying off different prices, this could be performed in a cost-efficient manner. For network operators which aim to decrease capital expenditure and operational expenditure costs (CAPEX/OPEX), this can turn into a very attractive solution.

1.1.1 Admission Control

The novel network slicing paradigm is envisioned to deeply impact the mobile market ecosystem, breaking the traditional business model of a single network infrastructure ownership, and opening the door to new and unexplored sources of revenue, finally allowing telco operators to offer virtualized slices of infrastructure resources on-demand to heterogeneous 3^{rd}-party industry verticals or Over-the-Top (OTT) service providers, referred as *tenants*. The network slicing business model envisions on the one side the Infrastructure Provider (InP), which owns the physical networking equipment composing the mobile infrastructure, and on the other, the tenants, which may be willing to acquire a dedicated slice of the mobile network from the InP, in order provision their customers with mobile services characterized by guaranteed performance metrics. Nevertheless, network infrastructure resources are limited, and this calls for novel admission control schemes to manage incoming slice requests. Upon receiving a network slice request, the infrastructure provider should map slice request SLAs onto own network resource availability. Several admission control strategies can be implemented, addressing diverse objectives, such as revenue maximization, optimization of network capacity multiplexing gain, high responsiveness and so on. This sheds the light on novel online selection schemes which must carefully take into account previous taken decisions, current system utilization and upcoming future slice requests. The state-of-the-art lacks such an improved mechanism able to optimally take online decisions. We envision this solution implemented into a novel entity interposed between external tenants and the network infrastructure, dealing with external slice requirements coming from vertical industries as well as with the network management requisites. We name this entity as *Network Slice Broker*, which collects all incoming network slice requests accommodating them while pursuing the goal of system resource utilization maximization. As soon as tenants are not fully utilizing network slice resources, an exploitable gap could be further identified by the network operator for resource over-provisioning mechanisms. The idea behind is to constantly monitor the effective slice resources utilization and assigns more resources than the current overall system availability. This feature enhances our final goal by considering the system multiplexing gain as a new target. The larger the network slice resource gap, the higher the multiplexing gain, the more profitable the network slicing brokering solution.

1.1.2 Network Orchestration

Upon admission in the mobile infrastructure, network slices should be provided with enough resources to provision their vertical service. From an infrastructure perspective, given the softwarization trend encompassing the mobile ecosystem these resources will be prevailing of computational nature, e.g., CPU at the cloud of the network where mobile core functionalities like the Evolved Packet Core (EPC) can run as a set of VNFs or containers, or CPU at the RAN where baseband units (BBU) run signal processing functions to decode the information exchange occurring over the radio wireless channel.

In this context, the network operator is in charge to proactively define the amount of resources that should be dedicated to each network slice, ensuring that all the hosted slice are satisfied while pursuing the minimization of operational expenses. The highly heterogeneous set of modern mobile services, each one characterized by specific and stringent networking requirements, together with the variety of networking resources that need to be efficiently shared among multiple tenants, e.g., radio spectrum, transport bandwidth, computing power, etc., exacerbate the need for novel orchestration solutions able to cope

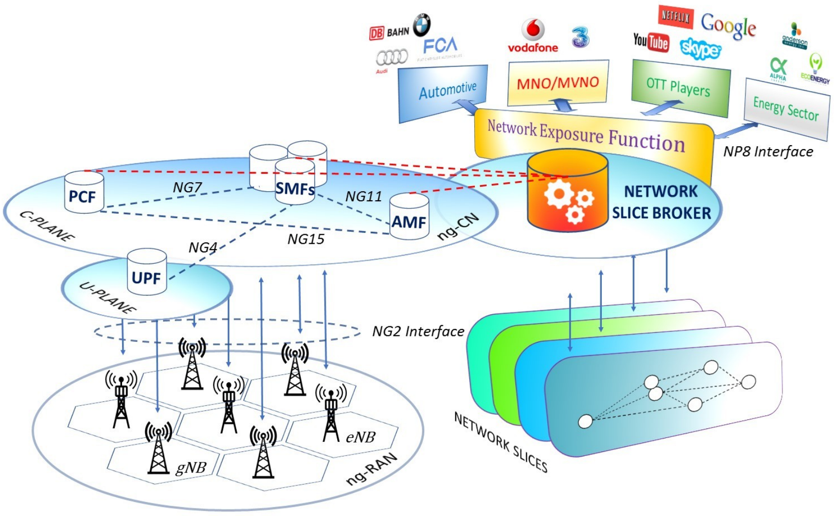


Figure 1.1: *3GPP Network Slicing Architecture*

with such resource allocation problem. Artificial Intelligence (AI) and Machine Learning (ML) have been identified as key-technologies in this context, providing the means to efficiently automate the management of mobile network resources in response to real-time trigger events, e.g., user mobility or traffic load variations. This thesis investigates the applicability of AI into slice-enabled mobile network, as a means to seamlessly optimize resource management and allocation aspects.

1.1.3 Network Slice Broker

To allow the coexistence of many and very different kinds of slices, each one with a specific traffic footprint, such as throughput, latency and QoS, the next-generation network design envisions the introduction of advanced mechanisms for slice admission control and dynamic resource orchestration. One of the critical aspect is the network slices isolation. It must be guaranteed to prevent security threats as well as to bound the slice resource utilization within a single slice to avoid service degradation for other running slice traffic flows.

3GPP has defined a novel network architecture for network slicing support. In particular, the 3GPP working group SA2 [18] has already defined the basis for building an evolved core network infrastructure managing multiple slices on the same network infrastructure. The envisioned architecture is depicted in Fig. 1.1 which clearly differentiates between control plane (C-Plane) and user plane (U-Plane). In the control plane, new components are introduced to manage user authentication and registration (AMF), support multiple connection sessions (SMF), and instruct different routing policies (PCF). On the other hand, the traditional core user plane functionalities are decentralized towards multiple dedicated User Plane Functions (UPFs) managing distinct data networks (DNs) through the next-generation Radio Access Network (ngRAN), therefore allowing for packet processing and traffic aggregation at the network edge. This new architecture builds on network functions virtualization and also enables flexible multi-tenant deployments. In fact, RAN resources can be virtualized and dynamically chained to provision end-to-end slices with a dedicated SMF [19]. Interestingly, AMF (and PCF) can be still be shared among multiple slices when presenting service requirements commonalities.

Based on this architecture, the Network Exposure Function (NEF) can be used as a direct interface between the mobile network operator and the network slice tenants to access the virtualized network functions [20]. NEF is envisioned to expose a list of available slice templates defining specific functions to be instantiated for given service requirements. Network slice request coming through the NP8 interface will then indicate the requested slice template based on the available ones. At this point, an arbitration entity is needed to grant (or deny) network slice requests. Once a network slice request is granted, a Network Slice Selection Assistance Information (NSSAI) indicator is propagated through all network components and advertised to incoming UEs through the RAN [21]. Based on the NSSAI, the AMF will select the SMF and a network slice will be successfully installed. Associated UEs might then indicate in the RRC signaling the NSSAI to be used for serving its traffic. In Fig. 1.1 we depict the proposed location of the arbitration entity in charge of granting or denying network slice requests, referred as *Network Slice Broker*.

1.1.4 Privacy and security

As detailed in the following of this thesis, the significant amount of monitoring and operational data generated by the different networking domains composing the mobile infrastructure allows ML-based models to gather substantial insights of real-time networking processes, including spatio-temporal characteristics of cellular traffic and end-users mobility, as well as complex relationships relating different mobile network metrics, which may be adopted to perform more accurate decisions. Thanks to the latest development in the field, mobile network resource management decisions that previously took slow human interactions can now be performed in an autonomous way by algorithms with a holistic view of the network, enabling software components to directly contribute to decision-making activities in a faster and automatic way. Through ML-based techniques, accurate resource planning can be enforced based on both historical and forecast information, therefore allowing for significant energy reduction and resource consumption savings. Real-time and low-granularity data information play a central role in the adoption and successful implementation of such solutions, however privacy issues may arise. In the context of this work, whenever aggregated end-user and/or mobile infrastructure level data were used, privacy and data management security regulations defined by the EU General Data Protection Regulation (GDPR)[1] have been strictly followed.

---

[1] General Data Protection Regulation-GDPR. Available online: https://gdpr.eu/ (Accessed on the 1st of September 2021).

1.2 Contributions

This thesis investigates resource orchestration and management in next generation mobile networks, as well as the applicability of machine-learning based solutions towards zero-touch management and automatic mobile network resource provisioning. The ideas, figures, tables and results included in this thesis are taken from several scientific papers, listed at the beginning of this thesis, published in international peer-reviewed conferences and journals.
More in details, the main contributions of this thesis can be summarized as follows:

1. **A framework for network slices admission and control.** We design and implemented a platform for the management of network slice requests, including multiple mobile network domains. Our solution allows to maximize the revenue of mobile operators when dealing with the admission of new network slices in a shared mobile infrastructure.

2. **An end-to-end framework for slice resource orchestration.** We design a hierarchical control plane to manage the orchestration of slices in an end-to-end fashion, including radio access, transport network, and distributed computing infrastructure.

3. **A blockchain-based solution for mobile network resource brokering**. We propose a novel network slicing brokering (NSB) solution, which leverages on the widely adopted Blockchain technology to address the new business models needs beyond traditional network sharing agreements. NSBchain defines a new entity, the Intermediate Broker (IB), which enables Infrastructure Providers (InPs) to allocate network resources to IBs through smart contracts, as well as to assign and re-distribute such resources among tenants in a secure, automated and scalable manner.

4. **A MEC-specific solution for multi-tenant platform administration.** We introduce the concept of MEC broker as an entity exposing administration and management capabilities while handling heterogeneous tenant privileges. Our concept is validated by developing an orchestration solution, namely M2EC, to optimally allocate requested resources in compliance with the tenants service level agreements.

5. **A machine-learning based solution for RAN slicing with latency control.** We propose a novel radio slicing orchestration solution that simultaneously provides latency and throughput guarantees in a multi-tenancy environment, that makes adaptive resource slicing decisions with no prior knowledge on the traffic demand or channel quality statistics.

6. **A data-driven analysis of RAN metrics in crowded scenarios.** Based on data provided by a major European carrier during mass events in a football stadium, we performed a data-driven analysis of the radio access network infrastructure dynamics during such events. Given the insights obtained from the analysis, we developed a

model-free deep learning Radio Access Network (RAN) capacity forecasting solution that, taking as input past network monitoring data and events context information, provides guidance to mobile operators on the expected RAN capacity needed during a future event.

7. **A deep learning framework for road event classification and emergency slice orchestration.** Based on real RAN traces from a major italian highway, we develop a deep learning framework to automatically learn regular mobile traffic patterns along roads, detect non-recurring events and classify them by severity level enables operators to proactively instantiate dedicated Emergency Network Slices (ENS) as needed while re-dimensioning the existing slices according to their service criticality level.

1.3 Thesis Outline

The outline of the thesis is organized as follows:

Chapter 2 details the design of a hierarchical framework to manage the orchestration of network slices end-to-end, i.e., including radio access, transport network, and distributed computing infrastructure. In particular, Section 2.1 first introduces the overbooking concept, including a solid mathematical model formulation guiding the overall procedures. Additionally, it also showcases our implementation of an experimental proof-of-concept composed by real networking devices, assessing the performances of our approach both experimentally and via simulations. Afterwards, in Section 2.2 we build on the emerging blockchain technology and describe a solution that enables seamless network resource brokering among network slice tenants, evaluating its performances in realistic scenarios.

Chapter 3 focuses on domain specific solutions. In particular, Chapter 3.1 introduces the concept of *MEC broker* as an entity exposing administration and management capabilities while handling heterogeneous tenant privileges in Multi-Access Edge Computing scenarios, while Chapter 3.2 focuses on the RAN domain, and introduces LACO, an orchestration solution for radio slicing that simultaneously provides latency and throughput guarantees. Leveraging on a solid mathematical framework, we exploit the exploration-vs-exploitation paradigm by means of a multi-armed-bandit-based (MAB) orchestrator that makes adaptive resource slicing decisions with no prior knowledge of the traffic demand and channel quality statistics.

Chapter 4 provides an analysis of RAN monitoring data taken from operational mobile networks owned by major european operators, and introduce ARENA and π-ROAD as concrete examples of the advantages that machine-learning can bring when dealing with mobile network resource orchestration and anticipatory resource forecasting. In Section 4.1, we focus on one of the most challenging scenarios for mobile networks, i.e, mass events, and detail ARENA, a model-free deep learning RAN capacity forecasting. In Section 4.2, we focus on vehicular traffic and emergency scenarios devising a deep learning framework, π-ROAD, which automatically learns regular mobile traffic patterns along roads, detects non-recurring events and classifies them by severity level, eventually enabling proactive slice resource allocation for emergency slices.

Finally, **Chapter 5** concludes this thesis summarizing the presented work and discussing the limitations of the contributions, as well as possible directions for future research and extensions of the topics addressed in this thesis.

Chapter 2

Network Slicing Admission Control and Resource Brokering

2.1 Overbooking Network Slices End-to-End

The hype around software-defined networking (SDN) and network function virtualization (NFV) is the projection of a trend towards network *softwarization* and *programmability* that is blending together telecommunication and computing industries. This combination has a deep impact on mobile communications infrastructure that is yielding a transformation from relatively complex monolithic architectures into a flurry of *commoditized* networking, computing and radio resources [22, 23]. Clearly, the need of mobile operators to augment their revenue is a strong pull towards said convergence, spawning uncharted sources of monetization as a result. Namely, the availability of *cloudified* networking, computing, and radio resource pools can now be offered, via proper abstractions, to *vertical* sectors (e.g., automotive, health, construction)—traditionally alien to the telco sector—as a means to enable new services such as remote-controlled machinery, augmented/virtual reality (AR/VR), etc. [24, 25].

In this context, *Network Slicing* appears as a key solution to accommodate these emerging business opportunities in next generations of mobile systems [26]. The Next Generation Mobile Networks (NGMN) Alliance defines a network slice as "*a set of network functions, and resources to run these network functions, forming a complete instantiated logical network to meet certain network characteristics required by the service instance(s)*" (c.f. [27]). Inspired by recent advances on SDN and NFV, this concept shall provide the required tools to allocate (virtual) resources to 3$^{\text{rd}}$-parties in an isolated, flexible and guaranteed manner. It thus becomes evident that the orchestration of resources *end-to-end*[1] is, albeit challenging, a requirement in order to provision network slices with (i) spectrum at radio sites, (ii) transport services in the backhaul and (iii) computing/storage at

---

[1] With *end-to-end*, we refer to all domains of the mobile network ecosystem, including network/storage/computing/radio resources. Domains beyond the ownership of a mobile operator, e.g., Internet Service Providers (ISPs), are not considered.

distributed computing clouds. Nevertheless, its benefits are compelling. Network Slicing leads mobile operators towards business models that, perhaps surprisingly, have a similar nature to successful yield management strategies popular in areas such as airline or hotel industries, and promise substantial gains in the revenue attained to mobile investments.

In particular, in this chapter we explore the concept of *slice overbooking*, accommodating the common practice in airline services of intentionally allocating more cargo than available capacity to the allocation of mobile network slices for 3$^{\text{rd}}$-party services.

The challenge to adopt an orchestration system based upon the concept of slice overbooking is threefold: (i) when doing overbooking, resource deficit (and thus violations of system-level agreements) may occur; and so, in order to maintain the incentives for 3$^{\text{rd}}$-parties (users) to join the system, a balance between overbooking and potential service disruption must be taken care of; (ii) we need to untangle the coupling between resource reservation *and* slice admission control decisions, which is further compounded by the heterogeneous nature of the resources required to build a slice across the whole system; (iii) we need to make an appropriate use of monitoring information to be able to adapt to behavioral dynamics of 3$^{\text{rd}}$-party services embedded in network slices.

2.1.1 System Design and Model

We now introduce the design of our system and a mathematical model that allows us to make orchestration decision. Our system has decoupled control and data planes. The data plane is comprised of base stations, switches and computing infrastructure. In the control plane, we have a hierarchical architecture where local domain controllers are governed by an end-to-end (E2E) orchestrator.

As depicted in Fig. 2.1, we consider a system with a radio access network (RAN) comprised of $\mathcal{B} := \{1, \ldots, B\}$ base stations (BS), a distributed computing fabric with $\mathcal{C} := \{1, \ldots, C\}$ computing units (CUs), and a transport network connecting BSs and CUs that we model as an undirected graph where the edges, collected in set $\mathcal{E}$, are network links.

2.1.2 Service model

We allow tenants to deploy their services, dubbed vertical services (VSs), within a slice of the system. Such VSs are provided by the tenant in an offline on-boarding phase, e.g., as virtual machines (VMs). The first task to create a slice is to construct a *network service* (NS) with sufficient computing resources allocated to the VS (and related mobile functions), connectivity in the transport network, and spectrum resources at radio sites to enable VS access to the tenant's users. To this aim, we model such network service as an ETSI NFV NS [28], with a chain of physical network functions (PNFs, e.g., slices of BSs and switches), the VS and all virtual network functions (VNFs) that connect end-users and VS (e.g., GTP gateways, MME, etc.). This is shown in Fig. 2.1.

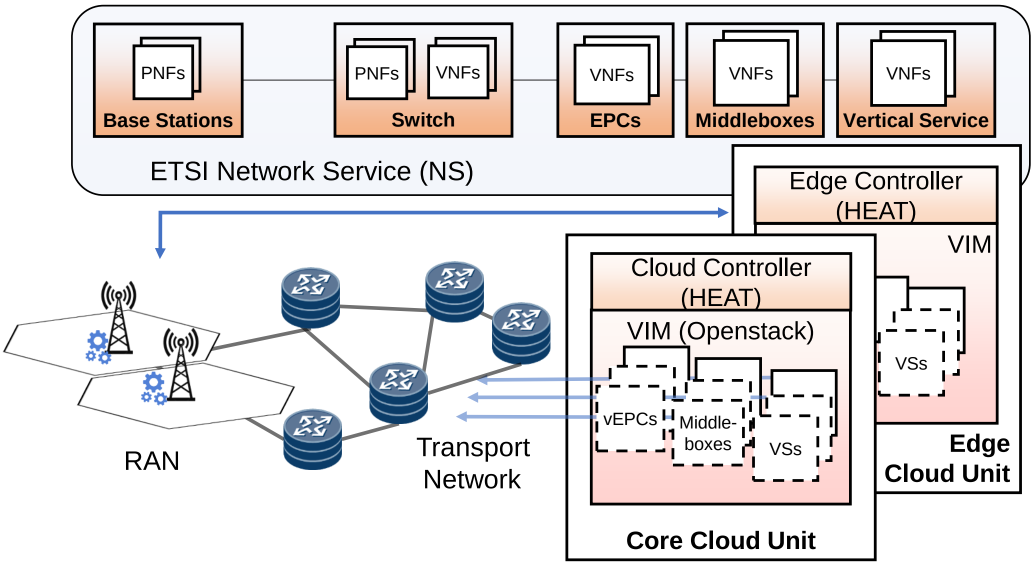


Figure 2.1: *Data plane*

2.1.3 Resources

We assume BSs with RAN sharing or slicing support (e.g. [29]), an SDN-based transport network and OpenStack as compute infrastructure manager (although other cloud managers can be accommodated). BSs, network links and CUs are characterized by a *capacity* value C_b, C_e and $C_c \in \mathbb{R}_+$ indicating, respectively, the maximum amount of radio resources (spectrum chunks), transport network resources (bits per second) and computing resources (shares of aggregated CPU pools)[2] that can be allocated to a service in BS $b \in \mathcal{B}$, network link $e \in \mathcal{E}$ and CU $c \in \mathcal{C}$. To keep our problem tractable, we assume that the microscopic problem of selecting a server for a VNF within a CU is handled locally by a cloud orchestrator (e.g., Heat),[3] and focus in this chapter on the macroscopic problem of jointly optimizing (i) slice access control, (ii) CU selection, and (iii) reservation of resources across the system for the NS. Now, we let $p_{b,c} = \langle e_1, e_2, \cdots \rangle$ be a sequence of links $e_i \in \mathcal{E}$ connecting BS b and a CU c (i.e., a *path*) and $\mathcal{P}_{b,c}$ be a set with all possible available paths $p_{b,c}$. This can be readily computed offline using, e.g., k-shortest path methods based on Dijkstra's algorithm. Each path $p \in \mathcal{P}_{b,c}$ is further characterized with a delay D_p.

Middleboxes

We rely on an overbooking mechanism that adapts the reservation of resources to the actual demand of each slice (or a prediction of it) as explained later on. However, we may violate service-level agreements (SLAs) when making overly optimistic predictions (slice overbooking). In these cases (which we strive to minimize), it is important to avoid

---

[2] To avoid notation clutter, we focus on compute resources only; however our model can be readily extended to consider others such as storage.

[3] We refer the reader for more details on the microscopic issue to [30].

perturbations of the transmitter's behavior. If we simply delayed or dropped packets, TCP's transmission control of end-users would react in an undesirable manner. Hence, we need a scheme to under-provision resources that is also transparent to the tenant's users.

TCP proxies are nowadays common in many service gateways and load balancers in operational networks to improve throughput performance, enhance security, perform network analysis and traffic control [31, 32]. In our system, we exploit basic TCP proxy functionality in a *middlebox* as depicted in Fig. 2.1. Our proxy creates a TCP overlay network splitting each connection into two as per Split TCP [33]: the former between the service of the slice and the middlebox, and the latter between the middlebox and the end-user(s) of the slice *where we do rate control*. If the slice's (aggregate) load exceeds the SLA, packets are randomly dropped to adjust the rate to the SLA. If the load is within the SLA parameters *and* below the maximum network capacity reserved for the slice (as detailed later), the middlebox simply forwards packets transparently. Finally, if the load is within the SLA parameters *but* it exceeds the network capacity reserved for the slice, the middlebox buffers packets to adjust the rate to the reserved capacity. Buffered packets are immediately acknowledged back to the service and then transmitted to the final user upon capacity availability. This avoids that the rate controller of the transmitter's TCP implementation reacts to our traffic control actions when the load is within the tenant's SLA.

2.1.4 Control Plane

Our control plane is depicted in Fig. 2.2. At the top of the hierarchy, a *slice manager* interacts with the tenants and oversees the setup of a NS for the slice. In the middle, the *end-to-end orchestrator* embeds most of our system's intelligence and is in charge of performing access control and resources reservation activities for the slices all across the mobile system, and interacts with domain *controllers* (RAN, transport, cloud) to deploy the NS, accordingly.

2.1.5 Slice Manager

We consider a time slotted system whereby time is divided into *decision epochs* $\langle 1, 2, \dots \rangle$. Tenants issue slice requests to the slice manager at any time within one decision epoch.[4] We then let $\mathcal{T}^{(t)}$ be the set of tenants requesting a slice in epoch t.

Each slice request is characterized by $\Phi_\tau := \{s_\tau, \Delta_\tau, \mathbf{\Lambda}_\tau, L_\tau\}$. s_τ is a function that maps the network load received by tenant τ's VS into computing requirements (details later). Δ_τ describes the latency tolerance between τ's service and any BS, and $\mathbf{\Lambda}_\tau = \{\Lambda_{\tau,p} \mid \forall p \in \mathcal{P}_{b,c}, b \in \mathcal{B}, c \in \mathcal{C}, \Lambda_{\tau,p} \in \mathbb{R}_+\}$ captures the service bitrate requested for τ's service at each radio site. Finally, L_τ is the duration of the slice. Should the slice be accepted, Φ_τ becomes an SLA between the tenant and the operator during L_τ intervals.

---

[4] We assume it as an adjustable parameter, e.g., based on (off-)peak hours [34, 35] that may trade off the forecast accuracy and speed of reaction.

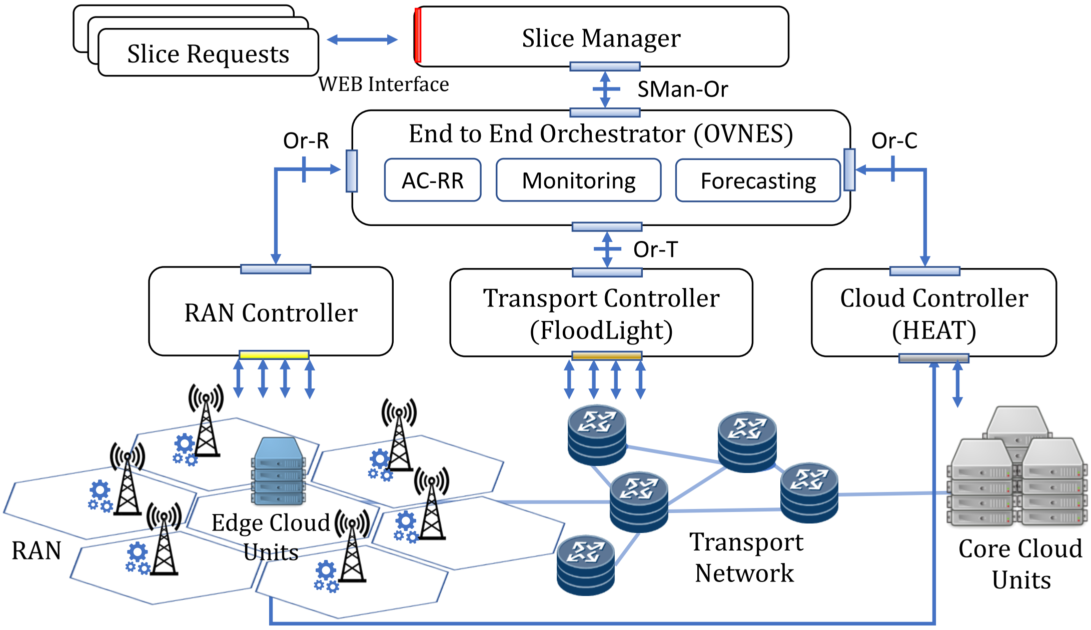


Figure 2.2: *Control plane*

From the implementation perspective, we build our slice manager as a web app where tenants can introduce their Φ_τ requests. Internally, we use `TOSCA` templates to model NSs as shown in Fig. 2.1, and send it down to the E2E orchestrator using a `REST` interface.

2.1.6 E2E Orchestrator

This is the main building block of our system. On the one hand, it processes monitoring data provided by each controller and provides data aggregation functions and forecasting algorithms. On the other hand, it makes judicious decisions regarding resource reservation and admission control, and interacts with the different controllers in order to enforce such decisions. From a software perspective, and to prove our concept, we develop our own orchestrator in `Java`.[5] This is the only entity that maintains system *state* information. All the remaining entities (i.e., slice manager, controllers) are stateless in order to guarantee consistency. As shown in Fig. 2.2, the main functional sub-blocks (connected by means of a `REST` interface) are the following:

Admission Control and Resource Reservation (AC-RR). At the beginning of each decision epoch t the AC-RR engine has to (i) decide which slices are accepted among those requests arrived during the previous decision interval, (ii) select a CU to instantiate the VNFs/VS of each NS, and (iii) make radio/transport/compute resource reservations

---

[5] We acknowledge the fact that there exists a plethora of software projects developing NFV orchestration tools (Tacker, OSM, Cloudify, etc.). We advertise that none of the tools accommodate our needs in full and thus we develop our own for the purpose of this chapter. As future work, we aim to integrate our concept within a mainstream orchestration platform.

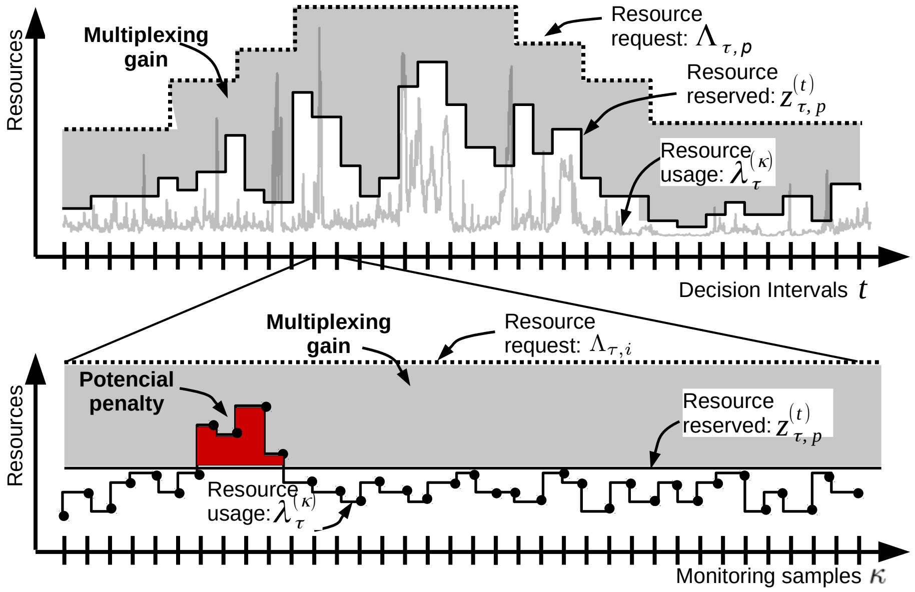

Multiplexing gain
Resource request: $\Lambda_{\tau,p}$
Resource reserved: $z^{(t)}_{\tau,p}$
Resource usage: $\lambda^{(\kappa)}_{\tau}$
Resources
Decision Intervals t
Multiplexing gain
Resource request: $\Lambda_{\tau,i}$
Potencial penalty
Resource reserved: $z^{(t)}_{\tau,p}$
Resource usage: $\lambda^{(\kappa)}_{\tau}$
Resources
Monitoring samples κ

Figure 2.3: *Resource dynamics and resource (under-)provisioning.*

across the system (i.e., make an infrastructure slice) *in order to maximize the net revenue* obtained from the tenants. To this aim, we let $x_{\tau,p}^{(t)}$ denote whether tenant τ is granted access to path p ($x_{\tau,p}^{(t)} = 1$) or not ($x_{\tau,p}^{(t)} = 0$); if slice Φ_τ is rejected, then $\sum_p x_{\tau,p}^{(t)} = 0$. Let us also define $z_{\tau,p}^{(t)}$ as the resource reservation for tenant τ, in terms of bitrate, when using path p, as illustrated in Fig. 2.3 (top).

Importantly, $z_{\tau,p}^{(t)}$ is not necessarily the amount of transport resources reserved in path p (there are transport overheads we need to account for), but the bitrate associated to the service when using this path. Based on $z_{\tau,p}^{(t)}$, however, we derive the reservations of radio, transport and compute resources for slice Φ_τ. For notation convenience, we vectorize $x_{\tau,p}^{(t)}$ and $z_{\tau,p}^{(t)}$ into $\boldsymbol{x}^{(t)} \in \{0,1\}^{\mathcal{S}^{(t)}}$ and $\boldsymbol{z}^{(t)} \in \mathbb{R}_+^{\mathcal{S}^{(t)}}$, where $\mathcal{S}^{(t)} := \sum_{b\in\mathcal{B}}\sum_{c\in\mathcal{C}}\sum_{p\in\mathcal{P}_{b,c}} |\mathcal{T}^{(t)}|$.

In order to make decisions, we formalize our problem as a yield management problem (§2.1.8) and devise two algorithms to solve it (§2.1.13). As a result, the `TOSCA` NS descriptors are modified accordingly and passed down to the different domain controllers through a `REST` interface that follows closely the ETSI GS NFV-IFA 005 specification.

Monitoring and Feedback. We further divide the time window between two decision epochs into $\kappa^{(t)} := \langle 1, 2, \dots \rangle$ *monitoring samples*. As depicted in Fig. 2.3 (bottom), the monitoring function collects VS network load samples in sequences $\langle \lambda_{\tau,p}^{(\theta)} \mid \theta \in \kappa^{(t)} \rangle$ for every epoch t. With a slight abuse of notation, we let $\lambda_{\tau,p}^{(t)} = \max\left\{ \lambda_{\tau,p}^{(\theta)} \mid \theta \in \kappa^{(t)} \right\}$ denote the maximum demand of resources during epoch t. This value can be computed for past epochs $\{1, \dots, t-1\}$ but it is unknown in the current one. Note that we use `max` to account for peak aggregate loads that will allow us to minimize our under-allocation footprint. Therefore, we let $\hat{\lambda}_{\tau,p}^{(t)}$ denote the estimated (predicted) value for epoch t, and $0 < \hat{\sigma}_{\tau,p}^{(t)} \leq 1$ denote the level of uncertainty of such prediction. This is performed by the Forecasting sub-block, explained below.

In addition to service demand, another source of uncertainty is the wireless channel capacity. To model this, we let $\eta_{\tau,b}^{(t)}$ be a factor that maps radio spectrum (physical resource blocks (PRBs)) into actual load injected into the transport network (bits per second) for tenant τ and BS b at epoch t. Note that $\eta_{\tau,b}^{(t)}$ depends mostly on the average signal quality between users and BS, which can be monitored with conventional utilities and then estimated using standard radio models.

From our implementation perspective, we use `sFlow` to collect service load samples, `OpenStack Ceilometer/Gnocchi` to collect computing/storage monitoring data, and a proprietary protocol to gather signal quality samples from the RAN. Finally, we exploit `InfluxDB` to store time-series data and a `MySQL` database to save additional control plane information, e.g., current state of each slice.

Forecasting. This block processes the measurements (observations) performed during previous decision epochs t and provides the forecasting information to drive the system towards optimal states. In particular, we focus on a specific class of machine-learning algorithms that learn and predict the future traffic behaviors $\hat{\lambda}_{\tau,p}^{(\delta)}$ for the next N decision intervals, i.e., $\delta \in \{t+1, \ldots, t+N\}$. Exponential smoothing methods are common to properly handle future resource provisioning in cloud computing environments. However, the main drawback of (double) exponential smoothing is the inability to account for seasonabilities. Hence, our forecasting algorithm is based on a three-smoothing function.[6] This accurately applies to our problem as mobile data has periodicity features [37] that can be exploited to provide predicted traffic levels with a certain accuracy $\hat{\sigma}_{\tau,p}^{(\delta)}$. Therefore, we rely on the multiplicative version of Holt-winters (HW) algorithm [38], where the forecasting function f_{HW} is defined as $f_{HW} : \mathbb{R}^{|t-1|} \to \mathbb{R}^{|t+\delta|} \mid \lambda_{\tau,\mathbf{p}} \to \hat{\lambda}_{\tau,\mathbf{p}}$.

2.1.7 Controllers

As depicted in Fig. 2.2, our orchestrator interacts with domain controllers to enforce orchestration decisions and to retrieve monitoring information. At the northbound of the Cloud controller, we translate the received `TOSCA` descriptor into a `Heat` template and send it down to a driver that interfaces with `OpenStack Heat` and `Keystone` for proper instantiation and CPU reservation (using CPU pinning [39]). Similarly, at the northbound of the Transport controller we translate the `TOSCA` descriptor into a series of OpenFlow instructions that are processed with `Floodlight` SDN controller to set up paths between BSs and CUs with appropriate capacity. Finally, we use the same descriptor file to configure radio shares of commercial LTE base stations, wherein each slice is connected to a different mobile core.

2.1.8 Admission Control & Resource Reservation (AC-RR) Problem

Maximization of a business' revenue falls into the category of *yield management*, a mainstream business theory that studies fare management, access control and resource allo-

---

[6] Naturally, we can seamlessly plug in alternative forecasting methods, e.g., recent approaches based on neural networks [36].

cation [40]. In the airline industry, the problem is to decide, based on the number of seat reservations, whether to accept or reject new requests considering that passengers may cancel, or even be "no-shows", prior to the flight departure. Thus, overbooking is performed with associated penalties determined by a penalty-cost function. Owning to similar business nature, we cast our slice orchestration problem into a stochastic yield management optimization problem.

2.1.9 Design of the objective function

Analogously to the airline example, we exploit the fact that users rarely consumes *all* the resources they request [41]. This gives us the opportunity to allocate more tenants than those presumably allowed by the leftover capacity, and gain additional revenue from slice multiplexing (overbooking). Clearly, an overly aggressive strategy may lead to resource deficit, discouraging potential users to join the system. We address this by (i) considering (forecasted) peak loads at each interval and (ii) designing a proper penalty-cost function. Consequently, we define

$$\psi^{(t)} := \sum_{\tau \in \mathcal{T}^{(t)}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in \mathcal{C}}} \overbrace{K_\tau \Pr\left[z_{\tau,p}^{(t)} < \lambda_{\tau,p}^{(t)}\right] x_{\tau,p}^{(t)}}^{\text{Expected penalty}} - \overbrace{R_\tau x_{\tau,p}^{(t)}}^{\text{Reward}}$$

as the expected *instantaneous* cost in epoch t, and define

$$\min_{\boldsymbol{x} \in \{0,1\}^{S}, \boldsymbol{z} \in \mathbb{R}_{+}^{S}} \quad \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \psi^{(t)} \tag{2.1}$$

as our optimization problem, where R_τ is the reward obtained from accepting slice Φ_τ (e.g., subscription fee) and K_τ is a penalty paid to tenant τ when we violate its SLA,[7] which happens with probability $\Pr\left[z_{\tau,p}^{(t)} < \lambda_{\tau,p}^{(t)}\right]$. The target is to asymptotically minimize the aggregate cost or, equivalently, maximize the net reward.

A possible approach to solve this problem is to model $\lambda_{\tau,p}$ as a random variable with known distribution, and estimate its parameters looking at the realizations. This falls into the realm of stochastic programming where the aim is to balance reward maximization (right-hand side of $\Psi^{(t)}$) with the cost of a recourse action (left-hand side). However, in practice, $\lambda_{\tau,p}$ may be characterized by an intractable distribution and/or discretization may lead to overly complex computation. Hence, we adopt a more practical approach.

First, we assume that the duration of a slice L_τ is relatively small compared to the system's time horizon. Therefore, solving Eq. (2.1) is equivalent to minimizing $\psi^{(t)}$ at each decision epoch. This also allows us to drop the superscript (t) to simplify the notation and mitigate clutter in our analysis.

---

[7] These coefficients K_τ and R_τ shall be designed to balance user incentives and revenue. We refer the reader to related economy literature [42].

Second, we substitute $\Pr\left[z_{\tau,p}^{(t)} < \lambda_{\tau,p}^{(t)}\right]$ with *a risk cost function* $\rho(z_{\tau,p}, \hat{\sigma}_{\tau,p}, L_\tau) := P_{\tau,p} \cdot \xi_{\tau,p}$ that depends on the resource reservation $z_{\tau,p}$, forecast uncertainty $\hat{\sigma}_{\tau,p}$ and slice duration L_τ, where the term

$$P_{\tau,p} := \frac{\Lambda_{\tau,p} - z_{\tau,p}}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}}, \quad 0 \leq P_{\tau,p} \leq 1,^{8}$$

captures *the risk of resource deficit* due to overly aggressive under-provisioning, and

$$\xi_{\tau,p} := \hat{\sigma}_{\tau,p} L_\tau, \quad 0 < \xi_{\tau,p} \leq L_\tau,$$

is a *scaling factor* that accounts for the uncertainty in our prediction ($\hat{\sigma}_{\tau,p} > 0$) and the duration of the slice request ($L_\tau > 0$). In this way, we can rewrite our problem as:

$$\min_{\boldsymbol{x}\in\{0,1\}^{\mathcal{S}}, \boldsymbol{z}\in\mathbb{R}_+^{\mathcal{S}}} \Psi := \sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B}, c\in\mathcal{C}}} \overbrace{K_\tau \rho(z_{\tau,p}, \hat{\sigma}_{\tau,p}, L_\tau) x_{\tau,p}}^{\text{Estimated penalty}} - \overbrace{R_\tau x_{\tau,p}}^{\text{Reward}}$$

We next introduce the constraints of our problem.

2.1.10 Constraints

We first formulate the system capacity constraints as

$$\sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B}}} a_\tau + z_{\tau,p} b_\tau \leq C_c, \qquad \forall c \in \mathcal{C} \tag{2.2}$$

$$\sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B}, c\in\mathcal{C}}} z_{\tau,p} \eta_e \mathbb{1}_{e\in p} \leq C_e, \qquad \forall e \in \mathcal{E} \tag{2.3}$$

$$\sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall c\in\mathcal{C}}} z_{\tau,p} \eta_{\tau,b} \leq C_b, \qquad \forall b \in \mathcal{B} \tag{2.4}$$

describing capacity constraints of CU resources, transport links, and BSs, respectively. Parameters $a_\tau, b_\tau \in s_\tau$ in Eq. (2.2), characterize the linear relationship between network load arriving at the service of tenant τ and its computing requirements.[9] a_τ models a baseline consumption associated to, e.g., the VS operative system, the mean number of users of the tenant, etc., and b_τ models the amount of computation required to serve the allocated bitrate. In Eq. (2.3), we let η_e model the overhead of the specific transport protocol used in link $e \in \mathcal{E}$ (e.g. VLAN/MPLS tags, GTP tunnels, etc.); and $\mathbb{1}_{e\in p}$ is equal to 1 only if link e belongs to path p. Finally, in Eq. (2.4), $\eta_{\tau,b}$ maps bitrate resources into

---

[8] We later impose $\hat{\lambda}_{\tau,p} \leq z_{\tau,p} \leq \Lambda_{\tau,p}$, which yields $0 \leq P_{\tau,p} \leq 1$.

[9] This model is motivated by the strong linear correlation between network load and storage/compute usage in network services evinced in several works, e.g. [43, 44], and our own measurements. We assume the model parameters are *learnt* during an offline on-boarding phase.

radio resources, which can be estimated with readily available radio models.

We also add the following constraints:

$$\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall c\in\mathcal{C}}} x_{\tau,p} \leq 1, \qquad \forall \tau \in \mathcal{T}, \forall b \in \mathcal{B} \tag{2.5}$$

to prevent multipath connections;[10]

$$\sum_{p_1\in\mathcal{P}_{m,c}} x_{\tau,p_1} \leq \sum_{p_2\in\mathcal{P}_{n,c}} x_{\tau,p_2}, \qquad \forall m \neq n \in \mathcal{B}, \forall c \in \mathcal{C}, \forall \tau \in \mathcal{T} \tag{2.6}$$

to guarantee that accepted slices are given a slice of *all* BSs and that each BS slice belonging to the same system slice Φ_τ is connected to the same CU; and the delay constraint

$$\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall c\in\mathcal{C}}} x_{\tau,p} D_p \leq \Delta_\tau, \qquad \forall \tau \in \mathcal{T}, \forall b \in \mathcal{B}. \tag{2.7}$$

Finally, we formulate the constraints that *couple* the resource reservation decisions ($\boldsymbol{z}$) and the routing/function placement and access control decisions ($\boldsymbol{x}$) as follows:

$$\boldsymbol{z} \preceq \boldsymbol{x}\boldsymbol{\Lambda} \tag{2.8}$$

$$\boldsymbol{x}\hat{\boldsymbol{\lambda}} \preceq \boldsymbol{z} \tag{2.9}$$

that yield $\hat{\boldsymbol{\lambda}} \preceq \boldsymbol{z} \preceq \boldsymbol{\Lambda}$, if Φ_τ is accepted, or $\boldsymbol{z} = \boldsymbol{0}$, otherwise.

2.1.11 AC-RR Problem

Consolidating the above, our problem becomes:

Problem 1 (AC-RR Problem).

$$\min_{\boldsymbol{x}\in\{0,1\}^{\mathcal{S}}, \boldsymbol{z}\in\mathbb{R}_+^{\mathcal{S}}} \quad \Psi(\boldsymbol{x}, \boldsymbol{z})$$

$$\text{s.t.} \quad (2.2), (2.3), (2.4), (2.5), (2.6), (2.7), (2.8), (2.9).$$

We note that $\Psi(\boldsymbol{x}, \boldsymbol{z})$ is a quadratic function. Fortunately, the structure of our problem yields the following conventional linearization technique. First, we create an auxiliary variable $y_{\tau,p} := z_{\tau,p} \cdot x_{\tau,p}$ and then rearrange the terms in Ψ to be linear with $\boldsymbol{x}$ and $\boldsymbol{y}$ as

---

[10] This constraint is motivated by the reluctance of operators to deploy multipath systems due to additional expenditures and delay (due to packet reordering) [45] but it can be relaxed if a multipath protocol is implemented [46].

follows. $\Psi(\boldsymbol{x},\boldsymbol{z}) = \Psi(\boldsymbol{x},\boldsymbol{y}) =$

$$\sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B},c\in\mathcal{C}}} \left(\frac{\Lambda_{\tau,p}\xi_{\tau,p}K_\tau}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}} - R_\tau \right) x_{\tau,p} - \frac{\xi_{\tau,p}K_\tau}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}} y_{\tau,p}.$$

Second, we add the following constraints to maintain the linearized problem equivalent to the original Problem 1:

$$\boldsymbol{y} \preceq \boldsymbol{\Lambda}\boldsymbol{x} \tag{2.10}$$

$$\boldsymbol{y} \preceq \boldsymbol{z} \tag{2.11}$$

$$\boldsymbol{z} + \boldsymbol{\Lambda}\boldsymbol{x} \preceq \boldsymbol{y} + \boldsymbol{\Lambda} \tag{2.12}$$

As a result, our AC-RR problem can be formulated as the following mixed integer linear problem (MILP):

Problem 2 (AC-RR MILP).

$$\min_{\boldsymbol{x}\in\{0,1\}^{\mathcal{S}},\boldsymbol{y}\in\mathbb{R}_+^{\mathcal{S}},\boldsymbol{z}\in\mathbb{R}_+^{\mathcal{S}}} \quad \Psi(\boldsymbol{x},\boldsymbol{y})$$

$$\text{s.t.} \quad (2.2), (2.3), (2.4), (2.5), (2.6), (2.7), (2.8), (2.9), (2.10), (2.11), (2.12).$$

We next establish the complexity of our problem.

Theorem 1. *Problem 2 (and so Problem 1) is NP-Hard.*

Proof. The proof goes by reduction. Consider a restricted instance of Problem 2 (or Problem 1) with n tenants with no associated penalty ($K_\tau = 0, \forall\tau$), 1 CU c_1 with unlimited capacity $C_{c_1} \to \infty$, 1 BS b_1 with capacity $C_{b_1} = B$, and a simple transport network with a direct link e_1 connecting c_1 and b_1 with unlimited capacity $C_{e_1} \to \infty$ and no delay. Given this setting, it is trivial to cast this problem (in polynomial time) into the well-known knapsack problem [47], which is NP-hard. Adding multiple BSs and CUs increases the complexity of the problem, making it even harder to solve. This proves that Problem 2 is NP-Hard. □

2.1.12 Practical Considerations

There are a few additional practical details we need to consider. In particular, if tenant τ is accepted in t, we need to ensure that τ is also accepted in epochs $\{t+1, t+2, \ldots, t+L_\tau\}$. This can be done by adding the following constraint to Problem 2:

$$\sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B},c\in\mathcal{C}}} x_{\tau,p} \nVdash_{\Omega_\tau\in\mathbb{Z}_{>0}} = 1, \forall\tau \in \left\{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(t-1)}\right\} \tag{2.13}$$

where Ω_τ is a state variable of slice Φ_τ indicating the time the slice has left till expiration (for all previously accepted tenants).

However, (2.13) may render unfeasibility. Imagine a scenario where two slices have been accepted in t_1 for a duration equal to L. Now, if the load forecast of any tenant exceeds the capacity of some resource in t_2, $t_2 < t_1 + L$, we would encounter a deficit of resources that represents an unfeasible setting due to constraint (2.13). To address this, we relax the capacity constraints (2.2)-(2.4) as follows,

$$\sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B}}} a_\tau + z_{\tau,p} b_\tau \le C_c + \delta_c, \qquad \forall c \in \mathcal{C} \tag{2.14}$$

$$\sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B}, c\in\mathcal{C}}} z_{\tau,p} \eta_e \not\Vdash_{e\in p} \le C_e + \delta_b, \qquad \forall e \in \mathcal{E} \tag{2.15}$$

$$\sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \in\mathcal{C}}} z_{\tau,p,i} \eta_{\tau,b} \le C_b + \delta_r, \qquad \forall b \in \mathcal{B} \tag{2.16}$$

and Problem 2 as follows

$$\min_{\substack{\boldsymbol{x}\in\{0,1\}^{\mathcal{S}}, \boldsymbol{y}\in\mathbb{R}_+^{\mathcal{S}}, \boldsymbol{z}\in\mathbb{R}_+^{\mathcal{S}} \\ \delta_r\in\mathbb{R}_+, \delta_b\in\mathbb{R}_+, \delta_c\in\mathbb{R}_+}} \quad \Psi(\boldsymbol{x}, \boldsymbol{y}) + M(\delta_r + \delta_b + \delta_c)$$

$$\text{s.t.} \qquad (2.14), (2.15), (2.16), (2.5), (2.6), (2.7), (2.8), (2.9), (2.10), (2.11), (2.12),$$

where $\delta_r, \delta_b, \delta_c \in \mathbb{R}_+$ are auxiliary variables accounting for the deficit of radio, transport and computing resources, respectively, and M is a large value accounting for the cost of leasing these resources (e.g., via federation) or the penalties that we would have to pay (also sometimes known as "big M method"). This method fixes the unfeasibility issue as the resource deficit potentially caused by Eq. (2.13) is absorbed by the new auxiliary variables (at a high cost M). While we consider this in our implementation (as shown in §2.1.21), we omit these details in the following analysis to keep our presentation simple.

2.1.13 Algorithms

We next present two algorithms to solve Problem 2: an optimal method based on *Benders decomposition*, designed for small to medium-scale networks, and a suboptimal *heuristic* that expedites solutions in medium to large-scale networks.

2.1.14 Benders Method

Our first methodology to solve Problem 2 lies on the observation that constraints (2.8), (2.9), (2.10) and (2.12) couple the real-valued resource reservation decision variables ($\boldsymbol{z}$, $\boldsymbol{y}$), and the binary placement and path selection decision variables ($\boldsymbol{x}$). We relax these constraints and decouple the slack problem into two subproblems by means of Benders decomposition [48]: one that involves the so-called "complicated" variables and one that involves only continuous variables. We first describe our *slave* subproblem as follows:

Problem 3 (Slave problem $P_S(\bar{\boldsymbol{x}})$).

$$\begin{array}{lll}
\min\limits_{\boldsymbol{y}\in\mathbb{R}_+^{\mathcal{S}},\boldsymbol{z}\in\mathbb{R}_+^{\mathcal{S}}} & \displaystyle\sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall b\in\mathcal{B},c\in\mathcal{C}}} -\frac{\xi_{\tau,p}K_\tau}{\Lambda_{\tau,p}-\hat{\lambda}_{\tau,p}} y_{\tau,p} & \\
\text{s.t.} & (2.2), (2.3), (2.4), (2.11) & \\
 & \boldsymbol{z} \preceq \bar{\boldsymbol{x}}\boldsymbol{\Lambda} & (2.17)\\
 & \bar{\boldsymbol{x}}\hat{\boldsymbol{\lambda}} \preceq \boldsymbol{z} & (2.18)\\
 & \boldsymbol{y} \preceq \boldsymbol{\Lambda}\bar{\boldsymbol{x}} & (2.19)\\
 & \boldsymbol{z} + \boldsymbol{\Lambda}\bar{\boldsymbol{x}} \preceq \boldsymbol{y} + \boldsymbol{\Lambda} & (2.20)
\end{array}$$

which can be solved with standard linear programming solvers, and define its dual problem as $P_{DS}(\bar{\boldsymbol{x}})$.

Problem 4 (Dual slave problem $P_{DS}(\bar{\boldsymbol{x}})$).

$$\begin{array}{ll}
\max\limits_{\boldsymbol{\mu}\in\mathbb{R}_+^{N}} & g\left(\bar{\boldsymbol{x}},\boldsymbol{\mu}\right)\\
\text{s.t.} & \displaystyle -b_\tau\mu_{1,c} - \sum_{e\in p}\eta_e\mu_{2,e} - \eta_{\tau,p}\mu_{3,b} - \mu_{4,\tau,p} + \mu_{5,\tau,p} +\\
 & + \mu_{7,\tau,p} - \mu_{8,\tau,p} \leq 0, \qquad \forall b\in\mathcal{B}, \forall c\in\mathcal{C}, \forall p\in\mathcal{P}_{b,c}, \forall\tau\in\mathcal{T}\\
 & \displaystyle -\mu_{6,\tau,p} - \mu_{7,\tau,p} + \mu_{8,\tau,p} \leq -\frac{\xi_{\tau,p}K_\tau}{\Lambda_{\tau,p}-\hat{\lambda}_{\tau,p}},\\
 & \qquad\qquad\qquad\qquad\qquad \forall b\in\mathcal{B}, \forall c\in\mathcal{C}, \forall p\in\mathcal{P}_{b,c}, \forall\tau\in\mathcal{T}
\end{array}$$

where $g\left(\bar{\boldsymbol{x}},\boldsymbol{\mu}\right) =$

$$\begin{aligned}
&\sum_{c\in\mathcal{C}}\mu_{1,c}\left(\sum_{\tau\in\mathcal{T}}\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall b\in\mathcal{B}}} a_\tau - C_c\right) - \sum_{e\in\mathcal{E}}\mu_{2,e}C_e - \sum_{b\in\mathcal{B}}\mu_{3,b}C_b +\\
&+\sum_{\tau\in\mathcal{T}}\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall b\in\mathcal{B},c\in\mathcal{C}}}\Big(-\mu_{4,\tau,p}\bar{x}_{\tau,p}\Lambda_{\tau,p} + \mu_{5,\tau,p}\bar{x}_{\tau,p}\hat{\lambda}_{\tau,p} -\\
&\qquad -\mu_{6,\tau,p}\Lambda_{\tau,p}\bar{x}_{\tau,p} + \mu_{8,\tau,p}(\Lambda_{\tau,p}\bar{x}_{\tau,p} - \Lambda_{\tau,p})\Big)
\end{aligned}$$

and $\boldsymbol{\mu}$ is the vector of $N = C + |\mathcal{E}| + B + 5\mathcal{S}$ dual variables.

We then formulate our *master* subproblem as follows:

Algorithm 1 Benders method

```
$k \leftarrow 1$
Initialize $\mathcal{C}_1 = \mathcal{C}_2 = \varnothing$, $UB^{(1)} = -LB^{(1)} >> 1$
while $UB^{(k)} - LB^{(k)} > \epsilon$ do
    $LB^{(k)}, \boldsymbol{x}^{(k)}, \theta^{(k)} \leftarrow PM(\mathcal{C}_1, \mathcal{C}_2)$
    $\boldsymbol{\mu}^{(k)} \leftarrow_{DS} (\boldsymbol{x}^{(k)})$
    if $P_{DS}(\boldsymbol{x}^{(k)})$ is unbounded then
        $\boldsymbol{\mu}^l \leftarrow$ extreme ray
        $\mathcal{C}_2 \leftarrow \mathcal{C}_2 \cup \{\boldsymbol{\mu}^l\}$
    else
        $\boldsymbol{\mu}^m \leftarrow$ extreme point
        $\mathcal{C}_1 \leftarrow \mathcal{C}_1 \cup \{\boldsymbol{\mu}^m\}$
    end if
    $\Gamma = \sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in \mathcal{C}}} \left( \frac{\Lambda_{\tau,p} \xi_{\tau,p} K_\tau}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}} - R_\tau \right) x_{\tau,p}^{(k)} - g\left(\boldsymbol{x}^{(k)}, \boldsymbol{\mu}^{(k)}\right)$
    if $UB^{(k-1)} > \Gamma$ then
        $UB^{(k)} = \Gamma$
    end if
    $k \leftarrow k + 1$
end while
$\boldsymbol{x}^* = \boldsymbol{x}^{(k)}$
$\boldsymbol{y}^*, \boldsymbol{z}^* \leftarrow P_{DS}(\boldsymbol{x}^{(k)})$
```

Problem 5 (Master problem $P_M(\mathcal{C}_1, \mathcal{C}_2)$).

$$\min_{\boldsymbol{x} \in \{0,1\}^{\mathcal{S}}, \theta \in \mathbb{R}_+} \quad \sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in \mathcal{C}}} \left(\frac{\xi_{\tau,p} K_\tau}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}} \Lambda_{\tau,p} - R_{\tau,p} \right) x_{\tau,p} + \theta$$

$$\text{s.t.} \quad (2.5), (2.6), (2.7)$$

$$g\left(\boldsymbol{x}, \boldsymbol{\mu}^m\right) \leq \theta, \qquad \forall \boldsymbol{\mu}^m \in \mathcal{C}_1 \tag{2.21}$$

$$g\left(\boldsymbol{x}, \boldsymbol{\mu}^l\right) \leq 0, \qquad \forall \boldsymbol{\mu}^l \in \mathcal{C}_2 \tag{2.22}$$

where θ is a surrogate variable substituting the "cost" of the resource reservation decisions, and equations (2.21) and (2.22) correspond to the optimality and feasibility cuts, respectively, added iteratively by Algorithm 1. We then use the iterative Algorithm 1 to solve Problem 2. The optimality of this approach is formalized in the following theorem.

Theorem 2 (Algorithm 1 Optimality). *Algorithm 1 converges to the optimal solution of Problem 2 in a finite number of iterations.*

Proof. The proof follows from the Partition Theorem in [48]. Let us consider the abstract formulation of Problem (5):

$$\min_{\boldsymbol{x}, \theta} \; c_1^T \boldsymbol{x} + \theta \quad \text{s.t.} \quad (\boldsymbol{x}, \theta) \in \mathcal{G}\,, \tag{2.23}$$

where $\mathcal{G}$ is the set of constraints, created by the intersection of the constraints in $\mathcal{X}$ and the convex hull of the extreme halflines resulting from the dual slave problem (which is a polyhedral cone $\mathcal{C}$). Algorithm 1 is initialized with empty sets $\mathcal{C}_1$ and $\mathcal{C}_2$ and thus $\mathcal{G}^{(1)}$ corresponds to a minimal set of constraints. At each iteration $k > 1$, the algorithm appends a point of the dual slave problem into set $\mathcal{C}_1$ or $\mathcal{C}_2$, which results in the addition of one extreme halfline of the cone $\mathcal{C}$ in $\mathcal{G}^{(k)}$. As a result, set $\mathcal{G}$ is iteratively reconstructed and, given that there is a finite number of them, convergence to the optimal solution is guaranteed because, in the worst case, the algorithm will reconstruct the full set $\mathcal{G}$. □

2.1.15 Heuristic Algorithm

While Benders method provides an optimal solution, it might take long time to converge. For larger scale systems, we propose a heuristic to solve Problem 5 by casting it into a classical multi-constrained 0-1 Knapsack problem model [49]:

Problem 6 (Multi-constrained Knapsack Problem).

$$\min_{\boldsymbol{x}\in\{0,1\}^{\mathcal{S}}} \quad \sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B},c\in\mathcal{C}}} \gamma_{\tau,p}\, x_{\tau,p}$$

$$\text{s.t.} \quad \sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B},c\in\mathcal{C}}} w_{\tau,p}^{(k)}\, x_{\tau,p} \leq W^{(k)}, \quad \forall k \tag{2.24}$$

$$\sum_{j\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B},c\in\mathcal{C}}} \mathbb{1}_{j=\tau}\, x_{j,p} \leq 1, \quad \forall \tau \in \mathcal{T}; \tag{2.25}$$

where $\gamma_{\tau,p}$ and $w_{\tau,p}^{(k)}$ in constraint (2.24) are the cost and the weight of item $x_{\tau,p}$, respectively, whereas $W^{(K)}$ is the total capacity of the knapsack. They are defined as follows.

$$\gamma_{\tau,p} = \left(\frac{\xi_{\tau,p} K_{\tau}}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}} \Lambda_{\tau,p} - R_{\tau,p} \right) \tag{2.26}$$

$$w_{\tau,p}^{(k)} = -\mu_{4,\tau,p}\Lambda_{\tau,p} + \mu_{5,\tau,p}\hat{\lambda}_{\tau,p} - \mu_{6,\tau,p}\Lambda_{\tau,p} + \mu_{8,\tau,p}\Lambda_{\tau,p} \tag{2.27}$$

$$W^{(k)} = -\sum_{c\in\mathcal{C}} \mu_{1,c} \left(\sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B}}} a_{\tau} - C_c \right) + \sum_{e\in\mathcal{E}} \mu_{2,e} C_e + $$

$$+ \sum_{b\in\mathcal{B}} \mu_{3,b} C_b + \sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B}}} \mu_{8,\tau,p} \Lambda_{\tau,p}. \tag{2.28}$$

Note that constraints are dynamically added by Algorithm 1 at each iteration $k \geq 1$. The constraint set (2.25) accounts for constraint (2.5) in Problem 5. When devising a lightweight solution to solve the above-mentioned problem, we rely on classical heuristics

Algorithm 2 Knapsack-Solver($\bar{W}, \bar{\boldsymbol{w}}$)

```
Initialize $H = 0, \mathcal{C} = \{e\}$ where $\{e\} = \{\tau, p\}, \forall \tau, p$
**Calculate** $w_{\tau,p}$ and $W$ based on (2.29)
$H = \bar{W}$
**for** $e \in \mathcal{C}$ **do**
    $\phi_{\tau,p} = \frac{\gamma_{\tau,p}}{\bar{w}_{\tau,p}}$
**end for**
**Sort** $\mathcal{C}$ based on $\phi_{\tau,p}$ in a decreasing order
**while** $(H > 0 \wedge |\mathcal{C}| > 0)$ **do**
    **Pool** the first $e \leftarrow \mathcal{C}$
    **if** $H - w_{\tau,p} \geq 0$ **then**
        $x_{\tau,p} = 1$
        $H = H - w_{\tau,p}$
    **end if**
**end while**
```

proposed for knapsack problems. We name our proposal Knapsack Admission Control (KAC) algorithm and we show the details in Algorithm 2. First, we combine together different weights $w_{\tau,p}^{(k)}$ into one single value per item $x_{\tau,p}$ and we calculate the overall system capacity W as follows

$$\bar{w}_{\tau,p} = \sum_{k} \epsilon_k w_{\tau,p}^{(k)}, \quad \text{and} \quad \bar{W} = \sum_{k} \epsilon_k W^{(k)}, \tag{2.29}$$

where ϵ_k is recursively defined as follows

$$\epsilon_k = \left| \epsilon_{k-1} W^{(k)} - \sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}}} \epsilon_{k-1} w_{\tau,p}^{(k)} \right|, \quad \forall k > 0, \tag{2.30}$$

assuming that $\epsilon_0 = 1$. This translates the problem into a classical 0-1 Knapsack problem with one single capacity constraint. Thus, we compute the ratio $\phi_{\tau,p} = \frac{\gamma_{\tau,p}}{\bar{w}_{\tau,p}}$ per item $x_{\tau,p}$. Based on such ratio, we sort all the items in a decreasing order and we try to fit them into our system capacity $\bar{W}$, following the classical first-fit decreasing (FFD) algorithm [50].

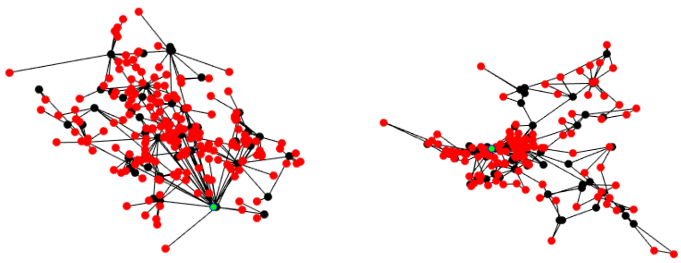
Algorithm 2 is a heuristic that allows us to expedite solutions of Problem 5. Then, by combining Algorithm 2 and removing the optimality cuts from our Benders approach, we can design a fast method to solve our orchestration Problem 2 in larger scale scenarios. We describe such method, descriptively labeled Knapsack Admission Control (KAC), in Algorithm 3.

2.1.16 Simulation Results

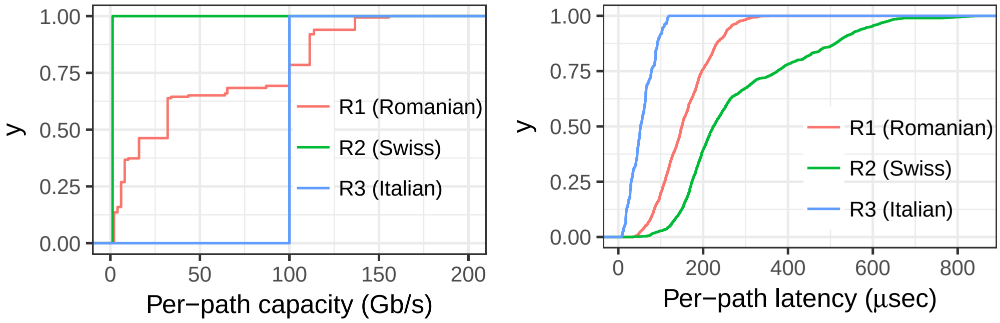
We now evaluate, with emulated data planes from real operators, the revenue gains achievable by our approach under different slice types, traffic patterns and penalties/rewards.

Algorithm 3 Knapsack Admission Control (KAC)

```
$k \leftarrow 1$
Initialize $\bar{W} = \varnothing, \bar{\boldsymbol{w}} = \varnothing, \epsilon_0 = 1$
$\boldsymbol{x}^{(k)} \leftarrow$ Knapsack-solver($\bar{W}, \bar{\boldsymbol{w}}$)
while $P_{DS}(\boldsymbol{x}^{(k)})$ is unbounded do
    $\boldsymbol{\mu} \leftarrow$ extreme ray
    Compute $\boldsymbol{w}^{(k)}$ and $W^{(k)}$ based on (2.27) and (2.28)
    $\bar{\boldsymbol{w}} = \bar{\boldsymbol{w}} + \epsilon_k \boldsymbol{w}^{(k)}, \quad \bar{W} = \bar{W} + \epsilon_k W^{(k)}$
    Compute $\epsilon_k$ based on (2.30)
    $\boldsymbol{x}^{(k)} \leftarrow$ Knapsack-solver($\bar{W}, \bar{\boldsymbol{w}}$)
    $k \leftarrow k + 1$
end while
$\boldsymbol{x}^* = \boldsymbol{x}^{(k)}$
$\boldsymbol{y}^*, \boldsymbol{z}^* \leftarrow P_{DS}(\boldsymbol{x}^{(k)})$
```



(a) *Romanian topology (N1)* **(b)** *Swiss topology (N2)* **(c)** *Italian topology (N3)*



(d) *Path Capacity Distribution* **(e)** *Path Delay Distribution*

Figure 2.4: *(a)-(c): Networks from 3 European operators: red dots indicate the BSs' locations, black dots the routers/switches, and the green dot an edge CU (placed at the most central position). (d)-(e) Path capacity and delay distribution for the 3 networks.*

2.1.17 Infrastructure

We consider real urban networks from 3 different operators in Romania (N1), Switzerland (N2) and Italy (N3), shown in Fig. 2.4(a)-(c). First, we observe that they *do not have canonical structure*. Some BSs are as far as 20Km from the edge CU (in N3), while others are within 0.1Km range. There is therefore high path diversity across networks. N1 has high path redundancy (mean of 6.6 paths), while in N3 several BSs have only 1 path (mean 1.6). As a result, the delay[11] distribution differs across networks. Second, they use heterogeneous link technologies. N3 uses mainly fiber, N2 wireless and N1 fiber, copper and wireless. This induces high diverse link capacities (from 2 to 200 Gb/s). This diversity, illustrated in Fig. 2.4(d)-(e), evinces that a one-size-fits-all orchestration policy may be arbitrarily inefficient.

Romania (N1) and Switzerland (N2) have $N = 198$ and $N = 197$ BSs, respectively. We consider $C_b = 20$ MHz for all BSs b that, assuming ideal channel conditions and 2x2 MIMO, yield $\eta_b = 20/150$.[12] Conversely, Italy (N3) has 1497 radio units clustered in 200 groups of 5-10 radio units. We consider each cluster as one BS with capacity equal to the aggregate capacity of the cluster (between $C_b = 80$ and $C_b = 100$ MHz). Finally, we connect the edge CU (green dot in Fig. 2.4(a)-(c)) with a core CU (not shown in the figure) with a link with unlimited bandwidth and a latency equal to 20 ms. We let the edge CU have a capacity equal to $20N$ CPU cores, i.e., enough capacity to accommodate one mMTC tenant (the more compute-hungry, as we show later) at maximum load, and the core CU have five times as much. Moreover, to ease presentation, we neglect transport overheads and so $\eta_e = 1$.

2.1.18 Scenarios

Based on 3GPP guidelines on 5G network design [18], 3 different slice types may be specified in Network Slice Selection Assistance Information (NSSAI): enhanced/extreme Mobile BroadBand (e/xMBB), massive Machine-Type-Communications (mMTC) and ultra reliable low-latency communications (URLLC). We rely on such 3 heterogeneous slice types to account for diverse delay/throughput requirements, summarized in Table 2.1. The reward R gained when accepting a tenant differs across slice types to reflect such heterogeneity. Slice requests Φ_τ are generated with a fixed $\mathbf{\Lambda}_\tau = \{\Lambda_{\tau,p} = \Lambda \mid \forall p \in \mathcal{P}_{b,c}, \forall b \in$

Slice type	R	Δ (ms)	Λ (Mb/s)	σ (Mb/s)	$s = \{a, b\}$ (CPUs)
(x)eMBB	1	30	50	variable	$\{0, 0\}$
mMTC	$(1 + b)$	30	10	0	$\{0, 2\}$
URLLC	$(2 + b)$	5	25	variable	$\{0, 0.2\}$

Table 2.1: *End-to-end network slice template*

---

[11] Assuming store-and-forward and $12000/C_e$, 4 or 5μs/Km (cable or wireless), and 5μs for transmission, propagation, and processing delay.

[12] We consider ideal conditions to ease the analysis. In practice, however, radio models can be used to make a more accurate estimation.

$\mathcal{B}, \forall c \in \mathcal{C}\}$. Then, the actual traffic demand $\boldsymbol{\lambda}_\tau^{(\theta)}$ follows a Gaussian distribution with variable mean $\bar{\lambda}$ and standard deviation σ. The only exception is the mMTC template that has a deterministic load (i.e., $\sigma_{\text{mMTC}} = 0$). Finally, the service model parametrization s is also shown in the table.

We compare both (`Benders` and `KAC`) against a baseline approach labeled `no-overbooking`. For the latter, we solve the same AC-RR problem but we replace constraint (2.9) with $\boldsymbol{x}\boldsymbol{\Lambda} \preceq \boldsymbol{z}$. As a result, accepted slices upon the `no-overbooking` policy are allocated the amount of resources agreed in their SLA. Note that we use our optimal `Benders` method to find the `no-overbooking` policy and so it is *an upper-bound benchmark*. All slice requests are issued at the beginning of each simulation, which runs until the mean revenue has a standard error lower than 2%. This is almost immediate for `no-overbooking` but it requires longer for our overbooking methods due to the time needed to *learn* slice load patterns.

We present results for a variable setting of mean load $\bar{\lambda}$, load variability σ, and penalty $K_\tau = K, \forall \tau$. In our results, depicted in Fig. 2.5 and 2.6, different colors represent different penalties such that $K = \frac{m}{\Lambda} R$, where $m = \{1, 4, 16\}$. In this way, if $m = 1$, failing to serve 10% of the SLA incurs in a penalty equal to 10% of the reward payed by the tenant (40% if $m = 4$ and so on). Finally, we set $\sigma = \{0, \bar{\lambda}/4, \bar{\lambda}/2\}$ with different line types (for `Benders`) or shapes (for `KAC`). We consider a total number of 10 tenants for "Romanian" and "Swiss" and 75 tenants for "Italian" (with more radio and transport capacity). In this way, our simulations span not only realistic topologies but also a wide set of parameters.

2.1.19 Homogeneous cases

We first let all the slices use the same template and have equal (but independent) service demand statistics ($\bar{\lambda}$ and σ). Fig. 2.5 depicts the relative net revenue gain (percentage) with our approaches and with `no-overbooking` for all slice types and all topologies described above. In the x-axis, we use parameter $0 \leq \alpha \leq 1$ to control the mean load of each slice such that $\bar{\lambda} = \alpha\Lambda$ (e.g., if $\alpha = 1$ the mean load of Φ_τ is equal to Λ_τ).

We note that both `KAC` and `Benders` provide equal performance when all slices are eMBB, regardless of the topology. This is remarkable because `Benders` may take a few hours to converge with some settings whereas `KAC` boils down this number to a few seconds. In case of mMTC and URLLC slices, `KAC` under-performs when compared to `Benders`, though it still provides between 200% and 75% additional revenue w.r.t. `no-overbooking` in low to medium load regimes. However, as above-mentioned, we use an optimal method to implement `no-overbooking` and it thus suffers from convergence times similar to our optimal method.

Let us focus on the eMBB slices in "Romanian" (top left plot of Fig. 2.5). In this setting, `no-overbooking` obtains a revenue equal to 3 monetary units irrespective of the conditions of the system (not shown due to space limitations). Regarding our approaches, we obtain up to 220% additional revenue (i.e., up to 10 monetary units) when the mean load is low (relative to the SLA). This is intuitive because the lower the ratio between

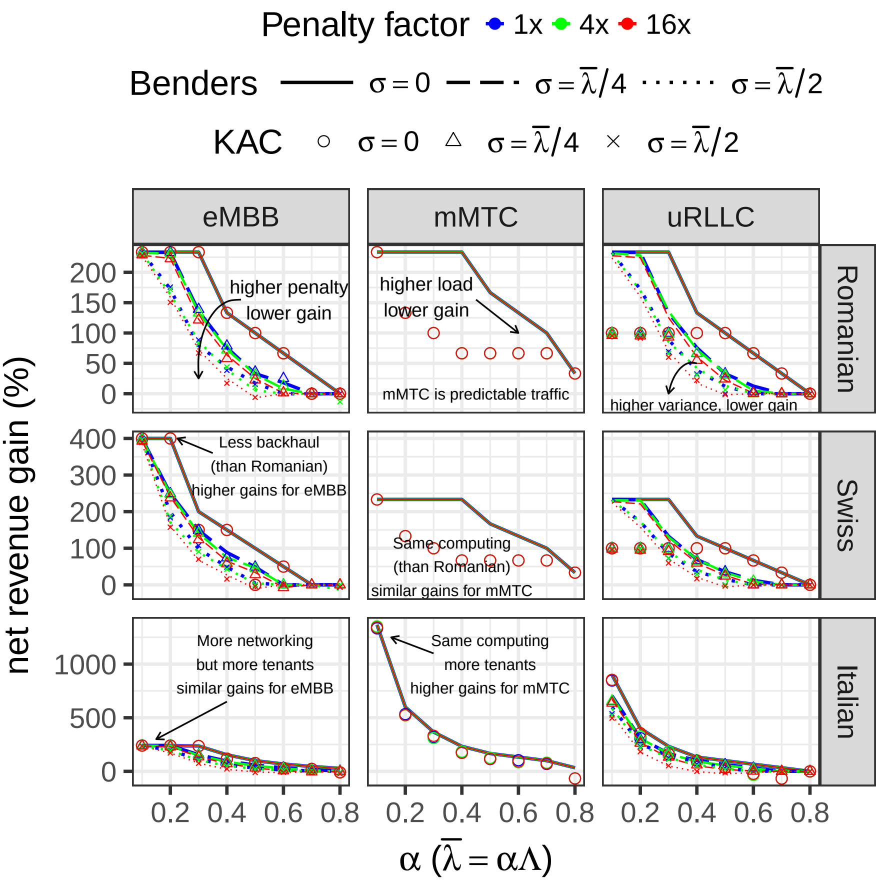


Figure 2.5: *Relative revenue (percentage) of our approaches over* `no-overbooking` *in homogeneous scenarios. Variable mean load* $\bar{\lambda}$.

mean load $\bar{\lambda}$ and Λ, the larger the chances for multiplexing load. The second observation is that, when $\sigma = 0$ (no traffic variability), our approach obtains the same revenue gains independently from the penalty factor imposed. This results in overbooking with no risk as the forecasting process is performed with high certainty.

The third due observation is that higher slice load variability leads to less revenue gains. The rationale behind is that higher variability incurs in a higher risk of committing an SLA violation and so our mechanism overbooks more conservatively. Finally, when $\sigma > 0$, higher penalty factors also negatively affect the potential revenue gains due to a conservative behavior.

The net revenue attained to mMTC or URLLC is higher (up to 30 and 25 units in "Romanian", respectively) due to their higher reward. However, we can observe that the relative gains remain very similar for all slice types in "Romanian". This is not the case for "Swiss", where the maximum gain of eMBB is twice its gain in "Romanian" (and twice the gain for mMTC and URLLC). The reason is that the transport of "Swiss" is constrained by low-capacity wireless links whereas the computing capacity (used by URLLC and specially mMTC) remains the same. As a result, `no-overbooking` obtains less net revenue when there are eMBB slices only w.r.t. “Romanian”. However, our approaches are capable of accepting more eMBB tenants when their actual load is limited.

Last, “Italian” has considerably more radio and transport resources than both Romania and “Swiss”, whereas the computing capacity remains the same. Indeed, `no-overbooking` obtains up to 25 monetary units when all slices are eMBB (8x more than the same scenario in “Swiss” and “Romanian”), and very similar net revenue when slices are mMTC and URLLC (because they mostly depend on computing, which keeps constant across topologies). Given that we have 75 tenants (instead of 10), the relative gains when applying overbooking are similar for eMBB as in the other topologies. This is due to the fact that increasing radio and transport capacity benefits both `no-overbooking` and our approaches. However, these gains are substantially higher when the mean load of the slices is mild to low with mMTC and URLLC as computing is severely constrained thereby substantially helping in these load regimes.

Notably, the gains shown in Fig. 2.5 come at a negligible cost on the tenants. Specifically, a tenant’s SLA violation occurred with a probability lower than 0.0001% in the most aggressive configuration ($\sigma = \bar{\lambda}/2$ and $m = 1$), and even in such rare cases, the dropped traffic is as much as 10%. As a sanity check, a more aggressive overbooking ($\sigma = 3\bar{\lambda}/4$ and $m = 0.01$) increases the chances of violating an SLA to only 0.043% samples with as much as 20% of traffic drop.

2.1.20 Heterogeneous cases

We now consider mixed setups. To simplify the visualization of our results, we focus on scenarios that merge eMBB and mMTC, URLLC and eMBB, and mMTC and URLLC slices, respectively, and fix the mean load $\bar{\lambda}_\tau = 0.2 \cdot \Lambda_\tau$. Fig. 2.6 depicts the net revenue of our approaches and `no-overbooking` (with a black line) for the same range of σ and penalty parameters m used before. The scenarios have a fix number of slices (10 for “Romanian” and “Swiss”, 75 for “Italian”) and we vary the percentage of one type of slice w.r.t. the other using parameter β.

First, let us study the top left plot where we have $10\frac{\beta}{100}$ mMTC slices and $10\frac{100-\beta}{100}$ eMBB slices in “Romanian”. The revenue attained to `no-overbooking` grows as we increase the ratio of mMTC tenants until $\beta = 25\%$ onwards when the revenue remains flat. At that point, `no-overbooking` is not capable of accommodating computing resources to the increasing number of mMTC slices but there are sufficient eMBB slices to compensate. This occurs until $\beta = 75$ where there are not enough eMBB tenants and therefore the revenue falls as computing resources are fully consumed. In marked contrast, our approach obtains a linearly increasing revenue as we increase the number of mMTC slices that are all eventually accepted. Interestingly, the larger relative gains over `no-overbooking` occurs when the scenario is more homogeneous ($\beta = 0\%$ and $\beta = 100\%$). Similar observations can be obtained from the other two mixes of slice types. We obtain similar revenues also for “Swiss”. The main difference is that, given the constrained transport, higher values of σ and higher penalty factors incur in lower revenues compared to the “Romanian” topology.

Compared to “Romanian” and “Swiss”, similar revenue trends are observed for `no-overbooking`

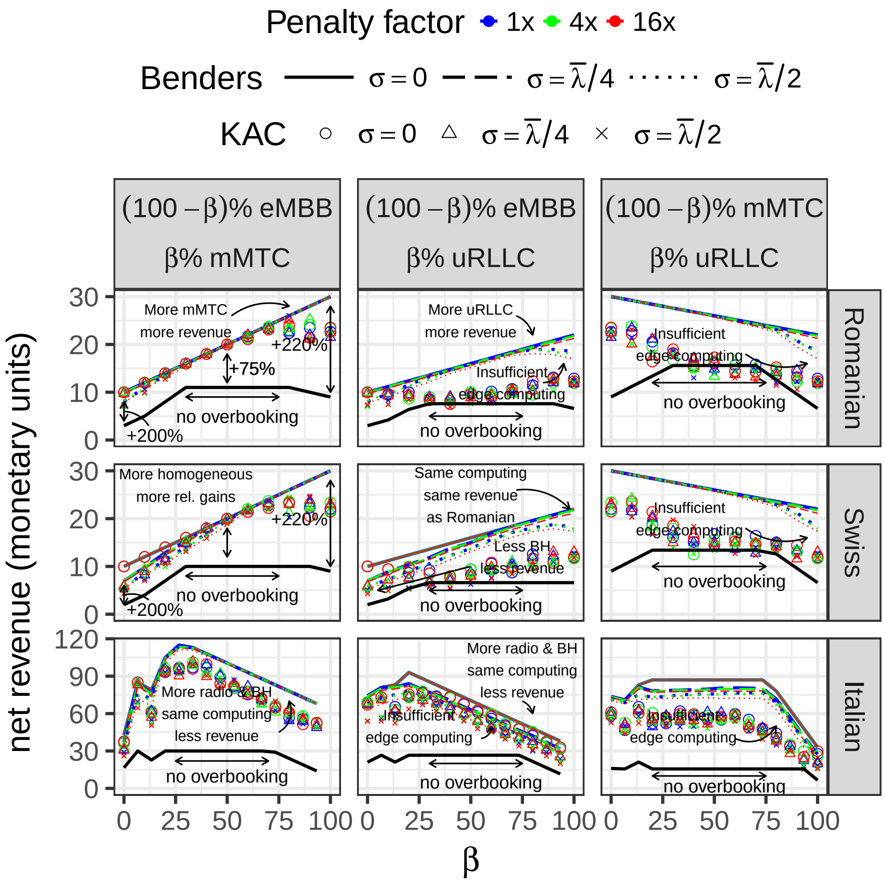


Figure 2.6: *Revenue of our approaches (colors) and* `no-overbooking` *(black) in heterogeneous scenarios. Mean load is* $\bar{\lambda} = 0.2\Lambda$.

but substantially different for our approaches in “Italian” taking the first case (eMBB and mMTC slices). The revenue of both `Benders` and `KAC` rapidly grows as we accept more mMTC slices while declining after we reach $\beta = 25\%$. Counter-intuitively, while “Italian” has substantially more radio and transport resources (and more slice requests) than the other two topologies, the computing resources are essentially the same, and there are not sufficient eMBB slices to compensate the rejected mMTC slices from $\beta = 25\%$ onwards. Similar observations can be made for “Italian” in the other two mixes of slices.

Importantly, our overbooking schemes cause SLA violations as often as in the homogeneous case (less than 0.001% samples with the most aggressive configuration) and so our gains come at a negligible cost for the tenants. In this way, we conclude that our system manages to trade off hard SLA guarantees of traditional systems for substantial revenue gains with minimal SLA violations and practically zero footprint from the overbooking scheme.

2.1.21 Experimental Proof-of-Concept

We evaluate our orchestrator[13] with a real data plane.

To this aim, we deploy the experimental testbed depicted in Fig. 2.7. The hardware components are summarized in Table 2.2.

In the RAN, we use 2 commercial BSs with RAN sharing support and we use different PLMN-Ids [56] to identify slices due to the lack of 5G network slicing-support equipment. The proprietary interface of the BSs allows us to grant shares of bandwidth, physical radio blocks (PRBs) specifically, to different mobile networks (associated with a different PLMN-id).[14] The BSs are set in 20-MHz channels (capacity equal to 100 PRBs). In the transport, we use a programmable `OpenFlow` switch to virtualize the backhaul topology shown in Fig. 2.1, comprised of 1-Gb/s Ethernet links. For computing, we connect two

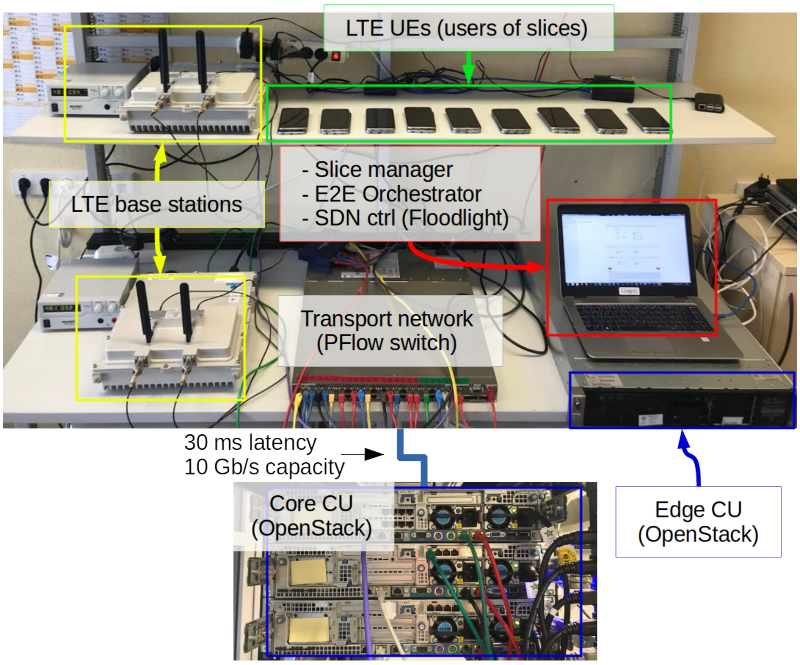


Figure 2.7: *Testbed*

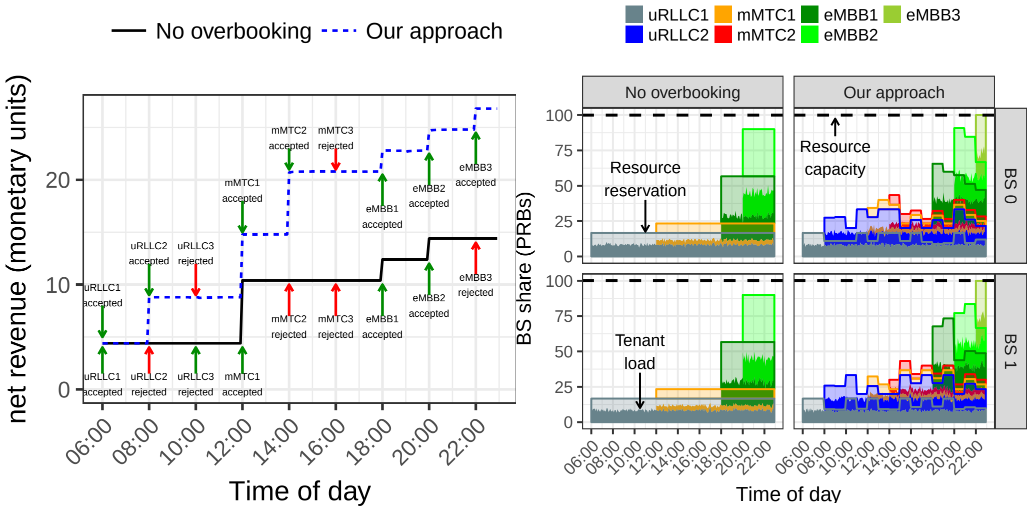
Device type	Description	Ref.
vEPCs	OpenEPC Rel. 7 (1x per slice)	[51]
UEs	Samsung Galaxy 7 (1x per slice and BS)	[52]
Transport	OpenFlow 1.5 switch with 48 1-gigabit ports	[53]
RAN	2x 20 MHz NEC small cell with RAN sharing @ band 3	[54]
CU	OpenStack Queens with 16 (Edge) and 64 (Core) CPUs	[55]

Table 2.2: *Detailed HW components in our testbed*

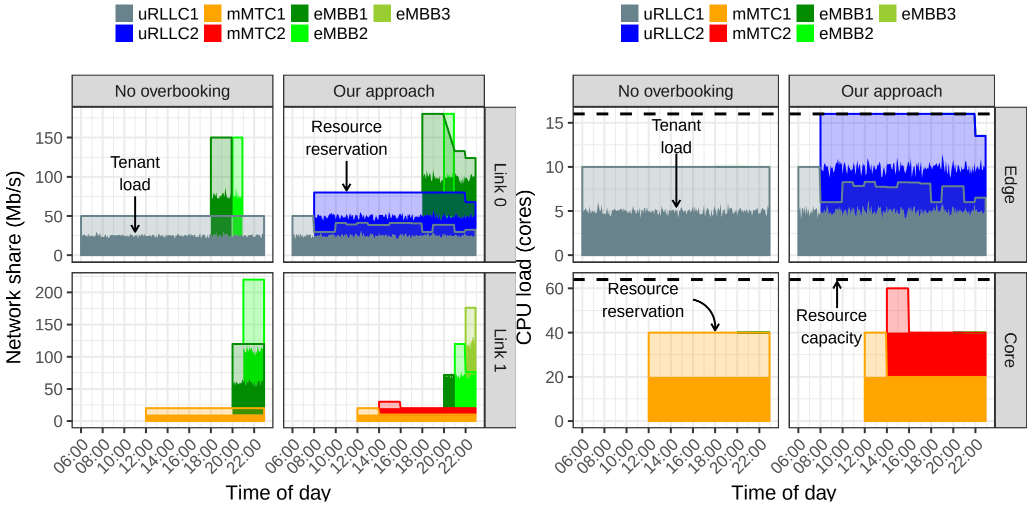
---

[13] The algorithm implementation has been carried out using the framework of `IBM ILOG CPLEX` and its `Python API`.

[14] We use commercial BSs for convenience; however, our approach is a natural fit to open source initiatives such as [29].



(a) *Net Revenue.* **(b)** *Radio utilization.*



(c) *Transport utilization.* **(d)** *Computation utilization.*

Figure 2.8: *Net revenue over time (a); and resource reservation and actual utilization across BSs (b), two transport links (c) and both CUs (d), respectively, for 9 heterogeneous slice requests arriving at different times.*

conventional servers with two 1Gb/s Ethernet links, respectively. The first server has 16 CPU cores and emulate an edge CU; the second has 64 CPU cores and we use `netem` to emulate 30 ms latency in its backhaul link, emulating a core CU. To construct each slice's network service (see Fig. 2.1), we create a VM instance of `OpenEPC` to connect the slice to the mobile system, a VM with our rate-control middlebox and an additional VM with `mgen` to generate traffic with custom traffic patterns, emulating the VS of the slice. Finally, we use one Android smartphone per slice and BS, connected to the BS with coaxial cables for isolation, to emulate a crowd of UEs receiving traffic from each VS.

We set up a dynamic scenario where slice requests arrive every 2 epochs for a total of 18 epochs (i.e., up to 9 slices). We take one monitoring sample every 5 minutes (which is conventional [34]), and collect 12 samples per epoch (i.e., 1 hour). The first three slice requests “uRLLC1”, “uRLLC2”, “uRLLC3” are URLLC (with the parameters described in Table 2.1), the next three “mMTC1”, “mMTC2”, “mMTC3” are mMTC and the remaining slices “eMBB1”, “eMBB2”, “eMBB3” are eMBB. To ease the analysis, we fix the mean load of each slice to be half its Λ (SLA) with a standard deviation equal to 10% of its mean, and a penalty equal to $K = \frac{R}{\Lambda}$ ($m = 1$ in Fig. 2.5 and 2.6). We repeat the experiment with our approach (using Benders) and with “no overbooking”. The results are summarized in Fig. 2.8(a)-(d). Fig. 2.8a shows the net revenue per BS of both approaches over time; and Fig. 2.8(c)-(d) show, with stacked areas, the utilization and the actual reservation made on each domain of the system. For the transport, we selected the two links that connect each CU to the rest of the system to guarantee that any possible path is represented.

The first 3 slice requests (URLLC) arrive at 6h, 8h and 10h, respectively, requesting an aggregate of 10 CPUs each in the edge CU. While “no overbooking” accepts only “uRLLC1”, our mechanism adapts the CPU reservation to the actual load of the slices and thus accepts also “uRLLC2” as shown by Fig. 2.8d. This results in twice the revenue we obtain at 10h. The next 3 slice requests are mMTC requesting up to 40 CPUs. Similarly, our approach adapts the CPU reservation to the actual load and allows us to accept an additional slice over “no overbooking”, which results in 100% revenue gain at 16h. From this time on, one eMBB slice request arrives every 2h requesting 50 Mb/s service SLA. This forces “no overbooking” to accept only 2 slices at the moment, since some radio resources are already used by URLLC and mMTC tenants. Conversely, our approach allows us to squeeze one extra eMBB slice, leading to an extra 86% revenue after 22h.

2.2 Blockchain for Mobile network resource Brokering

With the announced arrival of the 5^{th} generation of mobile networks (5G), vertical industries can embrace the mobile ecosystem and explore novel sources of revenues. Overcoming the traditional telecom stagnation on connectivity services, *network slicing* expands telecom services towards dedicated virtual network instances, or slices, customized to meet specific industry verticals service requirements. The advantage coming from the adoption of the network slicing paradigm is two-fold: i) Infrastructure Providers (InPs) will be able to reach greater levels of resource sharing, thus increasing the actual utilization of their physical infrastructure by means of statistical multiplexing of requests coming from $3^{\text{-}rd}$ party vertical industries [57]; ii) vertical industries will benefit from dedicated mobile network slices enabling advanced services to their final users, with specific Quality of Service (QoS) and Service Level Agreements (SLAs).

The novel network slicing paradigm, made available by the latest developments on virtualization and softwarization technologies, enables advanced and dynamic resource allocation schemes built on top of modular mobile architectures and commoditized platforms. Such advanced resource allocation mechanisms must deal with a heterogeneous and wide set of vertical requirements to satisfy per-slice performance guarantees. In this context, the figure of the *Network Slicing Broker (NSB)*, firstly introduced in [58], acts as an entity in charge of mediating between industry verticals' slice requests and the mobile infrastructure resource orchestrator.

In this chapter, we extend the NSBconcept towards further dividing the value chain and allowing the entrance of new players in a similar manner as mobile virtual network operators (MVNOs) did in telecom networks. MVNOs allowed InPs to address specific market niches, which they did not manage to tap into due to the *subscriber acquisition costs*. The new challenge here is that, while the number of MVNOs is rather small in established mobile markets, network slicing is expected to accommodate hundreds to thousands of new industry vertical tenants, ranging from full coverage connected car platforms to localized IoT deployments.

In order to achieve this, we introduce the figure of the *Intermediate Broker (IB)*, which leverages on *Blockchain* technology to develop the *network slicing brokering (NSB)* solution NSBChain, enabling *Infrastructure Providers (InPs)* to allocate network resources to IBs through smart contracts and IBs to allocate and re-distribute their resources among tenants in a *secure, automated and scalable* manner. While MVNO agreements with InPs have to go through a regular *offline contract signature* process, NSBChain enables a much faster, scalable and cost-efficient *secure online digital signature* process for the resource allocation transactions.

The contributions of this chapter can be summarized as follows:

- Introduction of a novel hierarchical network slicing brokering framework based on blockchain to support the evolution of the telecom business model.

- Design of a Blockchain-based smart contracts and consensus system that allows, in sliced networks, dynamic resource exchange among tenants.

- Evaluation of our NSBChain framework building on top of the HyperLedger platform [59] and analysis of key performance features, e.g. transaction throughput, communication latency and platform scalability.

2.2.1 The Network Slicing Brokerage Process

The network slicing business model revolves around three main entities [58]: *i)* the *Infrastructure Provider (InP)*, which is the owner of the mobile network physical infrastructure and responsible for its maintenance, *ii)* the *Network Slice Tenants*, which are those business entities, e.g., Over-The-Top (OTT) service providers or 3^{rd}-party vertical industries, interested in renting a slice of the mobile network from the InP to provide tailored services to their customers through allocation of dedicated resources, *iii)* the *Network Slice Broker (NSB)*, which is in charge of mediating between tenants' requirements and network resource availability, and instructing the physical infrastructure to accommodate requests.

In more detail, upon slice requests arrivals, the NSB is in charge of running an admission control mechanism, and if granted, deploying the new slices in the system. Such admission control mechanism involves the evaluation of the slice resource requirements against the resource availability over the different network domains, Radio Access Network (RAN), transport, and core. Keeping running slices SLAs isolated from newcomers is of paramount importance in this scenario as it shall avoid resource shortage that might impact the service delivery. As different tenants may require a diverse set of network resources, the admissibility of each slice request depends on an elaborated multi-domain optimization problem, see for instance [5]. To ease this task, a common solution accounts for the usage of a predefined set of Network Slice Templates (NSTs)[60]. Each template specifies static parameters and functional components of different network slice types as well as the relevant attribute's value in terms of resource allocation requirements necessary to satisfy the service provisioning. An illustration of the workflow is depicted in Fig. 2.9.

2.2.2 Related Work

Both network slicing and blockchain paradigms recently attracted wide interest as a consequence of the hype around 5G mobile networks and cryptocurrencies. Therefore, perhaps not surprisingly, several research efforts started investigating how to combine these two emerging technologies.

In [61] the authors present a study on the leasing ledger concept proposing the blockchain technology as a means to overcome absence of trust in data management and satisfy the need for automated solutions in industrial network facilities. One of the key-features inherited by the blockchain technology is indeed the capability of providing trust in a distributed way. The authors of [62] exploit this feature to minimize (discourage) overcommitting issues during the negotiation of SLAs and radio frequency channels assignments between InP and Mobile Virtual Network Operators (MVNOs). Differently from our

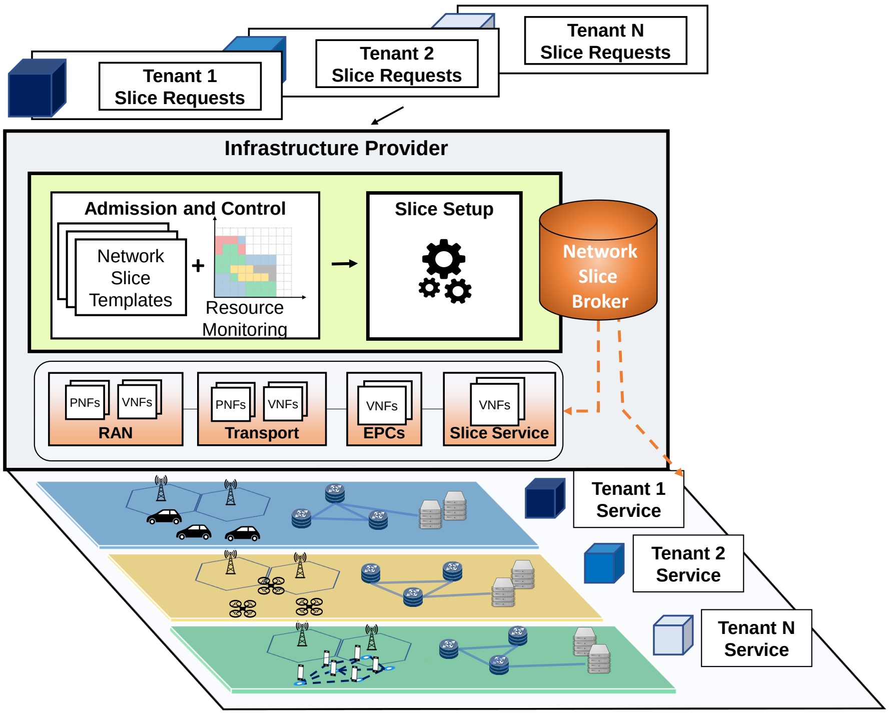


Figure 2.9: *Network slice brokerage overview*

work, they only focus on RAN-specific resource negotiation without considering other network domains. Recent related work along our proposal has been presented by [63] in the context of vehicular ad-hoc network communication. Despite a complete analysis about security and performance aspects, no guidance is provided regarding functions and/or consensus protocols to achieve the complete solution. Finally, [64] proposes to extend the NFV-MANO architecture to account for a dedicated API through which network slices can be configured and orchestrated according to the negotiated transactions. As future work, [64] highlights the need for a consensus algorithm able to manage, in an efficient manner, the huge number of interactions expected in slicing systems.

The key novelty of our framework is the capability to support the network slice resource brokerage process in an end-to-end fashion, embracing the multi-domain nature of the network slicing paradigm and its need to guarantee heterogeneous tenants' requirements, even at fine-grained granularity. Conversely, none of these prior works fully investigates a multi-tenant multi-domain scenario, limiting their analysis at domain-specific implementations.

2.2.3 The Hierarchical Architecture to evolve the Network Slicing Market

While the network slice market is envisioned to unlock a wide set of business opportunities[65], the management of a multitude of relatively small network slices—if geographically con-

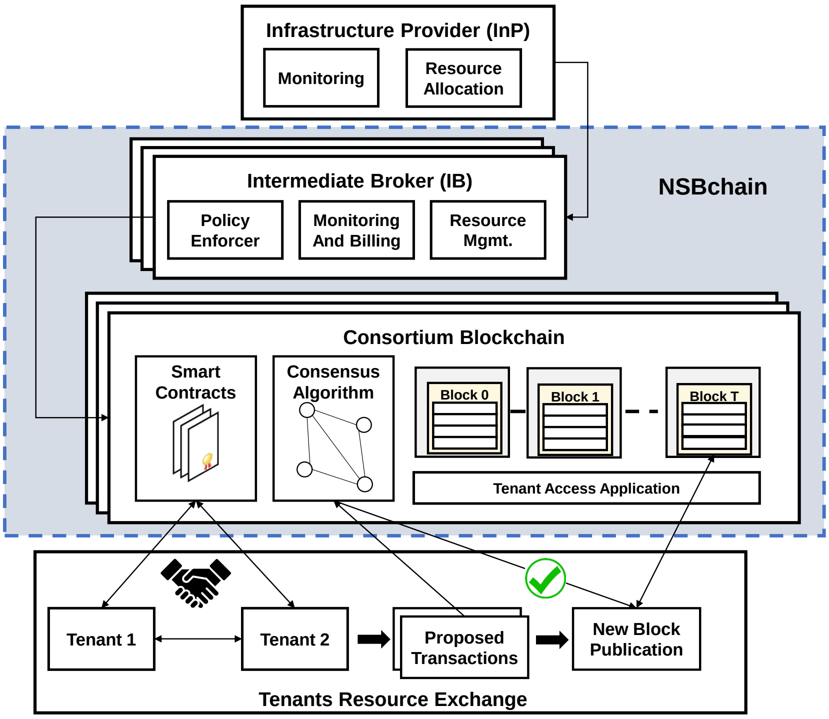


Figure 2.10: *A distributed hierarchical architecture for network slicing.*

strained like small business industries and factories—introduce additional complexity in the orchestration process when performed in a centralized fashion: operators might not want to undertake it.

The network slice ecosystem is envisioned to support dynamic and real-time resource allocation over the mobile network. In such a fast-changing scenario, tenant requirements may vary as a result of external causes, e.g., end-users' mobility, possibly leaving tenants with under or over-provisioned network slices and the need of acquiring/releasing resources.

In this context, the roles of the InP and the wholesaler can be comparable. From this perspective, it is preferable to deal with the exchange of big quantities of goods to intermediate retailers rather than trading, with a significant increase of management costs, small quantities directly with the end-users. Thus, this opens up to new marketing opportunities for 3^{rd}-party entities willing to play the role of retailers, e.g., Mobile Virtual Network Operators, municipalities in case of public events, highway operators and factories, which may buy a quota of network resources from the InP and re-sell it to final tenants. We define such business entities as *Intermediate Brokers* (IBs). We envision the network slicing economy as an open market where tenants can select the IB that best suits their requirements, e.g., better price, thus leading to the creation of consortia of tenants under the management of the same IB. The proposed architecture is depicted in Fig. 2.10.

Technical challenges. In order to support the hierarchical structure above-described as well as the additional management and security complexity inherited by this enhanced business model, several challenges must be considered: flexibility and scalability are key-features for next generation mobile networks.

Network slices should meet tailored and fast service provisioning requirements as dictated by the service diversity foreseen in the 5G era. Fully automated solutions are thus necessary to keep efficient network operations and management while reducing costs. End-user mobility aspects and interference management bring additional complexity in the network slicing context, especially for real-time use-cases. The assignment of network resources to tenants in such cases requires the resource allocation process to evolve dynamically following tenant demand variations.

At the same time, the chain of network resource loans must be negotiated in a secure, transparent and fast way [66, 67], such that the lifecycle of each slice is not affected. Current mobile network sharing solutions require long negotiation processes that hardly fit within short time-to-market deployments of the 5G use-cases.

Due to its decentralized nature, the blockchain technology well suits these requirements. The distributed ledger allows all members of the system to be aware of the current (and past) network resource availability as well as to be informed, in real-time, about the dynamic exchange of resources through a public hash-chain of blocks provided with valid transactions. A secure resource exchange is guaranteed by smart contracts and distributed consensus algorithms, allowing the system to evolve autonomously without the need of centralized authorities.

Blockchain

Despite becoming famous for the hype around cryptocurrencies, the blockchain technology applicability is not limited to that scope. In its simplest definition, a blockchain is a distributed data structure shared among the members of the network. Each block stores information about a set of transactions e.g., timestamp, amount of good exchanged, partners involved and most importantly a reference to the previous block of the chain (usually the hash of its content). The creation of new blocks involves secure cryptographic mechanisms that make the chain unalterable and safe against fraudulent attacks. The content of a blockchain database is broadcast and updated in a decentralized manner, being the absence of a centralized control an advantage for data transparency.

However, the decentralized architecture implies synchronization issues, for example, when dealing with the insertion of new blocks in the chain. This calls for the introduction of a *consensus* mechanism to keep the information contained in the ledger coherent within the network. Several algorithms are available in the literature showing advantages and drawbacks, e.g., proof of elapsed time, proof of work, and so on [68].

We can identify two types of blockchain: i) *Permissionless* chains allow anyone to read, to write and to participate in the creation of the ledger, ii) *Permissioned* blockchains pose restriction on who is allowed to participate in the network activities, e.g., limiting the kind of transactions. Considering the enterprise facility represented by a mobile network

Algorithm 4 Smart Contract implementing an auction-based resource allocation scheme.

```
Input: AuctionEndTime, ResourceSet
Initialize: HighestBidderID = 0x00, HighestBid = 0
while Now() $\leq$ AuctionEndTime do
    ListOfBids = CollectBids();
    for CurrentBid $\in$ ListOfBids do
        if CurrentBid.value $>$ HighestBid then
            HighestBidderID = CurrentBid.peerID;
        end if
    end for
end while
Notify(HighestBidderID, ”Your Bid was the highest.”);
Assign(ResourceSet, HighestBidderID);
```

infrastructure, permissioned access is preferable to maintain high security levels. To this aim, permissioned blockchains often exploit Trusted Execution Environments (TEE) to securely onboard participants and assist with the establishment of the consortium that composes the blockchain network. Such scheme also avoids the need of energy consuming activities related to block validation process, which has been identified as one of the main drawbacks of public blockchain systems [69]. Therefore, we can assume that peer nodes admitted in the system are not malicious and rational, i.e., profit driven.

In the blockchain context, smart contracts are often used to automatize the exchange of goods in reply to trigger events. A smart contract can be defined as an agent that translates contractual clauses into self-enforcing software that minimizes the need of trusted intermediaries. SCs are stored in the blockchain and provided with a unique address, making it easy to be reached from all the peers in the network and inheriting useful security features like distributed consensus agreements to prevent fraudulent usages. The implementation of smart contracts often implies the usage of high-level programming languages, which are then compiled into low-level byte-coded languages and loaded into the blockchain to ensure immutability.

In our framework, we exploit SCs to guarantee reliable auditing and enforce IB-specific policies in the management of requests. For example, one IB may decide to auction his share of resources in different ways [70, 71] or simply sell them to the first coming tenant. The pseudocode of an auction-based SC implementation is shown in Algorithm 4. Peer nodes can invoke a SC by sending transactions to its address. In more detail, if a new transaction is proposed in the system, the contract address can be inserted as recipient address of the transaction. To validate the resource exchange, all the peer entities execute the code using, among the others, transaction payloads and current system state as input arguments of the call [72]. The participation in the consensus protocol finally assures that the new output ledger comes from valid transactions.

2.2.4 The NSBChain framework

Hereafter, we introduce our novel framework, namely NSBChain, showing the main advantages and limitations when implemented in real deployments.

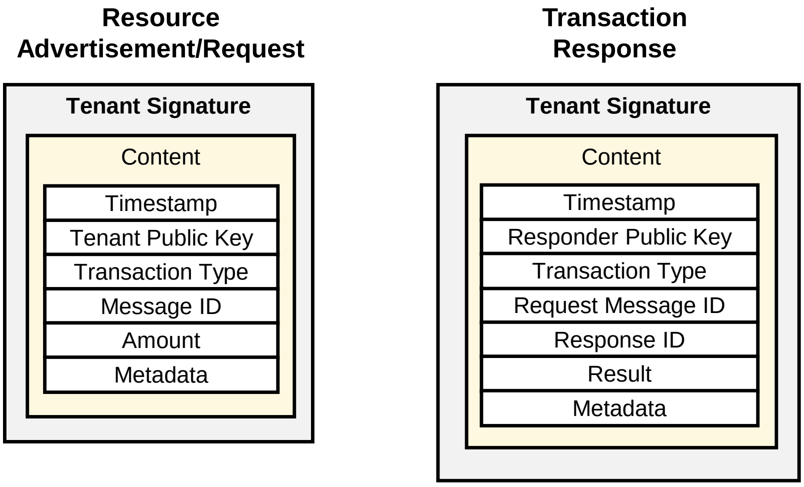


Figure 2.11: *Example of transaction message exchange within NSBChain.*

Smart Contracts

Analytically, let us introduce $\mathcal{B} = \{b_1, b_k, \dots, b_K\}$ as the set of IBs allowed to trade network resources, and $\mathcal{T}_k = \{\tau_1, \tau_t, \dots, \tau_T\}$ as the set of tenants admitted within the consortium of IB b_k. Being a permission-based system, our framework requires an invitation for participation[15]. To guarantee secure message exchange, each entity is provided with a cryptographic key pair $\{K_{priv}, K_{pub}\}$. The usage of group signature schemes and the generation of new key pair for every message exchange is preferable to avoid reply attacks [73].

We detail in the following the main steps involved in the creation and management network slices on a blockchain-based platform providing a mathematical background for the consensus process and the overall revenue maximization. **System Setup**. In order to enable dynamic resource exchange among tenants, a dedicated blockchain must be set up for each consortium of tenants. Each IB b_k deploys the first block of the chain and loads a registry of resources $\mathcal{R}_k = \{r_1, r_i, \dots, r_I\}$ into such a block, which reflects the amount of i-type resources, with $i \in \mathcal{I}$, originally assigned by the InP. This step is required to avoid over-selling so as to limit the availability of resources in the blockchain. Each IB b_k can define leasing policies and code them into a set of SCs, which are then available to all tenants in the consortium. Finally, each IB b_k is in charge of assigning the initial share of resources to admitted tenants.

Message Exchange. Upon private exchange domain creation, network slice requests can be dispatched among the network of peers. According to their real-time requirements, tenants may decide to publish a resource advertisement or a resource request message. In the former case, the current owner of resources decides to release some of his shares making them available on the market. In the latter case, the tenant broadcasts its need to other tenants, which may be interested in providing their quota. To guarantee authentica-

---

[15] While the admission procedure is out of the scope of this work, it is assumed that such a mechanism is in place and managed by the InP to guarantee that only trustworthy entities are admitted.

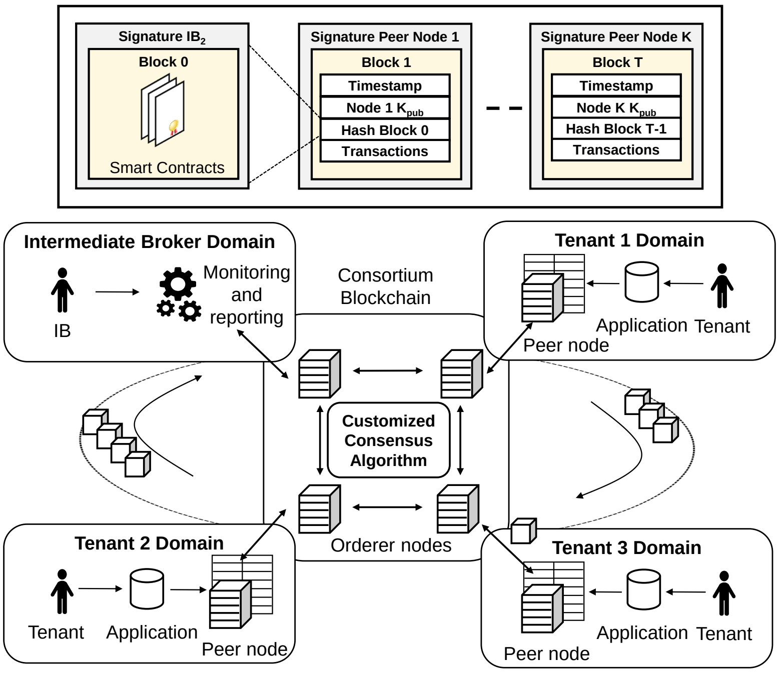


Figure 2.12: *Private blockchain architecture supporting network slicing.*

tion, each message is signed with the sender private key and uniquely identified by an ID number. A simplified message structure is depicted in Fig. 2.11.

The network slicing brokerage must deal with multi-domain resource allocation problems. In its simplest definition, a resource request from tenant τ can be defined as a tuple $\Psi_\tau = [\pi_1^{(\tau)}, \pi_i^{(\tau)}, \dots, \pi_I^{(\tau)} | \theta_1^{(\tau)}, \dots, \theta_I^{(\tau)}]$, where $\pi_i^{(\tau)}$ represents the required amount of i-type resources, and $\theta_i^{(\tau)}$ is the price to be paid. It should be noticed that we do not pose any limitation on the nature of exchanged resources, and that the proposed resource request scheme easily accommodates heterogeneous resource specifications. For example, a tenant could be more interested in trading only Radio Access Network (RAN) resources at the edge of the network, e.g., for delay sensitive applications, while others may be more interested in cloud resources, e.g., storage and processing power for data analytic applications in the context of the Internet of Things (IoT).

Billing Management. Interestingly, a blockchain can be viewed as a transaction-based state machine, wherein its state is updated every time consensus is reached on a set of transactions. To this aim, orderer nodes can be introduced and exploited to collect and sort proposed transactions by arrival time. Such nodes are usually not involved in the validation process, however they may allow decoupling and parallel processing of ordering and validation functionalities thereby improving the overall system efficiency [64].

We show the blockchain architecture with involved players supporting network slicing in Fig. 2.12. Specifically, the IB may join blockchain activities, i.e., it might read the blockchain results, participate to validation and consensus phases as an active member of the blockchain consortium. This implies that the IB can recursively apply confirmed (validated) transactions onto resource scheduling policies that might include (not limited to) RAN/transport and computational resources. Despite enabling a more dynamic resource trading market, the blockchain technology would easily allow to keep track of the different resource exchange over time. From the InP perspective, this also simplifies the billing management as each block of transactions stores precise information about the nature of the exchanged resources and the corresponding time window utilization. Moreover, tenants are directly responsible for the management of their requests: once issued, they could not be withdrawn. Clearly, each IB shows interest in managing properly its resource share with the objective of maximizing the overall final revenue while parsing and processing upcoming slice requests. Let us assume that each tenant τ can issue multiple slice requests so that the IB can collect all coming slice requests Ψ_j with $j \in \mathcal{J}$. Let us denote $x_{i,j}$ as a decision variable indicating whether i-resources of request j is assigned (to the tenant issuing such a request) whereas y_j is used to prevent from partially assigning resources to a single slice request: in other words, a slice request is accommodated only if all types of demanded resources can be assigned to the tenant thereby guaranteeing a correct end-to-end slice instantiation. We can formally write the following optimization problem:

Problem `IB-REVENUE-MAX`:

$$
\begin{aligned}
\text{maximize} \quad & \sum_j \mu_j y_j \\
\text{subject to} \quad & \sum_j \pi_i^{(j)} x_{i,j} \leq r_i, \quad \forall i \in \mathcal{I}; \\
& \sum_i x_{i,j} \geq I y_j, \quad \forall j \in \mathcal{J}; \\
& \sum_i x_{i,j} \leq I y_j, \quad \forall j \in \mathcal{J}; \\
& x_{i,j} \in \{0,1\}, \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}; \\
& y_j \in \{0,1\}, \quad \forall j \in \mathcal{J}.
\end{aligned}
$$

where $\mu_j = \sum_i \theta_i^{(j)}, \forall j \in \mathcal{J}$ represents the overall revenue from all types of resources included within the slice request j, and $\mathcal{I}$ represents the resource set with $I = |\mathcal{I}|$. The above optimization problem can be easily mapped onto a integer-linear programming (ILP) problem and solved by means of commercial solvers, e.g. [74].

Consensus algorithm. The ownership of a resource set can be transferred from one tenant to another by invoking a transaction on the blockchain. The transaction is validated only if all the relevant parties agree, namely, a consensus among peer nodes is reached. When dealing with consensus algorithms, a trade-off between transaction throughput and latency must be considered. We define transaction throughput as the number of transactions that the system can handle per unit of time. In realistic scenarios, this number can range over a wide range depending on the study use-case. For example, public BitCoin's network supports 7 transactions per second, while the financial networks of MasterCard and Visa handle up to 60000 [75]. Obviously, different consensus algorithms provide different latency. For this reason, we let each IB b_k choose the preferred method according to its service requirements. In general, being NSBChain a permissioned framework, we suggest the use of relatively light mechanisms, like Practical Byzantine Fault Tolerance (PBFT) consensus protocol [76], Kafka [77] or Raft [78] to allow fast convergence to a common agreement and speed up the resource exchange process.

2.2.5 Proof-of-Concept Evaluation

We implement NSBChain on top of Hyperledger Fabric [79], an open-source framework for developing permissioned blockchains within private enterprises, and make use of its benchmarking tool, namely Hyperledger Caliper, to evaluate the blockchain performance in network slicing scenarios.

Experimental setup. Our Proof-of-Concept (PoC) architecture consists of 3 IBs and a variable number of orderer nodes that depends on the adopted consensus algorithm. Such entities run as Docker containers on an Intel Xeon CPU E5-2630v3 32-Core @2.4GHz 64GB RAM shared platform.

We guarantee the isolation among consortia through the definition of dedicated and encrypted communication channels. Moreover, we set the maximum number of entries per block to 20 and the block timeout[16] to 300 ms. This last metric specifies the amount of time (after receiving the first transaction) each orderer waits before publishing a new set of proposed transactions to other peer nodes. Please note that the choice of those parameters may strongly affect the blockchain performance. In particular, although decreasing the block timeout improves the latency, setting it to low values may decrease the overall throughput as new blocks would not be filled up to their maximum capacity. To limit the impact of this trade-off on our results, we do not modify these settings throughout this section.

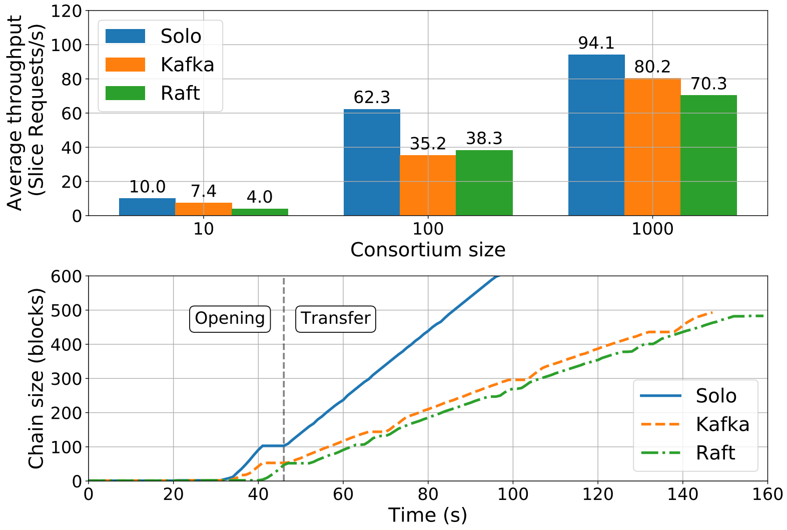
The benchmark process consists of two phases, dubbed as opening and transfer. In the initial phase, we create tenant instances and assign them with an equal amount of resources such that all available resources at IB side are assigned. Once assigned, each tenant might decide to free or seek additional resources based on a random value drawn from a uniform distribution between 0 and 30% of the initially assigned amount[17]. During the transfer phase, tenants issue Slice Requests (SRs), modeled as tuples $\Psi_\tau = \{\rho, \eta, \gamma\}$, where $\rho, \eta, \gamma \in \mathcal{R}_k$ represent the percentage of required radio access, transport and core cloud resources, respectively. In case the SR does not fit the availability or the need of the involved tenants (SR collision), it is automatically rejected and the respective transaction is dropped.

Full-scale evaluation. With the first experiment we evaluate the performance of our framework in terms of slice request throughput and latency. We compare two popular consensus algorithms (Kafka and Raft) against a single orderer configuration (Solo) that does not require any consensus process. The top of Fig. 2.13a shows the average SR

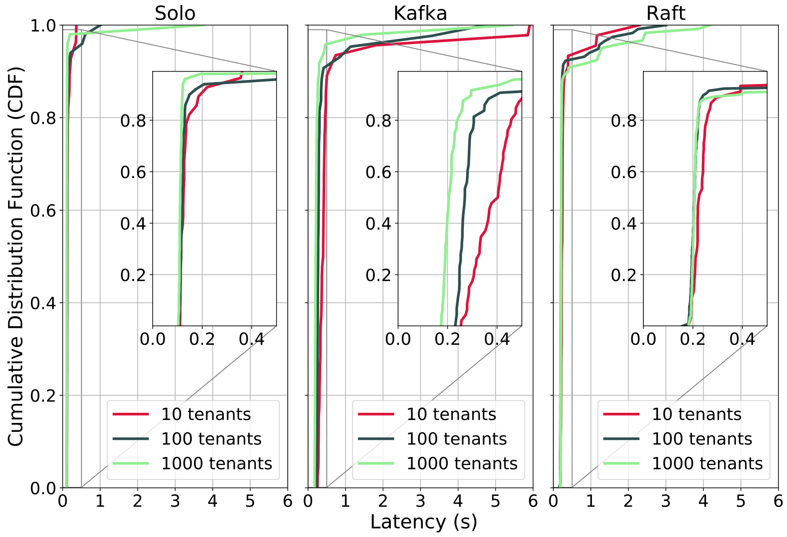
---

[16] We select such values as they maximize the throughput at a minimum latency cost as proved hereafter in the section.

[17] We empirically prove that the choice of this value leads to convergence within a reasonable time.



(a) *Slice request throughput and blockchain size growth for different consensus algorithms and consortium size.*



(b) *CDF of the slice request validation latency experienced by tenants for different consensus algorithms and consortium size.*

Figure 2.13: *Performance evaluation for different consensus algorithms and consortium size a) Slice request throughput and Blockchain size growth b) Validation latency.*

throughput of the platform in the transfer phase for an increasing consortium size and fixed SR rate of 150 SRs/s to emulate high load conditions. In these settings, especially for a small consortium size, the limiting factor of the blockchain performance throughput is the Multiversion Concurrency Control (MVCC) process. As we issue SRs at a very high rate, the same database entry, e.g. the resources assigned to a specific tenant, may be edited by a new request before the completion of the validation process involving it. This

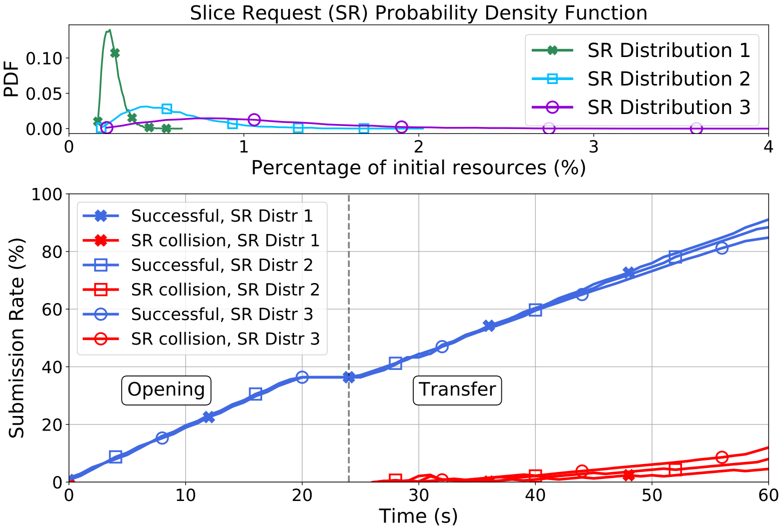


Figure 2.14: *Transaction acceptance and error rates for different scenarios.*

raises a database inconsistency, dubbed as Read/Write (RW) conflict, which prevents the current transaction to be successful. As shown in the figure, this problem is mitigated by an increasing consortium size.

Fig. 2.13b depicts the Cumulative Distribution Function (QoE) of the experienced latency by the successful SRs. As expected, the best latency performance is obtained when no distributed consensus mechanism is in place, i.e., Solo. However, despite being the fastest scheme, this single-node approach is not fault tolerant. It can be noticed that the transaction exchange and validation process introduce a small time overhead for the Kafka and Raft cases, which however has negligible impact, especially when compared to the on-boarding time required e.g., by virtualized infrastructures to setup virtual services [67]. The blockchain growth rate is also affected by the different consensus scheme, as shown at the bottom of Fig. 2.13a, which refers to the consortium size case of 1000 tenants. We plot the evolution of the chain size over time and mark the beginning of the transfer phase with a dashed vertical line. It can be noticed that the blockchain grows at a rate proportional to the average throughput since blocks are filled up to their maximum capacity.

Brokering scenario evaluation. The second experiment focuses on evaluating the capabilities of the system when dealing with the brokering scenario. To this aim, we consider 3 IBs managing a consortium of 1000 tenants, correspondingly. In light of the performances shown above, we select Kafka as consensus algorithm for its high fault-tolerance and scalability [80]. We assume that resource request values ρ, η, γ are drawn from a right-skewed distribution over a positive interval as resource requests must be non-negative. Such distributions are depicted at the top of Fig. 2.14 for different demand ranges, spanning from 0.1% to 4% of the tenant initial resources. Note that since we assume the same distribution for all resource requests within the same slice, it is dubbed as SR PDF.

The bottom of Fig. 2.14 illustrates the system behavior for a constant submission rate of 50 SRs/s so as to keep RW Conflicts to a minimum (around 2% of the submitted SRs). In such operational conditions, errors raise only in case of SR collisions. It is worth noting that SR collision rate increases along with the SR variance. Specifically, SR distributions with high variance leads to tenant satisfaction more quickly than with a lower variance. Additionally, the closer to tenant satisfaction, the lower the resource availability and, in turn, the smaller the likelihood of a request to be accepted by the system.

2.3 Conclusions

In this chapter, we have presented a novel yield-driven orchestration platform that explores the concept of *slice overbooking*. Notably, our solution is specifically designed for the orchestration of slices end-to-end, across multiple heterogeneous domains of the mobile ecosystem. To this aim, our design is based on a hierarchical control plane that governs multiple domain controllers across a mobile system and uses ETSI-compliant interfaces and data models. Our system embeds a control engine in charge of making (i) admission control and (ii) resource reservation decisions by exploiting monitoring and forecasting information. Our overbooking mechanism is grounded on an optimization formulation, providing provably-performing algorithms that achieve up to $3x$ revenue gains in several realistic scenarios built upon data from three real mobile operators. The feasibility of our approach has been validated over an experimental proof-of-concept composed by conventional mobile equipment and on top of available open-source software.

Network slicing has been identified as a key enabler for the development of novel business models in 5G and beyond mobile networks. Therefore, in the second part of this chapter, we introduced a hierarchical blockchain-based framework, NSBChain, that provides a brokering solution between the infrastructure provider and network tenants willing to pay for acquiring, exchanging and managing network and computational slice resources within the domain of an intermediate broker. We developed a Proof-of-Concept implementation leveraging on the open-source Hyperledger platform and showed that our approach is feasible and scalable (up to 1000 tenants were considered) with different state-of-the-art consensus algorithms.

Chapter 3

Network Slicing Resource Orchestration

The results disclosed in the following chapter are partially taken from the granted patent [13].

3.1 Network Slicing at the Edge

The convergence of Information Technology and networking finds a realization when it comes to Multi-access Edge Computing (MEC). It is widely recognized as a promising technology to bring cloud computing capabilities to the edge of network, where low latency and high bandwidth can be exploited by cloud applications in order to deliver added-value services to the end users. Targeted use cases include tactile Internet, augmented and virtual reality, live streaming, etc., and also those belonging to specific vertical industry segments, as industrial automation, eHealth, automotive, etc.

The European Telecommunications Standards Institute (ETSI) has chartered the Industry Specifications Group[1] precisely to define a multi-vendor standardized MEC system to allow third party software providers to install their applications in the network operator's premises. According to the ETSI architecture [81], an *MEC provider* (e.g., a mobile network operator) owns the system that comprises a set of IT hosts and the management entities, building the concept of Infrastructure as a Service (IaaS). Third party software providers (i.e., the *MEC tenants*) deliver their application package(s) to the provider as responsible for the deployment of such application instance(s) in the MEC hosts, configuring the appropriate parameters and traffic rules, and for the policies' enforcement to provide specific tenant Service Level Agreements (SLAs).

Beyond the IaaS model, network operators are exploring novel business means to monetize their infrastructure by offering *Network Slices* to external tenants [82]. From the MEC perspective, a slice is envisioned as a technological opportunity for the tenant to operate and manage the MEC system according to different privilege levels while gaining more control over the delivered service. This would result in an advanced multi-tenancy multi-access edge computing architecture, namely M^{2}EC. An example is represented by a Mobile

---

[1] Please refer to `http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing` for further information.

Virtual Network Operator (MVNO) willing to acquire a slice of MEC to expand its business model and operate on the network management in order to tailor the system according to its dynamic requirements.

The MEC slicing concept enables the infrastructure provider to facilitate a set of heterogeneous privileges to multiple tenants. However, when different tenants operate on a shared infrastructure, efficient mechanisms must be in place to validate upcoming tenant requests and policies and to enforce them while avoiding conflict states. In this context, our contributions may be summarized as follows.

- Introducing the *MEC Broker* from an architectural viewpoint as an entity capable of exposing heterogeneous administration privileges to MEC tenants, of processing tenant requests, and of resolving resource contentions.
- Devising an *Orchestration mechanism* compliant with the ETSI-defined operations, able to fulfill the tenant requests and to avoid SLA violations.
- Defining an *Optimization Framework* that supports the MEC slicing paradigm pursuing the infrastructure resource efficiency maximization while accounting both end-to-end application delay requirements and platform computation/storage capabilities.

3.1.1 Background

We present a network slicing support mechanism built on top of the ETSI-defined MEC architecture [81], which simplified version is depicted in Fig. 3.1 whereas a brief description is given below.

In the service model envisioned by the MEC architecture, the tenants deliver their application software and the descriptor files to the system provider. The MEC host is then the IT infrastructure where such applications run as virtual machines (VMs). The MEC platform sitting in each host enables the applications by providing them access to the MEC service API endpoints, handling DNS procedures and enforcing the traffic rules and policies on the data plane [83].

In order to get the applications running in the desired location of the network, the MEC provider's Operations Support System (OSS) is the highest level management entity to accomplish the task, upon the requests coming from an MEC tenant. This is reflected on top of Fig. 3.1, where *Tenant 1* is represented as an entity similar to an OSS/BSS (Business Support System) which uses the Customer Facing Services (CFS) portal in order to request the instantiation and termination of an MEC application. The MEC provider's OSS receives the requests from the CFS portal and operates the MEC orchestrator and the MEC platform manager to fulfill them. Additionally, the MEC system allows applications running in the user's device to interact with the MEC system to perform the requests through the User app LCM proxy, which is connected to the MEC provider's OSS and MEC orchestrator to validate and satisfy the requests. It is safe to assume that also the user application logically

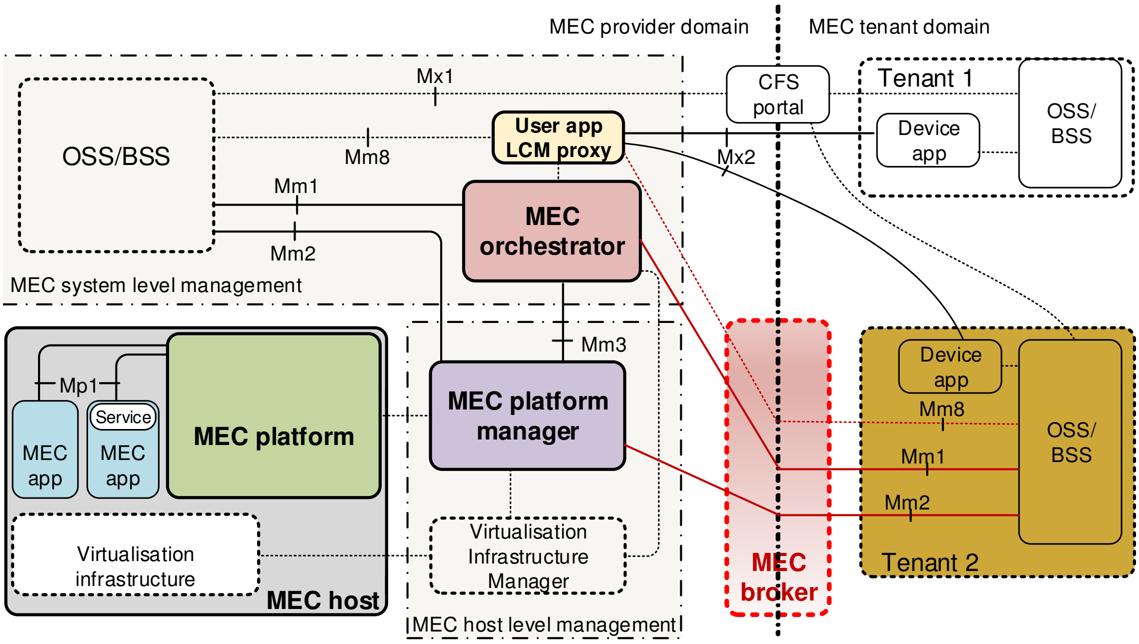


Figure 3.1: *Simplified Multi-access Edge Computing (MEC) architecture enhanced with the MEC broker. The diagram shows how different tenants can access the system via the legacy mechanism (Tenant 1) and the one proposed by the present solution (Tenant 2).*

belongs to the tenant, so that Tenant 1 using the MEC legacy system is allowed to interact with the MEC system only through the CFS portal and User app LCM proxy.

The role of the Tenant 1 is thus limited to controlling the application's logic (e.g., through remote access to the application's backports) once the application is up and running, whereas the MEC provider is responsible for the instance configuration and management, as per the following non-exhaustive list of operations, in fulfillment of the agreed SLAs and of the MEC service provider's own policies and capabilities:

- Application placement, i.e., the set of available MEC hosts where the application is installed and executed;
- Management of the application-assigned networking, computing and storage resources;
- Application LCM, including bootstrapping, termination, scaling in/out and up/down, migration (this latter assumes also validation and enforcement of the policies for application migration to other MEC hosts);
- DNS and traffic rules configuration in order to provide the appropriate connectivity.

While the IaaS model described above meets the expectations of several tenants, an MEC slice creation and management concept of different tenants attracts additional stakeholders having the capabilities and willing to gain full control over the delivered service.

In Section 3.1.2 we propose the architectural enhancements to MEC and the key concepts to enable MEC slicing. In the following paragraph we showcase some prior art on the topic.

Related Work

To the best of our knowledge, we pioneer the slice resource allocation framework and practical mechanisms in an ETSI-compliant MEC system. However, the network slice broker entity has been initially introduced in [58] with the objective of solving the admission and control problem of slice requests in mobile Radio Access Network (RAN) facilities. Performance evaluations there show promising results in terms of low SLA violations even in dynamic scenarios [84]. Bringing this entity in MEC environments opens the possibility to exploit results from other research domains, given the parallelism with bin packing problem in cloud computing and virtual function placement problem in virtual network embedding fields. In cloud computing context, a wide set of optimization criteria can be defined [85]. Energy consumption or average latency minimization, Quality of Experience (QoE) maximization as well as optimization of the number of migrations. All these metrics are also suitable for MEC systems, even if they are less performing and consume less energy due to limited capabilities with respect to cloud data centers. The authors of [86] address the VNF placement problem by considering the highly dynamic nature of cloud systems and by modeling requests as a continuous stream. Their solution considers two steps. The first consists in a continuous deployment decisional process, which is followed by a placement re-optimization phase that includes migrations. A way to optimize migrations is to consider consolidation of resources. [87] deals with initial VM placement including spatial and temporal awareness for consolidation purposes based on the forecast of resource demand in time. Similarly to our work, the authors of [88] and [89] present VNF placement and provisioning optimization strategies over edge and cloud infrastructure taking into account Quality of Service (QoS) requirements. Their objective is to solve the trade off between optimization of resource utilization and the minimization of SLA violations.

3.1.2 A Broker to Support Slicing in MEC

In the previous section a legacy mechanism to access the UE system by third parties and its limitations are shown. Advanced tenants willing to decide on the application LCM on their own, should be granted access to the management entities that are visible from the UE provider's OSS, i.e., the UE orchestrator, the UE platform manager and the User app LCM proxy.

We thence propose the *UE broker*, as a logically entity able to expose to a tenant's OSS the interfaces that connect the UE provider's OSS to the entities mentioned above (namely, the interfaces supported by MEC reference points Mm1, Mm2 and Mm8 [81, 90, 91]). In Fig. 3.1, one can see that *Tenant 2* is enabled with enhanced operations through the proposed MEC broker, as compared to Tenant 1 which accesses the MEC system with legacy mechanisms only.

By exposing the management interfaces to multiple tenants, the MEC broker enables them to manage the same system even simultaneously, thus it must ensure the consistency of the policies, configuration items and commands issued by the tenants.

Therefore, the MEC broker assigns tenants with *privileges* and *priorities*. Privileges refer to the set of allowed actions that a tenant is authorized to perform, which map directly to the usage of the MEC interfaces. Priorities refer to the validity in time of a privilege, and how commands issued by a tenant take precedence over those by the others.

Conflicts may occur when shared resources are simultaneously used or if the MEC system is running out of resources. The MEC broker will efficiently try to reduce the number of conflicts by promptly balancing the MEC tenants' applications over available MEC hosts. This is strictly related to the privileges issued to running MEC tenants and might be needed an optimal admission control in charge of filtering incoming MEC slice requests based on the current availability.

In order to carry out the operations above described, the MEC broker is logically equipped with functional elements that:

- Implement the exposure of the interfaces that are transported over the MEC reference points connected to the OSS entity, namely Mm1, Mm2 and Mm8;

- Grant, revoke, modify and check privileges and priorities, including the communication to the tenant to enable such operations; This block may consist of a login-based procedure through which a tenant acquires privileges and priorities, asks for updates and queries the status.

  A list of available privileges is provided by the MEC broker, which may update it and forward to MEC tenants (upcoming or currently connected).

- Record all the instructions issued by tenants over the exposed MEC reference points in order to validate and execute them. When any of the MEC operations is requested by the tenant, the priority associated to the privilege is looked up. The look-up determines if the operation is actually granted to the tenant and the conflicts associated to the operation (e.g., installing a DNS record with an already used IP address or domain name) are evaluated. If the check is successful, the operation can be executed. Otherwise, if a conflict is detected, the system generates an alarm to all the lower priority tenants, which are automatically disabled from executing such operations.

From a deployment point of view, the MEC broker can be realized in different ways, e.g., *i)* as an additional entity, like that depicted in Fig. 3.1, *ii)* by extending the capabilities of the MEC Orchestrator, the MEC platform manager and the User app LCM proxy, *iii)* by augmenting the MEC provider's OSS or the CFS portal.

The architectural enhancements proposed in this work enable tenants to access an MEC system with additional control over the MEC resources as compared to the legacy mechanism. In the following sections, we describe and validate an orchestration system able to allocate MEC resources when requested using both the legacy and our mechanism.

3.1.3 Multi-tenant Resource Orchestration

The role of the MEC orchestrator is to properly instantiate the tenant applications into the set of MEC hosts fulfilling the functional requirements of the applications and the agreed SLAs. Functional requirements might include virtual resources such as computing burden, storage and network throughput, as well as the dependency to particular *UE services*. An MEC service is a specific application in an MEC host able to expose an API to other applications providing enhanced information, e.g., radio conditions [92] or location of users [93]. Some services are built-in within the MEC platform whereas others may be installed on-demand. In addition, MEC applications run in fulfillment with tenant SLAs, e.g., the maximum tolerable delay experienced by the application client running in the user device, the maximum number of application instances running simultaneously in the MEC system, or a list of granted hosts for a specific application due to regulatory limitations.

In light of the above considerations, we define three different categories of MEC tenants based on heterogeneous application requirements and given privileges:

Basic. Tenants request to run one or more instances of the application on different hosts, with loose delay requirements and in absence of management privileges;

Premium. Tenants request to run one or more instances of the application on different hosts with *stringent* delay requirements and in absence of management privileges;

Gold. Tenants request direct access to an isolated slice of the MEC platform, which includes both management privileges and low-delay guarantees. This slice provides a guaranteed MEC applications deployment onto specific MEC hosts.

The orchestration system aims at allocating needed resources to run tenant applications while fulfilling application delay requirements, i.e., to accommodate applications onto available hosts. We devise an orchestration algorithm in charge of finding the optimal placement while pursuing the overall MEC hosts resources minimization. This opens up new opportunities for admitting additional MEC tenants (or MEC slices) and, in turn, increasing the overall system revenues.

Scenario Characterization and System Model

The MEC system is described as the set of deployed hosts H_i, where $i \in \mathcal{I} := \{1, \ldots, I\}$, connected to each other through provider backhaul links as per the Mp3 MEC reference point [81]. Thus, any host pair (H_i, H_j), $i, j \in \mathcal{I}$, is logically connected via a link which associated overall latency is $\delta_{i,j}$, regardless of the actual number of hops. In addition, each host is equipped with fronthaul links for user access (i.e., towards the base stations) which are modeled following the same principle. Thus, the variable λ_i accounts for the average delay between the users connected to host H_i through all its access link and the host itself (as shown in Fig 3.2). Given the purpose of our model to guarantee delay constraints, for the sake of simplicity we assume that each link has enough capacity to satisfy traffic requirements. Moreover, each host H_i is characterized by its computing resources, synthesized by the *total capacity* parameter c_i. Thus, each host can be described as $H_i = \{c_i, \lambda_i, \boldsymbol{\delta_i}\}$ where $\boldsymbol{\delta_i}$ comprises delays $\delta_{i,j}, \forall j \neq i \in \mathcal{I}$. Services offered by the MEC system are modeled as set S_w where $w \in \mathcal{W} := \{1, \dots, W\}$, and each service S_w consumes s_w resources from the host[2].

The infrastructure described above shall accommodate the set of applications A_k requested by tenants, where $k \in \mathcal{K} := \{1, \dots, K\}$. Without loss of generality, hereafter we refer to a k-th tenant through its application A_k. Although the admission and control procedure is out of the scope of this work, it is assumed that such a mechanism is in place and translates the incoming tenant requests into the following parametrization associated to the requested application A_k:

- a_k application's processing consumption;
- Δ_k maximum tolerable end-to-end delay between the application and the user consuming such an application;
- $b_{k,w}$ list of required services for each application.

Additionally, we assume each tenant asks for the deployment of its application in one or more hosts, thus requests are modeled as a set of binary variables $z_{k,i} \in \{0, 1\}$, where $z_{k,i} = 1$ if A_k is requested on host H_i, and 0 otherwise.

Overall, the above information allows to optimally define the placement strategy for applications and services over MEC facilities in a multi-tenant scenario. Obviously, the procedure becomes more challenging as the number of hosts and requirements stiffness increases.

MEC Slicing Problem

Multiple instances of the same application can run over different hosts in the network. However, given the limited capacity of MEC hosts and assuming a non-uniform distribution across the network, it could be useful from the provider's point of view to migrate some applications. For instance, an overloaded host can be offloaded by migrating some running instances to another location[94]. MEC systems could further benefit from migrations if we consider that different applications may require the same services. In particular, *Consolidation* of spread applications allows for an overall processing capacity utilization reduction and consequent operational cost savings [95]. Moreover, the saved capacity could be engaged to admit more requests in the future and to increase the acceptance rate given the same physical network, thus, from an economic standpoint, rising revenues.

Request arrivals/departures occur every time a tenant requests to modify locations and/or privileges or deploy new applications. Upon receiving the slice requests set, the provider has to deal with the applications placement problem, pursuing the objective of overall resource utilization minimization while honoring the past agreed guarantees. Fig. 3.2 depicts the workflow in case of K=2, W=2 and I=4. In this example, two tenants

---

[2] Please note that we assume a constant utilization of computing resources, irrespective of the actual load. A more accurate model should account for the consumption as a function of the load, being the derivation of such function a complex modeling problem by itself, which is out of scope of the present study.

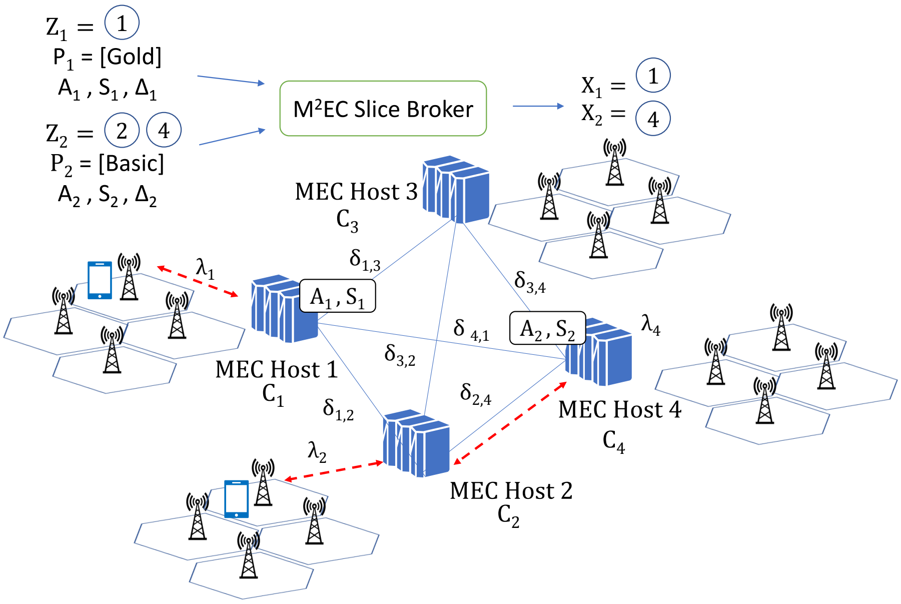


Figure 3.2: *M^{2}EC Slicing Architecture*

ask for the deployment of their application on specific hosts providing latency and service requirements. The tenant with higher privileges asks for the application deployment on the first host while the other tenant, with basic privileges, on the second and fourth hosts. At the end of the decisional process, *gold*-type requests are completely satisfied whereas *basic*-type requests are properly mapped by the M^{2}EC system so as to guarantee the delivery of the service while saving MEC resources. In particular, the second application has loose delay requirements, so it can be enabled in a less congested location, e.g., host H_4, without affecting the final service delivery. In this case, mobile users under the coverage area of host H_2 can access the service through host H_4. We can formulate our problem as the following.

Problem 7 (MEC Slicing Problem)**.**

$$\min_{\boldsymbol{x},\boldsymbol{y}} \sum_{i \in \mathcal{I}} \Big(\sum_{k \in \mathcal{K}} a_k x_{k,i} + \sum_{w \in \mathcal{W}} s_w y_{w,i} \Big) \tag{3.1}$$

s.t.

$$\sum_{k \in \mathcal{K}} a_k x_{k,i} + \sum_{w \in \mathcal{W}} s_w y_{w,i} \leq c_i, \ \forall i \in \mathcal{I}; \tag{3.2}$$

$$z_{k,j}(\lambda_j + \delta_{i,j}) \leq \Delta_k x_{k,i} + M(1 - t_{k,i,j}), \ \forall k \in \mathcal{K}, \forall i, j \in \mathcal{I}; \tag{3.3}$$

$$z_{k,i} p_k - x_{k,i} \leq 0, \ \forall k \in \mathcal{K}, \forall i \in \mathcal{I}; \tag{3.4}$$

$$b_{k,w} x_{k,i} - y_{w,i} \leq 0, \ \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \forall w \in \mathcal{W}; \tag{3.5}$$

$$\sum_{i \in \mathcal{I}} t_{i,j,k} \geq 1, \ \forall k \in \mathcal{K}, \ \forall j \in \mathcal{J}. \tag{3.6}$$

Decision Variables The decision variable $x_{k,i} \in \{0,1\}$ denotes whether an incoming tenant request for application A_k is placed on host H_i. The decision variable $y_{w,i} \in \{0,1\}$ establishes whether the service S_w is enabled on host H_i. Finally, $t_{i,j,k} \in \{0,1\}$ fictitiously models the choice to migrate the application A_k from H_i to H_j.

Objective Function The main goal is the definition of an optimal MEC applications placement, which allows the coexistence of heterogeneous tenants while minimizing the overall resource consumption. The result can be *a-posteriori* translated into operational costs reduction policies or used to efficiently drive the current admission and control strategy by increasing the request acceptance ratio.

Constraints Eq. (3.2) represents a capacity constraint that relates application and service consumptions with the hosts capability. Eq. (3.3) sets the maximum delay budget for each application and destination host. Tenant requests are represented by the variable $z_{k,j}$ as explained in Section 3.1.3, which takes into account the willing of the tenant to deploy the application A_k on a specific MEC host H_j. However, such a tenant might not have the privileges ($p_k \in \{0,1\}$) to demand for a guaranteed MEC applications deployment, and thus its instance might be automatically migrated to a more convenient location. With Eqs. (3.3) and (3.6), we assure that at least one delay request from each tenant is satisfied during the decisional process by exploiting the *Big M Method* [96] and the fictitious variable $t_{i,j,k}$. This also prevents from applying the straightforward solution of placing no tenant request in our MEC system. The value of M must be chosen sufficiently large so that the fictitious variable would not be part of any feasible solution (for e.g., $M = 1000$).

Application consolidation and migration are applied based on latency values of MEC links accounting for a delay cost $\delta_{i,j}$ between hosts H_i and H_j. As introduced before, tenants belonging to different categories are provided with diverse privileges. This is taken into account by Eq. (3.4), which ensures that *gold*-type requests ($p_k = 1$) will be entirely satisfied. Last, Eq. (3.5) enables the concurrent deployment of selected services required by all applications ($b_{k,w}$) running on specific hosts.

3.1.4 Solution Validation

We choose a real network topology from [97] for evaluation purposes. In particular, such network deployment, namely GARR, is composed by 37 hosts spread over the italian territory and more than 80 edges connecting them. Given the set of edges and node locations, the delay matrix with values $\delta_{i,j}$, $\forall i, j \in \mathcal{I}$, is easily obtained running the Dijkstra's algorithm. Let $\bar{\delta}$ and δ_{max} define respectively the average and maximum value of such delays. Without loss of generality, host capacities are equally distributed and normalized to a unitary value. The computational requirements for each application and service are expressed with respect to a fixed value γ, which represents $1/100$ of the single host capacity, as resumed in Table 3.1.

These values are small enough to fit into hosts, as it would be after the execution of the admission and control process. The relationship between services and applications mod-

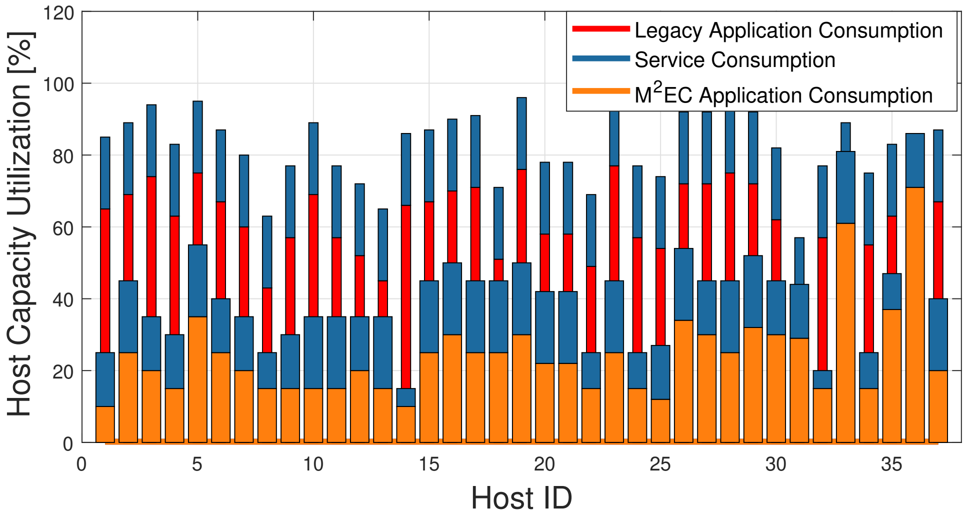


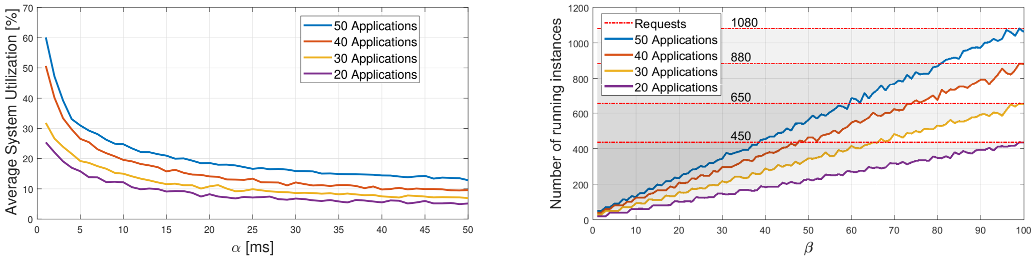
Figure 3.3: *Host capacity utilization for K=40 and W=4*

eled through the binary variable $b_{k,w}$ is obtained randomly at runtime. Given the lack of comparable MEC slicing solutions in the literature, we firstly evaluate our proposal against the baseline approach (*Legacy*) of placing the application and relative services exactly in the MEC hosts where each tenant demands. We run our simulations combining MATLAB and Optimization Programming Language (OPL), together with the optimization engine CPLEX. Fig. 3.3 shows the host utilization for a set of requests distributed among the available categories according to 40%, 40%, 20% ratios, respectively. The *legacy* solution results in an overall increase of resource utilization due to the unavoidable concurrent deployment of the same services and applications, even on nearby hosts. In the M^{2}EC scenario, the MEC applications deployment privilege is guaranteed only to *gold* tenants. It can be noticed that M^{2}E allows for a smarter deployment of resources accounting for services consolidation and leading to 40% of resources saving.

We investigate the MEC host capacity savings in case of variable delay requirements. By denoting with λ_{min} and λ_{max} the minimum and maximum fronthaul delay, respectively, the application delay requirement Δ_k cannot be $\Delta_k < \lambda_{min}$. We then let Δ_k vary as a uniform random variable within the interval $[\lambda_{min}, \bar{\delta} + \alpha]$, where $\alpha \in [0, \lambda_{max} + \delta_{max} - \bar{\delta}]$

Table 3.1: *Tenant Categories*

Tenant Category	Basic	Premium	Gold
Δ_k [ms]	[100, 150]	[50, 100]	[20, 50]
p_k	0	0	1
a_k	2γ	4γ	5γ
s_k	5γ		



(a) *Impact of delay on the consolidation capabilities.*

(b) *Management privileges impact on consolidation capabilities.*

Figure 3.4: *Simulation results.*

represents a tunable parameter aimed to increase the random interval up to the largest theoretical delay, $\lambda_{max} + \delta_{max}$. This setup provides heterogeneity in the incoming requests since both mean value and variance of Δ_k increase during the simulations. Fig. 3.4a shows the average MEC system utilization for increasing values of α. It can be noticed that with a Δ_k distribution close to λ_{min}, the overall capacity utilization is maximized, while it starts exponentially decreasing as the delay interval augments. With more stringent delay requirements, the possibility of finding a suitable host close enough to satisfy the incoming application delay request dramatically decreases. As a consequence, the same application must be deployed on multiple platforms without any consolidation opportunity. The scenario becomes less challenging as the delay requirement distribution spreads over a bigger interval. This further motivates the presence of heterogeneous tenant classes, as the stringent delay requirement deeply impacts on the general system consolidation capabilities.

Finally, Fig. 3.4b shows the impact of an increasing number of *gold*-type tenants on the system. The number of running application instances monotonically increases according to β, which represents the percentage of users belonging to the privileged category. The highlighted area between the curve of running instances and the curve of the requests is a measure of the M^{2}EC impact on the placement decisions with respect to the *legacy* straightforward solution of setting the instances only where demanded. Once again, M^{2}EC capabilities provide significant gains in terms of resource utilization savings, even with an increasing number of tenant requests. The number of running instances is minimized when $\beta = 0$ (all tenants belong to the *basic/premium* category) and maximized when $\beta = 100$ (all tenants belong to the *gold* category). During the admission and control phase, it must be taken into account that a majority of *gold*-type tenants limits the consolidation capability of the system and potentially interferes with the fulfillment of other agreed SLAs. Overall, our findings provide a guideline for the infrastructure providers: they might define ad-hoc solutions to prevent this issue, for instance, with different pricing labels to compensate the limited resources assignment flexibility.

3.2 RAN Slicing with Latency Control

The quest for new sources of revenue that revitalizes the mobile industry has spawned an unprecedented hype around the fifth-generation of mobile networks (5G) and, in particular, the network slicing concept. Enabled by software-defined networking (SDN) and network function virtualization (NFV), network slicing allows telco operators to offer virtualized slices of infrastructure resources *on-demand* to heterogeneous 3$^{\text{rd}}$-party services [98]. A high-level view of the system considered in this chapter is described in Fig. 3.5. The figure represents a series of *sliceable* base stations as a pool of radio resources (coloured cubes in the figure). The resource allocation process is considered hierarchical: while *bundles* of radio resources are assigned to different tenants (namely radio slices), each tenant autonomously schedules its bundle of radio resources to each individual user following classic radio scheduling policies. The difference between such operations is subtle but of paramount importance: a slice controller operates at a larger timescale and thus over a coarser granularity [5, 29]. While most prior work on network slicing focuses on average bit-rate guarantees [29, 99], *latency* considerations have received little attention. Latency aspects however are gaining more and more attraction as a quest to face advanced use-cases requirements, e.g., autonomous driving and platooning [100] in Vehicle-to-everything (V2X) enabled scenarios. In this context, accurate resource allocation schemes and inter-slice isolation aspects are fundamental features to support the provisioning of latency-constrained services.

Given the plethora of works on classic radio scheduling [101, 102], we keep this aspect out of the scope of this chapter and we focus instead on the former impelling need: a proper design of an orchestration solution that autonomously assigns chunks of radio spectrum (slices) in relatively larger time-scales pursuing the goal of *guaranteeing simultaneously latency and throughput constraints*. From the best of our knowledge, there is a non-negligible lack of works focusing on both aspects simultaneously in sliced-network environments.

To fill this gap, we design a LAtency-Controlled Orchestrator (LACO), a network slice controller that maps virtual radio resource allocations to physical resources while still guaranteeing latency requirements[3]. Specifically, LACO augments such prior work by accommodating resources to (granted) slices such that latency agreements are satisfied. This unlocks a new business opportunity for the telco operators that may apply customized pricing models according to the *elasticity* of offered slice latency constraints.

Technical challenges. While designing LACO, two sources of uncertainty need to be under control: i) the behavioral dynamics of the (aggregated) demand across involved tenants and ii) the inherent randomness of the wireless channel. These system dynamics have been traditionally modeled via either complex solutions that are hard to solve in re-

---

[3] Note that LACO does *not* compete with state-of-the-art throughput-only slice controllers—in fact, we purposely assume the presence of an admission controller that ensures that the aggregate load incurred by granted slices is within the system capacity region.

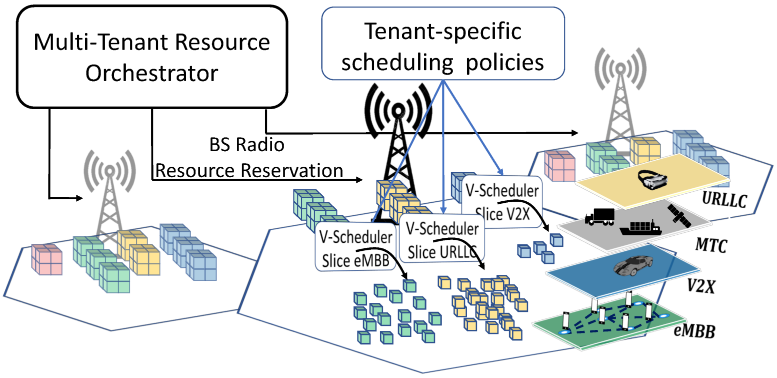

Multi-Tenant Resource Orchestrator
Tenant-specific scheduling policies
BS Radio Resource Reservation
V-Scheduler Slice V2X
V-Scheduler Slice URLLC
V-Scheduler Slice eMBB
URLLC
MTC
V2X
eMBB

Figure 3.5: *Illustration of the network slicing concept.*

alistic settings or via simplistic assumptions at the expense of low performance figures. In our work, we explore a novel approach by designing a scheme that *learns* the implications that allocation decisions have on per-slice latency without explicitly making assumptions on the underlying dynamics. To this aim, we first model our decision-making problem as a Markov Decision Process[4] (MDP), which allows us to neglect low-level details of the tenant demands and channel dynamics while letting us retain some knowledge on the consequences that a given action may have on the most immediate next system state.

An MDP model helps us to fully explore the problem features. However, the process of learning the state transition probability matrix of each of the embedded Markov chains incurs in prohibitive overhead as a reinforcement learning agent has to explore the whole space of state-action trajectories—the so-called *curse of dimensionality*. To address this, we resort to a Multi-Armed Bandit (MAB) model where the attained reward depends only on the action taken from a bounded set of possible actions. Importantly, in contrast to traditional MAB methods, LACO is *model-aware* (though not model-dependant), i.e., it exploits (abstracted) information regarding the underlying system to expedite the selection of highly rewarding actions, which is particularly attractive when dealing with dynamic non-stationary scenarios.

3.2.1 Related Work

The RAN design problem has always been at the forefront of the mobile operators and a vast amount of research has been devoted to novel RAN architectures [44, 103] and efficient radio resource schedulers [104, 105]. Recently, network slicing has been proposed as a new means for mobile operators to deploy isolated network services owned by different customers over a common physical infrastructure. However, as highlighted in [106], RAN needs additional functionalities to fully exploit SDN and NFV principles, specially in the partition and isolation of radio resources. The authors of [29] focus on efficient

---

[4] With a little misuse of nomenclature, we will refer to Markov Decision Process (MDP) rather than Semi-Markov Decision Process (SMDP) despite considering continuous time scales.

sharing of the RAN resources and proposed a RAN slicing solution that performs adaptive provisioning and isolation of radio slices. Their work is based on dynamic virtualization of base station resources, which gives to tenants the ability to independently manipulate each slice. Although the proposed architecture may guarantee isolation through different control planes, no mechanism is in place to ensure the satisfaction of delay requirements. [35] provides an empirical study of resource management efficiency in slicing-enabled networks through real data collected from an operational mobile network, considering different kinds of resources and including radio access, transport and core of the network. Similarly, the authors of [107] formulate an optimization framework to deal with resource partitioning problem, where inter-slice isolation is assured through a virtualized layer that decouples the reservation choice from the physical resource availability and proposing different abstraction types of radio resource sharing. In [108] the authors present an Earliest Deadline First (EDF) scheduling approach in the context of network slicing. Differently from us, their approach works on a single MAC scheduler and assumes for every TTI a complex fine-tuning of the quota of resources to be assigned to each slice, thus limiting the implementation of dedicated intra-scheduling solutions.

The exploration-vs-exploitation trade-off, typical of Multi-Armed Bandit (MAB) problems, is particularly suited to problems that require sequential decision-making. For this reason, a wide set of variations from the classical MAB model has been proposed in the literature [109, 110], together with novel algorithms to address them [111]. In this regard, the work of [112] investigates the MAB problems in case of Markovian reward distribution, where arms change their state in a two-state Markovian fashion. The authors addressed the problem assuming that the Markov chain evolves only when the arm is played, showing that the proposed sample mean-based index policy achieves regret performances comparable to the legacy UCB algorithm. The authors of [113] performed a complete regret analysis of the TS algorithm, generalizing the original formulation to distributions other than the Beta distribution. The MAB framework is also applied in [114] to deal with rate adaptation problem in 802.11-like wireless systems. The authors demonstrate that exploiting additional observations significantly improve the system performance. Similarly, [115] deals with scheduling transmissions in presence of unknown channel statistics. The proposed algorithm learns the channels' transmission rates while simultaneously exploiting previous observations to obtain higher throughput. This led to the design of a queue-length-based scheduling policy using the channel learning algorithm as a component in time-varying environment. The authors of [116] presented an algorithm for multivariate optimization on large decision spaces based on an innovative approach combining hill climbing optimization and Thompson sampling. While the scalability of their algorithm has been proven through exhaustive simulations, the framework lacks a complete analysis of regret bounds aimed at demonstrating the impact of hill climbing in combinatorial decision-making. Finally, similar to us, [117] deals with an MAB formulation where the reward distributions are characterized by temporal uncertainties. Interestingly, they were able to mathematically capture, in terms of reward, the added complexity embedded in the non-stationarity feature when compared to the legacy framework.

The key novelty of LACO relies on the exploitation of (abstract) information of the underlying system structure to expedite solutions. Conversely, prior works are blind to such type of information and need to spend substantial time exploring very bad decisions before achieving it.

3.2.2 LACO: The framework overview

Our solution relies on the concept of slicing-enabled networks wherein multiple network tenants are willing to obtain a network slice with predefined service level agreements (SLAs). Such SLAs may be expressed in terms of maximum slice throughput and average access latency. Within the context of this chapter, we define the average access latency as time the traffic belonging to a certain slice needs to wait before being served due to scheduling procedures. In particular, we focus on the radio access network (RAN) domain and design LACO, a RAN controller that dynamically provisions spectrum resources to admitted network slices while providing latency guarantees. In the following, we overview the main system building blocks with detailed notation and assumptions.

Business scenario

We consider different entities in our system: i) an *infrastructure provider* owning the physical infrastructure who offers isolated RAN slices as a service, ii) *tenants* who acquire and manage slices with given SLAs to deliver services to end-users, and iii) *end-users*, who demand radio resources from such tenants/slices.

Let us define $\mathcal{I}$ as the set of running network slices and $\mathcal{U}_i$ as the set of end-users associated to the i-th slice. The total amount of wireless resources (radio spectrum) is split into multiple non-overlapping network slices, each one belonging to one single tenant $i \in \mathcal{I}$.[5] Based on fixed SLAs, each network slice is characterized by *maximum throughput* and *expected latency* denoted by Λ_i and Δ_i, respectively. We assume that an admission control process[6] is concurrently running on a higher tier so that the average aggregate load can be accommodated within the overall system capacity.

Notation

We use conventional notation. We let $\mathbb{R}$ and $\mathbb{Z}$ denote the set of real and integer numbers. We use $\mathbb{R}_+$, $\mathbb{R}^n$, and $\mathbb{R}^{n\times m}$ to represent the sets of non-negative real numbers, n-dimensional real vectors, and $m \times n$ real matrices, respectively. Vectors are denoted as column vectors and written in bold font. Subscripts represent an element in a vector and superscripts elements in a sequence. For instance, $\langle \boldsymbol{x}^{(t)} \rangle$ is a sequence of vectors with

---

[5] We assume a one-to-one mapping between slices and tenants. Therefore, we use $i \in \mathcal{I}$ interchangeably throughout the chapter as a tenant identifier or its associated slice. Note that this assumption can be easily relaxed in the model.

[6] Given the plethora of solutions in the literature, the admission control design is out of the scope of this work. We refer the reader, for example, to [5, 99] for more details.

Table 3.2: *Notation table*

Notation	Description	Notation	Description	Notation	Description
y_i	Slice configuration	z_σ	Arm selection freq.	$\phi \in \Phi$	Action index
$n \in N$	Decision epoch index	$\mathcal{N}_i(\mu_i, \nu_i^2)$	Normal distribution	$\omega(\cdot)$	Latent var. weight
$u_i \in \mathcal{U}_i$	User index	$m \in \mathcal{M}$	MCS index	$R(\cdot)$	Reward function
$d \in \{0, 1\}$	Exceed delay flag	$i \in \mathcal{I}$	Slice index	$\psi(\sigma)$	Accuracy value
$\lambda_i^{(n)}$	Inst. traffic demand	$r_i^{(n)}$	Bits not served	L_i	Latency constraint
$f(x, \rho_i)$	Traffic demands distr.	$\bar{u}_i$	Aggregate user	Γ_m	Bits per subframe
$\zeta(\cdot)^{(n)}$	Throughput mapping	$g \in \mathcal{G}$	Channel level	$f(x, \theta_i)$	Channel distr.
$\sigma \in \Sigma$	MDP state index	$\gamma_i^{(n)}$	Inst. SNR	Δ_i	Latency tolerance
$w \in \mathcal{W}$	(Latent) Channel quality	C	Capacity of BS	τ	Rayleigh scale param.
$T(\cdot)$	Transition function	ϵ	Decision interval duration	Θ	PRB chunk

$\boldsymbol{x}^{(t)} = [x_1^{(t)}, \ldots, x_n^{(t)}]^T$ being a vector from $\mathbb{R}^n$, and $x_i^{(t)}$ the i'th component of the t'th vector in the sequence. Operation $[\cdot]^T$ represents the transpose operator while $[x_1, \ldots, x_n]_{\text{diag}}$ translates the vector into a diagonal matrix. Last, $\nVdash$ and $\nvdash$ indicate an all-ones and all-zeroes vector, respectively, and $\lceil \cdot \rceil$ is the ceiling operation.

Problem Definition

Assuming that an instance of LACO is executed per base station (BS) as shown in Fig. 3.5, we focus our problem design and performance evaluation on a single BS characterized by a capacity C, which is the sum of a discrete set of available physical resource blocks (PRBs) of fixed bandwidth. This resource availability must be divided into subsets of PRBs (i.e., slices), and our job is to dynamically assign such subsets to each network slice $i \in \mathcal{I}$. We refer to such assignment as the *configuration* of slice i, denoted by the variable y_i. Obviously, we shall guarantee $\sum_{i \in \mathcal{I}} y_i \leq C$. For the sake of clarity, we summarize all mathematical variables used throughout the chapter in Table 3.2.

We consider a time-slotted system where time is divided into *decision epochs* $n = \{1, 2, \ldots, N\}$. The decision epoch duration ϵ may be decided according to the infrastructure provider policies, ranging from few seconds up to several minutes. While the admission controller (pre-)selects a subset of slices that can co-exist without exceeding the capacity of the system *in average*, the dynamic nature of the slice's load and wireless channel may cause instantaneous load surges or channel quality fading effects and hence induce a non-zero mean delay. We denote the experienced instantaneous signal-to-noise ratio (SNR) of slice i (averaged out across all users of the slice) and the instantaneous aggregate traffic demand within time-slot n as $\gamma_i^{(n)}$ and $\lambda_i^{(n)}$, respectively. As each tenant i may show different behavior in terms of wireless channel evolution (according to θ_i) and traffic demands (according to ρ_i), we also assume $\gamma_i^{(n)}$ and $\lambda_i^{(n)}$ are drawn from different univariate probability density function, i.e., $\gamma_i^{(n)} \sim f(x, \theta_i)$ and $\lambda_i^{(n)} \sim f(x, \rho_i)$. Importantly, we do not assume any knowledge on such random variables; we exploit machine learning techniques to *learn* the inherent channel and demand models, which allow our system to dynamically adapt the slice configurations $y_i^{(n)}$ at every decision epoch n while mitigating latency constraint violations.

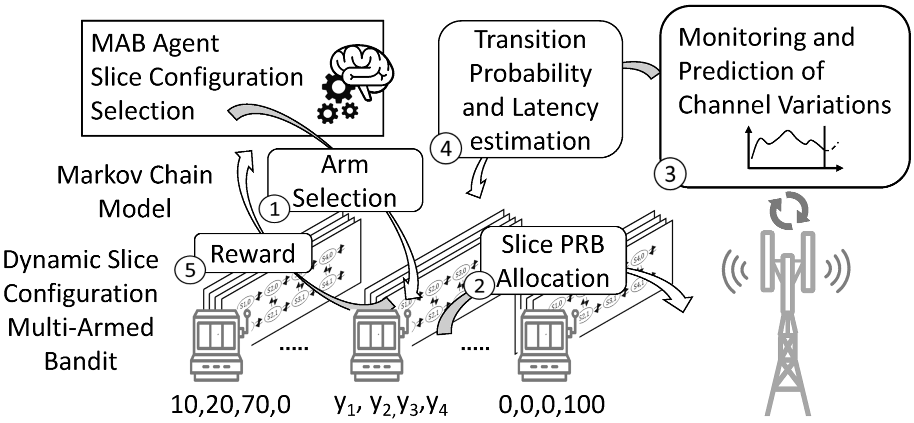


Figure 3.6: *Workflow illustration.*

Formally, the above-described problem becomes:

Problem `LATENCY-CONTROL`:

$$\begin{aligned}
\text{minimize} \quad & \lim_{N\to\infty} \sum_{n=1}^{N} \mathbb{E}\left[\sum_{i\in\mathcal{I}} r_i^{(n)}\right] \\
\text{subject to} \quad & \mathbb{E}\left[\frac{\lambda_i^{(n)}}{\zeta\left(y_i^{(n)},\gamma_i^{(n)}\right)+r_i^{(n)}}\right] \leq \Delta_i, \quad \forall i \in \mathcal{I}; \\
& \sum_i y_i^{(n)} \geq C, \quad \forall n; \\
& y_i^{(n)}, r_i^{(n)} \in \mathbb{Z}_+, \quad \forall i \in \mathcal{I};
\end{aligned}$$

where $\zeta(\cdot)^{(n)}$ is a mapping function that returns the number of bits that can be served using the allocated number of PRBs ($y_i^{(n)}$) and the current SNR level $\gamma_i^{(n)}$, as per [118, §7.1.7]. The traffic demand might not be satisfied within a single decision epoch incurring in packet queuing and additional delay. Therefore, in our formulation, we introduce $r_i^{(n)}$ as a deficit value indicating the number of bits not served within the agreed slice latency tolerance Δ_i during the time-slot n (i.e., dropped). The objective of Problem `LATENCY-CONTROL` is hence to find a sequence of $\langle y_i^{(n)} \rangle$ configurations such that the expected total non-served traffic demand is minimized. Hereafter whenever is evident from context, we drop the superscript (n) to reduce clutter. To address the problem, we rely on a two-layer scheduling approach commonly adopted in the network slicing context [29, 119]. On the one side, an inter-slice scheduler is in charge of defining the PRB allocation strategy to meet the networking requirements while ensuring resource isolation among slices. On the other side, a lower layer intra-slice scheduler enforces the assignment of the pre-allocated subset of PRBs to the connected end-users. Our work mainly focuses on the higher-level inter-slice scheduler, leaving the implementation of intra-slice scheduling strategies open to address tenant-specific requirements.

Working flow

For a given slot n, problem `LATENCY-CONTROL` can be easily linearized[7] and solved with standard optimization tools. However, this approach may exhibit suboptimal behavior in future epochs if the statistical distributions of $f(x, \theta_i)$ and $f(x, \rho_i)$ are not stationary. Hence, we propose a novel two-fold approach that: i) models channel and traffic demand variations based on previous observations, and ii) iteratively applies slice settings towards the goal of honouring SLAs.

Fig. 3.6 depicts the building blocks of our solution. LACO relies on the concept of Markov Decision Process (MDP) as described in Section 3.2.5 to decide which configuration y_i should be enforced to all active slices i, adapting its choice at every epoch n according to the observed *reward* function that measures the incurred latency. In turn, this information is asymptotically calculated within Discrete-Time Markov Chain (DTMC) model described in Section 3.2.3. The transition probabilities of such DTMC are updated according to previous observations in the *Monitoring and Prediction of Channel Variations* module, described in Section 3.2.4.

3.2.3 DTMC Model

Hereafter, we analyze the system dynamics through a Markov Chain-based (MC) model that computes expected channel conditions and violations on latency tolerance. It should be noted that channel variations and traffic demands are independently obtained according to each slice, thus each DTMC may be treated individually without the need to setup a Markov chain accounting for the overall system configuration. Such global DTMC could anyway be easily obtained as linear combination of the individual DTMCs. For the sake of tractability, we consider a single (virtual) user $\bar{u}_i$ with an aggregate traffic demand resulting from the set of users $u_i \in \mathcal{U}_i$ belonging to slice i.[8] We also assume a finite number of channel quality levels G, which may bound each instantaneous user channel quality γ_i, as depicted in Fig. 3.7. This is a system design choice and allows operators to trade off high accuracy for convergence speed, by ranging from a fine-grained scale (large G), e.g. by letting each channel quality level be equal to the modulation and coding scheme (MCSs) as defined in the 3GPP standard document [118], to a coarse-grained scale that may capture the channel variation behaviors with limited accuracy, as detailed in Section 3.2.4.

Let us consider a discrete-time stochastic process X_t[9] that takes values from a finite and discrete state space, which is denoted by $\mathcal{S} = \{S_{0,0}, \ldots, S_{g,d}, \ldots, S_{G,1} \mid 0 \leq g \leq G, d \in \{0,1\}\}$.[10] In particular, a realization of X_t when visiting state $S_{g,d}$ represents virtual user $\bar{u}_i$ experiencing channel level $g \in \mathcal{G}$ with an associated delay exceeding the

---

[7] Function $\zeta(\cdot)$ can be easily approximated with a linear function by applying piece-wise linearization.

[8] This assumption can be readily relaxed by considering the convolution of single cumulative distribution functions of every user channel and demand variation [120].

[9] The time scale t of DTMC state switch is much shorter than the decision epoch n used in the MDP described in Section 3.2.5.

[10] Each DTMC is defined within a state space $\mathcal{S}^i$. We remove the index i to limit the clutter, as the analysis can be easily extended to any other slice.

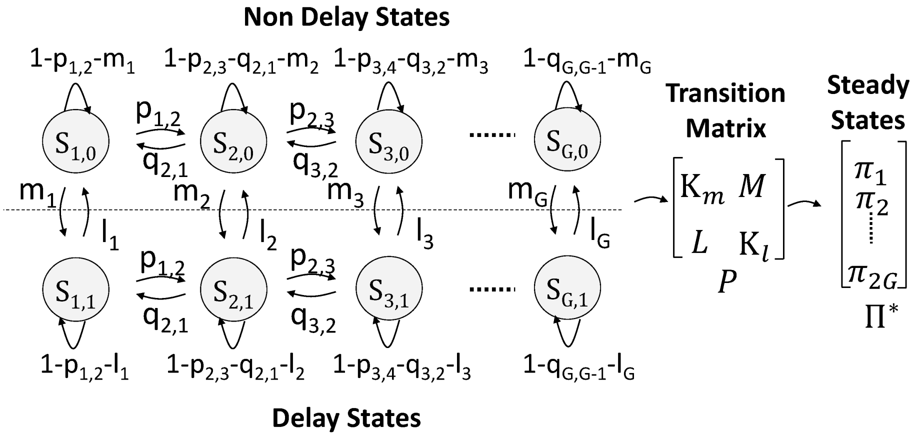


Figure 3.7: *Radio channel variations as Markov chain.*

one specified by the slice SLA ($d = 1$) or otherwise ($d = 0$). When considering wireless channel conditions as Rayleigh distributed, it is common practice to model the variations as a sequential visiting of consecutive states, as the channel does not vary faster than the Markov chain time-slot [121]. Hence, we define the probability to improve the user channel condition from level g to level $g + 1$ as $p_{g,g+1}$ whereas the probability to get a bad channel from level g to level $g - 1$ as $q_{g,g-1}$. As shown in recent works like [122] [123], accurate scheduling strategies might mitigate the interference effects coming from multiple base stations serving the same sets of slices thus improving the overall channel conditions. However, such schemes introduce additional complexity and synchronization overhead, which hardly fit with our view of a lightweight base station oriented solution. Last, given the available physical resource blocks assigned to a particular slice y_i, the channel quality level g and the overall traffic demand within the time-slot, we model the probability to incur in delay constraint violation as m_g and the probability to keep the access delay within the agreed bound as l_g. This process can be formulated as a two-dimensional DTMC $M := (\mathcal{S}, P)$, where P denotes the following transition probability:

$$P = \begin{vmatrix} K_m & M \\ L & K_l \end{vmatrix}, \text{where } K_{x=\{m,l\}} = \{k_{ij}^{(x)}\} \tag{3.7}$$

$$\text{with } k_{ij}^{(x)} = \begin{cases} 1 - p_{i,i+1} - q_{i,i-1} - x_i & \text{if } i = j, \\ q_{i,j} & \text{if } i = j+1, \\ p_{i,j} & \text{if } i = j-1, \\ 0 & \text{otherwise;} \end{cases}$$

$$\text{and } M = \{m_i\}_{\text{diag}}, L = \{l_i\}_{\text{diag}}.$$

Note that we assume $p_{G,G+1} = q_{1,0} = 0$ and each square block $K_{x=\{m,l\}}, M$ and L with $[G \times G]$ size so that the square matrix P has dimension $[2G \times 2G]$.

Without loss of generality, we assume that such transition probabilities do not depend on the particular time-slot we are evaluating. Thus, we define our DTMC as a time-homogeneous MC where the process X_t evolves based on $\Pi(t) = \Pi(0)P^t$ where the row vectors $\Pi(t)$ and $\Pi(0)$ represent the first order state probability distribution at time n and 0, respectively. In order to evaluate the long-term behavior of our system, we need to calculate the steady-state probability $\Pi^* = \{\pi_s^*\}$ of being in each of the defined states.

It yields that

$$\Pi^* = \lim_{n\to\infty} \Pi(t) = \Pi(0) \lim_{n\to\infty} P^t = \Pi(0)P^*. \tag{3.8}$$

The above-described Markov chain is irreducible, as each state may reach through available paths any other state. Therefore, by stochastic theory, if a Markov chain is irreducible and non-periodic, the steady-state probability distribution Π^* always exists, is unique and is independent of the initial conditions.

Recalling the total probability theorem and using Eq. (3.7), we calculate the steady-state probability distribution as the solution of the following equations

$$\begin{cases} (P^T - \nVdash_{\text{diag}})\,\Pi^* & = 0 \\ \nVdash \Pi^* & = 1 \end{cases} \tag{3.9}$$

where $\nVdash_{\text{diag}}$ is the identity matrix.

3.2.4 DTMC Monitoring and Prediction

The asymptotic behavior of a Markov chain depends on the transition probability matrix P, which in turn depends on the stochastic processes of the slice traffic demands and wireless channel variations. While several models have been already defined in the literature to derive such probabilities [124], the latency control objective and the need of an accurate estimation exacerbate the problem and render model-fitting approaches impractical. This brings additional complexity and delay the convergence process to the optimal solution.

We apply the concept of *unsupervised learning* to estimate the transition probabilities based on previous observations. In particular, we rely on the well-known theory of *probabilistic latent variable* [125]. Let us consider $w \in \mathcal{W}$ as the stochastic latent variable denoting the current channel quality level. Formally, we redefine the transition probability of the above-described DTMC as

$$\rho^g_{a,b} = Pr(X_t = S_{g,b} \mid X_{t-1} = S_{g,a}, g = w) \tag{3.10}$$

that is the probability to move from state $S_{g,a}$ to $S_{g,b}$ when the channel level is exactly $g = w$. To easily understand this, note that $\rho^g_{0,1} = m_g$, $\rho^g_{1,0} = l_g$ whereas $\rho^g_{0,0}$ and $\rho^g_{1,1}$ are the probabilities to stay within the same state $S_{g,0}$ and $S_{g,1}$, respectively. We use an expectation maximization technique to estimate such probabilities. To this aim, we enumerate the transitions between a and b upon g in $h^g_{a,b}$ based on the number of times X_t switches to another state (or stays within the same state) between t and $t+1$. We then derive the *a posteriori* probability as follows

$$Pr(g = w \mid X_t = S_{g,b}, X_{t-1} = S_{g,a}) = \frac{Pr(X_t = S_{g,b} \mid X_{t-1} = S_{g,a}, g = w)Pr(g = w)}{\sum\limits_{z\in\mathcal{W}} Pr(X_t = S_{g,b} \mid X_{t-1} = S_{g,a}, g = z)Pr(g = z)}, \tag{3.11}$$

and the likelihood probability as the following

$$Pr(X_t = S_{g,b} \mid X_{t-1} = S_{g,a}, g = w) = \frac{\sum\limits_{g\in\mathcal{G}} h^g_{a,b} Pr(g = w \mid X_t = S_{g,b}, X_{t-1} = S_{g,a})}{\sum\limits_{\{\alpha,\beta\}\in\{0,1\}^2} \sum\limits_{g\in\mathcal{G}} h^g_{\alpha,\beta} Pr(g = w | X_t = S_{g,\beta}, X_{t-1} = S_{g,\alpha})} \tag{3.12}$$

and

$$Pr(g = w) = \frac{\sum\limits_{\{\alpha,\beta\}\in\{0,1\}^2} h^g_{\alpha,\beta} Pr(g = w \mid X_t = S_{g,\beta}, X_{t-1} = S_{g,\alpha})}{\sum\limits_{\{\alpha,\beta\}\in\{0,1\}^2} h^g_{\alpha,\beta}} \tag{3.13}$$

The above system of equations can be solved using an iterative method that yields $\rho^g_{a,b}$. Finally, we calculate the weight of each latent variable based on a given set of previous observations as per the following equation

$$\omega(w \mid \hat{\mathcal{S}}_i) = \frac{\sum_{\{\alpha,\beta\}\in\hat{\mathcal{S}}_i} \rho^w_{\alpha,\beta}}{\sum_{g\in\mathcal{W}} \sum_{\{\alpha,\beta\}\in\hat{\mathcal{S}}_i} \rho^g_{\alpha,\beta}}, \tag{3.14}$$

where $\hat{\mathcal{S}}_i$ denotes the history of transitions (or lack thereof) across X_t among different states belonging to level 0 or 1 in the DTMC depicted in Fig. 3.7. We can generalize the probability to move from a state wherein the latency is under control $S_{g,0}$ to a state incurring unexpected latency $S_{g,1}$, i.e., exceeding the threshold defined in the slice SLA, using the following expression

$$\rho_{a,b} = Pr(X_{t+1} = S_b | X_t = S_a, \hat{\mathcal{S}}_i) = \sum_{w\in\mathcal{W}} \omega(w \,|\, \hat{\mathcal{S}}_i)\rho^w_{a,b}. \tag{3.15}$$

In the next section, we design a control-theory process by means of a Markov Decision Process (MDP) that optimally selects the best slice configuration y_i based on the probability to exceed the access latency constrained by the slice SLA.

3.2.5 Markov Decision Process

We model the decision problem as a Markov Decision Process (MDP) defined by the set of states $\Sigma = \{\sigma\}$, the set of actions $\Phi = \{\phi\}$, the transition function $T(\sigma, \phi, \sigma')$, and the reward function $R(\sigma, \phi)$. The set of states accounts for all the radio resource splitting options among different tenants, namely *slicing configuration* $c_\sigma = \{y_1, y_2, \dots, y_i, \dots, y_I\}$ expressed in terms of PRBs, where $\sum_{i\in\mathcal{I}} y_i = C$, i.e., the overall capacity is exactly split between running slices. We assume that each slicing configuration is issued at every decision epoch n. The transition function characterizes the dynamics of the system from state σ to state σ' through action ϕ. Analytically, $P(\sigma' \mid \sigma, \phi)$ is the probability to visit state σ' given the previous visited state σ and the action ϕ. Finally, the function $R(\sigma, \phi)$ measures the reward associated to the transition from the current state σ performing action ϕ.

We shall consider an MDP with an infinite time horizon. Future rewards will be discounted by a factor $0 < \chi < 1$ to ensure the total reward obtained is finite.

When dealing with MDPs is common practice to define a “policy” for the decision agent, namely a function $\mathcal{P}^{(n)} : \Sigma^{(n)} \to \Phi^{(n)}$ that specifies which action ϕ to perform at time n when in state σ. As soon as the Markov decision process is combined with a defined policy, this automatically fixes the next action for each state so that the resulting combination exactly behaves similarly to a Markov chain.

The final aim of the decision agent is to find the policy that maximizes the expected total reward, or, equivalently, to discover the policy $\mathcal{P}^*$ that maximizes the value function.

Reward Definition

Each state (or slicing configuration) is associated with a reward value that influences the agent during the decision process. The rationale behind is that we need to bind the action reward to the probability of exceeding the latency constraints defined in the slice SLA. In the following, we introduce the reward function used in our experiments with a detailed overview of its behavior.

Given a slicing configuration $c_\sigma = \{y_i \mid i \in \mathcal{I}\}$, we can analytically build a Discrete-Time Markov Chain, as described in Section 3.2.3. If the associated transition probability matrix P is perfectly known, we can also derive the steady-state probabilities $\Pi^* = \{\pi^*_s\}$ to be within any single state using Eq. (3.9). Thus, we can compute the probability to have the access latency of our system under control. This can be used to formulate the instantaneous reward value

$$R(\sigma^{(n)}, \phi^{(n)}) = \left(\sum_{s\in\mathcal{S}_g,0} \pi^*_s \right)^{\eta} \tag{3.16}$$

where s is the index of all states $S_{g,0}, \forall g \in \mathcal{G}$ such that the slice latency is under control, whereas $\eta \in [0, 1]$ is an adjustable value decided by the infrastructure provider to provide action fairness in the reward function when η tends to 0, or maximum likelihood of keeping latency under control when η tends to 1. Then our objective is to maximize the expected aggregate reward obtained as $\lim\limits_{N\to\infty}\sum\limits_{n=1}^{N}\mathbb{E}\left[\chi^n R\left(\sigma^{(n)},\phi^{(n)}\right)\right]$. However, given the fully-connected structure of our Markov Decision Process, i.e., all states are reachable from any MDP state, our objective is equivalent to maximize the instantaneous reward given by (3.16) at each decision epoch n.

Nonetheless, the assumption of perfect knowledge on the transition probability matrix P might be not realistic. Therefore, we need to rely on the transition probabilities $\rho_{a,b}$ inferred based on the previous observations, as explained in Section 3.2.4, Eq. (3.15). The larger the set of observations, the higher the accuracy of our probability estimation and the higher the reward attained to the instantaneous best action taken by the MDP.

Complexity analysis

Once we have fully characterized our proposed MDP, we can solve it by using dynamic programming solutions such as Value Iteration [126]. These approaches require exploring the entire state space of the MDP (several times) and the associated rewards.

Let us consider a scenario with I online slices running in our system. Assume that each slice configuration y_i can take values from integer multiples of a minimum PRB chunk size Θ and that the slicing configuration must be consistent, i.e., $\sum_{i\in\mathcal{I}} y_i = C$. Then, we can calculate the overall number of states equal to $\frac{(\frac{C}{\Theta}+I-1)!}{(I-1)!\frac{C}{\Theta}!}$. This poor state scalability, as well known as *the curse of dimensionality*, compromises the feasibility of MDP models under practical conditions. However, MDPs provide insights regarding the structure of the problem itself and are very helpful to design ausiliary solutions, such as Multi-Armed Bandit (MAB) models, which are better suited for functional deployments. Therefore, in the next section, we rely on a novel MAB design that exploits information from the underlying MDP to expedite the learning process while attaining near-optimal results.

Multi-armed Bandit problem

The online decision-making problem has been addressed in the past with several mathematical tools [109]. The limited information about real-time channel quality and effective traffic demand forces the operator to choose, like a gambler facing diverse options to play, the number of radio resources to assign to each running slice. This automatically falls in the fundamental *exploration-vs-exploitation* dilemma: the gambler needs to carefully balance the exploitation operations on known slicing configurations that provided the best revenues in the past against the exploration of new slicing configuration that might eventually produce higher revenues.

This class of decision process can be formulated as a Multi-Armed Bandit (MAB) problem, which emulates the action of selecting the best (single) bandit (or slot machine) that may return the best payoff. Each slot machine returns unpredictable revenues out of fixed statistical distribution, not known *a priori*, that is iteratively inferred by previous observations. This matches well the randomness of the channel quality and the traffic demand we aim to capture whereas each bandit can be mapped onto a state of the MDP, i.e., a specific slicing configuration. The final objective of such a problem is to maximize the overall gain after a finite number of rounds. This class of problems is usually assessed by a defined metric called *regret* Ω, which is defined as the difference between the reward that can be gained by an optimal oracle, i.e., using an optimal policy that knows the reward distributions *a priori*, and the expected reward of the myopic online policy.

Reusing notation from our MDP model, let us define each arm $\sigma \in \Sigma$ as a different slicing configuration $c_\sigma = \{y_i \mid i \in \mathcal{I}\}$. Once selected, each arm provides an instantaneous reward $R(\sigma)$ defined as the following

$$R(\sigma) = \sum_{i \in \mathcal{I}} \left(\zeta(y_i, \gamma_i) - \frac{\lambda_i}{\Delta_i} \right) \tag{3.17}$$

where the slicing configuration is $y_i \in c_\sigma$, $\zeta(\cdot)$ computes the number of bits that can be served using y_i configuration and given the current channel quality γ_i, and λ_i is the slice traffic demand, as described in Section 3.2.2.

While using such reward function requires low overhead, as it only needs to calculate the incurred latency after selecting a slicing configuration, it only converges to a near-optimal solution after exploring several configurations, which results in overly long training periods (as shown in Section 3.2.6). This is an inherent issue with classic MAB methods, which are *blind* to the underlying system structure. Conversely, in this chapter we resort to a novel model-assisted approach that exploits the system model of Section 3.2.5 to guide the exploration/exploitation process with (abstract) system information. In this way, as opposed to using the traditional reward model of Eq. (3.17), we define our bandit's reward as the expectation of access latency exceeding slice SLA defined in Eq. (3.16). This has a two-fold advantage: i) during the initial training period, the DTMC associated to each state of the MDP is updated (and enhanced) with more accurate values of the transition probabilities: this helps to find steady-state probabilities (and in turn an updated reward per slicing configuration) that reflect the real behavior of our system as time goes on; and ii) the slicing configuration selection accounts directly for stochastic behaviors of both channel quality and traffic demand, while reducing the state space to those that may benefit the entire system. Many algorithms have been proposed to optimally solve the MAB while learning from previous observations [127]. One of the main issues is that collecting rewards on a short-time basis may negatively impact on the decision of the best bandit. Thus, we rely on a modified version of the so-called *Upper Confidence Bound* (UCB) algorithm devised by [128] that overcomes this issue by measuring not only the rewards collected up to the current time interval, but also the *confidence* in the reward distribution estimations by keeping track of how many times each bandit has been selected $z_{\sigma,n}$. The pseudo-code is listed in Algorithm 5.

Initially, we explore all bandits, i.e., slicing configuration $\sigma \in \Sigma$, to get a consistent reward (line 2-6). Then we select the best configuration that maximizes the empirical distribution $\hat{\rho}_\sigma$ accounting for a confidence value. This confidence value depends on the number of times we have explored that particular configuration as well as the accuracy

Algorithm 5 LACO

```
1.  Input: $\Sigma, N, \Psi = \{\psi(\sigma)\}, \mathcal{I}, \omega, \epsilon, \mathcal{S}$
2.  Initialization: $z_\sigma, \hat{\rho}_\sigma = 0,\ \forall \sigma$
3.  Procedure:
4.  for all $n \in N$ do
5.      if $n = 0$ then
6.          for all $\sigma \in \Sigma$ do
7.              GET reward: $\hat{\rho}_\sigma = R_\sigma^{(n)}$
8.              $z_\sigma = z_\sigma + 1$
9.          end for
10.     else
11.         $\sigma^* = \arg\max_{\sigma \in \Sigma} \hat{\rho}_\sigma + \psi(\sigma)\sqrt{\frac{2 \log \sum_k z_k}{z_\sigma}}$
12.         UPDATE $\hat{\rho}_{\sigma^*} \leftarrow R_{\sigma^*}^{(n)}$
13.         $z_\sigma = z_\sigma + 1$
14.     end if
15.     for all TTIs $\in \epsilon$ do
16.         for all $i \in \mathcal{I}$ do
17.             UPDATE $\omega(w \mid \hat{\mathcal{S}}_i) \leftarrow \mathcal{S}_i$
18.         end for
19.     end for
20.     UPDATE $\psi(\sigma^*) \leftarrow \omega(\cdot)$
21. end for
22. End Procedure
```

of the transition probabilities we calculate for the associated DTMC. Note that this is different to traditional UCB algorithms. Specifically, we define a Markov accuracy value $\psi(\sigma) = (\frac{(\sum_w \omega(w|\hat{\mathcal{S}}_i))^2}{W \sum_w \omega(w|\hat{\mathcal{S}}_i)^2})$, where W represents the cardinality of the set $\mathcal{W}$. Note that $\psi(\sigma)$ depends on the weights $\omega(\cdot)$ obtained through the performed observations $\hat{\mathcal{S}}_i$, as reported in Eq. (3.14). Interestingly, $\psi(\sigma) \in (0, 1]$, i.e., when the DTMC has no relevant observations to build its transition probabilities this function returns $\psi(\sigma) = 1$ whereas, when a relevant number of observations allow to determine accurate transition probabilities, its value tends to 0. The value of $\psi(\sigma)$ is updated at the end of every decision interval (line 20) after monitoring the effects of the last decision on the Markov Latent variable distribution (lines $15 - 19$).

Regret analysis

Here, we mathematically calculate the bounds of our solution, LACO, for multi-armed bandit problems. Let us consider a player selecting an arm $\sigma \in \Sigma$ every decision epoch n. Every time arm σ is pulled down, it returns a reward $R_\sigma^{(n)}$ drawn from an unknown distribution with mean $\bar{\rho}_\sigma$ and empirical mean value calculated until time n as $\hat{\rho}_\sigma^{(n)} = \frac{\sum_{s=1}^{n} R_\sigma^{(s)}}{n}$. We denote σ^* as the arm providing the maximum average reward such that $\bar{\rho}_{\sigma^*} > \bar{\rho}_\sigma, \forall \sigma \neq \sigma^*$. If the arm selection is performed using LACO, it yields that the regret is obtained as

$$\begin{aligned}\Omega_N^{\text{LACO}}(\Sigma) &= N\bar{\rho}_{\sigma^*} - \mathbb{E}[\sum_{n=1}^{N} R_\sigma^{(n)} \mid \sigma \in \mathcal{P}^{\text{LACO}}] \\ &= N\bar{\rho}_{\sigma^*} - \sum_{\sigma=1}^{\Sigma} \bar{\rho}_\sigma \mathbb{E}[z_\sigma^{(n)}]; \end{aligned} \tag{3.18}$$

where $\mathcal{P}^{\text{LACO}} = \{\sigma_n\}$ is the policy as defined in Section 3.2.5 that consists of a set of moves that LACO will play at time n whereas $z_{\sigma,n}$ is the overall number of decision epochs arm σ has been pulled down till time instant n. Now consider LACO as a uniformly good policy, i.e., any suboptimal arm $\sigma \neq \sigma^*$ is chosen by our policy up to round n so that $\mathbb{E}[z_{\sigma,n}] = o(n^\alpha), \forall \alpha > 0$. It holds that

$$\lim_{\text{N}\to\infty} \sum_{\sigma=1}^{\Sigma} N^{-1} \bar{\rho_\sigma} \mathbb{E}[z_\sigma^{(N)}] = \Sigma \bar{\rho}_{\sigma^*}. \tag{3.19}$$

Hence, we can express the regret lower bound as the following

$$\lim_{\text{N}\to\infty} \inf \frac{\Omega_N^{\text{LACO}}(\Sigma)}{\log N} \geq \sum_{\sigma:\ \bar{\rho}_\sigma < \bar{\rho}_{\sigma^*}} \frac{\bar{\rho}_{\sigma^*} - \bar{\rho}_\sigma}{Div(\bar{\rho}_\sigma, \bar{\rho}_{\sigma^*})} \tag{3.20}$$

where $Div(\bar{\rho}_\sigma, \bar{\rho}_{\sigma^*})$ is the Kullback-Leibler divergence of one statistical distribution against the other and it is used to measure how one distribution might diverge from another probability distribution.

Now consider the Hoeffding's inequality for multiple i.i.d. variables x_n with mean μ. It yields that $Pr(|\frac{\sum_{i=1}^{n} x_i}{n} - \mu| \geq \delta) \leq 2e^{-2n\delta^2}$. Our algorithm LACO applies an upper confidence interval $\delta = \sqrt{\frac{2\log \sigma_k z_k}{z_\sigma}}$. Therefore, it yields that

$$Pr\left(|\hat{\rho}_{\sigma,n} - \bar{\rho}_\sigma| < \sqrt{\frac{2\log \sum_k z_k}{z_\sigma}}\right) \geq 1 - \frac{2}{n^4} \tag{3.21}$$

and also that

$$Pr\left(\mathcal{P}^{(n+1)} = \sigma \mid z_\sigma^{(n)} > \frac{4\log n}{\bar{\rho}_{\sigma^*} - \bar{\rho}_\sigma}\right) \leq \frac{4}{n^4}. \tag{3.22}$$

We can then derive the expectation of number of times suboptimal arm $\sigma \neq \sigma^*$ is pulled down as follows

$$\mathbb{E}[z_\sigma^{(N)}] \leq \frac{4\log N}{\bar{\rho}_{\sigma^*} - \bar{\rho}_\sigma} + 8 \tag{3.23}$$

and the regret upper bound as the following

$$\mathbb{E}\left[\Omega_N^{\text{LACO}}(\Sigma)\right] \leq \sum_{\sigma:\ \bar{\rho}_\sigma < \bar{\rho}_{\sigma^*}} \frac{4\log N}{\bar{\rho}_{\sigma^*} - \bar{\rho}_\sigma} + 8\left(\bar{\rho}_{\sigma^*} - \bar{\rho}_\sigma\right). \tag{3.24}$$

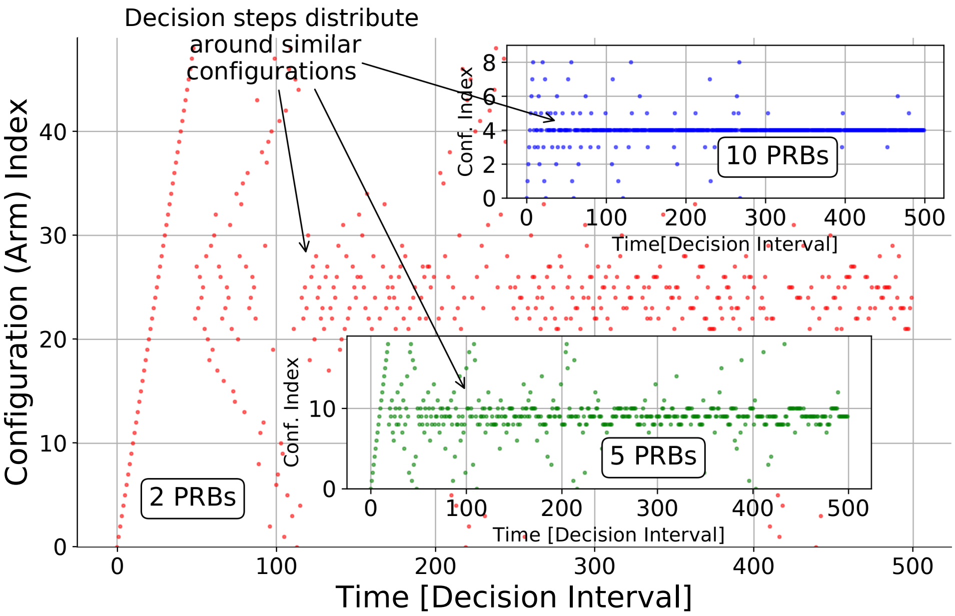


Figure 3.8: *Impact of different resource allocation chunk sizes.*

3.2.6 Performance Evaluation

In this section, we evaluate our solution through an exhaustive simulation campaign that takes into account complexity, revenue and SLA violation metrics.

Simulations setup

To assess heterogeneous slices, we simulate the network load demand of slice i at each time-slot (i.e., each transmission time interval (TTI) in Long Term Evolution (LTE) systems) by extracting a random value from a Normal distribution $\mathcal{N}_i(\mu_i, \nu_i^2)$, where μ_i and ν_i represent the mean value and standard deviation, and let L_i describe its latency constraint. Moreover, we model the SNR channel variation as another random variable drawn by a Rayleigh distribution and derive the probability distribution encompassing the whole SNR range.

For every channel instantiation, we extract the corresponding Modulation and Coding Scheme (MCS) as defined by the 3GPP standard.[11] The MCS index $m \in \mathcal{M}$ combines one possible modulation scheme and a predefined coding rate providing a compact way to represent a simple concept: the better the radio conditions, the more bits can be transmitted per time unit, and *vice versa*. Fixing the channel bandwidth, the expected average throughput achievable by one slice during one epoch depends on both the modulation and coding schemes used and, most importantly, on the number of PRBs reserved for the slice. In a

---

[11] We refer the reader to [118] for an exhaustive explanation of the mapping between SNR and MCS.

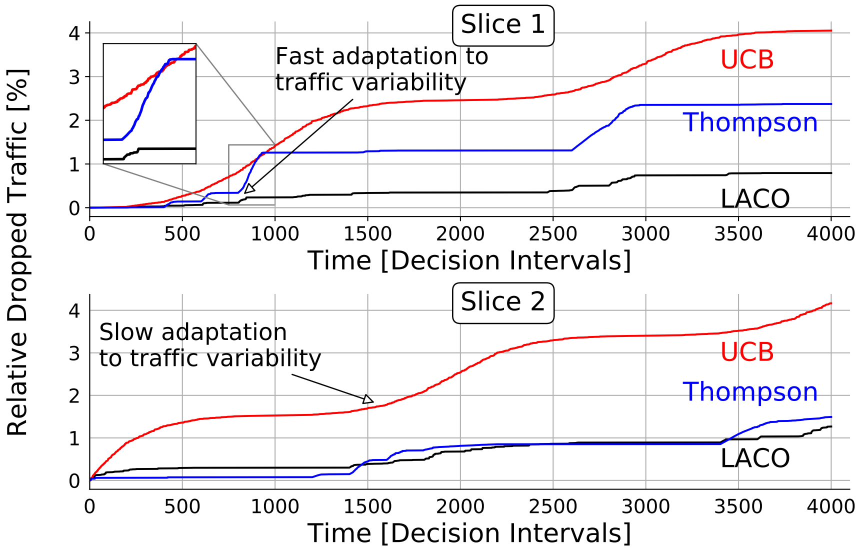


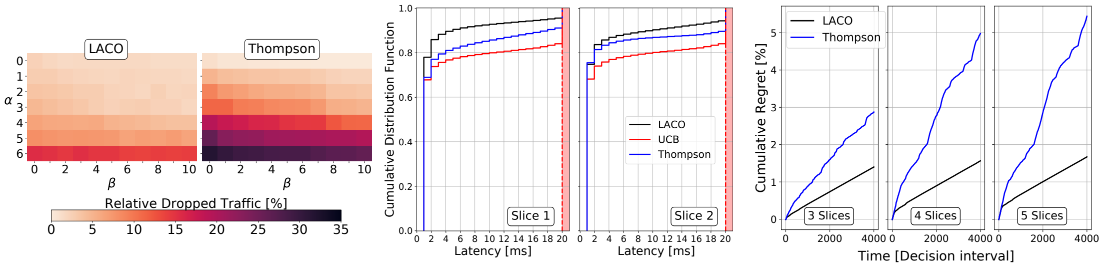
Figure 3.9: *Cumulative dropped traffic due to latency constraints violations.*

wider timescale [12], the average capacity can be approximated as $C_i = \left(\sum_m^M \Gamma_m \pi_{m,i}\right) T_i y_i$ where Γ_m represents the average number of bits per LTE subframe that can be transmitted using the m-th MCS index, $\pi_{m,i}$ is the steady-state probability distribution output of the first stage Markov chain model, T_i defines the decision interval size, and y_i accounts for the number of PRBs allocated to the i-th slice. We refer the reader to Table 3.2. In the LTE radio interface, the maximum amount of PRBs is fixed to 100 when operating at conventional bandwidth values of 20 MHz. In order to support massive type communication and Ultra-Reliable Low-Latency Communication (URLLC) use-cases, the 5G New Radio (NR) introduces significant enhancements in the radio frame composition. Not only 5G NR will support wider channel bandwidth (up to 100 MHz), but also introduce the support for multiple different types of subcarrier spacing. For back-compatibility reasons, even in 5G NR the time duration of radio frames and subframes are fixed to 10 ms and 1 ms, respectively[129]. The number of slots within each subframe however would change according to the subcarrier configuration, which eventually translates in shorter PRB time duration and thus a different PRB availability depending on the selected configuration. It must be noticed that all the subcarrier spacing are defined as $\Delta f = 2^j \cdot 15$ KHz, $j = \{0, \ldots, 4\}$, thus leading at the definition of time-frequency grids containing an amount of PRBs which is multiple of those contained in the traditional LTE grids. In this context, we assume a simple mapping function, as the one described in [105], implemented at intra-slice scheduler to homogenize the resources of potentially heterogeneous radio access technologies.

Traffic demands are compared with the current channel availability to derive the pos-

---

[12] Note that we assume a timescale larger than our epochs used in the decision-making process.



(a) *Effects of different slice reqs.* **(b)** *CDF of experienced latency.* **(c)** *Empirical cumulative regret.*

Figure 3.10: *(a) Effects of different slice requirements; (b) CDF of latency experienced by served traffic; (c) Empirical cumulative regret for a variable number of slices.*

sibilities to pass from one state to another. It must be noticed that the accuracy of the resulting steady-state distribution strictly depends on the precision of such comparison. For this reason, we constantly monitor and update the transition probabilities of the Markov chain based on the resource allocation adopted in the current decision interval. During the arm selection, if the chosen configuration does not provide enough resources to meet the latency requirements, the steady-states will be mostly distributed in the lower part of the Markov chain leading to a minor reward that, in turn, guides the MAB agent to take a different action (i.e., selecting a different arm) in the following decision round.

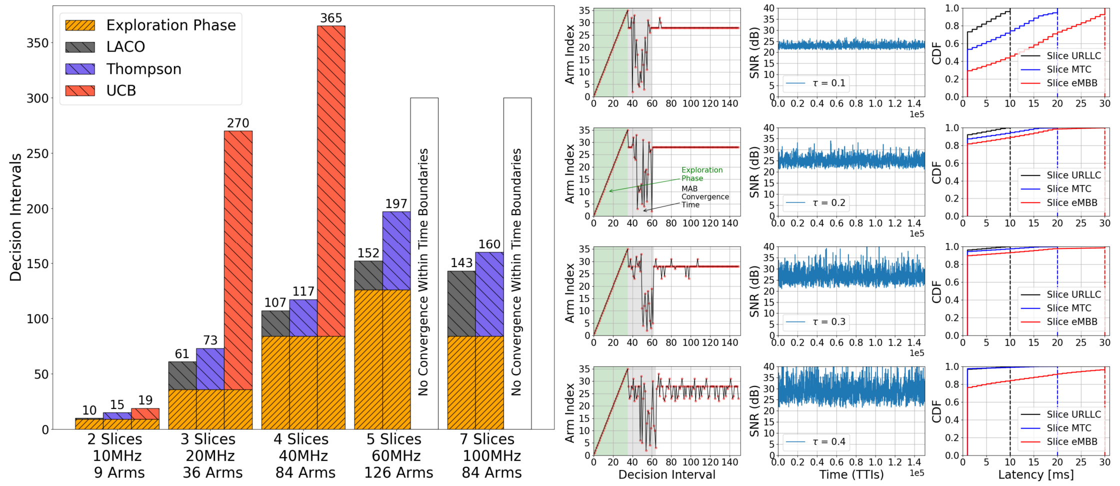
For benchmarking purposes, we implement two widely used MAB algorithms, namely “legacy” UCB and Thompson Sampling (TS)[13]. On the one hand, UCB adopts a deterministic approach to deal with the exploration-vs-exploitation dilemma, but its performance generally degrades as the number of arms increases. On the other hand, Thompson sampling adopts a probabilistic approach that scales better with the number of arms, but it may provide suboptimal results when the distribution of reward changes over time (i.e., in non-stationary scenarios). Conversely, LACO combines the advantages of them both by adopting a probabilistic model (MDP) guiding an exploration phase derived from UCB.

Multi-armed bandit problem behavior

We first explore the trade-off between action space (and its granularity) and the associated reward loss. To this aim, we set up a simple experiment with 2 slices with equal SLA requirements in a deterministic and static environment. We then apply LACO using 3 different action sets: $\{0, 2, 4, \ldots, 100\}$, $\{0, 5, 10, \ldots, 100\}$ and $\{0, 10, 20, \ldots, 100\}$ PRBs (with 50, 20 and 10 available configurations each), labelled “2 PRBs“, “5 PRBs“ and “10 PRBs“, respectively. The results, shown in Fig. 3.8 make it evident that the higher the granularity the longer the exploration phase(s): over 50 intervals for “2 PRBs“ whereas it takes around 10 intervals for “10 PRBs“. Interestingly, the loss in reward attained to the

---

[13] Due to space limits, we refer the reader to the literature introducing such algorithms, e.g. [113].



(a) *Effects of different number of slices and bandwidth (PRB) availability.*

(b) *Effects of increasing variability in the channel conditions (SNR).*

Figure 3.11: *Sensitivity analysis of bandwidth availability and SNR variability on the convergence time to the optimal slice resource allocation.*

latter configuration is only 2%. Therefore, due to a faster convergence time at the expense of minimal reward loss, we empirically select $\Theta = 10$ PRBs for our purposes.

Slice SLA violation analysis

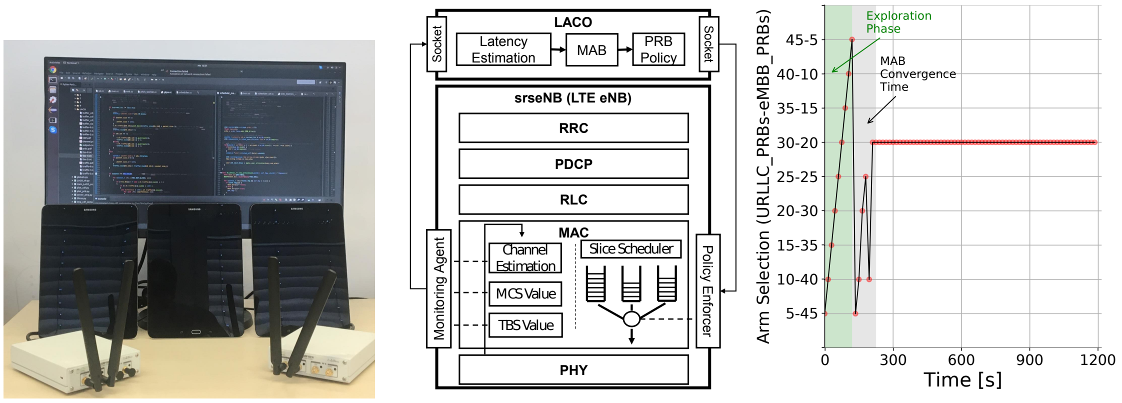
We thus grant spectrum-time resources in the granularity of chunks of 1 second $\times$ 10 PRBs. In the first scenario, we investigate the capacity of LACO to adapt the resource allocation at variable traffic loads. For this reason, we consider only two slices with equal requirements, i.e., $\nu_i^2 = 10$ Mb/s and $\Delta_i = 20$ ms for $i = 1, 2$. To assess real scenarios with non-stationary traffic patterns, we vary the mean load of each slice i following a sinusoidal curve in counter-phase between $\mu_i = 8$ Mb/s and $\mu_i = 40$ Mb/s. This forces the resource allocation process to span across the whole configuration set when dealing with SLAs guarantees. As shown in Fig. 3.9, the cumulative dropped traffic of each slice changes when different MAB algorithms are used. The behaviour of UCB shows high variability after few decision intervals. As soon as all the arms are selected, the agent starts learning about the statistics of the outcomes and builds a ranked list. The need for a comprehensive knowledge of all the arms leads to several ”bad” choices during the exploration phase. This slows down the convergence to the optimal configuration and penalizes performance. From the obtained rewards, TS builds a bivariate probability distribution across the expected reward of each arm, extracts a random sample and chooses the arm associated to the maximum value. This approach performs well in static scenarios as TS favours exploitation of the empirical results obtained in the first attempts; but in time-varying scenario as the one we are considering, the reward distribution associated to each arm fluctuates over time rendering TS unable to adapt fast enough in highly-dynamic scenarios. In contrast, the LACO's model-awareness allows for quicker convergence and so it accommodates real-time traffic requirements in dynamic environments and as a result reduces the amount of data violating delay deadlines.

Obviously, heterogeneous throughput/latency requirements impact the system differently. Fig. 3.10a shows the effect of such variations on the system extending the previous scenario and considering increasing values of resource requirements as $10 \cdot \alpha$ Mb/s, and $10 \cdot \beta$ ms, respectively. As expected, smoother delay requirements (horizontal direction in the figure) allow to serve more traffic within the latency bounds defined by the SLA, although the impact becomes negligible after few incremental steps. This is due to long decision intervals when compared to the timescale of fast channel variations. A proper resource configuration selection allows matching the offered traffic requirements with the expected channel capacity, allowing the incoming traffic to be served within few milliseconds. As the offered traffic approaches the channel capacity boundary (vertical direction in the figure), the same task becomes more challenging and the admission and control process should consider this aspect when granting/rejecting access to new network slices. LACO 's abilities to adapt to demand variations not only mitigates the amount of traffic violating delay requirements but also improves the distribution of data delivery delay overall. As shown by Fig. 3.10b, the empirical CDF of delay for each slice in the same scenario presented above remarkably improved with a mean delay equal to 2.6, 3.9 and 4.9 ms for LACO, TS and UCB, respectively.

Finally, we implement an optimal offline policy with full knowledge of the system, i.e., an oracle policy that knows the future with the corresponding latency violations. We compare both LACO and TS to this optimal policy for a variable number of slices. The aggregated demand is adapted to ensure we operate within the system capacity. In Fig. 3.10c, we depict the temporal evolution of the cumulative reward loss over time (regret) for both approaches. The figure illustrates how the regret increases with time much rapidly for TS, a difference that increases with the number of slices.

Convergence time

The next generation of mobile networks (5G) promises to support the provisioning of high throughput and low-latency services even in highly dense scenarios [5]. These capabilities are tightly bounded with the possibility to exploit higher communication frequencies together with wider spectrum bandwidth. In the 5G context, bandwidth is expected to increase up to 100MHz, leading to additional complexity in the management of radio resources. In order to assess LACO performances in such scenarios, we investigate the convergence time of our solution to the optimal slice configuration in different bandwidth settings. To enable more efficient use of the spectrum resources and reduce the power consumption at UE side, 5G New Radio (NR) introduces the concept of bandwidth parts (BWP) [129], where each BWP can be configured by different numerologies defining specific signal characteristic, e.g., in terms of subcarrier spacing. Without loss of generality, we assume all the end-users belonging to the same slice operating under similar numerol-



(a) *Experimental setup.* **(b)** *Architecture overview.* **(c)** *Arm selection.*

Figure 3.12: *Experimental setup (a); Architecture overview (b); Arm selection over time (c).*

ogy settings. Moreover, we keep the subcarrier spacing fixed to $\Delta f = 15$ KHz as in legacy LTE systems. Such coarse resource allocation scheme is mandatory to support LTE devices but, it can be easily mapped to finer resource block structures as defined within the 5G domain at lower layer intra-slice schedulers [105].

Fig. 3.11a compares the convergence time of different MAB algorithms for an increasing number of slices and bandwidth availability over a time period of $N = 1000$ decision intervals. It should be noticed that depending on channel statistics and real-time slice requirements, *multiple* resource allocation settings (namely arms) may provide *optimal performance* making unfeasible a single convergence point. Thus, we opted to simulate the worst-case scenario allowing for a unique optimal resource configuration in each simulation run. In line with previous observations, we fix $\Theta = 0.1\,C$. Despite a common initial exploration phase (highlighted in orange), from the picture it is evident how the curse of dimensionality affects the overall convergence time. This is more evident for the legacy UCB approach (depicted in red), which hardly copes with the increasing size of the action space and in some runs did not converge to a solution within the time boundary of our experiment. Focusing on LACO performances (depicted in black), the number of decision intervals necessary to converge to the optimal resource allocation outperforms Thompson Sampling (in blue) by scaling almost linearly with the number of slices (and PRB availability) after the initial exploration phase.

Convergence to the optimal slice configuration also shows its dependency on the radio channel statistics. To measure the sensitivity of the decision process at the SNR fluctuations, Fig. 3.11b considers a fixed number of slices (i.e., 3) deployed in a system characterized by average channel statistics with an increasing variance. In every scenario, the average (per slice) channel realization is derived from a Rayleigh distribution characterized by a scale parameter $\tau = \{0.1, 0.2, 0.3, 0.4\}$, respectively. This introduces an increasing level of variability in the SNR distribution according to the formula $\mathit{Var} = \frac{4-\pi}{2}\tau^2$, as depicted in the plots of the central column. On the left-hand side of the same picture, it can be noticed how higher SNR variability has very limited impact on the decision steps. This feature is inherited by the Markov Chain model described in Section 3.2.3. In particular, provided that the slice requirements fit within the admissibility region of the system, a higher SNR variability will simply map into a wider excursion over the Markov chain steady states without affecting the final reward of the same arm.

Finally, on the right-hand side of the picture, we depict the empirical CDFs of the overall latency occurred per slice. In (almost) static channel conditions, slices' latency distribution suffer from having poor channel conditions, which are barely sufficient to support requested data volumes. In this context, slices with less stringent delay requirements, namely the MTC and eMBB, are lightly penalized to meet the expected latency threshold w.r.t. the URLLC one. When increasing the channel variability, the average channel conditions improve easing the allocation resource task thus favouring the satisfaction of overall latency requirements.

3.2.7 Experimental Proof Of Concept

In order to illustrate, validate and analyze the performance of our LACO solution, we developed it as a standalone software module running on top of an open source platform that implements the LTE protocol stack, namely `srsLTE` [130], attached to a USRP[14] Software-Defined Radio (SDR) device as radio front-end. Our testbed is depicted in Fig. 3.12a and consists of one LTE eNB (a modification of `srseNB`) and commercial Android tablets[15] as UEs. Any single UE emulates the aggregated traffic of multiple UEs within one slice. We use `mgen`[16] to generate different downlink traffic patterns. Due to our LTE spectrum testing license restrictions, we use 10 MHz bandwidth in LTE band 7 and use SISO configuration for simplicity. This renders a maximum capacity of ~ 36 Mb/s with highest SNR. Finally, in accordance with the findings described in Section 3.2.6, we set the minimum PRB allocation value at 10% of the overall availability.

Implementation

The architecture of our software implementation and LACO's interfaces with `srseNB` are depicted in Fig. 3.12b. LACO interacts with the eNB's Medium Access Control (MAC) layer to implement two key features:

- **Monitoring agent**. This feeds LACO with real-time SNR reports generated by the physical (PHY) layer from feedback received from the UEs, the selected MCSs and corresponding transport block size (TBS) value used to encode information at the MAC layer, and other traffic statistics such as packet size and arrival times;

---

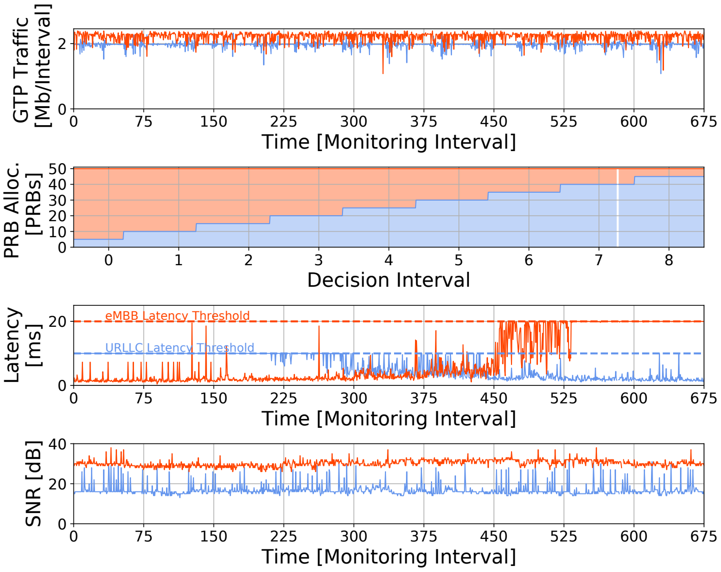
[14] USRP B210 from National Instruments/Ettus Research (`https://www.ettus.com/all-products/UB210-KIT/`).

[15] Samsung Galaxy Tab S2 (`https://www.samsung.com/de/tablets/galaxy-tab-s2-9-7-t813/SM-T813NZKEDBT/`).

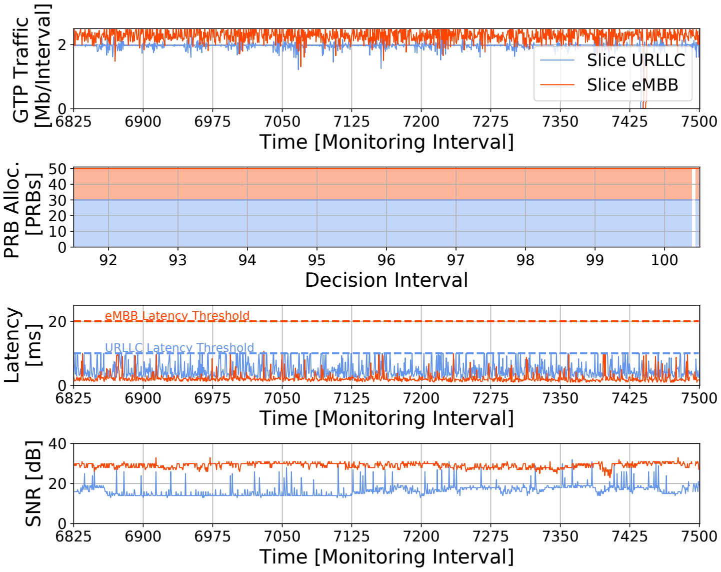
[16] mgen (`https://www.nrl.navy.mil/itd/ncs/products/mgen`).

- **Policy Enforcer**. This allows LACO to dynamically enforce the PRB allocation policies calculated by our MAB model, as described in Section 3.2.5.

The main feature of our implementation is the possibility to collect, with TTI granularity, the traffic arrival rate and the TBS values to be used in each transmission frame. This information, together with the scheduling buffer size and data arrival times, is essential to compute the latency experienced by the different slices running in the system.



(a) *System dynamics during MAB discovery phase.*



(b) *System dynamics at convergence.*

Figure 3.13: *Comparison of system dynamics during a) discovery phase and b) MAB convergence.*

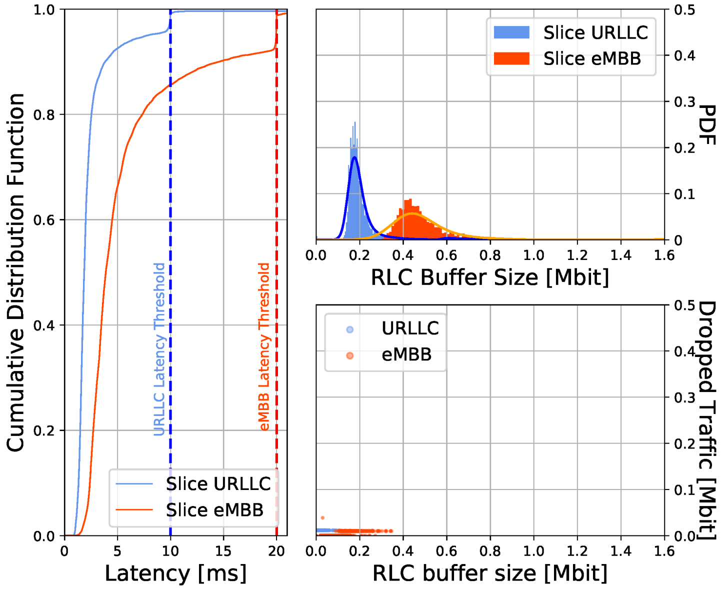
The different metrics are collected in a time series database, namely `InfluxDB`, and periodically reported to LACO which constructs a virtual queue (one per slice) to track the dynamics of packets arriving at the eNB, from their entrance into the scheduling buffer to their transmission. This approach is particularly useful as Internet Protocol (IP) packets are multiplexed while advancing the transmission path in the eNB, complicating the computation of slice latencies by external modules. Our approach aims to characterize the PRB allocation policy currently enforced into the system. In case of constant traffic and low latency requirements for example, poor channel conditions will result in lower TBS values and a sudden increase of the virtual queues size. Such event directly maps into an additional delay suffered by IP packets at the Radio Link Control (RLC) layer. Note that higher packet rates also lead to larger waiting times, which might result in exceeding slices SLAs boundaries. In such cases, the violation of pre-defined SLA latency boundaries triggers the DTMC model described in Section 3.2.3 to a *delay state* and the selected PRB allocation policy is assigned with a lower reward value. Conversely, in a stable system where serving rate and packet arrival rate are balanced, the size of the virtual queues get smaller and the DTMC model is mostly characterized by *non-delay states*.

Experimental results

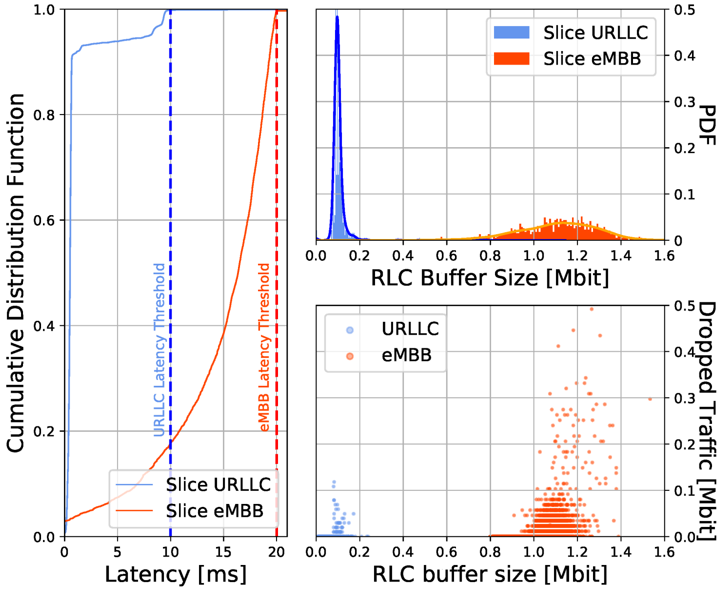
We consider a scenario accounting for two slices characterized by the following requirements. The first slice (labelled Ultra Reliable Low Latency Communications or URLLC) demands $\Delta_{URLLC} = 10$ ms communication delay and is characterized by a constant bit rate equal to 9.6 Mb/s. The second slice (labelled enhanced Mobile Broaband or eMBB) is characterized by a constant throughput equal to 11.2 Mb/s with a more relaxed latency requirements $\Delta_{eMBB} = 20$ ms. We set LACO's decision interval to 15 seconds and let our experiment run over the downlink direction for 100 decision intervals. Fig. 3.12c shows the evolution of the PRB allocation configuration decisions taken by LACO over this time span and how fast the convergence to a suitable layout is achieved. The monitoring information about incoming traffic at GTP level collected during the experiment are depicted in the upper plots of Figs. 3.13a and 3.13b. It should be noticed that these values represent aggregated values (sum) over monitoring intervals of 200ms. Latency and SNR information are depicted in the third and fourth plots of each figure. In this case, we use maximum and average as aggregation functions, respectively.

As described in Section 3.2.5, during the starting procedure the MAB algorithm explores all available arms with the aim of collecting an initial feedback on the system dynamics. Fig. 3.13a depicts the effects of these sequential choices on the latency experienced by the ongoing traffic flows. The initial steps drive the allocation of radio resources towards the eMBB slice thereby providing significant advantages in terms of experienced delay with respect to the URLLC one. In this phase, traffic coming from the URLLC might be dropped due to delay violation Δ_{URLLC}. The scenario changes after the 6-th decision interval, when the agent selects the configuration (35-15). Given the current channel quality, that arm does satisfy the URLLC radio requirements but does not reserve enough radio resources for the eMBB slice, thus increasing the latency experienced by its users. Subsequent arm selections within decision intervals 7 and 8, further reduce the radio resources assigned to the eMBB slice thus leading the traffic to violate Δ_{eMBB}. The MAB agent collects this information and quickly converges to a satisfactory configuration. In Fig. 3.13b, we focus on the system dynamics once the convergence is achieved and clearly notice how both the latency requirements are satisfied. Interestingly, despite similar traffic requirements, the algorithm selects the configuration (30-20), which assigns more resources to the first slice. This is justified by the lower SNR value experienced by such a slice during the experiment, as depicted in the bottom plot of Figs. 3.13a and 3.13b. The URLLC slice thus requires more PRBs to compensate for the lower MCS used during the communication and successfully meets the latency requirements. For illustration purposes, we select a vanilla PRB allocation policy, namely round-robin (RR), as a generic non-latency-aware benchmark and compare the performance of the two schemes running in the same scenario. The results of our experiments are summarized in Fig. 3.14, where both plots depict the empirical CDF of the latency, the RLC buffer density and the dropping rate incurred by each slice for the two allocation schemes.

The performances of the system when LACO is in place are depicted on the left-hand side picture, whereas the right-hand side shows the results of the RR-based slice scheduling scheme. In both plots, the URLLC slice is shown in blue and the eMBB one in orange. Based on these results, we can observe that LACO successfully meets both slices latency requirements. This is achieved by providing the required resources to the URLLC and eMBB slices (Fig. 3.12c) according to their different latency needs. This results in the URLLC slice allowing SLA latency requirements (≤ 10ms) at a very low average latency cost increase for the eMBB slice. In our experiments, very few traffic ($\sim 2\%$) experienced a latency above the 10 ms target of URLLC when using LACO, in contrast to $\sim 10\%$ experienced with RR. Despite of negligible impact, note that by our design choice parts of fragmented packets are sent even if above the latency threshold to avoid long HARQ based retransmission procedures [131], which may negatively affect the slice performance. Moreover, we wish to highlight that for LACO the amount of violations due to the exploration and convergence period could be significantly reduced if desired by introducing a policy aimed at minimizing such cases. The performance gap further increases when comparing the eMBB results. Given that RR sequentially allocates resources to the URLLC slice and, when the buffer is empty, to eMBB, it consistently favours the URLLC slice over the eMBB one. Thus, despite the higher channel quality condition experienced by the eMBB slice, in every scheduling period the resource availability for the eMBB slice is highly reduced. This provides a better performances for URLLC traffic, but at a significant degradation cost for eMBB users, as confirmed by Fig. 3.14b (bottom-right), which depicts the amount of traffic dropped during the experiment. The latency performance is strongly related with the traffic queue waiting in the transmission buffers. For this reason, the two figures depict the buffer size density distribution obtained during the experiments. It is clear from the comparison how different PRB allocation schemes affect the transmission buffer size at RLC layer. In the LACO case, they are generally lightly loaded, finally providing shorter serving time for incoming packets. In the RR scenario however, the eMBB traffic suf-



(a) *LACO*



(b) *Round Robin (RR)*

Figure 3.14: *Evaluation of different performance metrics for different scenarios.*

fers higher congestion, which leads to augment packet's waiting time, and consequently increases the rate of latency violations.

3.3 Conclusions

The key-enablers of 5G networks design are identified as Multi-Access Edge Computing (MEC) and Network Slicing capabilities, driven by the impelling need to provide high bandwidth as well as real-time access to mobile users in an isolated manner. In this chapter, we have first introduced the figure of the *MEC Broker* and proposed an orchestration solution, namely M^{2}EC, to deal with the concurrent deployment of MEC applications in multi-tenancy environments, with the objective of minimizing the overall capacity utilization exploiting applications consolidation over different MEC hosts. Considering the overall MEC system economy, our analysis shows significant benefits provided by the introduction of advanced resources allocation mechanisms into the slice management. This enables costs savings while providing ad-hoc solutions for external tenants willing to place their services over edge computing systems. Following the major efforts in the design of next-generation mobile systems around network slicing and (mobile edge) low-latency services, the second part of this chapter details LACO, a RAN-specific network slice orchestrator that considers network slice requests with *strict latency requirements*. Despite the efforts devoted by 5G researchers and engineers to network slicing, to the best of our knowledge, this is the first radio slicing mechanism that provides *formal delay guarantees*. To make network slicing decisions in environments with varying wireless channel quality and user demands, LACO builds on a learning Multi-Armed Bandit (MAB) method that is *model-aware* as opposed to classic MAB approaches that are blind to information regarding the underlying system. In addition, we exploit information from the system model to expedite the exploration-vs-exploitation process. Our results derived from an implementation with off-the-shelf hardware show that LACO is able to guarantee strict slice latency requirements at affordable computational costs.

Chapter 4

Data-driven Network Slicing Solutions

In this chapter we provide few concrete examples to demonstrate the capabilities of ML solutions applied to realistic mobile network scenarios. Building on operational mobile network data coming from real mobile network deployments, we showcase the capabilities of state-of-the art machine learning mechanism when dealing with multi-variate time series predictions, focusing our analysis on key mobile network metrics in non-trivial settings i.e., radio resource utilization within a football stadium premises, and end-user mobility along a major vehicular highway.

4.1 ARENA: A Data-driven Radio Access Networks Analysis of Football Events

Mass events such as sport events (e.g., football games), religious events (e.g., holy pilgrimages), political events (e.g., demonstrations) or entertainment events (e.g., concerts) are particularly challenging for mobile operators *despite being planned* with months or weeks ahead in most cases [132]. Indeed, operators know *when* a demand surge spawning from such events is coming but they are unable to adapt timely and appropriately [133].

Traditional delay tolerant [134] and information centric [135] approaches offer methods to smooth the traffic volumes during congestion time, e.g., postponing the transmission of delay tolerant traffic outside the busy time windows, or reducing the traffic flowing in the network by a proactive content placement at the edge of the network. However, due to generally scarce radio access resource availability, none of those approaches would avoid mobile traffic congestion in the event premises. To increase the spectrum availability, current approaches imply the on-demand deployment of, e.g., nomadic cells such as Cells on Wheels (CoWs) or Cells on Light Trucks (CoLTs), or fixed small (micro, pico, femto) cells for surge offloading. All these options are rather rudimentary and, needless to say, overly costly. Network slicing [5], which will allow an operator to request further slices of dormant resources (even from other operators) in a more flexible manner, can certainly reduce the cost tied to the hunt for capacity during these events. However, albeit planned ahead in time, the amount of additional resources required to cope with the demand during these situations is largely unpredictable.

Indeed, the amount of network load resulting from these events *depends on contextual features* such as the type of event—different events foster different mobile applications, such as real-time video streaming during concerts; and consumption patterns, such as data avalanches occurring during the breaks of a football match—or the ability of the event to attract attendance, such as the ranking of the matching teams in a football competition. However, although this contextual information is available, the model capturing the relationship between mobile traffic demand and the specific context of a given event *is inherently hard to build because mass events are rare and each is different in nature from one another*.

In this chapter, we advocate for the use of model-free deep learning techniques to take up on the challenge. Specifically, we first analyze data obtained from a major operator regarding the cellular coverage in the stadium of an important football team in Greece. It is well understood nowadays that the use of radio spectrum through mobile systems is closely related to human activity [136]. In this way, our analysis not only makes evident the aforementioned challenges, but it also sheds light on the behavioral dynamics of sport fans when attending major football events. Then, we formulate a Bandit Convex Optimization (BCO) model to formalize the problem we are addressing. Our model formalizes some of the aspects that make capacity forecasting during mass events complex, that is, the unknown relationship between user traffic patterns, spectrum usage and the context around the event. Then, to address this problem we design a deep neural network model to estimate the amount of extra spectrum resources required during the event to preserve the quality of service experienced by end users during regular days.

To summarize, the main contributions are the following:

- We provide a quantitative and qualitative analysis of the network statistics during football events taken from a real deployment;
- We formulate our problem by means of a Bandit Convex Optimization (BCO) model;
- We evaluate the performance of different state-of-the-art forecasting models while dealing with real-data;
- We propose a deep learning architecture, namely ARENA that takes as input monitoring metrics from Radio Access Network (RAN) devices as well as other contextual information to assess the extra spectrum resources required during the event;
- We validate the model and finally quantify the expected amount of resources to be proactively deployed in the stadium area to meet the target Quality of Service (QoS).

Privacy issues Our research activity does not violate user's privacy rights. The dataset contains only high-level aggregated and anonymous information.

4.1.1 Related Work

Mass events (sports, in particular) are gaining substantial attention for analysis in the last few years [137, 138, 139]. This is certainly a projection of the underlying social attraction to novel applications (and the business therein) that improve the experience for attendees and participants, e.g., virtual/augmented reality (VR/AR) technologies, in the upcoming 5G reign.

An accurate characterization of mobile network channel statistics is of paramount importance to assess the network capacity boundaries. However, as highlighted in [140], stable performances in mobile networks are much more complex to be achieved due to highly variable wireless channel statistics that affect the final service provisioning. The authors of [137] present a detailed analysis of spectator behavior with the goal of designing novel mobile applications for stadium-based sporting events. In their work, they exploit Bluetooth and GPS traces to derive information such as density, location and even travel speeds of crowds of people attending the UEFA world cup final that took place at the City of Manchester Stadium in May 2008. This early study hints at the importance of capturing contextual information to manage mobile services during mass events. A thorough analysis of mobile network performance during the 2013 Superbowl is presented in [138]. The authors conclude that the use of multicast cellular coverage in combination with content caching is paramount to solve the congestion of the uplink LTE channels during super-sized events. The authors of [139] analytically investigate the root causes of cellular network performance drops during crowded events, identifying the random access procedure as the main cause for QoE degradation and suggesting to leverage on device-to-device communication to cluster network access requests as to alleviate the problem. Similarly, [141] provides an in-depth analysis of the subscribers' quality of experience focusing on roaming scenarios.

Mobile traffic forecasting has fostered substantial research effort in its own merit. In [142, 143, 144, 145] the authors explore mobile traces collected from metropolitan areas to investigate and understand human mobility. In [34, 35] the authors focus on mobile resource utilization of individual services at a national scale. This work sheds the lights on interesting macroscopic properties and provides informative insights of todays' mobile networks, including spatio-temporal traffic patterns as well as the relationship between data volumes and urbanization levels. Moreover, several machine learning mobile traffic prediction solutions have been proposed in the literature. In [146], the authors use deep learning techniques for spatio-temporal modeling and prediction in cellular networks. The authors of [5] apply forecasting of mobile network resource utilization to investigate the network slicing concept in 5G systems. They design a dynamic resource allocation framework accounting for all the network domains and validate their proposal through exhaustive simulation campaigns. Deep models for long-term traffic prediction are also provided in [143, 147, 148]. Of particular interest from these works is how deep learning models substantially outperform traditional predictive methods. Finally, [149] proposes an encoder-decoder neural-network structure to forecast traffic demands on specific services (or network slices) and solve the trade-off between capacity over-

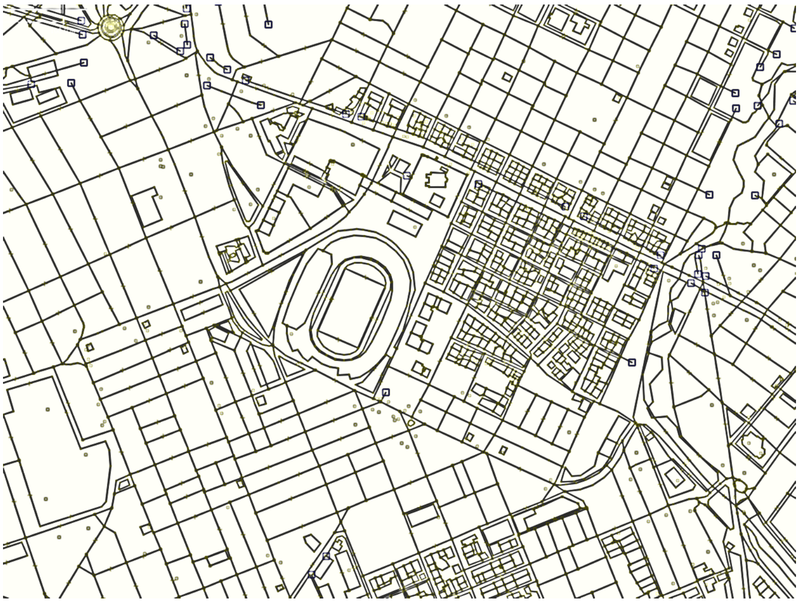


Figure 4.1: *Overview of the stadium area.*

dimensioning and service requirements guarantees. While this work on traffic forecasting provides insightful understanding on the dynamics of regular-user regular-day when interacting with mobile services, the model capturing such behavior is not valid during rare but massively crowded events.

In the sequel, we will first analyze the case of an important football stadium in Greece and the mobile coverage provided by a major mobile operator during a regular football season. A key observation of our analysis is that the relationship between active users and radio resources during sport events substantially differ from that of a regular day and so it requires special consideration to that given in [5, 34, 35, 146, 149].

4.1.2 Preliminary considerations

Despite frustrating, the poor network performance achieved in crowded initiatives like concerts or sport matches is commonly accepted by public and professionals attending such events who, after several initial unsuccessful trials, often postpone the upload (or download) of their photos and videos on social media due to network congestion. Similar poor conditions are mostly registered during an event in case of voice calls with a clear dropping rate increase [150]. The problem is well known and accounts for two separate root causes deriving from the need of widely guaranteeing access to the network, i.e., congestion in the radio access network (RAN) during the random access procedure (uplink traffic) and limited resource availability against a huge data demand surge (downlink traffic).

Standardization procedures

During the initial access, the User Equipment (UE) scans the system broadcast information to obtain the main system configuration parameters and synchronize with the network before attempting to attach through the physical random access channel (PRACH). The LTE standard defines two random access procedures: a contention-based and a contention-free. The former is mainly used for initial access, Radio Resource Control (RRC) layer connection (re-)establishment and for uplink data transmission in non-synchronized states or in absence of scheduled resource availability. The latter, less common, is mostly implemented to allow fast handover procedures. The RRC protocol of LTE system includes, among the others, connection establishment and connection release, system information broadcast and radio bearer setup and reconfiguration [151]. In LTE, the base stations or eNBs centralize most of the RRC functionality as well as the resource management and packet scheduling. The set of activities performed by the eNB varies according to the state of the radio connection—which can be either active (Connected) or not active (Idle)—between the UE and the network. In the $\texttt{RRC}_{\texttt{IDLE}}$ state, the UE passively monitors the system information broadcast by the network without the need to send monitoring reports or mobility updates. Conversely, UEs move into $\texttt{RRC}_{\texttt{CONNECTED}}$ state when performing a call or transmitting data, which increase the monitoring information reports sent to the radio access node in order to keep the session active. During the contention-based procedure, the UE randomly selects one of the $64 - N_{cf}$ orthogonal preamble signatures and the next available subframe for PRACH transmission, where N_{cf} is the number of preamble signatures allocated for the contention-free procedure (its value can change according to traffic load in the system). After sending the attachment request, the UE monitors the physical downlink control channel (PDCCH) and schedules a timer. In case of simultaneous RACH requests, the eNB uses the preamble sequence to differentiate among users. However, specially in crowded areas, different requests may collide. In this case, the eNB will not be able to decode the requests thereby triggering an exponential backoff procedure in the UE which delays the next access attempt. If no other user selects the same preamble sequence, the eNB is able to decode the request and reply with a Random Access Response (RAR) message through the PDCCH. At this point, the UE sends a RRC connection request message including a temporary identifier and the establishment cause. If accepted by the network, the access procedure terminates, and the UE moves into $\texttt{RRC}_{\texttt{CONNECTED}}$ state so that it is allowed to use the data communication services of the system.

Several optimization steps can be adopted to increase the performance of the RACH procedure. Typical deployments assume collision probability between UEs in the order of 1% with periodic random access occasions distributed every 10 ms. This setup translates into the possibility to handle an offered load of 128 attempts/second over 10MHz bandwidth [152]. Clearly, such a setup leads to low performance when dealing with crowded events. Typical approaches to solve this situation involve the reduction of the access occasion cycle duration and the increase of access opportunities within one frame, however this negatively impacts on the PRACH overhead finally reducing the scheduling opportunities for user data transmissions.

The second issue, data demand surges, is more intuitive and involves reduced rate of transmission opportunities (delay) and small chunks of resources scheduled per user within the Physical Downlink Shared Channel (PDSCH) (low throughput) due to a high number of users in $\mathtt{RRC}_{\mathtt{CONNECTED}}$ state.

Our work focuses on the latter issue. A simple approach to solve it is to smooth down traffic patterns by delaying load opportunistically to time instances with a less-congested network [134, 138]. However, this approach is only acceptable for delay-tolerant applications and hardly fits with the real-time nature of most use cases motivating 5G applications. The only solution upon such context is to increase the capacity of the system by deploying nomadic cells such as Cells on Wheels (CoWs) or Cells on Light Trucks (CoLTs), or offload traffic to fixed small cells, which effectively increases the capillarity of the network and hence the density of radio access points.

Unfortunately, this approach is overly expensive and slow, incapable of adapting to different traffic patterns occurring during events of different nature or events with different number of attendees. Certainly, network slicing opens the door for novel ways of increasing the capacity to the network opportunistically in a more agile manner (e.g., leasing additional spectrum through a network slice from neighbouring operators)—in the matter of few hours or even minutes. Still, the human relationship with mobile applications and the number of attendees substantially differ from one event to another. Summing up the fact that such massive events are rare (though planned), makes forecasting capacity requirements particularly daunting.

4.1.3 Data Analysis

In this section we present a data-driven analysis of sport events. Our dataset consists of weeks of monitoring data including large sport events taken place in a football stadium during a regular season under the coverage of 6 LTE eNBs, subdivided into 16 sectors, from a major operator in Greece. The set of eNBs covers an area of approximately 1 Km^2. Such concentration of radio access points is required to accommodate the traffic peaks generated by the almost 30000 people hosted in the stadium during sport events. For the same reason, the base stations are equipped with directive arrays of antennas and support

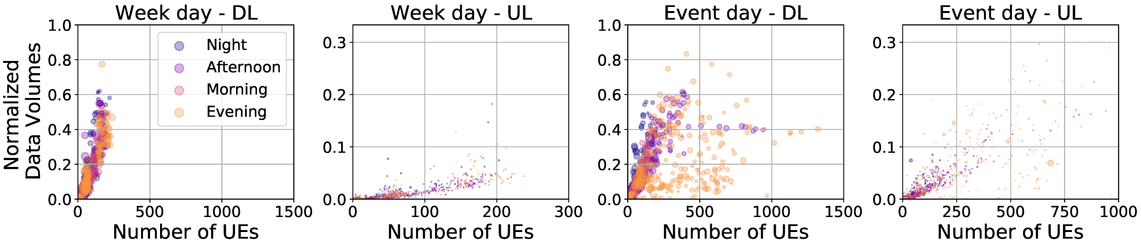


Figure 4.2: *Normalized data volume distribution as a function of the number of active users ($RRC_{CONNECTED}$) for downlink (DL) and uplink (UL) and different periods of the day (morning, afternoon, evening, night).*

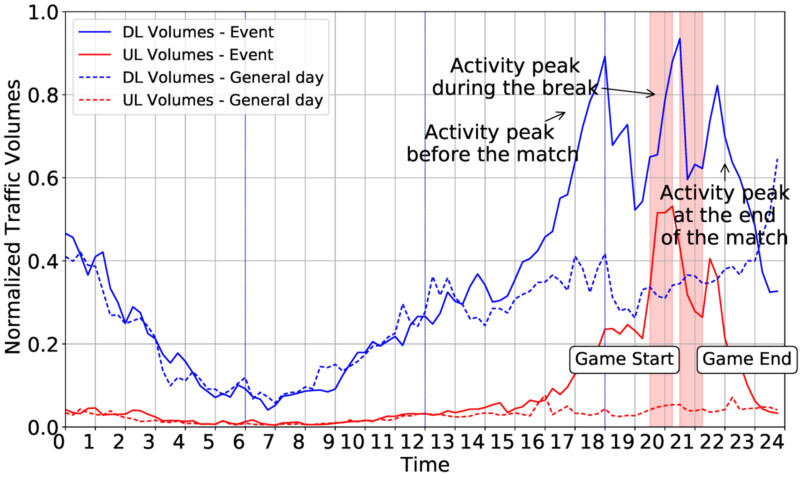


Figure 4.3: *Temporal evolution of the aggregated downlink (DL) and uplink (UL) traffic volume in the stadium area during a regular day (dashed line) and during the day of a large event (straight line).*

Carrier Aggregation (CA). As shown in Fig. 4.1, the surrounding district includes both residential neighbourhoods and vehicular streets, which highly characterize the spatio-temporal behaviour of the traces as later detailed in Section 4.1.3. We do not disclose the location of the eNBs due to privacy matters. Our goal is to characterize the relationship between active users during the events and network usage in contrast to regular days. Therefore, we focus our analysis on the average throughput per active user experienced by the event attendees in the stadium. Throughout the rest of the chapter, we will consider it as the key performance indicator (KPI) of the perceived quality of service (QoS) experienced by the end-users.

To ease our analysis, we focus on a particular sport match, taken place in February 10^{th}, 2019, which gathered 25097 attendees in the stadium as stated by official notes from the organizers.[1] The data collection exploits local information registered by the LTE eNBs for monitoring purposes, and include both averaged and aggregated features with a time granularity of 15 minutes. This ensures that the subscribers' privacy rights are preserved.

Traffic Volume Patterns

Fig. 4.3 shows the aggregated traffic volume generated within a 24-hour time period of a regular weekday (dashed line) and also during the day of the event (straight line), discerning between uplink (UL) and downlink (DL) traffic with red and blue lines, respectively. The event is scheduled to start at 19.30h and finish at 21.15h. We first observe that both UL and DL patterns are remarkably similar between a regular day and the day of the event up until 3 hours before the beginning of the event and the anomaly data usage persists until around 2 hours after the end of the event. This clearly reflects com-

---

[1] While our analysis is based on a single event, we have exhaustively evaluated other 10 different matches finding no relevant performance differences, which makes our approach generalizable.

mon human behavior as attendees usually gather in the surroundings of the stadium for social interactions before and after the event. This is evidenced by two volume peaks at around 18h and at around 21.30h. The former gradually vanishes until the event begins, coinciding with crowds moving towards the allotted seats in the stadium, whereas the latter peaks practically immediately after the end of the event and gradually vanishing as crowds move out of the stadium. Interestingly, there is a third peak, concomitant with the break of the event, with two drops in volume during the two halves of the event, which is expected as users are focusing on the game itself. We would like to remark that this pattern repeats across all events of the season with a gain factor dependent on the number of attendees and represents a signature of football games. Different events will leave different footprints and motivate the use of model-free approaches to forecast capacity requirements.

Fig. 4.2 compares the same distributions of traffic volume across different time periods, namely, morning (6-12h), afternoon (12-18h), evening (18-24h) and night (24-6h), as a function of the number of active users ($\mathtt{RRC}_{\mathtt{CONNECTED}}$) for both downlink and uplink in a regular weekday (first and second plots) and for the day of the event (third and fourth plots). As expected, the volume distribution during the day of the event is skewed with a long tail concentrating large number of users during the afternoon and the evening, precisely the new connections from the event attendees. This allows us to characterize the background traffic volume, generated by less than 250 active connections spawning less than 50% of the total system capacity following a strongly linear relationship. Such linearity is broken during the event for downlink traffic. The average data volume by active users within this regime (above 250 users) is lower than in normal conditions, which suggests poor quality of service experienced by the end users in the stadium.

Temporal Distribution of Mobile Users

Fig. 4.5 compares the activity of active users ($\mathtt{RRC}_{\mathtt{CONNECTED}}$) in the neighbourhood of the stadium during the day of the event (blue lines) with that of a regular weekday (dark lines), aggregated over all the eNBs in the area. The figure shows the average number of users within each 15-minute time window (dashed lines) and the maximum number

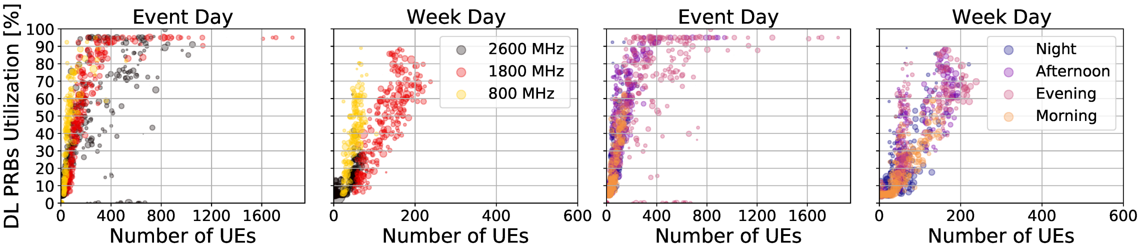


Figure 4.4: *Downlink PRB utilization as a function of the number of active users during the day of the event and during a regular day across three frequency bands managed by the eNB and different time periods (morning, afternoon, evening, night).*

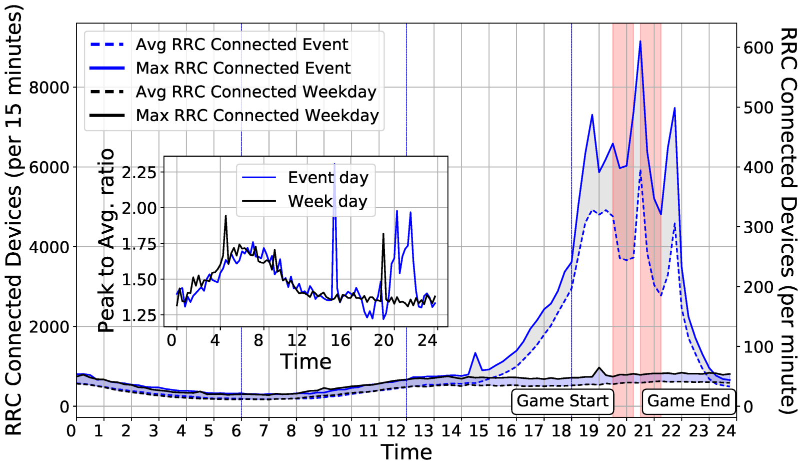


Figure 4.5: *Evolution of users activity. Peak and average number of active users ($RRC_{CONNECTED}$) (outer plot) and peak-to-average ratio (inner plot) during the day of the event and during a regular day.*

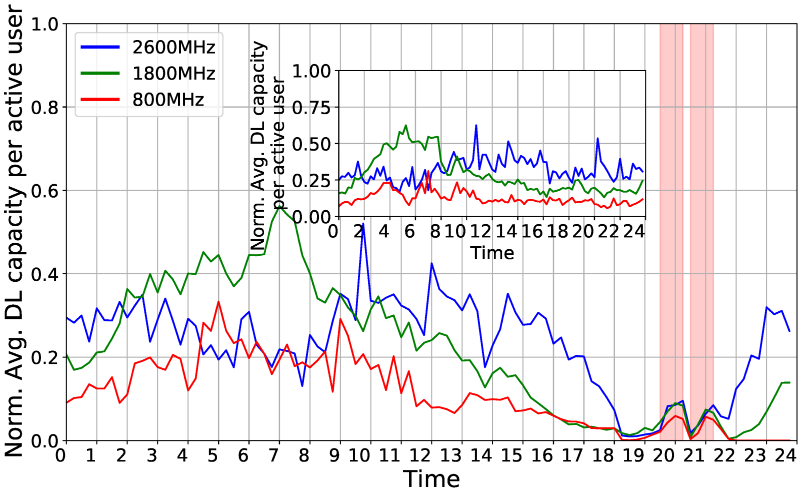


Figure 4.6: *Evolution of the average downlink (DL) capacity per user (our QoS metric) during the day of the event (outer plot) and a regular day (inner plot). Highlight areas indicate the two periods of time when the sport event is occurring.*

of active users within the same window (straight lines). As expected, the number of active connection peaks are highly correlated to the traffic volume time evolution shown in Fig. 4.3, with the number of concurrent connections ranging up to $600 \ \frac{\text{active UEs}}{\text{minute}}$, over 10 times higher than those during a regular day. The areas highlighted in light blue between straight and dashed lines provide an indication of the variability of the number of active UEs, i.e., the peak-to-average active connections, and evidence a substantial increase in volatility (around 2 times higher peak-to-average connections) during the course of the event. To visualize this, we create an inner plot depicting the peak-to-average ratio in both scenarios (event and regular day). Higher volatility in user activity is observed during the night and early in the morning, which is explained by the low number of net active connections (small variations yield higher peak-to-average ratios). Interestingly, however, the average-to-peak ratio increases during the event, which implies that the operator must deal with substantial control plane volatility in addition to data volume surges. It is clear that as soon as the base station reaches the saturation point, the perceived QoE decreases due to limited scheduling of resources. We further investigate this point in the following subsections.

Service Degradation

As mentioned above, we conjecture that the skewness of the volume distribution shown in Fig. 4.2 is due to poor quality of service. To confirm this hypothesis, we take a closer look to the physical spectrum usage, namely the downlink physical resource block (PRB) utilization measured at the eNBs. In this way, Fig. 4.4 compares the relative PRB utilization as a function of the number of active users during the event (first and third plot) and that of a regular day (second and fourth plot) for the different frequency bands used by the sectors of the eNBs (first two plots) and for different periods of time during the day (second two plots). We note that the 1800-MHz band saturates when the number of users grows over 250, which unavoidably translates into QoS degradation for the end users. The time analysis confirms that this event occurs during the time period of the football match.

This is further confirmed in Fig. 4.6, which compares the average throughput capacity per user over time during the day of the event (outer plot) against a regular weekday (inner plot), considering each of the three bands selected for our study. As expected, individual QoS substantially deteriorates during the sport event, particularly during a 2.5-hour time window before the beginning of the event, during the break of the event and during a 1.5-hour time window after the end of the event. With respect to standard working conditions, the average throughput capacity per user decreases up to 50 times.

Inter-Feature Time Correlation

We now study how different features relate to each other and across different neighbouring eNBs from a statistical perspective. To ease the presentation, we focus on two eNBs: an eNB that carries significant traffic volumes during both week and event days, labelled as ”High Load eNB” and represented with a blue color palette in Fig. 4.8; and an eNB that serves less significant amount of traffic, labelled as ”Low Load eNB” and represented with a red color palette. Specifically, Fig. 4.7 shows the temporal dynamics of different features collected by the two eNBs during a regular weekday (dashed lines) and during a match game (shaded area), while Figs. 4.8a and 4.8b depicts their correlation matrix. From Fig. 4.7, it clearly appears the performance gap with standard operating conditions. Specially for the UL case, during the event both the ”High Load” and ”Low Load” eNBs present traffic peaks 10 times higher than the traffic volumes achieved in normal conditions. Such UL activity peak can be justified by a sudden increase of social media activity and background cloud synchronization processes [138].

Let us first focus on the left matrices of Figs. 4.8a and 4.8b. As expected, since both the

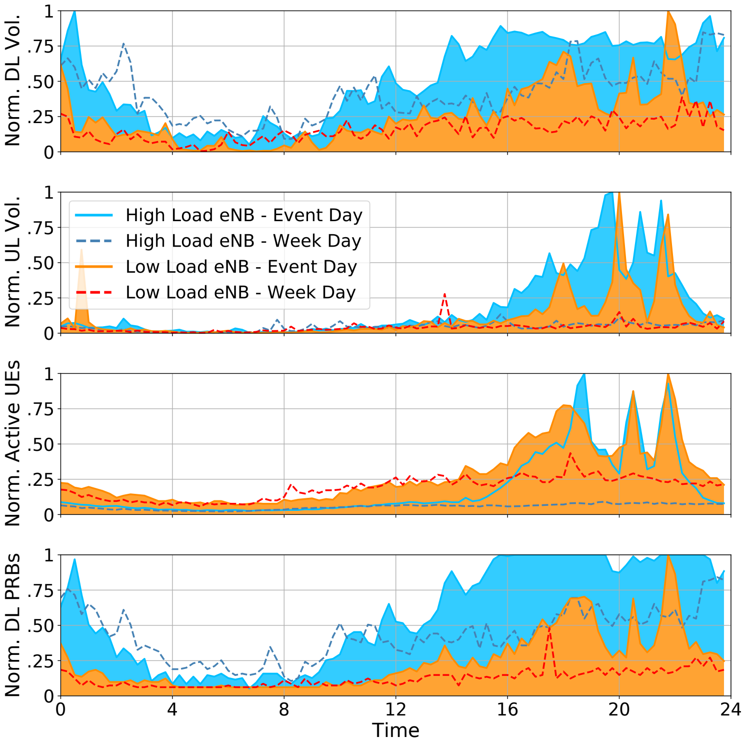
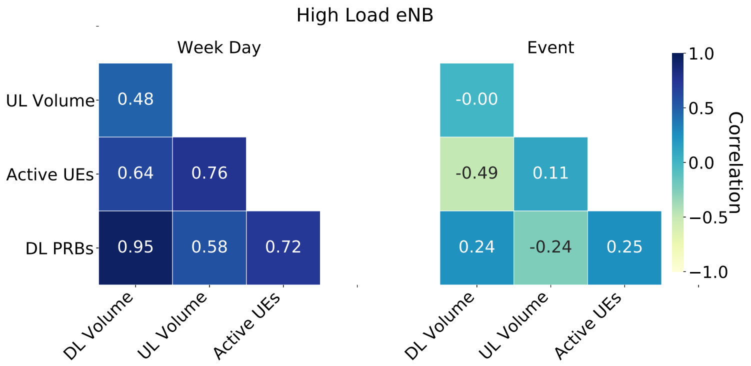


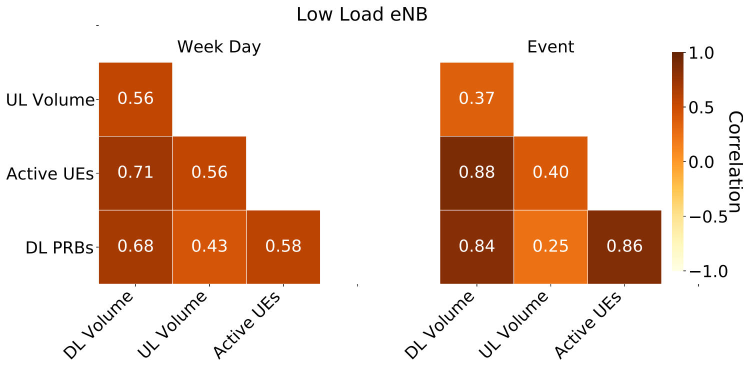
Figure 4.7: *Overview of different metrics collected by two eNBs in the stadium area during an event day and weekday (Normalized values).*

eNBs are deployed in the same area and the human activity corresponds to that of a regular weekday, the correlation value (Pearson's r coefficient) between each pair of features is positive and similar in both eNBs. This is true for every couple of features except the pair ”DL PRBs-DL Volume”, which appears strongly correlated for the highly loaded eNB and only moderately correlated for the lightly loaded eNB. It should be noticed that the latter exhibits a rather flat behavior during normal working conditions. More interesting are the correlation matrices we observe during the match, depicted by the two right-most plots of Figs. 4.8a and 4.8b. First, the lightly loaded eNB shows higher correlation values with respect to normal working conditions due to a more dynamic pattern enforced by the attendees in the area.

Second and more surprisingly, we observe that most of the correlations available during a weekday in the highly loaded eNB vanish during the match, eventually providing negative value for the couple ”Active UEs-DL Volume”. The latter represents a counterintuitive behaviour explainable by the fact that the considered eNB saturates its physical resources (PRBs) during most part of the game distorting the underlying relationship between the features. Together, these observations provide valuable insights not only on the fact that in normal working conditions features such as Active UEs and DL Volume are correlated (which is somewhat expected), but also that during mass events distortions may occur. This motivates us to design a mechanism that exploits not only the informa-



(a) *Correlation matrix for different traces collected by a highly loaded eNB during a weekday (left) and the event (right).*



(b) *Correlation matrix for different traces collected by a lightly loaded eNB during a weekday (left) and the event (right).*

Figure 4.8: *Temporal correlation of features in different working conditions.*

tion contained within each time series feature, but also the time-domain correlation that heterogeneous sequences might show among themselves. Moreover, as each eNB exhibits different behaviour, we advocate to model each cell individually taking into account both regular and crowded situations.

4.1.4 Model Design

In our data analysis, we showed that current network deployments may be insufficient during football events as the unexpected-high density of users can quickly saturate the available network resources. However, providing additional spectrum via increasing the density of radio access points is overly expensive, even with future network slicing mechanisms. Hence, proper mechanisms to accurately account for the required additional resources so that certain QoS targets are satisfied must be in place. Hereafter, we model the problem of network configuration during football events and suggest a new network setting to be dynamically applied by the operator.

Problem definition

Let us denote the set of base stations in the considered area as $b \in \mathcal{B}$, where $B = |\mathcal{B}|$ is the total amount of base station sectors. Without loss of generality, we consider each sector as an independent base station as each sector shows heterogeneous behaviour and each of them cover a different subset of the users in the area. We assume that time is split into different epochs $e \in E$ with duration T. Let r_u^e denote the amount of traffic served for user $u \in \mathcal{U}_b^e$ during epoch e (bits), where $U_b^e = |\mathcal{U}_{\lfloor}{}^{e}|$ is the average number of users under the coverage area of base station b connected concurrently within epoch e. Each base station b is provided with a spectrum capacity c_b^e (number of PRBs), which can be dynamically changed (every epoch e) based on the carrier aggregation policies of the telco operator. Note that in case of seriously-congested events, the operator may even decide to augment the overall spectrum capacity by placing portable base stations (e.g., Cells on Wheels or Cells on Light Trucks) or leasing or sharing a portion of the available bandwidth from other operators via, e.g., network slicing [153].

We now characterize the satisfaction of individual user connections by formally introducing the QoS metric used in Fig. 4.6 as the following expression:

$$q_b^e = \frac{\sum_{u \in \mathcal{U}_b^e} r_u^e}{\sum_{u \in \mathcal{U}_b^e} \tau_u^e}, \qquad \forall b \in \mathcal{B}. \tag{4.1}$$

where r_u^e represents the volume of traffic per user under the base station coverage served within the epoch e, and τ_u^e the effective downlink time per user, which accounts for the time when the first part of the PDCP SDU of the downlink buffer was transmitted until the buffer is emptied. It should be noticed that in presence of congestion and fair scheduling of resources, the average active time per user increases as a result of resource contention, thus leading to lower QoS. However, τ_u^e depends on the spectrum capacity covering the event *and* on the contextual data around the event (type of event, expected number of attendees) and the model capturing the behavior of τ_u^e is not known *a priori* and so is q_b^e's.

Objective. The target is to find the best PRB allocation on different deployed base stations such that the overall cost of such additional spectrum is minimized such that the aggregated QoS measured from all considered base stations is above a certain pre-defined threshold Γ. Of course, our focus is on the amount of spectrum required during mass events, which will typically exceed the capacity of the system and depends on the context of the event itself.

Constraints. We consider two sets of constraints. First, we need to impose that traffic queues are stable, that is, they do not grow infinitely over time, which would lead to dramatic drops in latency performance. We do not impose an instantaneous serving rate greater than the arrival rate; instead we consider that the long-term base station serving rate must be greater or equal to the long-term data arrival rate. Formally, this can be expressed by

$$\sum_{e\in\mathcal{E}}\sum_{u\in\mathcal{U}_b^e}(r_u^e-\zeta_u^e(c_b^e))\leq 0,\quad\forall b\in\mathcal{B}. \tag{4.2}$$

where ζ_u^e is the average throughput over epoch e for user u. Second, we shall impose certain threshold on individual QoS performance, which in turn can be succinctly described by

$$\sum_{e\in\mathcal{E}}(\Gamma_b^e-q_b^e(c_b^e))\leq 0,\quad\forall b\in\mathcal{B} \tag{4.3}$$

where $-q_b^e(\cdot)$ is the QoS of base station b during epoch e. While both functions $q_b^e(\cdot)$ and $\zeta_u^e(\cdot)$ are not known, we can get estimates at the end of each epoch e.

Convex functions and decision variables. Telco operators may apply different base station settings, i.e., the number of available spectrum resources (PRBs), and decide to increase the total number of available spectrum in order to tune users' performance. We assume that the functions capturing throughput and QoS performance are both convex with respect to the amount of spectrum or number of selected PRBs (the larger the spectrum, the higher the user throughput, the better the QoS). Note that a trivial solution to satisfy all user requirements, i.e., to keep the QoS at reasonable levels, is to provide an infinitely large amount of spectrum. However, deploying additional spectrum (e.g. via CoWs or network slicing) incurs in a cost δ^e for the operator that we shall minimize.

Bandit convex optimization

With the above information, we can mathematically formulate the following optimization problem

$$\begin{aligned}\min_{\boldsymbol{c}\in\mathbb{N}^{B\times E}} &\sum_{e\in\mathcal{E}}\sum_{b\in\mathcal{B}}\delta^e(c_b^e) && (4.4)\\ \text{s.t.} &\sum_{e\in\mathcal{E}}\sum_{u\in\mathcal{U}_b^e}\left(r_u^e-\zeta_u^e(c_b^e)\right)\leq 0,\quad\forall b\in\mathcal{B};\\ &\sum_{e\in\mathcal{E}}\left(\Gamma_b^e-q_b^e(c_b^e)\right)\leq 0,\quad\forall b\in\mathcal{B};\\ &c_b^e\in\mathbb{R}^+,\quad\forall b\in\mathcal{B},e\in\mathcal{E};\end{aligned}$$

where ζ_u^e, q_b^e and δ^e depend on the amount of spectrum (PRBs) c_b^e allotted to base station b during epoch e. We assume the following cost function

$$\delta^e(c_b^e)=\begin{cases}0, & \text{if } c_b^e\leq 1,\\ \alpha\,(c_b^e)^\beta, & \text{if } c_b^e>1;\end{cases} \tag{4.5}$$

so $\delta(c_b^e)$ captures the fact that base stations might be assigned with PRBs already available within the bandwidth provisioned to the ($c_b^e \leq 1$) and so it does not incur in extra cost, or additional spectrum is required ($c_b^e > 1$) with a cost parametrized by α and β.

Recalling the online convex optimization theory [154] and considering an invariant feasible set $\mathcal{C} = (c_b^e)$, we can rewrite the above optimization problem with its online Lagrangian version as follows

$$\mathcal{L}_e(\boldsymbol{c}^e, \boldsymbol{\lambda}) := \delta^e(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}^e(\boldsymbol{x}), \tag{4.6}$$

where $\boldsymbol{c}^e = (c_b^e)$ is the set of selectable PBRs during epoch e and $\boldsymbol{g}^e(\boldsymbol{c}^e) = \sum\limits_{e \in \mathcal{E}} (\boldsymbol{\Gamma}^e - \boldsymbol{q}^e(\boldsymbol{c}^e) + \boldsymbol{r}^e - \boldsymbol{\zeta}^e(\boldsymbol{c}^e))$. We can then write the recursive $\boldsymbol{c}^{e+1}$ given the prime iteration $\boldsymbol{c}^e$ as the following

$$\boldsymbol{c}^{e+1} = \arg\min_{\boldsymbol{x} \in \mathcal{C}} \nabla_{\boldsymbol{x}}^T \mathcal{L}^e(\boldsymbol{c}^e, \boldsymbol{\lambda}^e)(\boldsymbol{x} - \boldsymbol{c}^e) + \frac{1}{2\sigma}||\boldsymbol{x} - \boldsymbol{c}^e||^2, \tag{4.7}$$

where σ is a predefined constant and $\nabla_{\boldsymbol{x}}^T \mathcal{L}^e(\boldsymbol{c}^e, \boldsymbol{\lambda}^e) = \nabla \delta^e(\boldsymbol{c}^e + \nabla \boldsymbol{g}^e(\boldsymbol{c}^e)\boldsymbol{\lambda}^T$. This admits the following solution:

$$\boldsymbol{c}^{e+1} = \mathcal{P}_{\mathcal{C}}\left(\boldsymbol{c}^e - \sigma \nabla_{\boldsymbol{x}} \mathcal{L}^e(\boldsymbol{c}^e, \boldsymbol{\lambda}^e)\right), \tag{4.8}$$

where the projector operator is $\mathcal{P}_{\mathcal{C}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{c} \in \mathcal{C}} ||\boldsymbol{c} - \boldsymbol{y}||^2$, and the dual update as the following

$$\boldsymbol{\lambda}^{e+1} = \Big[\boldsymbol{\lambda}^e + \mu \left(\boldsymbol{g}^e(\boldsymbol{c}^e) + \nabla^T \boldsymbol{g}^e(\boldsymbol{c})^e)(\boldsymbol{c}^{(e+1)} - \boldsymbol{c}^e)\right)\Big]^+, \tag{4.9}$$

where $\mu > 0 \in \mathbb{R}^+$ is a step size.

Given that functions $\zeta(\cdot)$ and $q(\cdot)$ are not known, we need to construct a stochastic gradient estimate of unknown functions using the limited value information [155]. In particular, we can evaluate the function at a perturbated point $\boldsymbol{x} + \epsilon \boldsymbol{u}$ yielding that $\nabla f(\boldsymbol{x}) \approx E[\hat{\nabla}^1 f(\boldsymbol{x})]$, where $\hat{\nabla}^1 f(\boldsymbol{x}) = \frac{d}{\epsilon} f(\boldsymbol{x} + \epsilon \boldsymbol{u})\boldsymbol{u}$ is the one-point gradient [156]. Therefore, we need to recall the bandit online optimization theory to rewrite Eq. (4.8) as follows

$$\hat{\boldsymbol{c}}^{e+1} = \mathcal{P}_{(1-\epsilon)}\mathcal{C}\left(\hat{\boldsymbol{c}}^e - \sigma \hat{\nabla}_{\boldsymbol{c}}^1 \mathcal{L}^e(\boldsymbol{c}^e, \boldsymbol{\lambda}^e)\right), \tag{4.10}$$

where the projector operation is performed within a subset of $\mathcal{C}$ to ensure feasibility of the perturbed $\hat{\boldsymbol{c}}^e$. We can then write the dual update operation (Eq. (4.9)) as the following

$$\boldsymbol{\lambda}^{e+1} = \Big[\boldsymbol{\lambda}^e + \mu \left(\boldsymbol{g}^e(\hat{\boldsymbol{c}}^e) + \nabla^T \boldsymbol{g}^e(\hat{\boldsymbol{c}}^e))(\hat{\boldsymbol{c}}^{(e+1)} - \hat{\boldsymbol{c}}^e)\right)\Big]^+, \tag{4.11}$$

where $\hat{\boldsymbol{c}}^e$ is the learning iterate.

The above considerations suggest that the model may fail while trying to approximate functions $\zeta(\cdot)$ and $q(\cdot)$. In addition, we need the exact characterization of the mobility patterns influencing the number of connected user under each base station $\mathcal{U}_b^e$, for each time epoch e and base station b, as well as the expected traffic demand r_u^e. Unfortunately, all these variables present a strong relationship with respect to multiple real-world variables, making our objective hardly achievable within a reasonable time window. Therefore, in the next section, we propose a machine-learning-based approach to automatically learn and approximate these functions, and provide forecasts on the number of active users as well as their traffic request patterns based on a data-driven approach.

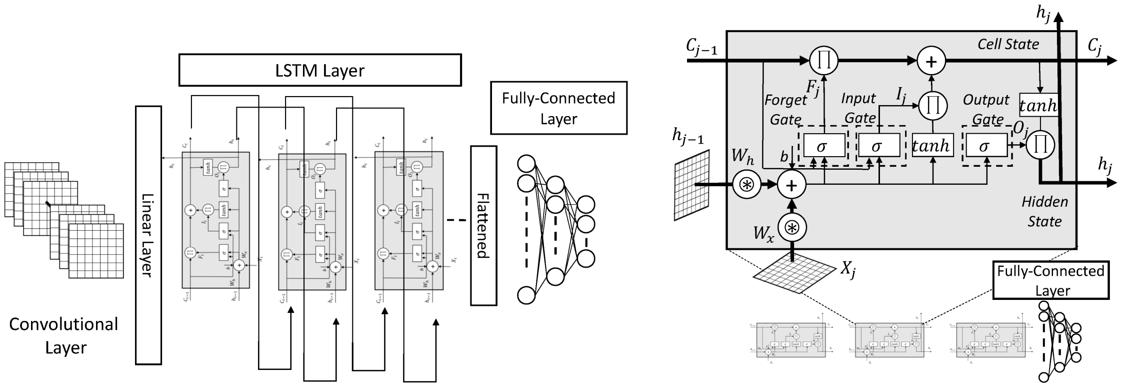
4.1.5 ARENA: Design and Performance

Our solution design, ARENA, requires two different ML-based models to estimate the number of active users and their traffic patterns. On the one hand, an LSTM-based approach allows to predict repetitive time patterns. On the other hand, a Deep-Learning model allows to approximate complex functions, e.g., those related to human mobility and cellular traffic loads. Thus, we first focus on inferring the number of active users in the stadium area, and then we exploit this information to estimate the additional resource utilization necessary to guarantee normal-condition QoE. Finally, we show the performance of ARENA applied to real traffic data.

Active users

Spatio-temporal characteristics of mobile traces unveil patterns that might be exploited to perform a forecasting process. In this context, Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs) [157] have attracted attention given their capacity to capture repetitive schemes within unstructured time series. An LSTM network is composed by a chain of units, which sequentially apply a linear combination of operations on the input data. Such a structure is fundamental to provide the network with the capability to *remember* useful information gathered from the past data samples and learn long-term trends in the input sequence making LSTM well-suited to handle mobile data traffic with significant spatio-temporal correlations [34]. LSTMs can be represented as a chain of J units, where each LSTM unit $j \in \mathcal{J}$ consists of 4 main elements, a memory cell and three gates, namely the input I_j, the output O_j and the forget gate F_j. The input and output gates are responsible to insert and read back the data into/from the memory cell itself, while the forget gate determines how much details should be kept or removed from the unit at each iterative step. The information stored within each unit $j \in \mathcal{J}$ is commonly referred as “state” of the unit, or C_j. To build more complex models, LSTM units are usually stacked and concatenated into consecutive layers, where the output of one layer feeds the input of the subsequent one, and the hidden state h_j of each unit holds the information about previously observed data. During the training phase, the input and the previous hidden state are combined to form a vector, which holds current and past information. Then, the vector passes through several activation functions, namely sigmoid and hyperbolic tangent functions depicted as σ and $tanh$ in Fig. 4.9b, which help regularizing the outcome of the linear combinations, finally adapting weight W and bias b values of each gate.

Although the LSTM model has proven powerful in handling temporal correlation, existing work in the literature suggests the use of LSTMs together with Convolutional Neural Network (CNNs) in order to increase the accuracy of the learning models [147]. The key-



(a) *CNN-LSTM architecture.*

(b) *ConvLSTM architecture.*

Figure 4.9: *Comparison of different LSTM-based architectures.*

idea is to exploit not only the information contained within each time series, but also the time-domain correlation heterogeneous sequences might show among themselves, as discussed in Section 4.1.3. For this purpose, we consider two well-known architectures, namely CNN-LSTM [158] and ConvLSTM [159], and compare their capabilities in predicting the number of $\texttt{RRC}_{\texttt{CONNECTED}}$ devices against the legacy LSTM model within a week time span.

As depicted in Fig. 4.9, the main difference about the two models is that ConvLSTM embeds the convolution steps into the LSTM unit, while CNN-LSTM models imply the concatenation of the two types of networks sequentially[2]. Both approaches therefore require the input data (and the output of the internal transfer functions) to be shaped as tensors to take advantage of the 2D representations of the network activities and include spatio-temporal information to learn valuable patterns within the dataset. The overall dataset has been split according to a $60/40$ ratio for the purposes of training and validation, respectively. We train the models exploiting a moving window of 6 days as input data and inferring on the consecutive one, over a time span of several weeks. Within our experimental setup, we considered several hyper-parameter settings. In particular, we account for a variable number of LSTM cells $J = \{200, 300, 400\}$ and convolutional filters $\Phi = \{64, 128, 256\}$ for each model architecture. The legacy LSTM model is composed of two consecutive layers of J units and a final fully-connected layer that provides the resulting output. The CNN-LSTM model accounts for two convolutional layers characterized by Φ filters each and a kernel size of 3 to learn spatial features, which are followed by an LSTM layer of J units and a final fully-connected layer to learn the correlation in time.

With respect to legacy LSTM models, the ConvLSTM architecture neglects the internal dense connection among gates in favour of a convolution operation, thereby reducing the number of model parameters and decreasing the chances of over-fitting. We adopt the mean-squared error (MSE) metric to train the models, and choose the Adam optimizer

---

[2] We refer the reader to state-of-the-art literature such as [158] and [159] for further implementation details.

Table 4.1: *Error characterization over different settings and models.*

	J / Φ	**LSTM**	**CNN-LSTM**	**ConvLSTM**	**ARIMA**
MSE	64/200	0.0039	0.0032	0.0036	0.055
	128/300	0.0041	0.0035	0.0040	
	256/400	0.0045	0.0039	0.0037	
MAX ABS Error	64/200	0.321	0.339	0.322	0.505
	128/300	0.296	0.322	0.318	
	256/400	0.321	0.328	0.321	

to optimize the loss function [160]. For the sake of comparison, we additionally include a baseline autoregressive integrated moving average (ARIMA) model characterized by parameters (p, d, q), where p is the order of the autoregressive term, d is the number of differencing required to make the time series stationary, and q is the moving average order [161].

This model requires the input time series to be stationary. Therefore, we applied simple differencing techniques to satisfy this requirement. Additionally, we implemented a grid search algorithm to optimize the ARIMA parameters choice, leading to the following settings $(p, d, q) = (5, 1, 0)$. We resume in Table 4.1 the resulting MSE and maximum absolute error measured over the forecasted sequences, averaged over 4 weeks of testing data. As expected, due to the non-stationarity of traffic traces (demonstrated by sudden peaks during the most busy time periods) the ARIMA predictor provides weak performances when dealing with the event time period, resulting in a maximum absolute error of 0.505. From our findings, we can conclude that the CNN-LSTM model with $J{=}200$ and $\Phi{=}64$ provides best performances in this scenario. Hence hereafter, we will adopt this model as our active user predictor.

Fig. 4.10 provides a comparative view of the models exploiting a walk-forward validation. Using such settings, the model is retrained as soon as new observations are made available thus expanding the time window horizon in each training step. Past predictions are then stored and evaluated against a longer list of observations, leading to more accurate forecasting performances. The different models exhibit good performance in forecasting the number of active users during normal working conditions thanks to the highly repetitive patterns. However, as highlighted by the inner-plot of Fig. 4.10, the prediction quality deteriorates during the event time period, resulting in an underestimation of the activity peaks. Due to its asymmetry with respect to standard working conditions, it is clear that temporal information alone is not enough to accurately predict the mobile traffic activity in the stadium during public events. In the following, we thus propose a deep learning model that in addition to spatio-temporal information also exploits contextual information to enhance the accuracy of the predictions.

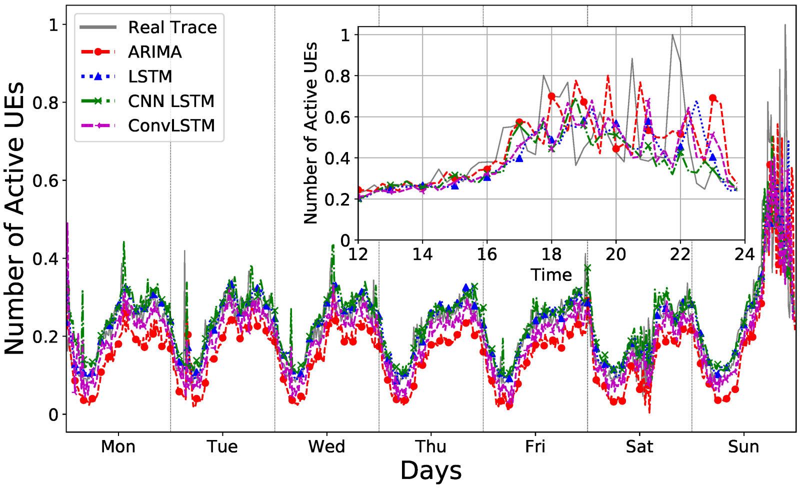


Figure 4.10: *Comparison of different LSTM-based forecasting models working on real data from a base station in the stadium area (Normalized values).*

Resource utilization

The relationship between QoS and availability of spectrum resources, characterized by functions $\zeta_u^e(c_b^e)$ and $q_b^e(c_b^e)$, is unknown and hard to construct due to the rarity and particularity of mass events. We hence focus on model-free deep learning methods to build our capacity forecasting mechanism.

To this aim, we design a neural network structure per base station, each receiving two sets of inputs pertaining each respective eNB: i) a representation of contextual information over the event (such as expected number of attendees, type of the event, etc.), and ii) network-specific monitoring data over time (historical time series), as depicted in Fig. 4.11. In order to reduce the dimensionality of the contextual data yet extract meaningful information we use a standard sparse autoencoder (SAE) [162, 163, Ch.14]. In brief, a SAE consists of two feed-forward neural networks: an encoder (with an output layer of size 1 in our case) and a decoder (with an output layer of size equal to the dimensions of the input of the encoder). They are trained together so that the reconstructed output of the decoder is as *similar* as possible to the input of the encoder. During exploitation, we simply use the encoder part of our SAE, which effectively compresses the contextual information into a 1-dimensional value. For the purpose of this study, we concentrate on the expected number of attendees to the event as contextual data, and empirically select the CNN-LSTM model as main forecasting module in light of the results discussed in Sec. 4.1.5. On the other hand, the second set of inputs concerns the historical traces of network measurements introduced in Section 4.1.3. In particular, we select the time evolution of average connected users U_b^e (for base station b during epoch e), volume of downlink traffic as well as our QoS metric q_b^e. Thus, we train a deep neural network for each base station in the stadium area using 60% of the football events in our dataset and validate the outcome using the remaining 40%.

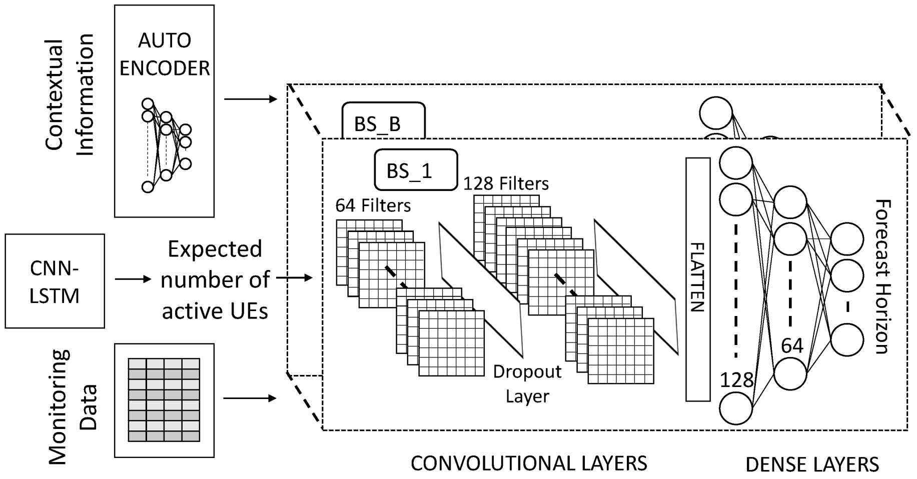


Figure 4.11: *Deep neural network architecture used in ARENA.*

Fig. 4.11 depicts the overall architecture of our system, consisting in two neural networks connected in series. The first neural network accounts for two layers of two-dimensional Convolutional Neural Network (2D-CNN) with η and μ filters respectively. The CNN is a class of deep neural network generally used in computer vision for their ability to detect patterns in input images. The workflow assumes each filter to be convolved along width and height of the input image to produce an activation map. Different filters detect distinct features so that a set of activation maps are passed to the next layer of the CNN. We leverage on this and select from the overall measurements matrix $\mathbf{M}$ the information related to a specific base station b, i.e., $\mathbf{M_b}$. Then, we split $\mathbf{M_b}$ in epochs, or snapshots s_b^e, generating smaller input matrices of size $(F \times T)$, where F is the number of features and T is forecast horizon. We adopt a sliding window over time as data augmentation technique. A normalization function $\mathcal{N}(\cdot)$ is applied at this point to favour the learning process.

The neurons of the 2D-CNN apply a filter $\mathcal{H}(\sum_{e \in \mathcal{E}} \mathbf{I_e} \circledast \mathbf{K} + \mathbf{b_e})$ where $\mathbf{I_e}$ is the input matrix at epoch e (i.e., $\mathbf{I_e} = \mathbf{s_b^e}$), $\circledast$ indicates the 2D convolution operator, $\mathbf{K}$ is the filter kernel, $\mathbf{b}$ is a bias vector and $\mathcal{H}(\cdot)$ is a non linear activation function. The significant development of deep learning methods lead to the definition of plethora of different activation functions in the literature [164]. We empirically select the Rectified Linear Units (ReLu) function to overcome the vanishing gradient problem and allow the models to learn faster. In order to reduce overfitting in favour of generalization, the inner layers of the CNN are interleaved with a Dropout layer. At each training stage, individual nodes and related incoming/outgoing edges are either dropped (with probability p) or kept (with probability $1 - p$). This step allows decreasing the co-dependency from adjacent neurons during the training phase. The dropout probability is fixed to 0.2. The output of our CNN consists on a Flatten layer, so we map the input matrices into a fixed-length vector that feeds the second stage of the architecture.

The second neural network consists in three feed-forward layers. In this structure,

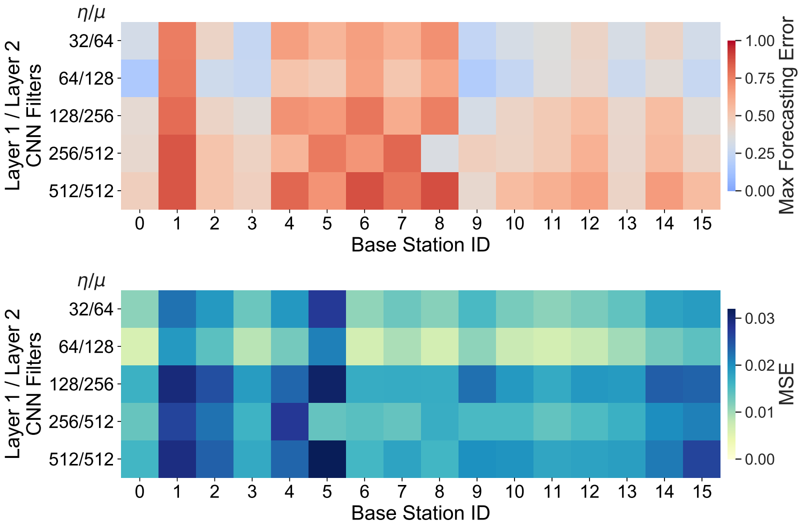


Figure 4.12: *Impact of different hyperparameter settings on ARENA forecasting performances.*

also called Multi-Layer Perceptrons (MLPs), the neurons of one layer are connected with all the neurons of the adjacent layer to build a finite and acyclic graph. The number of neurons in the first and second layer are fixed to 128 and 64 respectively, while the size of the output layer of the network matches with the number of samples to be predicted T for both output estimations: the active number of users and the PRB utilization during the event. Importantly, the MPLs inherit from standard multi-layer feed-forward networks the capability to approximate continuous n-dimensional functions $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ as stated by the universal approximation theorem [165].

Thus, the output of each MLP layer is a linear combination of weighted real-valued input and a nonlinear activation function, namely $y = \phi(\mathbf{w}^\mathbf{T}\mathbf{x} + \mathbf{b})$, where $\mathbf{w}$ denotes the vector of weights, $\mathbf{x}$ is the vector of inputs, $\mathbf{b}$ is the bias and $\phi(\cdot)$ is the non-linear activation function. Also in this case, we adopt the ReLu function for the hidden layers and a linear function for the output layer to retrieve the forecasting samples amplitude. The output of the neural network includes a T-dimensional vector predicting the amount of PRB utilization $\bar{c}_b$ during the T epochs covering the event and a T-dimensional vector with the respective prediction on the number of active users $\bar{U}_b$. The training phase exploits the adaptive moment estimation (Adam) optimizer, with learning rate 10^{-4}. At each iteration, the model parameters are adjusted to minimize the error between predictions and ground truth measured by means of a loss function. In our model, we exploit the Mean Squared Error (MSE) loss function [166].

The choice of the best hyperparameter settings for ARENA has been performed by means of an accurate optimization process aiming at reducing the forecasting error. In Fig.4.12 we investigate the effects of a variable number of convolutional filters η and μ over the two convolutional layers included in our proposed architecture. To deal with the highly heterogeneous traffic patterns, we have individually trained each model exclusively accounting for the traffic traces coming from the respective base station. The methodology to train each model follows the one previously presented. The training phase accounts for 50 epochs and runs over 60% of the overall dataset, while the validation phase encompasses the remaining 40%. Our aim is to deal with mass events within the stadium premises over the event days, therefore, in the upper plot of Fig.4.12 we depict the maximum forecasting error experienced while estimating the resource consumption of each base station. From the picture, we can notice that for some base stations the resulting performances are poor, regardless the configuration settings of our model. This outcome can be easily explained as follows. Such considered base stations are generally underutilized in regular days. This leads the forecasting models to provide biased estimations of the cell capacity utilization, even in case of significant traffic loads generated during the sport events.

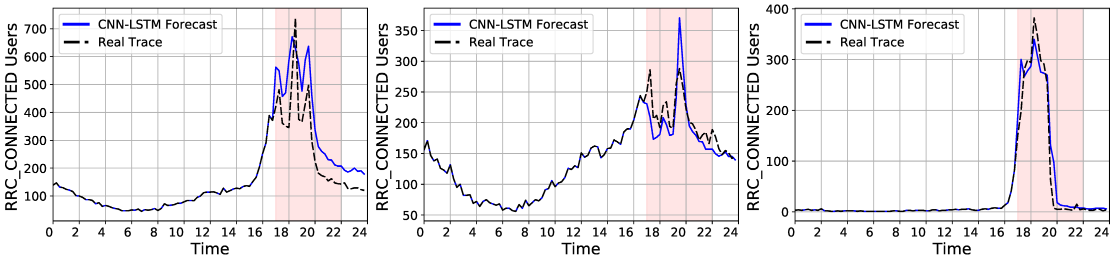
From the picture it can be noticed that an increase in the convolutional layer size yields a general performance degradation, both in terms of MSE and maximum forecasting error. Adding more layers may help to extract more features, but it also increases the number of parameters composing the model that, in turn, augments the chances of overfitted models. Empirically, we have identified the best performance with $\eta = 64$ and $\mu = 128$ settings. Hence, we adopt these values throughout this chapter.

With the above neural network structure, our system learns the relationship between active users and the incurred load. Our goal however is to provide additional spectrum capacity such that the same QoS experienced by the users during regular days is preserved during the event. To this aim, we let $\Delta_b^e = \frac{\bar{U}_b^e}{\tilde{U}_b}$ denote a scaling factor, where $\bar{U}_b^e$ is the predicted number of active users during epoch e for base station b (also marked as the output of our neural network) and $\tilde{U}_b$ is the number of users required to saturate resources of base station b according to a radio usage behavior from a regular day—this can be learned from previous history. The motivation behind is to compensate the expected radio resource usage with respect to the predicted impact that extra active users might have on individual QoS during the event. As a result, we finally devise the amount of spectrum required during the event as

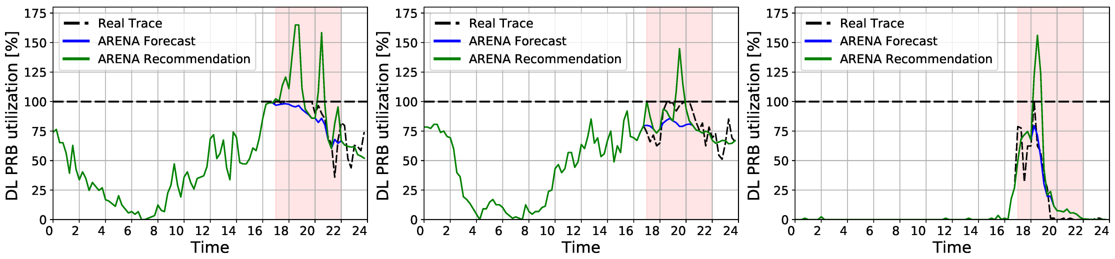
$$\boldsymbol{c}_b = \bar{\boldsymbol{c}}_b \cdot \boldsymbol{\Delta}_b^T, \qquad \forall b \in \mathcal{B}. \tag{4.12}$$

Performance evaluation

Hereafter, we evaluate ARENA considering the real traffic traces collected within the stadium neighbourhood. We leverage the neural network capability to approximate the relationship between QoE and PRB utilization as well as the temporal correlation of the measurements (as described in Section 4.1.4) in order to measure the amount of additional resources required to cope with the traffic demand during football events. To this aim, we set the time horizon from which our neural network starts making predictions to 2 hours before a real event and then we assess the ability of ARENA to predict the active number of users and spectrum usage after such time horizon. Based on this information, ARENA



(a) *Sector 1* **(b)** *Sector 2* **(c)** *Sector 3*



(d) *Sector 1* **(e)** *Sector 2* **(f)** *Sector 3*

Figure 4.13: *Forecast of the number of active users or* $RRC_{CONNECTED}$ *(*U_b*) during a mass event (blue line) and actual evolution (dashed black line) for three sectors chosen for illustration purposes. Forecast on the radio resource usage (blue straight line), actual usage evolution (dashed black line) and estimation of extra radio resource allocation (green line) for the same sectors.*

may take optimal spectrum allocation decisions. A complete snapshot of our validation results is depicted in Fig. 4.13. On the one hand, Figs. 4.13(a)-(c) (top three plots) show the evolution over time of the active users for three different sectors chosen for illustration purposes. The black dashed line represents the actual data whereas the blue straight line shows the predicted values. Like before, the area in light red highlights the period of time when the event takes place. Note that both lines overlap before the time horizon of the event as they represent known information from the history. We observe that our neural network structure follows the time evolution of active users during the event remarkably well thereby capturing the particularities of the football match as we have discussed in our analysis in Section 4.1.3. On the other hand, Figs. 4.13(d)-(f) (bottom three plots) show the evolution over time of the downlink spectrum utilization for the same sectors. Similarly, the dashed line represents the actual time series whereas the straight blue line shows the predicted values spawned by our neural network structure. In addition, we show the recommended PRB allocation with a straight green line (c.f. Eq. (4.12)), obtained varying the input QoS metric as in standard operating conditions. As it can be observed, our prediction on spectrum usage follows closely the real usage until the radio access network capacity limit is reached. In such cases, the recommended PRB allocation value (green line) follows the dynamics of the active users keeping the amount of time wherein extra capacity is required (PRB allocation above 100%) limited, thereby minimizing the incurred cost.

4.2 A Learn-as-You-Go Framework for On-Demand Emergency Slices in V2X Scenarios

Despite the high investment volume in public transportation, many people daily commute to work with their private vehicles over major roads around cities. Accordingly, both drivers and passengers consume and generate a large amount of data along the road mobile infrastructure for a wide variety of purposes: navigation systems, car sensors, infotainment but also phone calls, social media, streaming, etc. Along crowded roads, this may lead to network congestion and, in the worst case, service disruptions.

In order to avoid these situations, the next generation of mobile networks (5G) has defined the *Network Slicing* concept, which allows infrastructure providers to dynamically instantiate *on-demand customized virtual network instances* with dedicated Service Level Agreements (SLAs). Standardization bodies have defined the overarching architecture [18] to support such isolated slices, thus fostering research on dynamic resource orchestration mechanisms based on well-known technologies such as Network Function Virtualization (NFV) and Software Defined Networking (SDN) [167, 168].

As 5G gets rolled-out and advanced V2X services deployed, solutions to protect mission-critical services will become increasingly important. While significant road safety improvements have been introduced in the last decades, road fatalities are still a major cause of injuries and death worldwide [169]. Thus, guaranteeing public safety along roads is still a major challenge that requires novel solutions. Due to the combined effect of high-mobility patterns, traffic volumes and advanced automotive-related use cases (V2X), mobile network infrastructure along roads will face daunting challenges to guarantee mission-critical services in unexpected congestion scenarios. In this context, *Emergency Network Slices (ENS)* are expected not only to improve situation awareness during emergencies, but also to support the provisioning of enhanced communication schemes, e.g., Ultra-Reliable and Low-latency Communication (URLLC) for virtual and augmented reality (VR/AR) to first-responder teams, e.g., ambulances, police, firefighters, that have to reach the event location and manage emergencies in a faster and safer manner [170]. As ENS will get top priority, solutions need to be devised to re-dimension already running services according to their criticality level. Unexpected road events progressively cause traffic congestion to nearby areas and, in turn, saturate the networking resources of adjacent base stations. If such a *propagation effect* could be *predicted*, their congestion effects could be alleviated by means of *proactive slices resource management*.

We take on this challenge and propose *Passive Information-based Resource Orchestration in Automotive Driving scenarios (π-ROAD)* that relies on a deep learning framework to *analyze* passive information from real-time mobile network traffic statistics, *learn* regular traffic patterns, and accurately *detect* anomalous deviations due to unexpected road events.

4.2.1 Related Work

The ever-increasing vehicular traffic fosters the need to deeply understand the complex relationship that regulates mobility patterns which, in turn, affects mobile network operational conditions [171, 172]. From the one side, this effort requires constant monitoring of the communication infrastructure. From the other side, the highly heterogeneous set of monitoring KPIs demands for advanced solutions to automate the early detection of anomalous conditions.

The authors of [173] initially addressed this scenario proposing a Bayesian network working on a set of discrete metrics collected from an operational UMTS infrastructure. Similarly, the work of [174] focused on a large-scale cellular network scenario, where traffic traces are modeled into regular and random components. Their decomposition approach well suits predictable patterns, but fails in highly variable scenarios. More recently, state-of-the-art solutions start combining monitoring traces and heterogeneous contextual information to improve the effectiveness of model predictions and decisions. In [149, 175] the authors leverage spatio-temporal characteristics of mobile traces to predict resource utilization in the context of network slicing. In [176] the authors include weather conditions coming from social media sources to enhance the accuracy of their ARIMA-based model.

Most of the works in the literature address the urban environment and aims to mitigate the cause of traffic congestions by predicting traffic flows and offloading strategies to alternative paths [177], while less studies targeted the highway scenario. In [178] the authors present a spatio-temporal analysis of highways travel patterns exploiting per-vehicle data records collected by a centralized authority in China. Given the fine granularity of per-vehicle information, there is lack of a discussion about anomaly detection from a mobile network perspective. Additionally, note that the majority of the related works presented above tackle the vehicular traffic prediction by means of active systems, e.g., GPS location exchange, or networks of sensors deployed along the road, which clearly ease the prediction task but also increases the volume of information generated along the process. To the best of our knowledge, this is the first work considering the prediction of road events and their impact on the mobile network infrastructure, exclusively accounting for *aggregated* and *passive* RAN monitoring samples.

4.2.2 Data Analysis

We carry out an analysis of the network traffic dynamics occurring on the vehicular roads. The considered dataset consists of 6 months of real network data (February-July 2020) collected from an operational LTE network deployed alongside the Italian A4 highway shown in Fig. 4.14. The highway has a length of approximately 400 km and is located in the north of Italy, connecting the city of Turin with Venice, passing through the metropolitan area of Milan. Along this road segment, more than 1000 LTE cells (about 200 eNodeBs) are deployed to provide connectivity to the users commuting or traveling on this path, and to citizens living in the proximity of the highway. Available data exploit local moni-

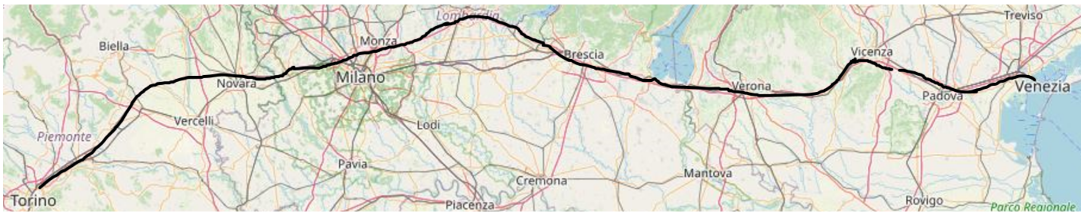


Figure 4.14: *Overview of the Italian highway (A4).*

toring information of LTE eNBs including Key Performance Indicators (KPIs) averaged and aggregated over 15 minutes.

Spatial consideration

Fig. 4.15 (top) depicts a snapshot of normalized DL and UL traffic volumes (per single base station, where the approximate base station location is highlighted in the upper part of the plot) aggregated over a month. We highlight areas of the main cities located nearby the A4 roadway. As expected, DL volumes increase in the proximity of major cities, due to urban traffic leakage of base stations covering the highway. Additional traffic peaks can be identified in specific geographical locations, e.g., intersections between different highways, main inter-urban roads, or train lines. Generally, these locations are characterized by a higher density of base stations, which means that the occurrence of any road events in these points will probably lead to a major impact on the mobile network. Moreover, numerous mobile terminals characterize such locations over time, due to significantly higher user mobility. UL traffic shows analogous behaviors but with a limited volume (about 10% of the overall DL traffic).

Temporal patterns consideration

Several works in the literature suggest a strong relationship that correlates end-users mobility patterns with cellular network statistics [35, 166, 179] in urban environments. Similarly, the mobility patterns identified on highways present repetitive trends following regular working routines. In this respect, we plot at the bottom of Fig. 4.15 normalized number of active users during working days and weekends for different areas of the highway, in particular two areas around metropolitan cities as well as two sections of the highway between two main cities. The signatures are extracted by calculating the median values of the same time periods over several weeks, separately for working days of the week and weekends, as proposed by [180].

The first and second subplots show that areas around big industrial cities, such as Milan and Turin, are characterized by commuting patterns with a presence of mobility peaks in the morning, noon and evening during the working days, which are absent on weekends. Conversely, mobility patterns of users during working and weekend days are comparable when considering highway segments interconnecting major cities (third and fourth subplots).

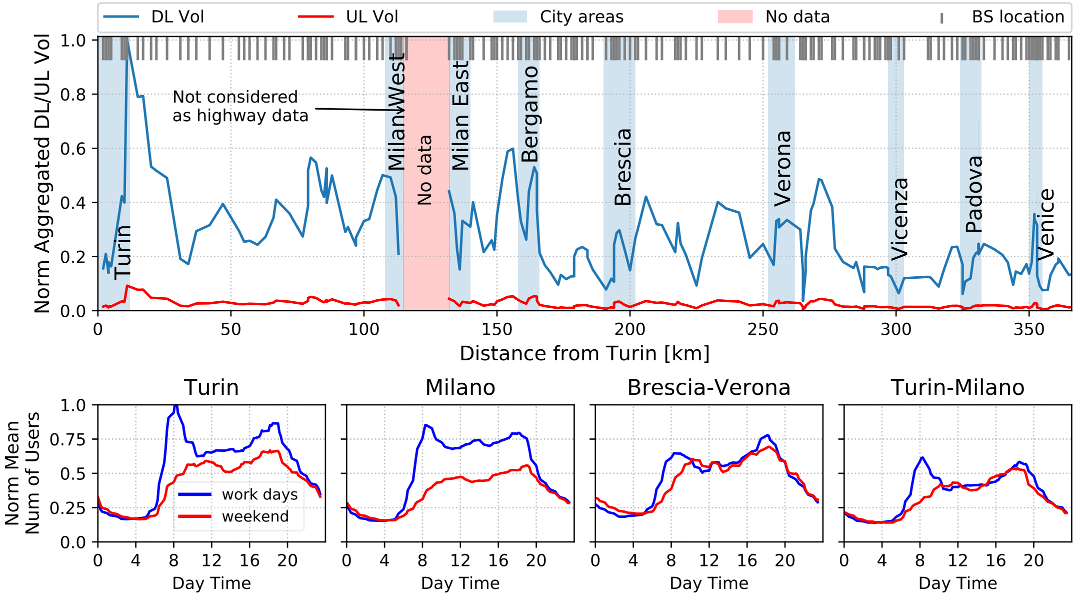


Figure 4.15: *Aggregated spatial distribution of DL/UL traffic volumes along the highway (top), with the focus on working day and weekend signatures for specific highway areas (bottom).*

Service-based QoS consideration

Different services with heterogeneous requirements might be seamlessly managed by the mobile network infrastructure. In real deployments, this is usually achieved by labelling each traffic flow with a specific QCI Class Identifier (QCI) to ensure that each traffic bearer [181] is allocated with the appropriate set of resources to guarantee an affordable Quality of Service (QCI).

In Fig. 4.16 (left-hand side), we depict the temporal distribution of the DL volumes differentiated by QCI traffic types. It can be noticed a dominance of non-guaranteed bit-rate (NON-GBR) traffic types (QCIs 6 to 9), mostly related to video-streaming and social media activities, and almost negligible volumes for guaranteed bit-rate (GBR) traffic types (QCIs 1 to 4). Note that the satisfaction of service requirements also depends on the network congestion level and on the instantaneous channel quality experienced by end-users, together with the corresponding modulation and coding scheme (MCS) selected at the eNodeB scheduler. Interestingly, DL traffic perceives, on average, a lower MCS index with respect to the UL one (right-hand side of Fig. 4.16). This is due to the high end-user mobility and longer data exchange sessions, which suffer from a wider communication distance along the path [182].

Road events consideration

The considered dataset takes into account monitoring information from the mobile network infrastructure, but it lacks information related to vehicular traffic and unexpected congestions/events that affect its dynamic. To fill this gap and help us to accurately relate

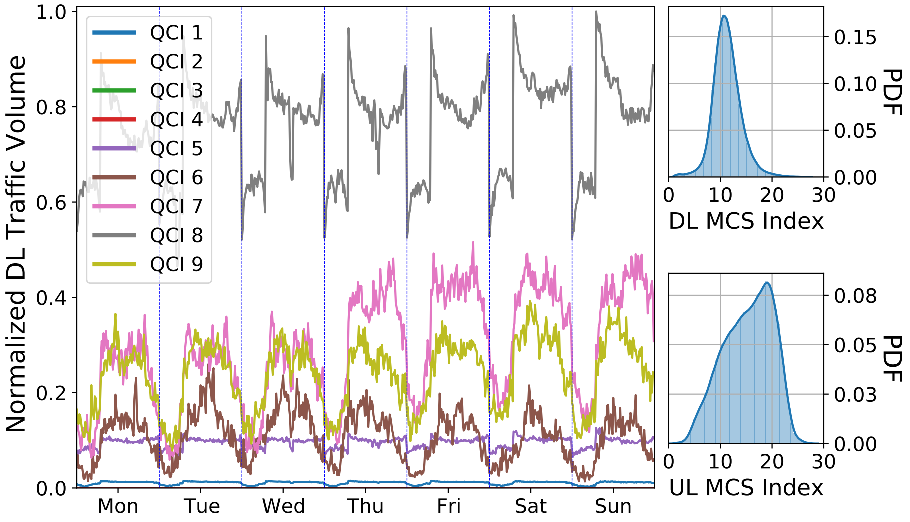


Figure 4.16: *Temporal distribution of different traffic types volumes.*

network traffic dynamics with road events, we rely on publicly available information coming from heterogeneous social media sources like Twitter notification service of *Autostrade per l'Italia*[3], Google Maps[4] and Waze[5], which facilitate an exhaustive catalog of real-time road event information and useful metadata, e.g., timestamp, exact location, and a short description.

After 6 months of data collection, the final dataset includes about 800 road events,[6] as shown in Fig. 4.17. As expected, road events mainly occur during morning and evening commuting periods and are geographically placed close to main cities, e.g., Milan and Verona. We will match the information contained in this supplementary dataset with the network geographical deployment information to assign each road event with the closest base station and obtain the ground truth of occurred events, as later detailed in Section 4.2.3.

Event propagation effect

Road events might have a consistent impact on the vehicular traffic conditions and, in turn, on overall network resource availability. However, it might be easily confused with common network traffic outliers and therefore be ignored. Current network deployments may come to help: the RAN deployment along the highway has quite a regular pattern and might reveal implicit information as monitoring data are simultaneously retrieved from different observation points. In particular, a road event might affect the end-user's

---

[3] Online available at: https://twitter.com/trafficoa

[4] Online available at: https://www.google.com/maps

[5] Online available at: https://www.waze.com/livemap

[6] Note that our experimental campaign has been carried out during the Covid-19 pandemic when the imposed lockdown limited the overall end-user mobility in the north of Italy, therefore, reported numbers may be biased and differ from yearly regular data.

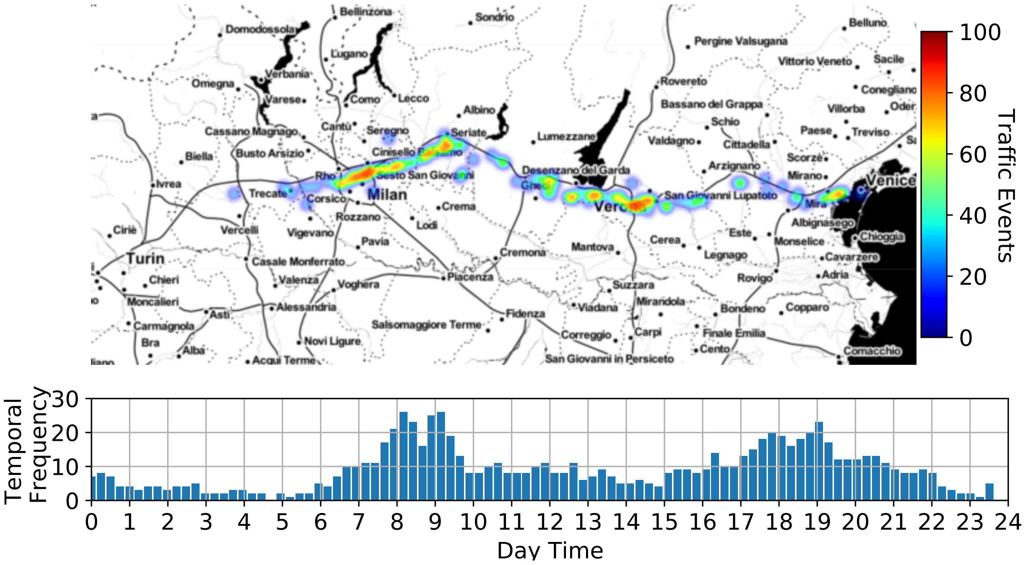


Figure 4.17: *Spatio-Temporal distribution of road events.*

activity or drop the mobility rate gradually for adjacent base stations. We call it *event propagation effect*. It usually depends on the severity of the road event, the day time, the specific location and the network deployment (e.g., base station density, etc.).

From the auxiliary information contained in our event dataset, we make a straightforward association between the location of each event and the closest base station, thereby identifying the source of the road event in the network. Then, by analyzing the propagation of anomalous traffic patterns onto nearby cells, we can infer which direction of the highway, or vehicular traffic flow, has been affected.

An example of this is shown in Fig. 4.18. The scenario accounts for a major accident occurred nearby Verona at $9{:}00$ AM. The red line in each plot represents the regular behaviour of the corresponding metric, while the blue line depicts the daily data. After the accident, we can notice a clear deviation from regular patterns for multiple metrics lasting a few hours. The higher number of users in RRC connected state indicates that multiple users are actively using mobile network resources, suggesting the presence of traffic congestions. Similarly, this affects DL volumes and average Physical Resource Block (PRB) utilization. Moreover, a higher number of affected base stations on the east side suggests that the accident occurred in east-west direction (Venice-Turin). Given the high variability and intrinsic characteristic of each monitoring metric, an accurate anomalous pattern characterization would involve time, space and metric-specific considerations. Therefore, road events can hardly be detected by simple anomaly detection algorithms, e.g., those based on outlier detection that may run on a single base station. This motivates us to investigate advanced deep learning solutions capable of dealing with such multidimensional information matrix while keeping a global view of the system.

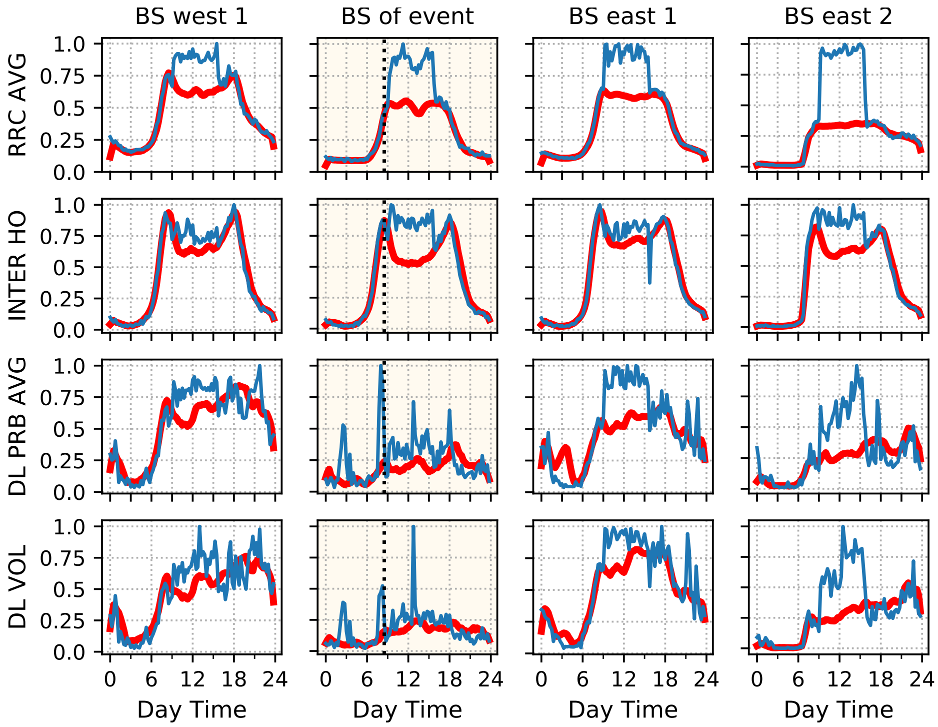


Figure 4.18: *Event propagation on nearby base stations in both directions for selected metrics.*

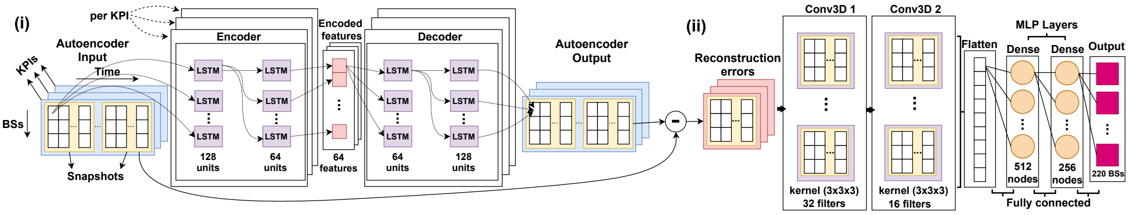


Figure 4.19: π*-ROAD model architecture*

4.2.3 Design and Model Formulation

Hereafter, we introduce the design of π-ROAD, a deep learning-based framework to detect road events. An overview of the model architecture is depicted in Fig. 4.19. The overall solution requires two different stages to deal with road event predictions: i) an autoencoder-based approach consisting of Long Short-Term Memory Recurrent Neural Networks (LSTM) layers to detect anomalies within highly dynamic and heterogeneous time patterns and ii) a Deep-Learning approach to approximate complex functions, e.g., those that relate road events occurrences with temporal and spatial distributions of anomalies detected in adjacent base stations. The overall 6 months dataset has been split according to a $60/20/20$ ratio for the purposes of training, validation, and testing procedures, respectively.

Input

Let us consider a time-slotted system whereby time is divided into time intervals $t = \{1, \ldots, \mathcal{T}\}$, and define $\mathcal{N}$ as the set of base stations deployed along the highway. As detailed in Section 4.2.2, each base station $n \in \mathcal{N}$ gathers information about a multitude of heterogeneous network statistics, defined as $\mathcal{M}$, which include, among the others, PRB utilization, physical channel quality information, traffic volumes, X2/S1 hand-over statistics, etc. In our experiments, we consider 25 different monitoring metrics.

Let $x_m^n(t)$ be the time series describing the m-th monitoring metric observed under the base station n at the time t. As suggested in [143], an accurate organization of the input traces helps enhancing deep learning models performances. In light of this, for all metrics $m \in \mathcal{M}$ and base stations $n \in \mathcal{N}$, we collect the $x_m^n(t)$ traces and order them by preserving their spatial location (from Turin to Venice) in a matrix $X_m(t)$ with dimensions $N \times L$, representing a metric-specific *snapshot* of the network, where N is the total number of base stations, and L is the number of monitoring samples within the observation period (or *lookback* time window), fixed to 4 hours throughout our experiments. The input matrices $X_m(t)$ are normalized with respect to the maximum values of each m in the training dataset.

LSTM-based anomaly detection

The possibility to detect changes in traffic conditions is subject to the capabilities of our model to correctly identify irregular statistics from the monitoring samples collected along the highway. Given the multidimensionality of our dataset, we build an autoencoder A_m for every collected metric thereby allowing for better scalability. Autoencoders imply the setup of an encoding-decoding architecture. The encoding part, commonly implemented as feed-forward neural networks, provides a compressed representation of the input data to subsequent layers [162]. The decoding phase follows the same steps over a symmetric architecture. Analytically, the two phases applied to the input metric X_m can be described as follows:

$$h_m(X_m(t)) = \eta(W_m^\eta X_m(t) + b_m^\eta), \tag{4.13}$$

$$\hat{X}_m(t) = \delta(W_m^\delta h_m(X_m(t)) + b_m^\delta), \tag{4.14}$$

where η and δ are the encoding and decoding functions with their corresponding weights W_m^η and W_m^δ, b_m^η and b_m^δ are bias vectors, and $h_m(X_m(t))$ and $\hat{X}_m(t)$ are the compressed input and the reconstructed output sequences, respectively.

The performance of autoencoders, defined as their capability to reconstruct the input sequence, depends on variability and complexity of data provided at the input. Differently, in our work we exploit autoencoders for anomaly detection [183] instead of simple dimensionality reduction.

The data analysis performed in Section 4.2.2 suggests the adoption of deep learning techniques able to deal with the spatio-temporal characteristics of mobile traces. For this reason, we implement our encoder-decoder architecture by means of Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs). Each autoencoder A_m accounts for 4 LSTM layers, two for each phase. LSTM is a type of RNN architecture that has proven its value when dealing with repetitive patterns and unstructured time series, while solving a problem of vanishing gradient for long-term dependencies present in other RNNs [157]. QCIs can be represented as a chain of G modules, or cells, each one applying a set of operations to the input data. In our case, the two QCI layers are composed by 128 and 64 cells, respectively [184].

The output of each cell $g \in G$, also known as cell state, is transferred to the subsequent cell in a recursive manner. The possibility of handling long-term trends in QCI is provided by structures, dubbed as gates, which carefully remove or add information to the cell state. Each cell has three gates, namely, input I_g, output O_g and forget gate F_g, which controls the amount of information that should be added (or dropped) to the cell state before transferring it to the next unit [185].

This effect is achieved by combining the influence of different non-linear activation functions, i.e., σ (sigmoid) and $\tanh$ (hyperbolic tangent function), at each gate. The impact of these functions on the input data needs to be learned during the training phase aiming to minimize a loss function [185], which in our case is the mean squared error (MSE). To accomplish the anomaly detection task, we make use of labeled data and train the autoencoders offline exclusively on *eventless* snapshots taken from historical data as to capture the behavior of the system without anomalies. Once trained, we feed the model with online monitoring traces. Given that anomalous patterns have not been part of the training phase, it is expected that the model will exhibit performance degradation during the reconstruction phase. Simple classifiers would mark the input snapshot as anomalous if the reconstruction error exceeds a given threshold. Instead, we derive the squared error matrix $e_m(t) = \|X_m(t) - \hat{X}_m(t)\|^2$ from each autoencoder A_m and combine the individual error matrices into a 3D tensor $e(t) = \{e_1(t), e_m(t), \dots, e_M(t)\}$ which is passed to the second stage of our model.

Road Event Localization

The function linking anomalies detected by the first stage of our model with the occurrence of road events along the highway is unknown and hard to be characterized due to the multitude of system-related and external variables affecting the entire detection process. The simple detection of anomalies in monitoring statistics does not imply the occurrence of an emergency, as uncorrelated events—like hardware failures and/or unexpected traffic peaks—may trigger alerts leading to erroneous estimations. For this reason in the following, we exploit the information described in Section 4.2.2 as ground-truth to train a Convolutional Neural Network (CNN)-based model that captures the spatio-temporal correlation of different anomalies and maps them into geographical information [186]. Our design choice is further motivated by the fact that in case of road events, as shown in Fig. 4.18, affected base stations present significant levels of correlation between each other, which further improves the learning task.

More in details, the second stage of π-ROAD consists of two stacked 3D-CNN layers and a final Multi-Layer Perceptron (MLP) fully connected layer. In order to exploit local correlation from nearby base stations, each neuron of the 3D-CNN layers has a limited *receptive field*, or kernel, whose size determines its observation area. For a given input tensor, convolutions with 3D kernels are iteratively applied to provide the subsequent layers with a compressed representation of the input information. Through extensive hyper-parameters optimization, we used two 3D-CNN layers with filter sizes of 32 and 16 neurons, respectively, and corresponding convolutional kernel sizes of $(3, 3, 3)$ and $(3, 3, 3)$ [187]. Each neuron of the CNN runs a filter $H(\sum_t e(t) * k(t) + b)$, where $e(t)$ is the input tensor at time t, $k(t)$ is the kernel filter, $*$ is the convolution operator, b is a bias, and $H(\cdot)$ is a non-linear activation function, in our case Rectified Linear Unit (ReLU) [188].

In order to map the encoded representation of the anomalies into geographical information about the emergency, we make use of a final Multi-Layer Perceptron layer. MLP is a class of neural networks with fully-connected neurons among layers which has the capabilities to approximate, through supervised learning, the function that relates the input with the output. In our case, we are interested in the function that links the encoded spatio-temporal anomalies of multiple metrics and different base stations with the exact event location. The MLP classifier consists of three layers with 512, 256 and 220 neurons, respectively, where the output layer matches the number of base stations [189]. To regularize the output and reduce over-fitting, we introduce a dropout rate of 0.2. Overall, the second stage of the model is trained using Adam optimizer with learning rate 10^{-4}, and adopting cross-entropy as loss function over 150 training epochs. Due to high system variability caused by user social behaviors and external causes affecting it (e.g., weather conditions), it clearly rises the need to design a model capable of adapting to new (anomalous) patterns. Therefore, we ensure that the model is retrained as soon as new observations are made available.

Road Event Classification

Upon detection, it is important to classify the magnitude of each event to promptly alert first aid responders and, if necessary, identify the set of networking requirements to be allocated for an emergency slice setup. We rely on the impact of the *propagation effect* over the set of base stations near to the road event to provide an empirical classification. Let us introduce $\tilde{\mathcal{N}} \subset \mathcal{N}$ as the set of base stations affected by the road event. Due to the irregular density that characterizes the Radio Access Network (RAN) deployment along the highway, we argue that a simple road event classification based on the cardinality of $\tilde{\mathcal{N}}$ would not be accurate, as road events occurring in the proximity of main cities would involve a wider number of base stations than those occurring in rural areas. Therefore, we define our classification metric μ as $\frac{\tilde{N}}{\psi}$, where $\tilde{N}$ is the cardinality of $\tilde{\mathcal{N}}$, and ψ is the number of base stations deployed in the area within a radius of 10 km surrounding the road event. We empirically select 10 km as this value represents the longest vehicular queue registered in our event dataset. Clearly, ψ should be re-dimensioned to generalize our findings within other network topologies.

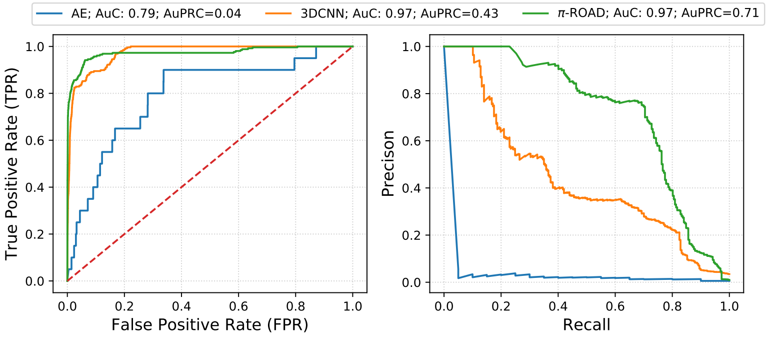


Figure 4.20: *ROC and PR diagrams with indicated AuC and AuPRC scores for different anomaly detection and classification approaches.*

Finally, we differentiate among three different categories of events: *Light*, *Moderate*, and *Severe*. The event duration of each category reflects the most common scenarios contained in our real dataset, while networking throughput requirements are meant to support the provisioning of Augmented and Virtual Reality (AR/VR) streaming in mission-critical scenarios [190]. The details about each road event class and corresponding ENS requirements are summarized in Table 4.2.

Performance Comparison and Practical Considerations

Hereafter, we compare the performance of π-ROAD against state-of-the-art models trained to perform similar anomaly detection and classification tasks, highlighting each model drawback (compared to our solution) through practical considerations. Benchmarks include a simple threshold-based autoencoder classifier (AE) [184], and a more advanced 3D-CNN-based classifier [187, 189].

Fig. 4.20 resumes the overall performances using Receiver Operating Characteristics (ROC) and Precision-Recall (PR) metrics. On the left-hand side, the ROC metric provides a compact representation of the capabilities of the model to deal with the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR).

On the right-hand side, the PR curve states the performance of a classifier in terms of Precision and Recall, where Precision is a measure of result relevancy, and Recall is a measure of how many truly relevant results are returned by the classifier. To ease the comparison, we also quantify the Area under the ROC Curve (AuC) and Area under the PR Curve (AuPRC). The unsupervised approach adopted in AE requires the definition of

Table 4.2: *Event characterization and ENS Requirements.*

Event category	Θ - Time Duration	NS Requirements	μ Value
Light	20min	10Mbps	$(0, 1]$
Moderate	40min	15Mbps	$(1, 2]$
Severe	60min	25Mbps	≥ 2

a-priori thresholds to mark anomalous patterns from the reconstruction errors. Clearly, the definition of these settings depends on the intrinsic variability of the traces in input, and requires considerations over the statistics of each feature. This approach hardly scales when considering multiple heterogeneous metrics, as in our case. Despite accurate tuning of the parameters, it clearly appears how this simple approach fails to reveal the majority of events, achieving the lowest precision score. This further emphasizes how detection schemes based on simple threshold/outlier detection are not enough to address the scenario considered in our work. Conversely, the 3D-CNN classifier adopts a supervised approach, which prevents from specifying user-defined thresholds for decision-making. The resulting ROC and PR curves show better performances when compared against the baseline AE approach. This result can be explained through the ability of 3D-CNN network to capture spatio-temporal correlations between different network measurements [3].

However, when Recall value increases over a certain level (i.e., moving from left to right on the PR diagram), the Precision score drops drastically, suggesting poor performances when differentiating among different types of anomalies. In other words, the model detects only some types of anomalies, e.g., those deriving from major events with very strong impact on monitoring traces, and fails to generalize over probably smaller ones. Conversely, π-ROAD outperforms the two stand-alone approaches in both ROC and PR metrics. The advantage deriving from the two-stage approach adopted by π-ROAD is two-fold $i)$ the initial anomaly detection task performed by first stage of the model, together with an accurate input organization, facilitates the learning task of the 3D-CNN layer and $ii)$ the capabilities of the 3D-CNNs to deal with bare reconstruction errors removes the need to provision user-defined thresholds which may bias the final results.

4.2.4 ENS orchestration

The outcome of π-ROAD can tackle a variety of public safety issues along the highway. For example, in case of road accidents, it is important to enable dynamic RAN resource allocation in order to provide first responder teams with enough communication capabilities, regardless of active user sessions or services. To this aim, in what follows, we introduce our formulation to address the Emergency Network Slice (ENS) orchestration problem.

Radio Access Network. Let us consider a RAN deployment with network slicing support covering the area surrounding the highway and comprising a set of base stations $\mathcal{N}$. Each base station $n \in \mathcal{N}$ is characterized by a capacity C_n, defined as the sum of its available physical resource blocks (PRBs). As result of the road event localization process executed by π-ROAD and described in Section 4.2.3, we identify the subset of base stations $\widetilde{\mathcal{N}} \subset \mathcal{N}$ that will host the emergency slice.

Active slice services. Let us assume a set of network slices $\mathcal{S}$ supporting V2X services, being already installed and active on the considered RAN deployment, whereas let us mark $s = 0$ the Emergency Network Slice (ENS) to be temporarily installed. We assume each V2X slice $s \in \mathcal{S}$ described by means of a predefined network slice template that suggests slice requirements in terms of throughput Δ_s, and latency Λ_s. Typical values for V2X services are shown in Table 4.3.

Table 4.3: *V2X Slice requirements (c.f. [15, 16, 17])*

V2X category	Latency	Data rate	Reliability
Autonomous driving	10ms	10Mbps	99.999%
Tele-operated driving	20ms	25Mbps Sensor Data Streaming	99.999%
Vehicular Internet/Infotainment	>100ms	15Mbps Video Streaming	Not Specified
Road Safety	100ms	1Mbps	99%

Problem design

In real scenarios, when setting up an emergency slice, advanced orchestration operations are required to still guarantee service level agreements (SLAs) of active slices. Assuming that an admission control process has been executed to accept and install V2X slices on the network with the aim of maximizing the resource efficiency (while still honouring expected service requirements), it might appear challenging to add on top of active slices an additional high-priority service, such as ENS. However, an efficient admission control will accommodate slices with heterogeneous requirements to compensate unexpected slice behaviors or traffic peaks [149]. Analytically, we assume that predefined SLAs include for each slice $s \in \mathcal{S}$ and base station $n \in \mathcal{N}$ a minimum number of reserved PRBs, namely $Q_{n,s}^{(t)}$ [7]. Without loss of generality, we assume that the overall resource availability is assigned to the set of running slices.

In addition, we translate the slice latency requirements Λ_s into a tolerance value λ_s that defines the maximum number of time intervals packets shall wait into the queue before being served, or dropped due to lack of resource availability within the latency requirements. Note that if all running V2X slices have high priority, i.e., very low delay tolerance, it might be infeasible to install the ENS without impairing other active services. ENS parameters depend on the severity of the emergency road event which can be obtained as output of a π-ROAD execution. We resume the corresponding settings in Table 4.2, in light of the discussions of Section 4.2.3. We define Θ as the envisioned emergency time duration, and assume, for each ENS to be deployed, a fixed amount of PRBs allocation $Q_{n,0}^{(t)}, \forall t \in \{0, \cdots, \Theta\}$ and the lowest delay tolerance $\lambda_0 = 1$. If not properly scheduled, the additional ENS may lead in the worst case to resource deficit and service disruption in one or multiple base stations. Therefore, we aim at minimizing the instantaneous resource deficit $\pi_n^{(t)}$, while still guaranteeing defined SLAs for other active running slices. Our problem can be formulated as follows:

---

[7] When SLAs disclose information on expected throughput or user rate, the PRB requirements could be obtained by monitoring the traffic demand of the different slices. For further details, we refer the reader to [5, 175].

Problem 8 (π-Orchestrator).

$$
\begin{aligned}
\text{minimize} \quad & \sum_{n\in\tilde{\mathcal{N}}}\sum_{t\in\mathcal{T}} \pi_n^{(t)} \\
\text{subject to} \quad & \sum_{s} z_{n,s}^{(t)} \leq C_n + \pi_n^{(t)}, \qquad \forall n \in \tilde{\mathcal{N}}, t \in \mathcal{T}; \\
& \sum_{t=0}^{\Theta} Q_{n,s}^{(t)} - z_{n,s}^{(t)} \leq \sum_{t=\Theta+1}^{\Theta+\lambda_s} Q_{n,s}^{(t)}, \ \forall s \in \mathcal{S}, n \in \tilde{\mathcal{N}}; \\
& z_{n,0}^{(t)} = Q_{n,0}^{(t)}; \quad \forall n \in \tilde{\mathcal{N}}, t \in \mathcal{T}; \\
& z_{n,s}^{(t)} \in \mathbb{N}^+, \qquad \forall n \in \tilde{\mathcal{N}}, s \in \mathcal{S}, t \in \mathcal{T}; \\
& \pi_n^{(t)} \in \mathbb{N}^+, \qquad \forall n \in \tilde{\mathcal{N}}, t \in \mathcal{T}.
\end{aligned}
$$

where (integer) decision variables are $\pi_n^{(t)}$ and $z_{n,s}^{(t)}$ indicating the number of PRBs to be assigned to slice s on base station n at time interval t. Note that $z_{n,0}^{(t)} = Q_{n,0}^{(t)}$ is due to the highest priority assigned to the ENS ($s = 0$). The first set of constraints introduces flexibility into the problem by adding a non-zero fictitious value (i.e., the resource deficit π_n) to avoid infeasible solutions. The second set of constraints specify that the slice resource reservation must be performed fulfilling the slice delay tolerance, i.e., waiting traffic still in the queue must be scheduled within λ_s time slots. We run Problem 8 for a set of time slots $\mathcal{T}$ that includes the time window Θ required by the ENS. Problem 8 is an Integer Linear Programming (ILP), which can be efficiently approximated by means of relaxation techniques (e.g., [191]), and commercial solvers to provide near-optimal results[8].

Proposition 1. *Problem 8 is NP-Hard and difficult to approximate within $N^{1-\epsilon}$, for $\epsilon > 0$.*

Proof. The NP-Hardness proof goes by reduction. Problem 8 clearly belongs to the NP problems. Let us consider an instance of a bin packing problem (BPP) with arbitrary constants a and B, and binary decision variables x_{ij} and y_j, where $\mathcal{I}$ and $\mathcal{J}$ are the sets of items and bins, respectively [192].

Problem 9 (BPP).

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j} y_j \\
\text{subject to} \quad & \sum_{i} a x_{ij} \leq B y_j, \qquad \forall j \in \mathcal{J}; \\
& \sum_{j} x_{ij} \leq 1, \qquad \forall i \in \mathcal{I}.
\end{aligned}
$$

If we consider $N = 1$ base station, and $z_{n,s}, \pi_n$ as binary decision variables, Problem 8 reduces to a BPP that shows that for all $\epsilon > 0$, packing items within the minimum number

---

[8] The implementation of the problem has been carried out using the framework of `IBM ILOG CPLEX` and its `Python API`.

of bins within $N^{1-\epsilon}$ is NP-hard [192]. Since such a problem is trivial compared to our original Problem 8—that includes a number of base stations $N > 1$ and integer decisions variables—this result is rather strong. $\square$

V2X traffic scheduling

The instantaneous resource deficit $\sum_{n\in\tilde{\mathcal{N}}}\pi_n^{(t)}$ depends on the severity of the detected road event and, in turn, on the specific ENS resource reservation and time duration settings. Therefore, Problem 8 considers the worst case scenario, i.e., when each active slice fully utilizes reserved resources. However, as shown in Section 4.2.2, network congestions rarely occur outside the commuting time periods and some of reserved network resources may be underutilized. So, the impact of the ENS on the system can be further mitigated by relaxing the fixed PRB allocation scheme envisioned to assure slice isolation at RAN scheduling level [2, 29]. Specifically, if a network slice is underutilized, other slices with pending traffic can use some of its network resources (PRBs) [9].

This allows us to devise a practical algorithm that reduces the overall slice resource deficit. In particular, the algorithm sorts slice traffic requests based on the slice priority (or traffic delay tolerance). Slices with higher priority will use assigned PRBs as per the solution of Problem 8 ($z_{n,s}^{(t)}$) within time interval t to serve incoming traffic requests. Once all requests have been served, remaining slice PRBs ($z_{n,s}^{(t)} > 0$) are used to serve traffic requests of the next slice in the priority-ranked list. When all PRBs are used ($\sum_s z_{n,s}^{(t)} = 0$), the algorithm proceeds to the next time interval t keeping unserved traffic requests in the slice queues. If slice traffic requests are not served within the slice delay tolerance λ_s, these are dropped. The resulting resource deficit is further minimized as the RAN slice scheduler efficiently assign resources to slices with pending traffic. Results are shown and discussed in Section 4.2.5.

4.2.5 Performance Evaluation

In the following, we investigate the performance of π-ROAD while detecting road events, and use this information to assess our solution's capabilities in orchestrating radio resources, involving different types of V2X network slices and realistic traffic conditions.

Localization of the road event

In order to evaluate event localization performances, we provide as input the test set of monitoring samples taken from our dataset. In Fig. 4.21, we summarize the results, focusing on the set of base stations and a representative time period. Each point on the map represents a pair of base station and time interval, and its color indicates the detected probability of event occurrence given the monitoring information. Circular markers on the same map highlight the ground-truth location of road accidents. From the picture, it

---

[9] While this concept may fail to comply with the network slice isolation principle, we assume that the isolation is still guaranteed at higher (abstracted) scheduling layers [29].

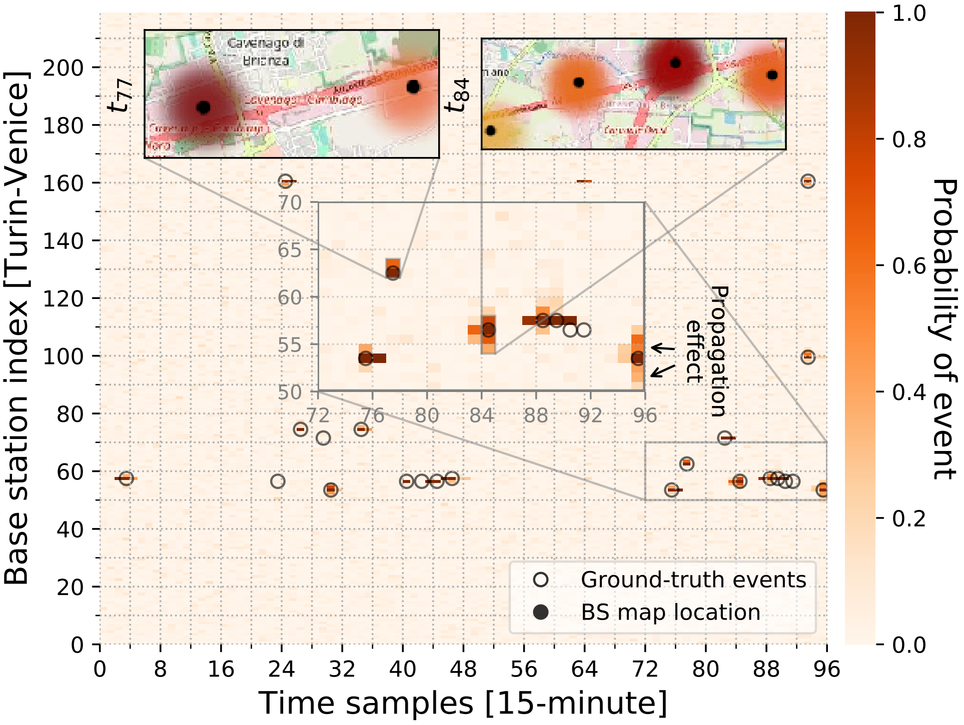


Figure 4.21: *Example of road event detection and localization using π-ROAD model along the highway within one day, compared against the event ground-truth information.*

can be noticed how π-ROAD detects most of the events, not only revealing their temporal duration (x-axis), but also suggesting how much the propagation effect influences adjacent base stations (y-axis). Interestingly, the heatmap also reveals some cases in which π-ROAD detects the occurrence of events before online notification services and social media platforms. For visualization purposes, at the top of the figure, we highlight two geographical areas of the highway covering impacted base stations in two different event occurrences happening at time t_{77} and t_{84}, respectively. In the first case, π-ROAD classifies the road event as *Light* since only two base stations reported anomalous statistics. In the second case, given the wider number of affected eNodeBs, the event is classified as *Moderate*. Finally, few events are not detected. Either this can be due to very limited impact of such events on the monitoring traces or, they may be approximated with patterns that are unknown to the model. In the latter case, we expect that similar events would be detected in the future as the model follows the learn-as-you-go approach to continuously train with the latest data.

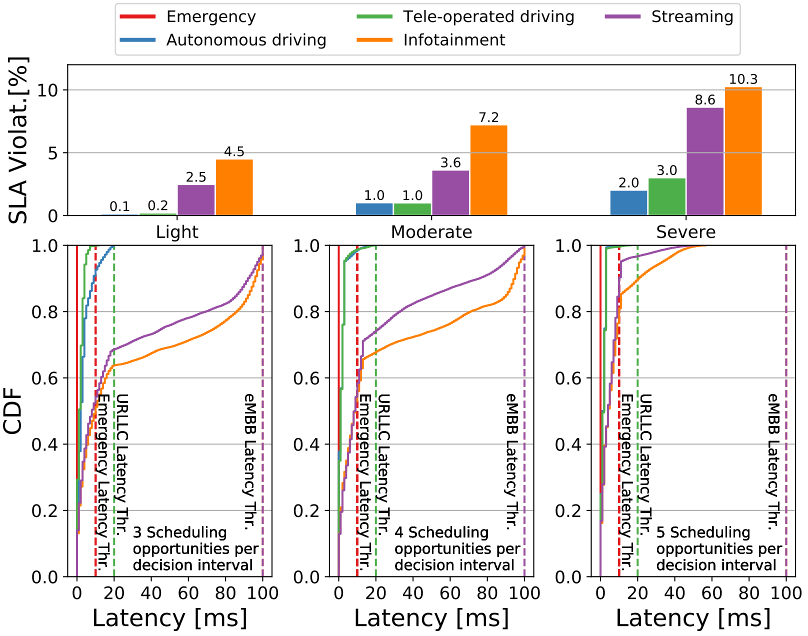
Resource Orchestration

Hereafter, we test our ENS orchestration solution exploiting π-ROAD's outputs and the realistic monitoring information contained in our dataset. We consider a 4-slices network scenario, including two enhanced Mobile Broadband (eMBBs) slices dedicated to streaming and infotainment services (with high delay tolerance), and two Ultra-reliable low-latency communication (URLLC) slices dedicated to Autonomous and Tele-operated driving, with corresponding networking requirements as detailed in Table 4.3. We assume that DL slice traffic volumes are generated under the reference set of base stations in the considered event period, according to the specific service QCI and corresponding traces as depicted in Fig. 4.16. In particular, eMBB slices are mapped to QCIs traces 7 and 8, while URLLCs to QCIs 1 and 6, respectively.

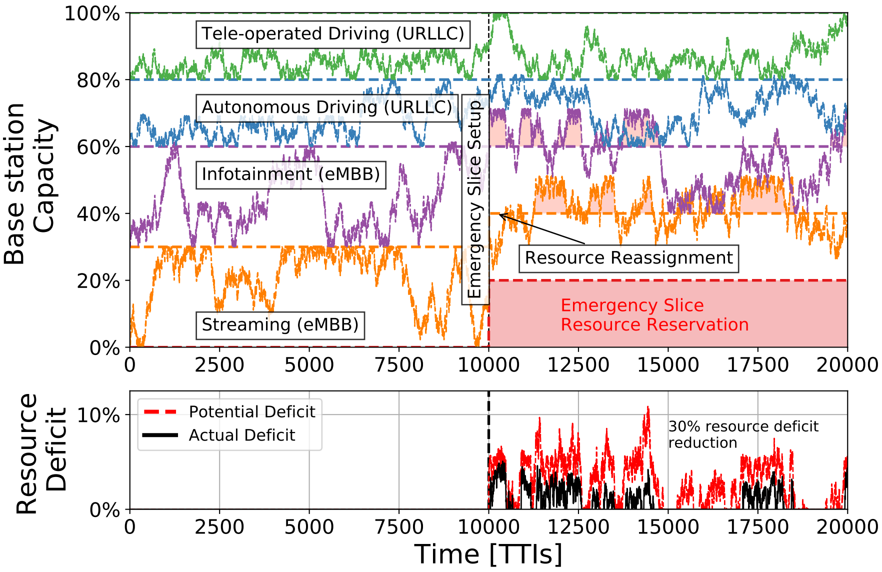
Intuitively, the impact of an ENS installation over the radio access network depends, among the others, by the networking capacity deployed in the area of the road event, instantaneous traffic demands, slice requirements, and user density. In Fig. 4.22a (top) we provide a quantitative measure in case of different road event types and corresponding duration. We consider channel conditions as perceived by the base station during the event, thereby limiting the base station capacity (due to lower assigned MCS) when compared to the ideal scenario. The results of each category are averaged over 10 road event occurrences taken from our real dataset. It can be noticed that eMBB slices unveil worse performance (w.r.t. URLLC) due to their higher throughput requirements and lower slice priority (i.e., higher delay tolerance). However, an advanced slice orchestration solution may alleviate this issue.

Fig. 4.22a (bottom) focuses on the *Moderate* road events evaluating the impact of different slice types (delay tolerance parameters λ_s) with a fixed decision interval t duration of 100 ms [193]. Specifically, π-Orchestrator may suggest different solutions with corresponding slice configurations that translate into variable slice scheduling opportunities per decision interval. Therefore, we evaluate the Cumulative Distribution Functions (CDFs) of traffic latency considering different scheduling opportunities. Despite the resource deficit introduced by the ENS, the average slice latency improves when the number of scheduling opportunities increases. Fig. 4.22b depicts the temporal evolution of the overall effects caused by the ENS slice setup at transmission time interval (TTI) level. All traffic traces, MCS and channel quality values are generated with a millisecond granularity according to their temporal evolution and statistical distributions resulting from our monitoring dataset. The initial scenario accounts for a fixed resource allocation scheme (without any ENS running) which allows satisfying all the different slice requirements, as expected in normal traffic conditions thanks to the adoption of calibrated admission and control algorithms, e.g. [5]. In particular, among the 4 slices populating the system, eMBB slices have a fixed quota of 30% of available resources while URLLCs have 20%, respectively.

After 10 seconds, an ENS is instantiated. Its higher priority implies the execution of the π-Orchestrator to minimize service disruption. From the upper plot, it can be noticed how our solution reduces the quota of resources assigned to non-delay sensitive services which, unavoidably, suffer from a resource deficit (highlighted in red). However, in emergency scenarios the RAN slice scheduler may assign unused reserved resources to active slices with traffic pending to be served, further reducing the resource deficit and, in turn, the SLA violation of certain slices. The lower plot depicts the overall traffic potentially violating the SLAs with a red dashed line, whereas the traffic actually exceeding it with a black



(a) *Resource orchestration at system level.*



(b) *Resource orchestration at base station level.*

Figure 4.22: *Performance evaluation of a 4-slices scenario when an ENS is installed.*

continuous line. Results show how scheduling relaxation would help in reducing the SLA violations up to 30%. Clearly, this result assumes that none of the resources allocated for ENSs can be consumed by other slices, even in absence of emergency traffic.

4.3 Conclusions

In the first part of this chapter, we performed a data analysis of the network dynamics during sport events in the neighbourhood of a football stadium. The mobile infrastructure covers a 1-Km^2 area with 16 base station sectors and is operated by a major European carrier. The events contained traffic peaks generated by about 30.000 people. Based on the insights obtained, we took up on the mass events RAN capacity forecasting challenge and designed ARENA: a model-free deep learning solution that predicts the expected number of connected users during a future event along with the corresponding RAN spectrum capacity (PRBs) needed per sector for providing an average user experience. ARENA takes as input the average number of connected users, downlink traffic volume, downlink throughput and other contextual event information to infer the future traffic loads. Our results show that ARENA $i)$ closely predicts the actual number of connected devices during mass events in time and $ii)$ that it can provide mobile operators guidance on the actual RAN capacity needed during an event to provide a user experience similar to a regular network situation.

In the second part of this chapter, we presented π-ROAD, a deep learning-based solution for the analysis and detection of road events over the highway. The model accounts for two complementary stages $i)$ an autoencoder-based stage to identify anomalous patterns in the temporal evolution of operational mobile network data, and $ii)$ a 3D-CNN-based stage to overcome simple threshold detection schemes and automatically learn the relationship that links multiple metric-specific anomalies to road event occurrences. The output of π-ROAD may be used in a multitude of emergency scenarios. We focused on the design and validation of a slice orchestration solution dealing with the setup of Emergency Network Slice (ENS) in V2X scenarios. Considering real mobile traffic distributions from a major highway in Italy, our results show that the information provided by π-ROAD can significantly reduce, up to 30%, the impact of Emergency Network Slice setup, also decreasing the probability of service disruption on other running network slices.

Chapter 5

Conclusions and Future Work

Network Slicing is widely regarded as a game-changer technology in the upcoming 5G mobile networks and beyond, bringing new players into the mobile ecosystem and enabling novel business models. By means of state-of-the-art technologies and advanced resource orchestration mechanisms taking into account the dynamic nature of the mobile environment, the Infrastructure Provider (InP) can dynamically provide slices of its own physical infrastructure to 3^{rd}-party vertical industries, traditionally alien to the mobile ecosystem, concurrently ensuring the demanded SLAs and maximizing the revenues.

In the first part of this thesis, we identified the main challenges and research problems introduced by the adoption of network slicing paradigm into the mobile network domain, highlighting the need for AI-enabled solutions able to cope with the envisioned multi-tenant and multi-domain scenario. The heterogeneity of networking requirements of modern use-cases drove the design and validation of the proposed algorithms.

Building on these novel concepts, in Sec. 2.1 we presented a hierarchical framework to manage the orchestration of network slices in an end-to-end fashion, i.e., including for radio access, transport network, and distributed computing infrastructure, jointly considering the admission control of incoming network slice requests, as well as their dynamic resource management according to real-time traffic demands. Our experimental implementation of the overall platform, composed by real networking devices, assesses the capabilities of our approach considering a heterogeneous set of network slices. Additionally, in Sec. 2.2 we considered the blockchain technology, and investigated its applicability into the network slicing resource brokerage scenario. To prove the generality of our approach, we developed a Proof-of-Concept implementation leveraging on the open source Hyperledger platform. Our results showed that our approach is feasible and scalable even in large settings including up to 1000 tenants simultaneously active in acquiring, exchanging and managing network and computational slice resources.

In the second part of this thesis, we focus on domain-specific solutions to refine the orchestration and management of mobile network resources. In particular, in Sec. 3.1 we have first designed a MEC-specific solution for multi-tenant platform administration, introducing the concept of the MEC broker as an entity exposing administration and management capabilities to heterogeneous tenants sharing a common MEC platform. Our analysis shows significant benefits provided by the introduction of advanced resource allocation mechanisms into the slice management, enabling costs savings while providing ad-hoc solutions for external tenants willing to place their services over edge computing systems. Then, in Sec. 3.2, we designed a solution for RAN resource orchestration, focusing our efforts on those slice types characterized by stringent latency requirements. Our results derived from an implementation with off-the-shelf hardware show that LACO is able to guarantee, without prior knowledge of the wireless channel conditions, strict slice latency requirements at affordable computational costs.

In the third part of this thesis, we provided an analysis of RAN monitoring metrics taken from operational mobile networks owned by major European operators, evaluating the advantages that machine learning based approaches can bring to mobile network resource orchestration and anticipatory resource forecasting. In Sec. 4.1, we addressed football events and studied how crowded scenario affects the mobile infrastructure, as well as its resource utilization. Inspired by recent advances in deep learning for image processing, ARENA exploits convolutional neural-networks to evaluate base station-level traffic demand (in the form of periodic snapshots) to provide an indication of the radio resources that the operator would need to deploy as to accommodate traffic loads with throughput performances comparable to those in standard working conditions. Finally in Sec. 4.2, we rely on RAN measurements taken from a wide geographical area to investigate vehicular traffic and emergency network slice scenarios. In particular, π-ROAD is a deep learning-based solution for the analysis and detection of road events over the highway. Building on state-of-the-art autoencoders and convolutional neural-networks models, π-ROAD exploit passive RAN monitoring statistics to identify anomalous patterns in the temporal evolution of operational mobile network data, and automatically link RAN metric anomalies to road event occurrences, therefore allowing for proactive Emergency Network Slices setup and corresponding resource allocation.

The research conducted in this thesis provides a first step towards the integration of Artificial Intelligence-based solutions in the management of next generation mobile networks' networking resources. Future research directions aimed at extending the results disclosed in this work are briefly described in the following.

1. We have showcased the benefits that AI-based solutions can bring into the network slicing ecosystem by means of realistic testbed implementations. Nevertheless, most of the proposed solution require centralized controllers, which may introduce performance bottlenecks in real deployments. Future extensions of the proposed works can investigate distributed approaches, as well as the resulting need for real-time information exchange to assist local decision-making processes.

2. AI-based approaches proved their effectiveness when dealing with heterogeneous problems related to network slicing statistics, including large traffic volumes, high end-users' mobility, and different (generally large) timescales. Nevertheless, the dynamics of the mobile network ecosystem poses stringent delay requirements into the

resource allocation process. In fact, the QoS experienced by the end-users depends on the instantaneous resource orchestrations decisions, which further exacerbate the complexity of the managing entities. In this context, extensive work is still necessary to investigate and better characterize the additional overhead introduced by machine learning-based solutions in mobile network management. This should be compared against the potential benefits in critical scenarios, specially those involving short-time scale decision-making processes.

Bibliography

[1] L. Zanzi, F. Cirillo, V. Sciancalepore, F. Giust, X. Costa-Pérez, S. Mangiante, and G. Klas. Evolving Multi-Access Edge Computing to Support Enhanced IoT Deployments. *IEEE Communications Standards Magazine*, 3(2):26–34, Sept. 2019.

[2] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, H. D. Schotten, and X. Costa-Pérez. LACO: A Latency-Driven Network Slicing Orchestration in Beyond-5G Networks. *IEEE Transactions on Wireless Communications*, 20(1):667–682, Oct. 2021.

[3] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, X. Costa-Pérez, G. Agapiou, and H. D. Schotten. ARENA: A Data-Driven Radio Access Networks Analysis of Football Events. *IEEE Transactions on Network and Service Management*, 17(4):2634–2647, Oct. 2020.

[4] L. Zanzi, F. Giust, and V. Sciancalepore. M^{2}EC: A multi-tenant resource orchestration in multi-access edge computing systems. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, Apr. 2018.

[5] J. Xavier Salvat, L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore, and X. Costa-Pérez. Overbooking Network Slices Through Yield-driven End-to-end Orchestration. In *Proceedings of the 14th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pages 353–365, Nov. 2018.

[6] L. Zanzi, A. Albanese, V. Sciancalepore, and X. Costa-Pérez. NSBchain: A Secure Blockchain Framework for Network Slicing Brokerage. In *IEEE International Conference on Communications (ICC)*, pages 1–7, Jun. 2020.

[7] A. Okic, L. Zanzi, V. Sciancalepore, A. Redondi, and X. Costa-Pérez. π-ROAD: a Learn-as-You-Go Framework for On-Demand Emergency Slices in V2X Scenarios. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pages 1–10, May 2021.

[8] L. Zanzi and V. Sciancalepore. On Guaranteeing End-to-End Network Slice Latency Constraints in 5G Networks. In *International Symposium on Wireless Communication Systems (ISWCS)*, pages 1–6, Aug. 2018.

[9] X. Costa-Pérez, V. Sciancalepore, L. Zanzi, and A.Albanese. Blockchain for Mobile Networks. In Anwer Al-Dulaimi, Octavia Dobre, and Chih-Lin I, editors, *Blockchains: Empowering Technologies and Industrial Applications*. IEEE/Wiley, to appear in 2021.

[10] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, and X. Costa-Pérez. Overbooking Network Slices End-to-End: Implementation and Demonstration. In *ACM SIGCOMM '18 (Poster and Demo Session)*, Aug. 2018.

[11] L. Zanzi, J. Xavier Salvat, V. Sciancalepore, A. Garcia-Saavedra, and X. Costa-Pérez. OVNES: Demonstrating 5G network slicing overbooking on real deployments. In *IEEE INFOCOM - IEEE Conference on Computer Communications Workshops*, pages 1–2, Apr. 2018.

[12] L. Zanzi, J. X. Salvat, V. Sciancalepore, A. Garcia-Saavedra, and X. Costa-Pérez. Latency-driven Network Slices Orchestration. In *IEEE INFOCOM - IEEE Conference on Computer Communications Workshops*, pages 965–966, Apr. 2019.

[13] L. Zanzi F. Giust, V. Sciancalepore. System and Method to Support Network Slicing in an MEC System Providing Automatic Conflict Resolution Arising from Multiple Tenancy in the MEC Environment, U.S. Patent App. 17/102,515, Mar. 2021.

[14] V. Sciancalepore, L. Zanzi, A. Albanese, and X. Costa-Pérez. Multi-Resource and Autonomous Hierarchical Brokering Platform to Enable Slice Resource Exchange Among Heterogeneous Network Tenants, U.S. Patent App. 16/819,218, Sept. 2021.

[15] Third Generation Partnership Project (3GPP). Study on enhancement of 3GPP Support for 5G V2X Services (Release 16) . 3GPP TR 22.886 V14.0.0, Dec. 2018.

[16] Third Generation Partnership Project (3GPP). Feasibility Study on New Services and Markets Technology Enablers, Stage 1 (Release 14) . 3GPP TR 22.891 V14.2.0, Sep. 2016.

[17] J. Mei, X. Wang, and K. Zheng. Intelligent Network Slicing for V2X Services Toward 5G. *IEEE Network*, 33(6):196–204, 2019.

[18] Third Generation Partnership Project (3GPP). System Architecture for the 5G System (Release 15). 3GPP TS 23.501 v15.2.0, June 2018.

[19] X. Li, R. Ni, J. Chen, Y. Lyu, Z. Rong, and R. Du. End-to-End Network Slicing in Radio Access Network, Transport Network and Core Network Domains. *IEEE Access*, 8:29525–29537, Feb. 2020.

[20] V. Sciancalepore, L. Zanzi, X. Costa-Pérez, and A. Capone. ONETS: Online Network Slice Broker From Theory to Practice. *IEEE Transactions on Wireless Communications*, pages 1–1, Jul. 2021.

[21] Y. J. Liu, G. Feng, Y. Sun, S. Qin, and Y. C. Liang. Device Association for RAN Slicing Based on Hybrid Federated Deep Reinforcement Learning. *IEEE Transactions on Vehicular Technology*, 69(12):15731–15745, Oct. 2020.

[22] J. G. Andrews et al. What Will 5G Be? *IEEE Journal on Selected Areas in Communications*, 32(6):1065–1082, Jun. 2014.

[23] Z. Guan and T. Melodia. The Value of Cooperation: Minimizing User Costs in Multi-Broker Mobile Cloud Computing Networks. *IEEE Transactions on Cloud Computing*, 5(4):780–791, Oct 2017.

[24] B. Chen et al. Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges. *IEEE Access*, 2017.

[25] A. Zanella et al. Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, 1(1):22–32, Feb 2014.

[26] GSMA. Smart 5G Networks: enabled by network slicing and tailored to customers' needs, Sep. 2017.

[27] NGMN Alliance. Description of Network Slicing Concept. *White Paper*, 2016.

[28] X. Li et al. 5G-Crosshaul Network Slicing: Enabling Multi-Tenancy in Mobile Transport Networks. *IEEE Communications Magazine*, 55(8):128–137, 2017.

[29] X. Foukas, M. K. Marina, and K. Kontovasilis. Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture. In *ACM International Conference on Mobile Computing and Networking*, Oct. 2017.

[30] S Agarwal et al. Joint VNF Placement and CPU Allocation in 5G. In *IEEE INFOCOM*, Apr. 2018.

[31] F. Le et al. Understanding the Performance and Bottlenecks of Cloud-Routed Overlay Networks: A Case Study. CAN '16, pages 7–12, New York, NY, USA, 2016. ACM.

[32] I. Maki et al. Performance analysis and improvement of tcp proxy mechanism in tcp overlay networks. In *IEEE ICC '05*, 2005.

[33] F. Le, E. Nahum, V. Pappas, M. Touma, and D. Verma. Experiences Deploying a Transparent Split TCP Middlebox and the Implications for NFV. In *ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, Hot-Middlebox '15, pages 31–36, 2015.

[34] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, C. Ziemlicki, and Z. Smoreda. Not All Apps Are Created Equal: Analysis of Spatiotemporal Heterogeneity in Nationwide Mobile Service Usage. In *ACM CoNEXT '17*, 2017.

[35] 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 *ACM International Conference on Mobile Computing and Networking*, MobiCom '18, pages 191–206, Oct. 2018.

[36] C. Zhang and P. Patras. Long-term mobile traffic forecasting using deep spatio-temporal neural networks. In *ACM Mobihoc '18*, Mobihoc '18, 2018.

[37] Feng Qian et al. Periodic transfers in mobile applications: Network-wide origin, impact, and optimization. In *Proceedings of WWW '12*, pages 51–60, 2012.

[38] J. W. Taylor. Exponentially weighted methods for forecasting intraday time series with multiple seasonal cycles. *International Journal of Forecasting*, 2010.

[39] A. Kavanagh. OpenStack as the API framework for NFV: the benefits, and the extensions needed. *Ericsson Review*, 2, 2015.

[40] J. Subramanian et al. Airline yield management with overbooking, cancellations, and no-shows. *Transportation science*, pages 147–167, 1999.

[41] J. Huang et al. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. *SIGCOMM Comput. Commun. Rev.*, 43(4):363–374, Aug. 2013.

[42] Haim Mendelson and Seungjin Whang. Optimal incentive-compatible priority pricing for the M/M/1 queue. *Operations research*, 38(5):870–883, 1990.

[43] J. Kuo, *et al.* Service chain embedding with maximum flow in software defined network and application to the next-generation cellular network architecture. In *IEEE INFOCOM*, May 2017.

[44] A. Garcia-Saavedra, G. Iosifidis, X. Costa-Pérez, and D. J. Leith. Joint Optimization of Edge Computing Architectures and Radio Access Networks. *IEEE Journal on Selected Areas in Communications*, 36(11):2433–2443, Nov. 2018.

[45] S. K. Singh et al. A survey on internet multipath routing and provisioning. *IEEE Communications Surveys Tutorials*, 17(4):2157–2175, Fourthquarter 2015.

[46] A. Garcia-Saavedra, M. Karzand, and D. J. Leith. Low Delay Random Linear Coding and Scheduling Over Multiple Interfaces. *IEEE Transactions on Mobile Computing*, Nov 2017.

[47] P. C. Chu and J.E. Beasley. A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics*, 4:63–86, Jun. 1998.

[48] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Num. Mathematik*, 4(1):238–252, Dec 1962.

[49] R. Nauss. The 0 1 knapsack problem with multiple choice constraints . *European Journal of Operational Research*, 2(2):125–131, 1978.

[50] R. E. Korf. A new algorithm for optimal bin packing. In *Eighteenth National Conference on Artificial Intelligence*, pages 731–736. American Association for Artificial Intelligence, 2002.

[51] OpenEPC. `http://www.openepc.com/`.

[52] Samsung Galaxy S7. `http://www.samsung.com/global/galaxy/galaxy-s7/`.

[53] NEC PFlow. `http://www.nec.com/en/global/prod/pflow/pf5240.html`.

[54] NEC MB4420. `https://uk.nec.com/en_GB/global/solutions/nsp/sc2/`.

[55] HP DL380p Gen8 server. `https://www.hpe.com/h20195/v2/default.aspx?cc=za&lc=en&oid=5177957`.

[56] D. Xenakis et al. Mobility Management for Femtocells in LTE-Advanced: Key Aspects and Survey of Handover Decision Algorithms. *IEEE Communications Surveys Tutorials*, 16(1):64–91, July 2014.

[57] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Pérez. Multi-Tenant Radio Access Network Slicing: Statistical Multiplexing of Spatial Loads. *IEEE/ACM Transactions on Networking*, 25(5):3044–3058, Oct. 2017.

[58] K. Samdanis, X. Costa-Pérez, and V. Sciancalepore. From network sharing to multitenancy: The 5G network slice broker. *IEEE Communications Magazine*, 54(7):32–39, Jul. 2016.

[59] Hyperledger - Open Source Blockchain Technologies, Oct. 2019.

[60] 3GPP. Management and orchestration; Provisioning. TS 28.531, 3rd Generation Partnership Project (3GPP), Sep. 2019.

[61] J. Backman, S. Yrjölä, K. Valtanen, and O. Mämmelä. Blockchain Network Slice Broker in 5G: Slice leasing in factory of the future use case. In *Internet of Things Business Models, Users, and Networks*, Nov. 2017.

[62] D. B. Rawat and A. Alshaikhi. Leveraging Distributed Blockchain-based Scheme for Wireless Network Virtualization with Security and QoS Constraints. In *International Conference on Computing, Networking and Communications (ICNC)*, pages 332–336, Mar. 2018.

[63] V. Ortega, F. Bouchmal, and J. F. Monserrat. Trusted 5G Vehicular Networks: Blockchains and Content-Centric Networking. *IEEE Vehicular Technology Magazine*, 13(2):121–127, Jun. 2018.

[64] G. A. F. Rebello et al. Providing a sliced, secure, and isolated software infrastructure of virtual functions through blockchain technology. In *IEEE 20th International Conference on High Performance Switching and Routing (HPSR)*, May 2019.

[65] Arthur D. Little. Opportunities for Telecom Operators to Facilitate New Business Ecosystems. White Paper, Jun. 2017.

[66] T. Neudecker and H. Hartenstein. Network Layer Aspects of Permissionless Blockchains. *IEEE Communications Surveys Tutorials*, 21(1):838–857, Sep. 2019.

[67] G. A. F. Rebello, I. D. Alvarenga, I. J. Sanz, and O. C. M. B. Duarte. BSec-NFVO: A blockchain-based security for network function virtualization orchestration. In *IEEE International Conference on Communications (ICC)*, May 2019.

[68] L. S. Sankar et al. Survey of consensus protocols on blockchain applications. In *International Conference on Advanced Computing and Communication Systems (ICACCS)*, Jan. 2017.

[69] A. de Vries. Bitcoin's Growing Energy Problem. *Joule*, 2(5):801–805, May 2018.

[70] L. Liang, Y. Wu, G. Feng, X. Jian, and Y. Jia. Online Auction-Based Resource Allocation for Service-Oriented Network Slicing. *IEEE Transactions on Vehicular Technology*, pages 8063–8074, Aug. 2019.

[71] M. Jiang, M. Condoluci, and T. Mahmoodi. Network slicing in 5G: An auction-based model. In *IEEE International Conference on Communications (ICC)*, pages 1–6, May 2017.

[72] L. Luu, D. Chu, H. Olickel, P. Saxena, and A. Hobor. Making Smart Contracts Smarter. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 254–269, 2016.

[73] C. Lin et al. HomeChain: A Blockchain-Based Secure Mutual Authentication System for Smart Homes. *IEEE Internet of Things Journal*, pages 1–12, Sep. 2019.

[74] IBM CPLEX OPL. `https://www.ibm.com/products/ilog-cplex-optimization-studio`.

[75] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A Secure Sharding Protocol For Open Blockchains. In *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[76] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186, 1999.

[77] J. Kreps, N. Narkhede, and J. Rao. Kafka: a Distributed Messaging System for Log Processing. In *International Workshop on Networking Meets Databases (NetDB)*, 2011.

[78] D. Ongaro and J. Ousterhout. In Search of an Understandable Consensus Algorithm. In *Proceedings of the USENIX Annual Technical Conference*, pages 305–320, Jun. 2014.

[79] E. Androulaki et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In *Proceedings of the ACM Thirteenth EuroSys Conference*, pages 1–15, Apr. 2018.

[80] P. Thakkar, S. Nathan, and B. Viswanathan. Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform. In *IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 264–276, Sep. 2018.

[81] ETSI MEC ISG. Mobile Edge Computing (MEC); Framework and reference architecture. DGS MEC 003, ETSI, April 2016.

[82] P. Rost *et al.* Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks. *IEEE Communications Magazine*, May 2017.

[83] ETSI MEC ISG. Mobile Edge Computing (MEC); Mobile Edge Platform Application Enablement. DGS MEC 011, ETSI, July 2017.

[84] V. Sciancalepore, K. Samdanis, X. Costa-Pérez, D. Bega, M. Gramaglia, and A. Banchs. Mobile traffic forecasting for maximizing 5G network slicing resource utilization. In *IEEE Conference on Computer Communications - INFOCOM'17*, May 2017.

[85] R. Cohen, *et al.* Near optimal placement of virtual network functions. In *IEEE INFOCOM*, pages 1346–1354, Apr. 2015.

[86] N. M. Calcavecchia, O. Biran, E. Hadad, and Y. Moatti. VM placement strategies for cloud scenarios. In *5th International Conference on Cloud Computing*, pages 852 – 859. IEEE, Dec. 2012.

[87] L/ Chen and H. Shen. Consolidating Complementary VMs with Spatial/Temporal-awareness in Cloud Datacenters. In *IEEE INFOCOM*, pages 1033–1041, Apr. 2014.

[88] F. Ben Jemaa, G. Pujolle, and M. Pariente. QoS-aware VNF Placement Optimization in Edge-Central Carrier Cloud Architecture. In *Global Communications Conference (GLOBECOM)*. IEEE, Dec. 2016.

[89] M. F. Bari et al. Orchestrating virtualized network functions. *IEEE Transactions on Network and Service Management*, 2016.

[90] ETSI MEC ISG. Mobile Edge Computing (MEC); Mobile Edge Management; Part 1: System, host and platform management. DGS MEC 010-1, ETSI, 2017.

[91] ETSI MEC ISG. Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management. DGS MEC 010-2, ETSI, July 2017.

[92] ETSI MEC ISG. Mobile Edge Computing (MEC); Radio Network Information API. DGS MEC 012, ETSI, July 2017.

[93] ETSI MEC ISG. Mobile Edge Computing (MEC); Location API. DGS MEC 013, ETSI, July 2017.

[94] W. Qiu, Z. Qian, S. Lu. Multi-objective virtual machine consolidation. In *10th International Conference on Cloud Computing*, pages 1346–1354. IEEE, Apr. 2017.

[95] Z. Huang and D. H. Tsang. SLA guaranteed virtual machine consolidation for computing clouds. In *International Conference in Communications (ICC)*, pages 1314––1319. IEEE, 2012.

[96] Griva I., Nash S. G., and Sofer A. *Linear and Nonlinear Optimization*. Society for Industrial Mathematics, 2009.

[97] S. Knight *et al.* The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[98] 5GPPP Alliance. 5G Innovations for New Business Opportunities. White Paper, Mar. 2017.

[99] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, and X. Costa-Pérez. A Machine Learning Approach to 5G Infrastructure Market Optimization. *IEEE Transactions on Mobile Computing*, 19(3):498–512, Mar. 2020.

[100] H. J. Lee, J. Flinn, and B. Tonshal. RAVEN: Improving Interactive Latency for the Connected Car. In *ACM International Conference on Mobile Computing and Networking*, MobiCom '18, pages 557–572, 2018.

[101] G. Sharma, R. R. Mazumdar, and N. B. Shroff. On the Complexity of Scheduling in Wireless Networks. In *ACM International Conference on Mobile Computing and Networking*, MobiCom '06, pages 227–238, Sept. 2006.

[102] S. Lee, I. Pefkianakis, A. Meyerson, S. Xu, and S. Lu. Proportional Fair Frequency-Domain Packet Scheduling for 3GPP LTE Uplink. In *IEEE INFOCOM'09*, pages 2611–2615, Apr. 2009.

[103] A. Rostami, P. Öhlén, M. A. S. Santos, and A. Vidal. Multi-Domain Orchestration Across RAN and Transport for 5G. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 613–614, Aug. 2016.

[104] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra. A Game Theoretic Approach for Distributed Resource Allocation and Orchestration of Softwarized Networks. *IEEE Journal on Selected Areas in Communications*, 35(3):721–735, Mar. 2017.

[105] L. Diez et al. LaSR: A Supple Multi-Connectivity Scheduler for Multi-RAT OFDMA Systems. *IEEE Transactions on Mobile Computing*, 19(3):624–639, Mar. 2020.

[106] I. da Silva, G. Mildh, A. Kaloxylos, P. Spapis, E. Buracchini, A. Trogolo, G. Zimmermann, and N. Bayer. Impact of Network Slicing on 5G Radio Access Networks. In *IEEE European Conference on Networks and Communications (EuCNC)*, pages 153–157, Jun. 2016.

[107] C. Y. Chang, N. Nikaein, and T. Spyropoulos. Radio Access Network Resource Slicing for Flexible Service Execution. In *IEEE Conference on Computer Communications - Workshops*, pages 668–673, Apr. 2018.

[108] T. Guo and A. Suárez. Enabling 5G RAN Slicing With EDF Slice Scheduling. *IEEE Transactions on Vehicular Technology*, 68(3):2865–2877, Jan. 2019.

[109] S. Bubeck and N. Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. Foundations and Trends in Machine Learning, 2012.

[110] A. Mahajan and D. Teneketzis. *Multi-Armed Bandit Problems*, pages 121–151. Springer US, Boston, MA, 2008.

[111] V. Kuleshov and D. Precup. Algorithms for the Multi-Armed Bandit Problem. *Journal of Machine Learning Research*, 1:1–48, Oct. 2000.

[112] Mingyan L. Cem T. Online Algorithms for the Multi-Armed Bandit Problem with Markovian Rewards. In *Proceedings of the 48th Annual Allerton Conference*, volume 2, Sep. 2010.

[113] S. Agrawal and N. Goyal. Near-Optimal Regret Bounds for Thompson Sampling. *Journal of the ACM*, 64(5):30.1–30.24, Sep. 2017.

[114] D. Yun, A. Proutiere, S. Ahn, J. Shin, and Y. Yi. Multi-armed Bandit with Additional Observations. *Proceedings Measurement and Analysis of Computing Systems*, pages 13/1–22, Apr. 2018.

[115] T. Stahlbuhk, B. Shrader, and E. Modiano. Learning Algorithms for Scheduling in Wireless Networks with Unknown Channel Statistics. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Mobihoc'18, pages 31–40, 2018.

[116] Daniel N. Hill, Houssam Nassif, Yi Liu, Anand Iyer, and S.V.N. Vishwanathan. An Efficient Bandit Algorithm for Realtime Multivariate Optimization. In *ACM SIGKDD*, KDD '17, pages 1813–1821, Sept. 2017.

[117] O. Besbes, Y. Gur, and A. Zeevi. Stochastic Multi-Armed-Bandit Problem with Non-Stationary Rewards. In *Conference on Neural Information Processing Systems*, pages 199–207. ACM, Dec. 2014.

[118] Third Generation Partnership Project (3GPP). Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures. 3GPP TS 36.213 V15.1.0, Mar. 2018.

[119] A. Ksentini, P. A. Frangoudis, A. PC, and N. Nikaein. Providing Low Latency Guarantees for Slicing-Ready 5G Systems via Two-Level MAC Scheduling. *IEEE Network*, 32(6):116–123, 2018.

[120] Aleksandar Antonić, Martina Marjanović, and Ivana Podnar Žarko. Modeling Aggregate Input Load of Interoperable Smart City Services. In *ACM International Conference on Distributed and Event-based Systems*, pages 34–43. ACM, Jun. 2017.

[121] R. Munos and A. Moore. Influence and Variance of a Markov Chain: Application to Adaptive Discretization in Optimal Control. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 2, pages 1464–1469, Sept. 1999.

[122] M. Zambianco and G. Verticale. Interference Minimization in 5G Physical-Layer Network Slicing. *IEEE Transactions on Communications*, pages 1–11, 2020.

[123] S. D'Oro, F. Restuccia, A. Talamonti, and T. Melodia. The Slice Is Served: Enforcing Radio Access Network Slicing in Virtualized 5G Systems. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 442–450, 2019.

[124] H. Shen Wang and N. Moayeri. Finite-state Markov Channel - A Useful Model for Radio Communication Channels. *IEEE Transactions on Vehicular Technology*, 44(1):163–171, Feb. 1995.

[125] T. Hofmann. Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Journal on Machine Learning*, 42:177–196, Jan. 2001.

[126] R. Howard. Dynamic Programming and Markov Processes. MIT Press, Cambridge, MA, 1960.

[127] J. Vermorel and M. Mohri. Multi-armed Bandit Algorithms and Empirical Evaluation. In *Proceedings of the 16th European Conference on Machine Learning*, pages 437–448, 2005.

[128] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time Analysis of the Multi-armed Bandit Problem. *Machine Learning*, 47(2-3):235–256, May 2002.

[129] Third Generation Partnership Project (3GPP). Technical Specification Group Radio Access Network; NR; Physical channels and modulation. 3GPP TS 38.211 V16.1.0, Mar. 2020.

[130] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith. srsLTE: An Open-source Platform for LTE Evolution and Experimentation. In *Proceedings of the 10th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, WiNTECH '16, pages 25–32, Oct. 2016.

[131] Third Generation Partnership Project (3GPP). Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification. 3GPP TS 36.321 V13.2.0, Jun. 2016.

[132] A. Abdallah, A. B. MacKenzie, V. Marojevic, J. Kalliovaara, R. Bacchus, A. Riaz, D. Roberson, H. Juhani, and R. Ekman. Detecting the Impact of Human Mega-events on Spectrum Usage. In *13th IEEE Annual Consumer Communications Networking Conference - CCNC'16*, pages 523–529, Jan. 2016.

[133] L. Rabieekenari, K. Sayrafian, and J. S. Baras. Autonomous relocation strategies for cells on wheels in environments with prohibited areas. In *IEEE International Conference on Communications - ICC'17*, May 2017.

[134] K. Sakai, M. Sun, and W. Ku. Data-Intensive Routing in Delay-Tolerant Networks. In *IEEE Conference on Computer Communications - INFOCOM'19*, pages 2440–2448, Apr. 2019.

[135] C. Xu, M. Wang, X. Chen, L. Zhong, and L. A. Grieco. Optimal Information Centric Caching in 5G Device-to-Device Communications. *IEEE Transactions on Mobile Computing*, 17(9):2114–2126, 2018.

[136] S. D'Oro, F. Restuccia, A. Talamonti, and T. Melodia. The Slice Is Served: Enforcing Radio Access Network Slicing in Virtualized 5G Systems. In *IEEE Conference on Computer Communications - INFOCOM'19*, pages 442–450, Apr. 2019.

[137] M. Morrison, M. Bell, and M. Chalmers. Visualisation of Spectator Activity at Stadium Events. In *International Conference on Information Visualisation*, pages 220–226, Jul. 2009.

[138] J. Erman and K.K. Ramakrishnan. Understanding the Super-sized traffic of the Super Bowl. In *Proceedings of the 2013 Conference on Internet Measurement Conference - ACM IMC'13*, pages 353–360, 2013.

[139] P. Castagno, V. Mancuso, M. Sereno, and M. A. Marsan. Why Your Smartphone Doesn't Work in Very Crowded Environments. In *IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9, Jun. 2017.

[140] A. S. Khatouni, M. Mellia, M. A. Marsan, S. Alfredsson, J. Karlsson, A. Brunstrom, O. Alay, A. Lutu, C. Midoglu, and V. Mancuso. Speedtest-Like Measurements

in 3G/4G Networks: The MONROE Experience. In *29th International Teletraffic Congress (ITC 29)*, volume 1, pages 169–177, 2017.

[141] A. M. Mandalari et al. Experience: Implications of Roaming in Europe. In *International Conference on Mobile Computing and Networking*, MobiCom '18, pages 179—189, New York, NY, USA, 2018. Association for Computing Machinery.

[142] D. Zhang, J. Huang, Y. Li, F. Zhang, C. Xu, and T. He. Exploring Human Mobility with Multi-Source Data at Extremely Large Metropolitan Scales. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking - ACM MobiCom '14*, pages 201—-212, Sept. 2014.

[143] X. Wang, Z. Zhou, F. Xiao, K. Xing, Z. Yang, Y. Liu, and C. Peng. Spatio-Temporal Analysis and Prediction of Cellular Traffic in Metropolis. *IEEE Transactions on Mobile Computing*, 18(9):2190–2202, Sep. 2019.

[144] X. Wang, Z. Zhou, Y. Zhao, X. Zhang, K. Xing, F. Xiao, Z. Yang, and Y. Liu. Improving Urban Crowd Flow Prediction on Flexible Region Partition. *IEEE Transactions on Mobile Computing*, pages 1–15, Aug. 2019.

[145] F. Xu, Y. Li, H. Wang, P. Zhang, and D. Jin. Understanding Mobile Traffic Patterns of Large Scale Cellular Towers in Urban Environment. *IEEE/ACM Transactions on Networking*, 25(2):1147–1161, Apr. 2017.

[146] 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 Conference on Computer Communications - INFOCOM'17*, May 2017.

[147] C. Zhang and P. Patras. Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Mobihoc '18, pages 231–240, 2018.

[148] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang. Urban Traffic Prediction from Spatio-Temporal Data Using Deep Meta Learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1720—-1730, 2019.

[149] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez. DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning. In *IEEE Conference on Computer Communications - INFOCOM'19*, pages 280–288, Apr. 2019.

[150] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang. A First Look at Cellular Network Performance during Crowded Events. In *ACM SIGMETRICS '13*, pages 17–28, Jun. 2013.

[151] Third Generation Partnership Project (3GPP). Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC) Protocol specification. 3GPP TS 36.331 V15.4.0, Feb. 2019.

[152] S. Sesia, I. Toufik, and M. Baker. *LTE, The UMTS Long Term Evolution: From Theory to Practice*. Wiley Publishing, 2009.

[153] V. Sciancalepore, X. Costa-Pérez, and A. Banchs. RL-NSB: Reinforcement Learning-Based 5G Network Slice Broker. *IEEE/ACM Transactions on Networking*, 27(4):1543–1557, Aug. 2019.

[154] T. Chen, Q. Ling, and G. B. Giannakis. An Online Convex Optimization Approach to Proactive Network Resource Allocation. *IEEE Transactions on Signal Processing*, 65(24):6350–6364, Dec. 2017.

[155] A. D. Flaxman, A. T. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of ACM SODA*, pages 385–394, Jan. 2005.

[156] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, May 2015.

[157] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997.

[158] Xiaolei Ma, Zhuang Dai, Zhengbing He, and Yunpeng Wang. Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction. In *Sensors*, 2017.

[159] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *International Conference on Neural Information Processing Systems - NIPS'15*, volume 1, pages 802—-810, Dec. 2015.

[160] S. Vani and T. V. M. Rao. An Experimental Approach towards the Performance Assessment of Various Optimizers on Convolutional Neural Network. In *International Conference on Trends in Electronics and Informatics - ICOEI'19*, Oct. 2019.

[161] O. Ozyegen, S. Mohammadjafari, E. Kavurmacioglu, J. Maidens, and A. B. Bener. Experimental Results on the Impact of Memory in Neural Networks for Spectrum Prediction in Land Mobile Radio Bands. *IEEE Transactions on Cognitive Communications and Networking*, 6(2):771–782, Jun. 2020.

[162] B. Liao, J. Zhang, C. Wu, D. Mcllwraith, T. Chen, S. Yang, Y. Guo, and F. Wu. Deep Sequence Learning with Auxiliary Information for Traffic Prediction. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 537–546, Jun. 2018.

[163] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[164] H. K. Vydana and A. K. Vuppala. Investigative Study of Various Activation Functions for Speech Recognition. In *National Conference on Communications - NCC'17*, Mar. 2017.

[165] G. Cybenko. Approximations by Superpositions of Sigmoidal Functions. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, Mar. 1989.

[166] G. Khodabandelou, V. Gauthier, M. Fiore, and M. A. El Yacoubi. Estimation of Static and Dynamic Urban Populations with Mobile Network Metadata. *IEEE Transactions on Mobile Computing*, 18(9):2034–2047, Sept. 2019.

[167] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker. Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks. *IEEE Communications Magazine*, 55(5):72–79, May 2017.

[168] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira. Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges. *IEEE Communications Magazine*, 55(5):80–87, May 2017.

[169] World Health Organization, Jul. 2020.

[170] K. Donoughe, J. Whitestone, and H. Gabler. Analysis of Firetruck Crashes and Associated Firefighter Injuries in the United States. In *Annals of advances in automotive medicine - Annual Scientific Conference*, volume 56, pages 69–76. Association for the Advancement of Automotive Medicine, Oct. 2012.

[171] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang. The Learning and Prediction of Application-Level Traffic Data in Cellular Networks. *IEEE Transactions on Wireless Communications*, 16(6):3899–3912, Jun. 2017.

[172] J. Zhang, Y. Zheng, J. Sun, and D. Qi. Flow Prediction in Spatio-Temporal Networks Based on Multitask Deep Learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(3):468–478, Mar. 2020.

[173] R. M. Khanafer, B. Solana, J. Triola, R. Barco, L. Moltsen, Z. Altman, and P. Lázaro. Automated Diagnosis for UMTS Networks Using Bayesian Network Approach. *IEEE/ACM Transactions on Vehicular Technology*, 57(4):2451–3058, Oct. 2008.

[174] F. Xu, Y. Lin, J. Huang, D. Wu, H. Shi, J. Song, and Y. Li. Big Data Driven Mobile Traffic Understanding and Forecasting: A Time Series Approach. *IEEE Transactions on Services Computing*, 9(5):796–805, Aug. 2016.

[175] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez. AZTEC: Anticipatory Capacity Allocation for Zero-Touch Network Slicing. In *IEEE Conference on Computer Communications - INFOCOM'20*, pages 280–288, Jul. 2020.

[176] A. Essien, I. Petrounias, P. Sampaio, and S. Sampaio. A Deep-learning Model for Urban Traffic Flow Prediction with Traffic Events Mined from Twitter. *World Wide Web*, 19(23):1573–1413, Mar. 2020.

[177] C. Ide, F. Hadiji, L. Habel, A. Molina, T. Zaksek, M. Schreckenberg, K. Kersting, and C. Wietfeld. LTE Connectivity and Vehicular Traffic Prediction Based on Machine Learning Approaches. In *IEEE Vehicular Technology Conference (VTC-Fall)*, pages 1–5, Sept. 2015.

[178] Wenhui Ji, Zhilong Lu, and Tongyu Zhu. Mining Spatial-Temporal Travel Patterns from Highway Transaction Data. In *ACM Symposium on Computer Science and Intelligent Control*, Sept. 2018.

[179] A. Okic, A. Redondi, I. Galimberti, F. Foglia, and L. Venturini. Analyzing different mobile applications in time and space: a city-wide scenario. In *IEEE Wireless Communications and Networking Conference*, pages 1–6, 2019.

[180] A. Furno, M. Fiore, R. Stanica, C. Ziemlicki, and Z. Smoreda. A Tale of Ten Cities: Characterizing Signatures of Mobile Traffic in Urban Areas. *IEEE Transactions on Mobile Computing*, 16(10):2682–2696, Oct. 2017.

[181] Third Generation Partnership Project (3GPP). Technical Specification Group Services and System Aspects; Policy and charging control architecture (Rel. 16). 3GPP TS 22.203 V16.2.0, Dec. 2019.

[182] C. E. Andrade, S. D. Byers, V. Gopalakrishnan, E. Halepovic, D. J. Poole, L. K. Tran, and C. T. Volinsky. Connected Cars in Cellular Network: A Measurement Study. In *ACM Internet Measurement Conference - IMC'17*, page 235–241, Nov. 2017.

[183] S. Tariq, S. Lee, Y. Shin, M. S. Lee, O. Jung, D. Chung, and S. S. Woo. Detecting Anomalies in Space Using Multivariate Convolutional LSTM with Mixtures of Probabilistic PCA. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 2123–2133, Jul. 2019.

[184] H. D. Trinh, E. Zeydan, L. Giupponi, and P. Dini. Detecting Mobile Traffic Anomalies Through Physical Control Channel Fingerprinting: A Deep Semi-Supervised Approach. *IEEE Access*, 7:152187–152201, Oct. 2019.

[185] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai. What to Do Next: Modeling User Behaviors by Time-LSTM. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 3602–3608, 2017.

[186] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, Jan. 2013.

[187] Y. Yang, Z. Hu, K. Bian, and L. Song. ImgSensingNet: UAV vision guided aerial-ground air quality sensing system. In *IEEE Conference on Computer Communications - INFOCOM'19*, pages 1207–1215, Jul. 2019.

[188] A. Zappone, M. Di Renzo, and M. Debbah. Wireless Networks Design in the Era of Deep Learning: Model-based, AI-based, or both? *IEEE Transactions on Communications*, 67(10):7331–7376, Jun. 2019.

[189] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury. ORACLE: Optimized radio classification through convolutional neural networks. In *IEEE Conference on Computer Communications - INFOCOM'19*, pages 370–378, Jul. 2019.

[190] J. Pérez-Romero, I. Vilà, O. Sallent, B. Blanco, A. Sanchoyerto, R. Solozábal, and F. Liberal. Supporting Mission Critical Services through Radio Access Network Slicing. In *International Conference on Information and Communication Technologies for Disaster Management*, pages 1–8, Dec. 2019.

[191] Weihua Wang, P. B. Luh, J. H. Yan, and G. A. Stern. An Improved Lagrangian Relaxation Method for Discrete Optimization Applications. In *IEEE International Conference on Automation Science and Engineering*, pages 359–364, Aug. 2008.

[192] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., USA, 1990.

[193] X. Ge. Ultra-Reliable Low-Latency Communications in Autonomous Vehicular Networks. *IEEE Transactions on Vehicular Technology*, 68(5):5005–5016, Mar. 2019.