NFV Orchestration in Edge and Fog Scenarios

by
Jorge Martín Pérez

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Telematics Engineering

Universidad Carlos III de Madrid

Advisor:
Dr. Carlos J. Bernardos

October 2021

NFV Orchestration in Edge and Fog Scenarios

by
Jorge Martín Pérez

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Telematics Engineering

Universidad Carlos III de Madrid

Advisor:
Dr. Carlos J. Bernardos

October 2021

Quien mucho abarca, poco aprieta.
— refranero popular español

Acknowledgements

After my years as a PhD student I only have words of gratitude for every colleague and professional I had the chance to work with.

First of all, thanks to my supervisor Carlos Jesús, who gave me the opportunity of doing this PhD. I owe him the trust and support that he has always put on everything I have done, the experience of collaborating in top european research projects, and doing an internship at POLITO with Francesco Malandrino, and Carla Fabiana Chiasserini. My gratitude to both of them during my months in Torino, thanks to their patience and dedication I have learned optimization tricks and tools that I have been using throughout my thesis.

Second, I want to also thank my PhD colleagues for the moments shared all over these years. Starting from Luca Cominardi, who was like a second supervisor for me during my first years, and from whom I have learned a lot about networking thanks to his patience and assistance. I also want to thank my colleagues from the second floor, for the moments shared, their warmness and support in the good and the bad during lunch time, breaks, and other great moments I will preciously keep in my memories. Thank you Nuria, Kiril, Milan, Guimarães, Sergio, Patricia, Donato, Ginés, Marco, Iñaki, Stefano, Pelayo, Winnie, Óscar, Luis Félix and Cristina. No me olvido de los compañeros con los que he compartido despacho estos años: Antonio, Borja, José, y Víctor. Los cuatro han sido como una familia para mi, y he aprendido mucho de ellos tanto a nivel profesional como personal. Tengo la certeza de que hacer el doctorado habría sido más duro de no ser por su alegría.

En tercer lugar, quiero dar las gracias a mis amigos de Alcorcón y Jarafuel. Todos ellos me han ayudado a desconectar de la investigación, muchas veces teniendo que tirar de mi. Gracias por vuestro amor, compañía, y los buenos momentos durante estos años.

Y para concluir estos agradecimientos, quiero dar las gracias a mi familia por su apoyo y amor. Para mi son el ejemplo de esfuerzo en el que pienso cada día. A ellos les debo todo.

Published and submitted content

In compliance with the principles referring to plagiarism in Law 14/2011 and in the Code of Good Practices of the UC3M Doctoral School, I hereby report a bibliography of articles or other contributions I have (co)authored that are included as part of the thesis and that have been published or submitted for publication.

Published content

Martín-Pérez, J. and Carlos J. Bernardos. 'Multi-Domain VNF Mapping Algorithms'. In: *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. 2018, pages 1–6. DOI: `10.1109/BMSB.2018.8436765`

- This work is wholly included and its content is reported in Chapter 4;
- The author did all the modelling, heuristics design and implementation in this work;
- The material from this source included in this thesis is not singled out with typographic means and references.

Martín-Pérez, Jorge, Luca Cominardi, Carlos J. Bernardos, Antonio de la Oliva and Arturo Azcorra. 'Modeling Mobile Edge Computing Deployments for Low Latency Multimedia Services'. In: *IEEE Transactions on Broadcasting* 65.2 (2019), pages 464–474. DOI: `10.1109/TBC.2019.2901406`

- This work is wholly included and its content is reported in Chapter 3;
- The author did the design of the algorithm, and the theoretical analysis of the Poisson Point Process (PPP)s used to generate the Base Station (BS)s;
- The material from this source included in this thesis is not singled out with typographic means and references.

Martín-Pérez, Jorge, Luca Cominardi, Carlos J. Bernardos and Alain Mourad. '5GEN: A tool to generate 5G infrastructure graphs'. In: *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*. 2019, pages 1–4. DOI: `10.1109/CSCN.2019.8931334`

- This work is partly included and its content is reported in Chapter 3;
- The author did the implementation of the associated R package, so as the implementation, and evaluation of the placement algorithms reported to validate the generated graphs;
- The material from this source included in this thesis is not singled out with typographic means and references.

Contreras, Luis M. and Cominardi, Luca and **Martín Pérez, Jorge** and Bernardos, Carlos J. 'Applicability of SDN and NFV Techniques for a Virtualization-Based Roaming Solution'. In: *Journal of Network and Systems Management* 28 (July 2020), pages 576–604. DOI: `10.1007/s10922-020-09534-z`. URL: `https://doi.org/10.1007/s10922-020-09534-z`

- This work is partly included and its content is reported in Chapter 4;
- The author was encharged of the Multi Domain Orchestrator (MdO) deployment in the experiments done to measure the termination and creation time of the GiLAN network service, so as its integration with the Openstack Domain Orchestration. Moreover, the measurements reported were done by the author.
- The material from this source included in this thesis is not singled out with typographic means and references.

Balazs Nemeth, Nuria Molner, **Martín-Pérez, Jorge**, Carlos J. Bernardos, Antonio de la Oliva and Balazs Sonkoly. 'Delay and reliability-constrained VNF placement on mobile and volatile 5G infrastructure'. In: *IEEE Transactions on Mobile Computing* (2021), pages 1–1. DOI: `10.1109/TMC.2021.3055426`

- This work is wholly included and its content is reported in Chapter 5;
- The author did, in collaboration with Dr. Balázs Németh and Nuria Molner, the optimization problem formulation. Additionally, the author did the generation of the graph infrastructures used in the simulations, collaborated with the aforementioned co-authors in the implementation of the AMPL system model, and implemented & integrated the state of the art FMC algorithm for comparison.
- The material from this source included in this thesis is not singled out with typographic means and references.

Submitted content

Jorge Martín-Pérez, Kiril Antevski, Andres Garcia-Saavedra, Xi Li and Carlos J. Bernardos. 'DQN Dynamic Pricing and Revenue driven Service Federation Strategy'. In: (March 2020). Submitted

- This work is wholly included and its content is reported in Chapter 4;
- The role of the author of this thesis in the included work relates to the formulation of both the associated optimization and Markov Decision Process (MDP) problems, so as the implementation of the Deep Q-Network (DQN) agent and QtEx. Moreover, the experimentation was also done and implemented by the author.
- The material from this source included in this thesis is not singled out with typographic means and references;

Jorge Martín-Pérez, Koteswararao Kondepu, Danny De Vleeschauwer, Venkatarami Reddy, Carlos Guimarães, Andrea Sgambelluri, Luca Valcarenghi, Chrysa Papagianni and Carlos J. Bernardos. 'Dimensioning of Vehicle-to-Network (V2N) Services in 5G Networks through Forecast-based Scaling'. In: (2021). Submitted. arXiv: `2105.12527 [cs.NI]`

- This work is wholly included and its content is reported in Chapter 6;
- The role of the author of this thesis in the included work relates to the collection of the dataset, its cleanup, so as the implementation of LSTM and GRU forecasting techniques. Additionally, the author did the experimental comparison of every time-series technique, designed and implemented the proposed $M/M/c$ scaling solution.
- The material from this source included in this thesis is not singled out with typographic means and references;

Jorge Martín-Pérez, Francesco Malandrino, Carla Fabiana Chiasserini, Milan Groshev and Carlos J. Bernardos. 'KPI Guarantees in Network Slicing'. In: *IEEE/ACM Transactions on Networking* (2021). Accepted with minor changes

- This work is wholly included and its content is reported in Chapter 5;
- The role of the author of this thesis in the included work relates to the same tasks done in the former work [Mar+20], so as the integration of OKpi with the RoS testbed developed in collaboration with Milan Groshev. In particular, the author did the interface to interact with RoS, and the installation of OpenWrt and VXLANs used for the testbed experiment.
- The material from this source included in this thesis is not singled out with typographic means and references;

Other research merits

This chapter first provides a list of additional publications I have (co)authored which are related to this thesis but not included therein. Next, it provides an overview of the various EU-funded projects I was involved in throughout the lifetime of this thesis as well as my role and participation. Finally, it highlights my participation as editor of a book chapter.

Related publications and submissions

A. Sgambelluri, F. Tusa, M. Gharbaoui, E. Maini, L. Toka, **J. M. Perez**, F. Paolucci, B. Martini, W. Y. Poe, J. Melian Hernandes, A. Muhammed, A. Ramos, O. G. de Dios, B. Sonkoly, P. Monti, I. Vaishnavi, C. J. Bernardos and R. Szabo. 'Orchestration of Network Services across multiple operators: The 5G Exchange prototype'. In: *2017 European Conference on Networks and Communications (EuCNC)*. 2017, pages 1–5. DOI: `10.1109/EuCNC.2017.7980666`

- The role of the author of this thesis was to assist on the experimentation and illustration of both the instantiation time, and distribution of the Multi Domain Orchestrator (MdO) deployment time;
- The material from this source is not included in this thesis.

A. Muhammad, A. Sgambelluri, O. Dugeon, **Martin-Perez, J.**, F. Paolucci, O. G. De Dios, F. Ubaldi, T. Pepe, C. J. Bernardos and P. Monti. 'On the Scalability of Connectivity Services in a Multi-Operator Orchestrator Sandbox'. In: *2018 Optical Fiber Communications Conference and Exposition (OFC)*. 2018, pages 1–3

- The role of the author of this thesis was to perform the TADS robustness experiments;
- The material from this source is not included in this thesis.

Claudio Casetti, Carla Fabiana Chiasserini, Nuria Molner, **Martín-Pérez, Jorge**, Thomas Deiß, Cao-Thanh Phan, Farouk Messaoudi, Giada Landi and Juan Brenes Baranzano. 'Arbitration Among Vertical Services'. In: *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2018, pages 153–157. DOI: `10.1109/PIMRC.2018.8580852`

- The author, in cooperation with Nuria Molner, did the modelling of the arbitration resource assignment;
- The material from this source is not included in this thesis.

K. Antevski, **Martín-Pérez, J.**, Nuria Molner, C. F. Chiasserini, F. Malandrino, P. Frangoudis, A. Ksentini, X. Li, J. SalvatLozano, R. Martínez, I. Pascual, J. Mangues-Bafalluy, J. Baranda, B. Martini and M. Gharbaoui. 'Resource Orchestration of 5G Transport Networks for Vertical Industries'. In: *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2018, pages 158–163. DOI: `10.1109/PIMRC.2018.8581029`

- The author, in cooperation with Dr. Francesco Malandrino, and Dr. Carla Fabiana Chiasserieni; did the design of the clustering algorithm. Additionally, the author implemented such algorithm and the experimentation comparison with the other algorithms studied in the paper.
- The material from this source is not included in this thesis.

A. Sgambelluri, O. Dugeon, A. Muhammad, **J. Martín-Pérez**, F. Ubaldi, K. Sevilla, O. G. De Dios, T. Pepe, C. J. Bernardos, P. Monti and F. Paolucci. 'Orchestrating QoS-based Connectivity Services in a Multi-Operator Sandbox'. In: *J. Opt. Commun. Netw.* 11.2 (February 2019), A196–A208. DOI: `10.1364/JOCN.11.00A196`. URL: `http://jocn.osa.org/abstract.cfm?URI=jocn-11-2-A196`

- The role of the author of this thesis was to perform the TADS robustness experiments;
- The material from this source is not included in this thesis.

J. Mangues-Bafalluy, J. Baranda, I. Pascual, R. Martínez, L. Vettori, G. Landi, A. Zurita, D. Salama, K. Antevski, **J. Martín-Pérez**, D. Andrushko, K. Tomakh, B. Martini, X. Li and J. X. Salvat. '5G-TRANSFORMER Service Orchestrator: design, implementation, and evaluation'. In: *2019 European Conference on Networks and Communications (EuCNC)*. 2019, pages 31–36. DOI: `10.1109/EuCNC.2019.8802038`

- The author, collaborated in the writing to explain the how the placement algorithms API (developed by the author) interacted with the 5G-TRANSFORMER platform;
- The material from this source is not included in this thesis.

K. Antevski, **J. Martín-Pérez**, A. Garcia-Saavedra, C. J. Bernardos, X. Li, J. Baranda, J. Mangues-Bafalluy, R. Martnez and L. Vettori. 'A Q-learning strategy for federation of 5G services'. In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 2020, pages 1–6. DOI: `10.1109/ICC40277.2020.9149082`

- The author, did the design of the optimization problem, so as the implementation of the simulator. Additionally, the author assessed the experimental comparison in between the optimal, q-learning, and greedy solutions.
- The material from this source is not included in this thesis.

Milan Groshev, **Jorge Martín-Pérez**, Kiril Antevski, Antonio de la Oliva and Carlos J. Bernardos. 'COTORRA: COntext-aware Testbed fOR Robotic Applications'. In: *MobiSys2021: the 19th ACM International Conference on Mobile Systems, Applications, and Services*. Accepted in the 1st Workshop on Serverless mobile networking for 6G Communications. Association for Computing Machinery, 2021. arXiv: `2101.07676 [cs.RO]`

- This work is not included in the present thesis;
- The role of the author of this thesis in the included work relates to the mathematical formulation of the proposed system, so as the deployment of the OpenWrt routers, the VXLAN, artificial delay, and integration of OKpi as a COTORRA plug-in;

Milan Groshev, Carlos Guimarães, **Jorge Martín-Pérez** and Antonio de la Oliva. 'Towards Intelligent Cyber-Physical Systems: Digital Twin meets Artificial Intelligence'. In: *IEEE Communications Magazine* (2021). Accepted with major review

– This work is not included in the present thesis;
– The role of the author of this thesis in the included work relates to the implementation of the time-series forecasting algorithms for the robotic arm movement prediction, so as the experimental comparison of them using a dataset that the author previously cleaned.

Danny de Vleeschauwer, Jorge Baranda, Josep Mangues-Bafalluy, Carla Fabiana Chiasserini, Marco Malinverno, Corrado Puligheddu, Lina Magoula, **Jorge Martín-Pérez**, Sokratis Barmpounakis, Koteswararao Kondepu, Luca Valcarenghi, Xi Li, Chrysa Papagianni and Andres Garcia-Saavedra. '5Growth Data-driven AI-based Scaling'. In: *2021 European Conference on Networks and Communications (EuCNC)*. Accepted for publication. 2021

– This work is not included in the present thesis;
– The role of the author of this thesis in the included work relates to the collection and clean-up of the Torino vehicular dataset, the collaboration in the implementation of the original environment designed by Dr. Danny de Vleeschauwer, and the implementation of the Q-learning scaling agent.

Participation and role in collaborative international projects

5G Exchange

The *5G Exchange* (5GEx) project (2015-2018) was an EU-funded project (H2020-ICT-2014-2 grant agreement 671636) with the goal of enabling cross-domain orchestration of services over multiple administrations or over multi-domain single administrations. This would allow end-to-end network and service elements to mix in multi-vendor, heterogeneous technology and resource environments. To that end, 5GEx aimed at enabling collaboration between operators with the view of introducing unification via NFV/SDN compatible multi-domain orchestration based on open source software tools and extensions that can be utilized outside the scope of 5GEx.

The role and the activities of the author of this thesis in the project are the following:

– Actively participating in the project since September 2016 to June 2018;
– deployment of the 5GEx MdO for the GiLAN experiment – see Section 4.1;
– deployment of the 5GEx MdO in UC3M facilities to belong to the project sandbox;
– integration of the IPsec tunnels to interconnect the different partners' facilities within the project sandbox; and
– actively involved in the measurements of deployment times of both the 5GEx platform, and the developed network services as ATOS virtual Content Delivery Network (CDN).

5G-TRANSFORMER

The *5G-TRANSFORMER* project (2017-2019) is an EU-funded project (H2020-ICT-2016-2 grant agreement 761536) aiming at transforming today's mobile transport network into an SDN/NFV-based Mobile Transport and Computing Platform (MTP), which brings the *network slicing* paradigm into mobile transport networks by provisioning and managing MTP slices tailored to the specific needs of vertical industries. The technical approach of the project is twofold: (*i*) enable vertical industries to meet their service requirements within customised MTP slices, and (*ii*) aggregate and federate transport networking and computing fabric, from the edge all the way to the core and cloud, to create and manage MTP slices throughout a federated virtualised infrastructure.

The role and the activities of the author of this thesis in the project are the following:

– Actively participating in the project since July 2017 to 30 November 2019;
– development of the initial versions of the IFA-related Network Service Descriptor (NSD)s;

– design of the initial and theoretical Arbitrator strategy;
– implementation and design of the placement algorithms Application Programming Interface (API) to interact with the platform resource orchestrator; and
– implementation of the clustering placement algorithm, and integration with the resource orchestrator.

5G-CORAL

The *5G-CORAL* project (2017-2019) is an EU-Taiwan project (H2020-ICT-2016-2 grant agreement 761586) leveraging on the pervasiveness of edge and fog computing in the Radio Access Network (RAN) to create a unique opportunity for access convergence. This is envisioned by the means of an integrated and virtualised networking and computing solution where virtualised functions, context-aware services, and user and third-party applications are blended together to offer enhanced connectivity and better quality of experience. The proposed solution contemplates two major building blocks, namely (*i*) the Edge and Fog computing System (EFS) subsuming all the edge and fog computing substrate offered as a shared hosting environment for virtualised functions, services, and applications; and (*ii*) the Orchestration and Control System (OCS) responsible for managing and controlling the EFS, including its interworking with other (non-EFS) domains.

The role and the activities of the author of this thesis in the project are the following:
– Side participation in the project since September 2017 to August 2019; and
– development and integration of placement algorithms reported in [Mar+19a].

5Growth

The vision of the 5Growth project is to empower verticals industries such as Industry 4.0, Transportation, and Energy with an AI-driven Automated and Sharable 5G End-to-End Solution that will allow these industries to achieve simultaneously their respective key performance targets. Towards this vision, 5Growth will automate the process for supporting diverse industry verticals through (i) a vertical portal in charge of interfacing verticals with the 5G End-to-End platforms, receiving their service requests and building the respective network slices on top, (ii) closed-loop automation and SLA control for vertical services lifecycle management and (iii) AI-driven end-to-end network solutions to jointly optimize Access, Transport, Core and Cloud, Edge and Fog resources, across multiple technologies and domains. The main objective of 5Growth is the technical and business validation of 5G technologies from the verticals' points of view, following a field-trial-based approach on vertical sites (TRL 6-7). Multiple use cases of vertical industries (Comau, Efacec_S, Efacec_E, Innovalia) will be field-trialed on four vertical-owned sites in close collaboration with the vendors (Ericsson, Interdigital, NEC, Nokia) and the operators (Altice, Telecom Italia, Telefonica) in the project. 5Growth will leverage on the results of 5G-PPP Phase 2 projects where slicing, virtualization and multi-domain solutions for the creation and provisioning of vertical services are being developed and validated, e.g. 5G-TRANSFORMER and 5G-MONARCH. Two ICT-17-2018 5G End-to-End platforms, 5G EVE and 5G-VINNI, have been selected for the Trials to demonstrate the 5Growth specific vertical use cases. In addition to the impact on vertical-oriented standards (e.g., EN50126 (IEC62278) for railway signaling), the verticals in the consortium will be offered an opportunity to influence ongoing 5G standardization by leveraging the involvement of leading experts in the various relevant SDOs.

The role and the activities of the author of this thesis in the project are the following:
– Side participation in the project since June 2019 to present; and
– actively involved in the forecasting and scaling innovations, which led to the research papers [Jor+21a] and [Vle+21];

Books

Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning

The book is currently in publication process, and will be published by IEEE/Wiley.

- Editor of chapter 4 – *Self-Managed 5G Networks*

Open source contributions

AMPLPY

`AMPLPY` is a python interface to interact with the AMPL programming language, and it was used to feed the optimization solvers with the parameters related to the network infrastructure graphs described in chapter 3. During the development of the thesis the following pull requests have been accepted:

- Pull request #24
  - **Title:** Indexed sets print for exportData().
  - **Description:** currently exportData() does not support dumping the indexed sets. Such functionality is included in this pull request, and the idea comes from issue: #23.
  - **URL:** `https://github.com/ampl/amplpy/pull/24`
- Pull request #25
  - **Title:** Format set index and values in exportData().
  - **Description:** this pull request formats properly the indexes and values of indexed sets, i.e., it uses format_entry. This was not tackled at: #24.
  - **URL:** `https://github.com/ampl/amplpy/pull/25`

networkx

`networkx` is a python library used to work with graph structures, and it implements graph operations as shortest path functions. The library was used in the work done in this thesis to implement every Network Function Virtualization (NFV) orchestration algorithm of chapters 4 and 5, so as the generation of the network infrastructure graphs generated in section 3.

When looking for paths in multigraphs, `networkx` did not return the edge index in between 2 nodes, thus, being impossible to differentiate among edges starting and ending at the same nodes. This was necessary for the implementation of the OKpi NFV orchestration algorithm presented in Section 5.1.

- Pull request #3358
  - **Title:** index edges for multi graph simple paths.
  - **Description:** modifies _all_simple_paths_multigraph() to return the traversed edges rather than the nodes. The returned edges contain their associated keys, so user can distinguish across edges connecting same pair of nodes.
  - **URL:** `https://github.com/networkx/networkx/pull/3358`

Abstract

Current network infrastructures handle a diverse range of network services such as video on demand services, video-conferences, social networks, educational systems, or photo storage services. These services have been embraced by a significant amount of the world population, and are used on a daily basis. Cloud providers and Network operators' infrastructures accommodate the traffic rates that the aforementioned services generate, and their management tasks do not only involve the traffic steering, but also the processing of the network services' traffic. Traditionally, the traffic processing has been assessed via applications/programs deployed on servers that were exclusively dedicated to a specific task as packet inspection. However, in recent years network services have stated to be virtualized and this has led to the Network Function Virtualization (Network Function Virtualization (NFV)) paradigm, in which the network functions of a service run on containers or virtual machines that are decoupled from the hardware infrastructure. As a result, the traffic processing has become more flexible because of the loose coupling between software and hardware, and the possibility of sharing common network functions, as firewalls, across multiple network services.

NFV eases the automation of network operations, since scaling and migrations tasks are typically performed by a set of commands predefined by the virtualization technology, either containers or virtual machines. However, it is still necessary to decide the traf-

Resumen

Las infraestructuras de red actuales soportan una variedad diversa de servicios como video bajo demanda, video conferencias, redes sociales, sistemas de educación, o servicios de almacenamiento de fotografías. Gran parte de la población mundial ha comenzado a utilizar estos servicios, y los utilizan diariamente. Proveedores de Cloud y operadores de infraestructuras de red albergan el tráfico de red generado por estos servicios, y sus tareas de gestión no solo implican realizar el enrutamiento del tráfico, sino también el procesado del tráfico de servicios de red. Tradicionalmente, el procesado del tráfico ha sido realizado mediante aplicaciónes/programas desplegados en servidores que estaban dedicados en exclusiva a tareas concretas como la inspección de paquetes. Sin embargo, en los últimos años los servicios de red se han virtualizado y esto ha dado lugar al paradigma de virtualización de funciones de red (Network Function Virtualization (NFV) siguiendo las siglas en inglés), en el que las funciones de red de un servicio se ejecutan en contenedores o máquinas virtuales desacopladas de la infraestructura hardware. Como resultado, el procesado de tráfico se ha ido haciendo más flexible gracias al laxo acople del software y hardware, y a la posibilidad de compartir funciones de red típicas, como firewalls, entre los distintos servicios de red.

NFV facilita la automatización de operaciones de red, ya que tareas como el escalado, o la migración son típicamente llevadas a cabo mediante un conjunto de comandos previamente definidos por la tecnología de virtualización pertinente, bien mediante contenedores o máquinas virtuales. De todos modos, sigue siendo necesario decidir el en-

fic steering and processing of every network service. In other words, which servers will hold the traffic processing, and which are the network links to be traversed so the users' requests reach the final servers, i.e., the network embedding problem. Under the umbrella of NFV, this problem is known as Virtual Network Embedding (VNE), and this thesis refers as "NFV orchestration algorithms" to those algorithms solving such a problem. The VNE problem is a NP-hard, meaning that it is impossible to find optimal solutions in polynomial time, no matter the network size. As a consequence, the research and telecommunications community rely on heuristics that find solutions quicker than a commodity optimization solver.

Traditionally, NFV orchestration algorithms have tried to minimize the deployment costs derived from their solutions. For example, they try to not exhaust the network bandwidth, and use short paths to use less network resources. Additionally, a recent tendency led the research community towards algorithms that minimize the energy consumption of the deployed services, either by selecting more energy efficient devices or by turning off those network devices that remained unused. VNE problem constraints were typically summarized in a set of resources/energy constraints, and the solutions differed on which objectives functions were aimed for. But that was before 5th generation of mobile networks (5G) were considered in the VNE problem. With the appearance of 5G, new network services and use cases started to emerge. The standards talked about Ultra Reliable Low Latency Communication (Ultra-Reliable and Low Latency Communications (URLLC)) with latencies below few milliseconds and 99.999% reliability, an enhanced mobile broadband (enhanced Mobile Broadband (eMBB)) with significant data rate increases, and even the consideration of massive machine-type communications (Massive Machine-Type Communications (mMTC)) among Internet of Things (IoT) devices. Moreover, paradigms such as edge and fog computing blended with the 5G technology to introduce the idea of having computing devices closer to the end users. As a result,

rutamiento y procesado del tráfico de cada servicio de red. En otras palabras, qué servidores tienen que encargarse del procesado del tráfico, y qué enlaces de la red tienen que utilizarse para que las peticiones de los usuarios lleguen a los servidores finales, es decir, el conocido como embedding problem. *Bajo el paraguas del paradigma NFV, a este problema se le conoce en inglés como Virtual Network Embedding (VNE), y esta tesis utiliza el término "NFV orchestration algorithm" para referirse a los algoritmos que resuelven este problema. El problema del VNE es NP-hard, lo cual significa que que es imposible encontrar una solución óptima en un tiempo polinómico, independientemente del tamaño de la red. Como consecuencia, la comunidad investigadora y de telecomunicaciones utilizan heurísticos que encuentran soluciones de manera más rápida que productos para la resolución de problemas de optimización.*

Tradicionalmente, los "NFV orchestration algorithms" han intentado minimizar los costes de despliegue derivados de las soluciones asociadas. Por ejemplo, estos algoritmos intentan no consumir el ancho de banda de la red, y usar rutas cortas para no utilizar tantos recursos. Además, una tendencia reciente ha llevado a la comunidad investigadora a utilizar algoritmos que minimizan el consumo energético de los servicios desplegados, bien mediante la elección de dispositivos con un consumo energético más eficiente, o mediante el apagado de dispositivos de red en desuso. Típicamente, las restricciones de los problemas de VNE se han resumido en un conjunto de restricciones asociadas al uso de recursos y consumo energético, y las soluciones se diferenciaban por la función objetivo utilizada. Pero eso era antes de la 5ª generación de redes móviles (5G) se considerase en el problema de VNE. Con la aparición del 5G, nuevos servicios de red y casos de uso entraron en escena. Los estándares hablaban de comunicaciones ultra rápias y fiables (Ultra-Reliable and Low Latency Communications (URLLC) usando las siglas en inglés) con latencias por debajo de unos pocos milisegundos y fiabilidades del 99.999 %, una banda ancha mejorada (enhanced Mobile Broadband (eMBB) usando las siglas en inglés) con notorios incrementos en el flujo de datos, e incluso la consideración de comunicaciones masivas entre máquinas (Massive Machine-Type Communications (mMTC) usando las siglás en inglés) entre dispositivos IoT. Es más, paradigmas como edge *y* fog computing *se incor-*

the VNE problem had to incorporate the new requirements as constraints to be taken into account, and every solution should either satisfy low latencies, high reliability, or larger data rates.

This thesis studies the VNE problem, and proposes some heuristics tackling the constraints related to 5G services in Edge and fog scenarios, that is, the proposed solutions assess the assignment of Virtual Network Functions to resources, and the traffic steering across 5G infrastructures that have Edge and Fog devices. To evaluate the performance of the proposed solutions, the thesis studies first the generation of graphs that represent 5G networks. The proposed mechanisms to generate graphs serve to represent diverse 5G scenarios. In particular federation scenarios in which several domains share resources among themselves. The generated graphs also represent edge servers, so as fog devices with limited battery capacity. Additionally, these graphs take into account the standard requirements, and the expected demand for 5G networks. Moreover, the graphs differ depending on the density of population, and the area of study, i.e., whether it is an industrial area, a highway, or an urban area.

After detailing the generation of graphs representing the 5G networks, this thesis proposes several NFV orchestration algorithms to tackle the VNE problem. First, it focuses on federation scenarios in which network services should be assigned not only to a single domain infrastructure, but also to the shared resources of the federation of domains. Two different problems are studied, one being the VNE itself over a federated infrastructure, and the other the delegation of network services. That is, whether a network service should be deployed in a local domain, or in the pool of resources of the federation domain; knowing that the latter charges the local domain for hosting the network service. Second, the thesis proposes OKpi, a NFV orchestration algorithm to meet 5G network slices quality of service. Conceptually, network slicing consists in splitting the network so network services are treated differently based on the slice they belong to. For example, an eHealth network slice will allocate the network resources

poraron a la tecnología 5G, e introducían la idea de tener dispositivos de cómputo más cercanos al usuario final. Como resultado, el problema del VNE tenía que incorporar los nuevos requisitos como restricciones a tener en cuenta, y toda solución debía satisfacer bajas latencias, alta fiabilidad, y mayores tasas de transmisión.

Esta tesis estudia el problema des VNE, y propone algunos heurísticos que lidian con las restricciones asociadas a servicios 5G en escenarios edge *y* fog*, es decir, las soluciones propuestas se encargan de asignar funciones virtuales de red a servidores, y deciden el enrutamiento del tráfico en las infraestructuras 5G con dispositivos* edge *y* fog*. Para evaluar el rendimiento de las soluciones propuestas, esta tesis estudia en primer lugar la generación de grafos que representan redes 5G. Los mecanismos propuestos para la generación de grafos sirven para representar distintos escenarios 5G. En particular, escenarios de federación en los que varios dominios comparten recursos entre ellos. Los grafos generados también representan servidores en el edge, así como dispositivos fog con una batería limitada. Además, estos grafos tienen en cuenta los requisitos de estándares, y la demanda que se espera en las redes 5G. La generación de grafos propuesta sirve para representar escenarios federación en los que varios dominios comparten recursos entre ellos, y redes 5G con servidores* edge*, así como dispositivos* fog *estáticos o móviles con una batería limitada. Los grafos generados para infraestructuras 5G tienen en cuenta los requisitos de estándares, y la demanda de red que se espera en las redes 5G. Además, los grafos son diferentes en función de la densidad de población, y el área de estudio, es decir, si es una zona industrial, una autopista, o una zona urbana.*

Tras detallar la generación de grafos que representan redes 5G, esta tesis propone algoritmos de orquestación NFV para resolver con el problema del VNE. Primero, se centra en en escenarios federados en los que los servicios de red se tienen que asignar no solo a la infraestructura de un dominio, sino a los recursos compartidos en la federación de dominios. Dos problemas diferentes han sido estudiados, uno es el problema del VNE propiamente dicho sobre una infraestructura federada, y el otro es la delegación de servicios de red. Es decir, si un servicio de red se debe desplegar localmente en un dominio, o en los recursos compartidos por la federación de dominios; a sabiendas de que el

necessary to meet low latencies for network services such as remote surgery. Each network slice is devoted to specific services with very concrete requirements, as high reliability, location constraints, or 1ms latencies. OKpi is a NFV orchestration algorithm that meets the network service requirements among different slices. It is based on a multi-constrained shortest path heuristic, and its solutions satisfy latency, reliability, and location constraints. After presenting OKpi, the thesis tackles the VNE problem in 5G networks with static/moving fog devices. The presented NFV orchestration algorithm takes into account the limited computing resources of fog devices, as well as the out-of-coverage problems derived from the devices' mobility.

To conclude, this thesis studies the scaling of Vehicle-to-Network (V2N) services, which require low latencies for network services as collision avoidance, hazard warning, and remote driving. For these services, the presence of traffic jams, or high vehicular traffic congestion lead to the violation of latency requirements. Hence, it is necessary to anticipate to such circumstances by using time-series techniques that allow to derive the incoming vehicular traffic flow in the next minutes or hours, so as to scale the V2N service accordingly.

último caso supone el pago de cuotas por parte del dominio local a cambio del despliegue del servicio de red. En segundo lugar, esta tesis propone OKpi, un algoritmo de orquestación NFV para conseguir la calidad de servicio de las distintas slices *de las redes 5G. Conceptualmente, el* slicing *consiste en partir la red de modo que cada servicio de red sea tratado de modo diferente dependiendo del trozo al que pertenezca. Por ejemplo, una slice de eHealth reservará los recursos de red necesarios para conseguir bajas latencias en servicios como operaciones quirúrjicas realizadas de manera remota. Cada trozo (*slice*) está destinado a unos servicios específicos con unos requisitos muy concretos, como alta fiabilidad, restricciones de localización, o latencias de un milisegundo. OKpi es un algoritmo de orquestación NFV que consigue satisfacer los requisitos de servicios de red en los distintos trozos, o* slices *de la red. Tras presentar OKpi, la tesis resuelve el problema del VNE en redes 5G con dispositivos* fog *estáticos y móviles. El algoritmo de orquestación NFV presentado tiene en cuenta las limitaciones de recursos de cómputo de los dispositivos* fog*, además de los problemas de falta de cobertura derivados de la movilidad de los dispositivos.*

Para concluir, esta tesis estudia el escalado de servicios vehiculares Vehicle-to-Network (V2N), que requieren de bajas latencias para servicios como la prevención de choques, avisos de posibles riesgos, y conducción remota. Para estos servicios, los atascos y congestiones en la carretera pueden causar el incumplimiento de los requisitos de latencia. Por tanto, es necesario anticiparse a esas circunstancias usando técnicas de series temporales que permiten saber el tráfico inminente en los siguientes minutos u horas, para así poder escalar el servicio V2N adecuadamente.

Table of Contents

List of Figures

List of Tables

1. 5G Networks & Paradigms

The 5th generation of mobile networks does not only correspond to the new set of radio technologies promising higher bandwidth, and lower delays than its predecessor 4G. Although 5G-related radio technologies, and new devices are the most visible part; the new standard conveys a wider set of innovations related to new paradigms as edge computing, fog computing, Network Function Virtualization (NFV), or network slicing. These paradigms, are introduced in the following subsections to ease the understanding of the present thesis. And their target is to reduce the latency bringing resources closer to the end user (edge & fog computing), and to increase the flexibility and Quality of Service (QoS) using NFV and network slicing.

The present section starts with an overview of the different standardization bodies that have contributed to the definition of the 5G standards. The standardization process have consisted of two main phases, one after the other, that provide specifications for the different use cases and applications that the 5G technology is supposed to provide.

Second of all, this section overviews the NFV paradigm, which essentially states that 5G networks can benefit from embedding network functionality in virtualized resources. The corresponding subsection overviews the building blocks of the NFV paradigm, and the functionalities that each of them have.

Third, the section introduces the edge and fog computing paradigms, so as the related standardization bodies that have proposed both of them. Edge and fog computing, are paradigms that propose to bring resources closer to the end user, either by means of having servers and computing resources closer to the access network (edge computing), or by means of considering a whole continuum of resources between cloud and users/Internet of Things (IoT) devices.

Last of all, this section explains the network slicing paradigm; which essentially consists in differentiating coexisting 5G applications (as high-quality video streaming, or ultra-low latency services) within the network infrastructure. As later explained, such differentiation is achieved using traffic prioritization, and dedicated resource allocation for every application. The result is a network infrastructure meeting the QoS requirements of 5G applications, thanks to an appropriate network provisioning (slice) for each application.

1.1 5G Networks

The 5G technology arises as a set of standards with the aim of satisfying the increasing network bandwidth, faster and more reliable communications. Not only that, but it tries to provide the necessary definitions to meet the communication expectations of new services such as Virtual Reality (VR), Vehicle-to-Everything (V2X), or Industry 4.0. The standardization of the 5G technology is not something finished yet, as its definition is an iterative process where standardization consortiums, or even the industry itself, push forward the description of the different entities present.

As done with its predecessors technologies, known as 3G and 4G, the standardization of the 5$^{\text{th}}$ generation of mobile networks has been tackled by the 3rd Generation Partnership Project (3GPP), a standardization organization responsible of formalizing the requirements and features of the new standard. 3GPP assesses the definition of a technology by splitting the different tasks into Working Group (WG)s, which are consortiums of companies/academia members proposing enhancements to the technology in documents known as Technical Specification (TS). For example, [3GP17a] describes a set of enhancements for the support of V2X scenarios. The TS have a life cycle that lead to different versions of the document, so the technology aspect treated in each TS is open to corrections and improvements. However, the 3GPP defines “releases” serving as milestones for the definition of 5G (so as for its predecessors). In particular, the 5G technology has already closed[1] Release 15, and Release 16; and is currently working in the definition of Release 17. The former is referred as 5G Phase-1, while the latter as 5G Phase-2. Each release spans over a couple of years and passes along milestones setting short-term objectives (months) regarding the work being done by the ongoing TSs, e.g., Release 16 set a milestone in the first quarter of 2020 regarding the Radio Access Network (RAN) definition. The definitions of Release 15 and Release 16 were closed on December 2018 and June 2020, respectively. Nevertheless, closing a release does not imply that the technology is frozen, as the different WGs keep on working on further enhancements of their respective TSs. Thus, the 5G technology will be enhanced in future releases that will incorporate upcoming features that either have not been though of yet, or have been evolving since previous releases.

Roughly speaking, the job of these standards is to define the new RAN and its requirements, what is known as the 5G Core (5GC) (the successor of the Evolved Packet Core (EPC)), the spectrum over which the 5G RAN transmits, the new services that 5G offers (e.g., V2X), how these services interact with the 5GC, the integration and enhancements of Long Term Evolution (LTE) to support new 5G services, so as the exchange of information in between the EPC and 5GC. In particular, the Release 15 mainly focused on the definition of 5G services so as the integration of the 5G Access Network (5GA) with not only the 5GC, but as well with the existing 4G LTE/ EPC Core Network; this is known as Non-Stand Alone (NSA). Whilst Release 16 pushed forward the definition of the 5G services presented in Release 15, got into detail regarding the 5GA & backhaul of 5G networks, and it introduced Ultra-Reliable and Low Latency Communications (URLLC) enhancements. As a result, both documents provide a clear road map with pointers towards TSs that provide the requirements and architecture design of 5G networks. This is particularly useful for the industry, so companies responsible of building Radio Unit (RU)s know the frequencies (which depend on countries’ regulations) and bandwidth in which the antennae have to emit/receive, car manufacturers will know how to interact with the network to deploy V2X services, and even mobile phone chip vendors that will need to process the so called New Radio (NR) signals for the upcoming 5G cell phones.

The first thing coming to the mind of a citizen when she/he hears about 5G, is new base stations that just by their own will provide the promised services and speeds of 5G services. Although far from the reality, it is true that what is known as NR plays a key role in the definition of a 5G network as a whole. Users will connect to antennae in the frequency band between 30 and 300 GHz [Niu+15], which result in a higher transmission rate that goes up to the order of magnitude of Gbps. This is a a huge improvement with respect to the predecessor LTE technology, which

[1] this sentence was written in 17/02/2021

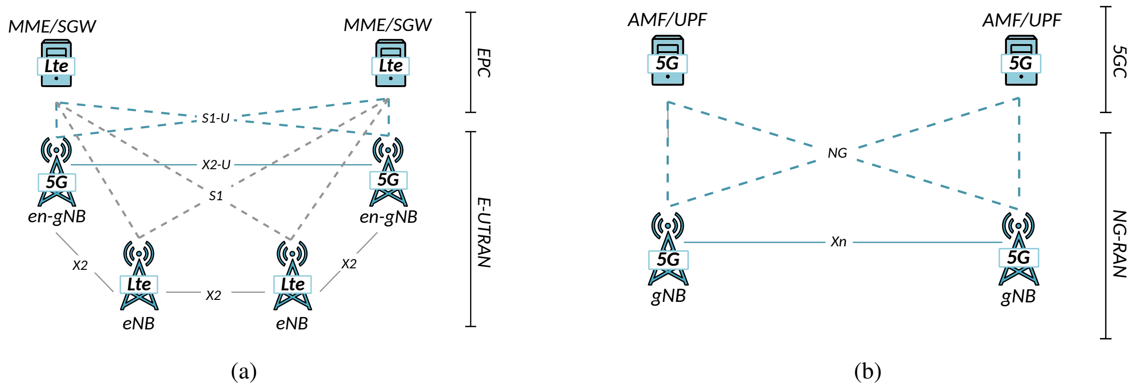


Figure 1.1: (a) NSA and (b) Stand Alone (SA) architecture overview – taken from [3GP19c]

provided data rates of up to 100 Mbps in the uplink, and 50 Mbps in the downlink using 20 MHz channels [HT11]; and peak downlink latency of 1 Gbps with the LTE advanced [3GP20i] technology. As a result the user will benefit by achieving higher data rates useful for 4K video streams, of either broadcasting services, or for video emissions that might exhaust the uplink channel using previous 4G technology.

The significant increase in data rates are a consequence of the research progress on the Millimeter Wave (mmWv) technology over the last years. With mmWv, RUs will transmit in the GHz band, but they have the side-effect of higher signal attenuation than its predecessor technologies working in the MHz bands. As a consequence, the signal emission should be more directional than in the past, and their coverage is reduced. This leads to a more dense deployment of antennae in the 5G network planning, and thus, a higher deployment cost for the infrastructure. Nevertheless, assuming a denser deployment of NR RU leads to other issues yet to be tackled such as interference in between the new antennae, and LTE- NR interference problems tackled in 3GPP Release 16 [3GP20h]. In any case, NR technology presents significant features such as a flexible Subcarrier Spacing (SCS), or allowing to emit data before the transmission slot finishes. The latter enhancement was defined in 3GPP Release 15 [3GP19c], and it is of high benefit for 5G services requiring low latency. Indeed, rather than consuming the 10 ms defined for a transmission slot, a User Equipment (UE) could use a slot fraction of 1 ms to transmit the latency-sensitive packet.

As in previous standardization documents of 3GPP, the 5G standardization also considers the coexistence of 5G with current LTE deployments of network operators. New network deployments can benefit from existing LTE infrastructure to provide wider coverage, and reuse RAN resources. The coexistence between LTE and 5G deployments is known as NSA (see Figure 1.1 (a)), and it assumes that NR base stations (known as Next Generation NodeB (gNB)) will forward traffic to the legacy EPC and its constituent functions, namely, the Mobility Management Entity (MME) and Serving Gateway (SGW). As LTE Evolved Node B (eNB) will coexist with the 5G network, there has still been ongoing work to improve the data rates and signaling of the legacy RUs. Indeed the work item [3GP17b] presented how to improve the Downlink (DL) capacity in the connectivity between LTE eNBs and UEs that remain stationary, as users connected to an antenna in the rooftop of their homes, or a fixed laptop with LTE connectivity. The modulation-based enhancement allows reaching DL data rates of up to 3.5Gbps. This enhancement proofs that a coexistence with yet to be improved LTE deployments, would be beneficial for future 5G infrastructures. However, the operators' goal is to achieve what it is known as SA architecture (see Figure 1.1 (b)) in which all RUs will be 5G gNBs connected to the 5GC rather than to the EPC.

The benefit of SA deployments is that the 5GC provides additional tools for traffic differentiation, QoS enforcement, multiple and diverse QoS flows over the same Protocol Data Unit (PDU). The 5GC is designed as a Service-Based Architecture (SBA) framework, and it is a compound of Network Function (NF)s that provide services to other NFs, or authorized users. The main NFs of the 5GC are (*i*) the User Plane Function (UPF), (*ii*) the Session Management Function (SMF); and

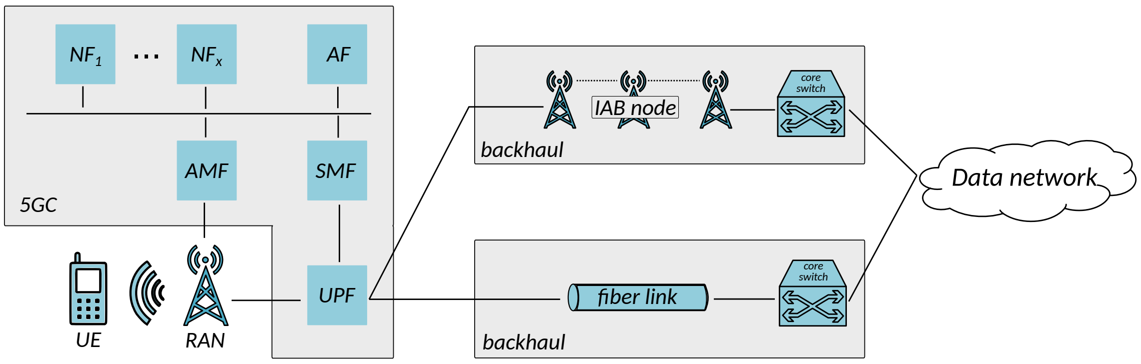


Figure 1.2: 5GC and two different backhaul connections to the Data network.

(*iii*) the Access and Mobility Management Function (AMF). The UPF handles the user data and performs tasks as traffic routing and inspection, the SMF keeps the users sessions and enforces QoS, and the AMF deals with the mobility support, and authentication tasks. On top of the aforementioned NFs, the SBA framework of the 5GC considers the interaction of a NF with an Application Function (AF), so applications have a more fine grained interaction with the 5G network. Moreover, AFs are functional elements designed to expose application-related information to consumers, and to send event notifications over subscription. The advantage of the 5G SBA framework is that its NFs do not have to be co-located and running under the same facilities, as specified in Release 15 [3GP19c]; which leads to the discussion of the split of such functions among different computing hardware nodes, i.e., the so-called functional split [3GP19c]. Not only that, but the 5GC has a specific NF to expose monitoring information of the existing connectivity, so as the possibility of issuing policy enforcement on the 5GC. This is the Network Exposition Function (NEF), and it is useful for services requiring strict latency and high reliability as V2X. For example, thanks to the NEF V2X services can monitor whether the connectivity of gNBs with cars is good enough to ensure that applications as car-platooning can operate without any risk – see [3GP20b].

Once the users' traffic have reached the 5GC, it is up to the UPF NF to perform the traffic steering. Moreover, within the 5GC, the UPF serves as reporter of the current traffic usage, and handler of QoS for the user plane [3GP19c]. This means that the UPF plays a key role to ensure the requirements of 5G traffic, specially the promised high bandwidth rates (e.g., 4K video streaming services), and high reliability requirements (e.g., eHealth services like remote surgery). Both reliability and bandwidth requirements are met thanks to functionalities such as packet filtering, or Uplink (UL) and DL rate enforcement.

Given the UPF traffic forwarding, it is still necessary to steer it towards the data network. Typically, the traffic goes over fiber links connecting the Baseband processing Unit (BBU)s and the switches of the core network, i.e., the backhaul portion of the network (see Figure 1.2). However, the deployment of fiber link relates to a high Capital Expense (CAPEX), as it requires digging in public facilities such as the sides of the road. When 5G started its standardization, operators and vendors wondered how to accommodate the high, reliable, and fast traffic promised by 5G without having to install the required fiber, all at once. This was of special urgency, as the 5G networks would be more dense than its predecessors, hence requiring more RUs and fiber-based backhaul connectivity from the latter towards the core network. To overcome the issue of having to deploy all the required backhaul fiber links at once, the Release 16 [3GP20h] of 3GPP introduced the concept of Integrated Access and Backhaul (IAB); a new approach of backhaul connectivity that uses wireless mmWv to forward backhaul traffic (see Figure 1.2). The idea is to have a chain of radio nodes interconnected among them, and the NR node providing connectivity to the UE. These radio nodes are seamlessly integrated with the defined 5GC NFs, and are easily updated with the target fiber-connected RUs, as they use as well Internet Protocol (IP) connectivity. With

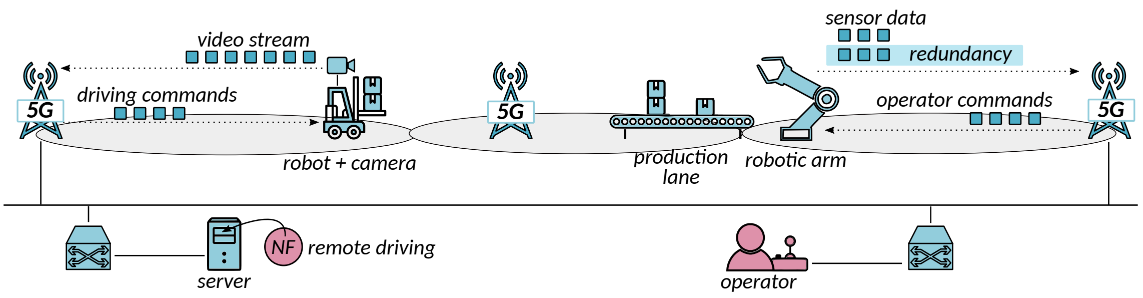


Figure 1.3: Smart manufacturing illustration

the IAB approach, current network deployments can shift easier to dense 5G networks relying on fiber wires, as the operators can deploy IAB in residential/suburban scenarios where the fiber deployment might imply a substantial economical effort. Rather, operators might deploy IAB nodes in such areas to steer the backhaul traffic via mmWv wireless links. Although it might seem like the transmission of backhaul through wireless channels is a fault-prone deployment, [3GP18c] accounts for such scenarios where IAB nodes are connected in a multi-hop fashion establishing mesh-alike connections that can recover upon link degradation due to seasonal circumstances such as the foliage increase of the trees. Furthermore, IAB considers the strict QoS requirements of 5G, and defines an adaption layer to assess QoS enforcement in the connections carrying the users' data.

Up until now, we have focused on the radio, core, and backhaul changes that 5G networks bring. All these changes are motivated by the services that 5G networks promise to offer, and it is their strict latency, bandwidth, and communications' reliability what motivates the upgrade of the network infrastructure. Among the different 5G service exemplary services, some of the most representative are the VR applications, connected drones, smart manufacturing, Machine Type Communication (MTC), Massive Machine-Type Communications (mMTC) with connected IoT devices, and V2X services. Each of them have their own requirements, which are specially demanding for the 4G network deployments by means of bandwidth, reliability, latency, location, or mobility; and even new requirements regarding battery constrained devices. The remaining of this subsection briefly introduces some of the 5G service use cases, and their network requirements.

One of the most appealing use cases of 5G, at least for industry, is the smart manufacturing – see Figure 1.3. Oftenly referred as the fourth industrial revolution, the industry 4.0 envisions the possibility of fully automating tasks performed on a daily basis on the facilities of every factory. For example, in an industrial warehouse it might be required to be constantly moving material towards the production line whenever new goods arrive to the factory. Such a task requires the coordination in between (*i*) the production lane, which might be about to run out of production goods; (*ii*) the cameras installed in the parking of the warehousing building; and (*iii*) the mobile robotics responsible of moving the goods between the truck in the parking, and the production lane or storage location. Note that the described scenario of mobile robotics in warehousing, must consider the battery levels of the operating robot that moves the goods, so as its mobility along the warehousing facility and the respective handovers. Additionally, this moving robot will be driven remotely by a NF that might take driving decisions using a video-stream of the robot with rates near $\sim$100 Mbps [3GP20k]. During its drive the robot communication with the remote driving NF must be reliable, a 99.9999% to be precise; and the radio coverage should be dense enough so the moving robot has connectivity throughout all the delivery path.

Although the remote driving use case is a use case benefiting from 5G technology, it is not as delay and jitter sensitive as other use cases like the motion control. The motion control might be the most demanding use case in this sense, as it embraces examples such as the closed-loop applications controlling sensor/actuator operations. This is the case of robot arms used for the assembly of pieces, such as doors in a car production lane. This requires a low latency in between the moving arm, and the NF that elaborates instructions using the sensor information of the robot, e.g., whether there are objects near itself. The latency must be below 1 ms in motion control scenarios [3GP20k], and that should be End-to-End (E2E) latency measured from the moment the robot issues a sensor measurement, until the reception of the remote NF instruction. The strict latency constraint of 1 ms leaves little room to the processing of the sensor information to assess the robot movements, and propagate the instruction packets back to the robot. A solution might be to have a high speed wired connection with the factory robot, however, the tendency of industry 4.0 is to have a wireless access network to allow the movement of robots and actuators across the installments depending on the daily needs. Thanks to the 5G NR, the latency is reduced in the connectivity with the antenna, and wireless deployments are considered for industry 4.0 scenarios. Nevertheless, in the aforementioned robotic arm scenario, the communications should not only be fast, but reliable. Indeed, the reliability requirements become very strict (a 99.9999% is mandatory according to [3GP20k]) so the operational errors are avoided. For example, if the robotic arm detects an assembly piece is broken, the packet containing such sensor information becomes crucial in the motion control; as upon this packet loss, the robotic arm might further damage the piece; thus the communication reliability.

This type of communications in which the network must ensure very low latency and extreme reliability, are known as URLLC; and there are different delay and reliability requirements depending on each scenario. The 3GPP gives the specific metrics in [3GP20k], where it talks about motion control, remote control, monitoring and remote control for process automation. These scenarios ask for latency of 1 ms, 5 ms, and 50 ms; respectively; and need to have communications with reliability of 99.9999%, 99.999%, 99.9999% and 99.9%; respectively. Thus, the most demanding scenario have to run over a network ensuring latency below the order of a millisecond, and with only a 0.0001% of error in the communication. To meet such low delays, 3GPP specifies in [3GP19c] Release 15 the new numerologies for the 5G NR so the SCS can take different values from 15 kHz up to 240 kHz. Additionally, [3GP19c] talks about mini slots of transmission, so 32 B packets can be transmitted in 1 ms rather than having to consume the whole Transmission Time Interval (TTI); so the URLLC packets can achieve a faster transmission. On top of these enhancements, in the Release 16 the TS [ETS20a] introduced presented an enhancement on the UL power control scheme to improve the prioritization/multiplexing of UEs transmission. Thanks to these enhancements, it is possible to achieve the URLLC delay constraints for 5G services. Regarding the reliability of URLLC packets within 5G networks, there are methods to enhance the transmission success via redundancy such as sending the same packet over several user plane paths [3GP20h].

Another relevant application service of 5G networks is the V2X services, which comprises several use cases such as car platooning, remote driving, cooperative awareness, hazard warnings, and vehicle safety as lane change warning/blind spot warning. Among these services some of them involve the communication in between cars Vehicle-to-Vehicle (V2V), or between the network and the cars Vehicle-to-Network (V2N). In the case the communication concerns a segment of the 5G network infrastructure, e.g., the car has to forward a packet to a server to report that it had an accident; then it becomes crucial that the packet informing of the accident arrives, and as fast as possible, to not only the server but the other vehicles/cars that might not be reachable in the V2V connectivity. Thus, some V2X communications lie within the category of URLLC communications. The Release 16 [3GP20h] contributed to the definition of the aforementioned use cases, so as the improvement of the 5G standard to technically support the V2X services. Note that services as the remote driving require delays below 5 ms, and such communication speed must be ensured throughout all the travel that the vehicle does. Thus, upon vehicle handovers in between RUs, the vehicle session must be maintained and carefully ensure that handover failures have recovery/backup mechanisms to mitigate packet loss that might lead to accidents.

For V2X services car manufacturers should collaborate closely with the network infrastructure so as to be aware of network conditions like congestion, radio interference, packet loss, or whether the given QoS meets the requirements of the provided V2X service. To this extent, the 3GPP Release 16 [3GP20h] came up with the concept of Service Enabler Architecture Layer for Verticals (SEAL). With it, 3GPP proposes to use an abstraction layer over the Internet to provide Verticals of

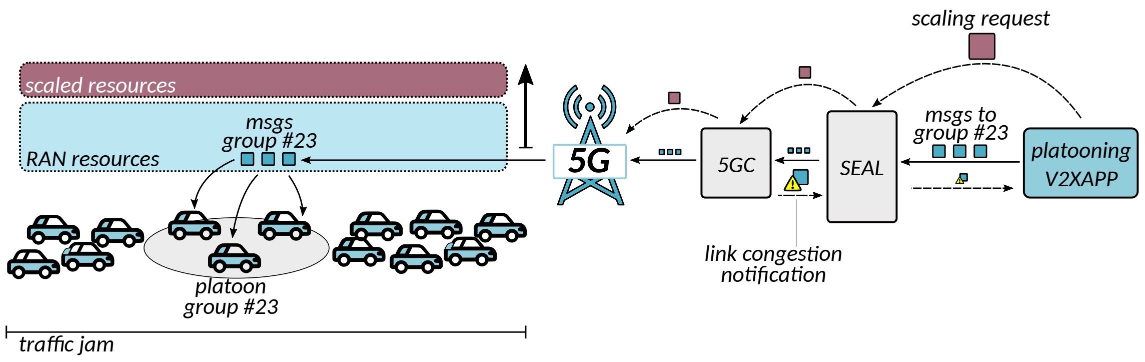


Figure 1.4: A car platooning V2XAPP asking for scaling of RAN resources

services satisfying the Key Performance Indicator (KPI)s promised to the users, e.g., latency below 10 ms for a collision avoidance service in V2X [3GP19b]. To this extent, SEAL offers a server providing different application layers to several verticals at the same time, and it is responsible of interacting with the network to ask/manage the unicast and multicast resources provided by either the EPC or 5GC. Additionally, V2X services can benefit of location and grouping functionalities that SEAL offers to verticals. But the purpose of SEAL was not only to provide of an application layer to V2X services, but other services provided by verticals, so applications that work closely with networks are managed by verticals, and not only the infrastructure owners of 5G networks. However, the definition of SEAL served as starting point to build the concept of V2XAPP [3GP20a], that is applications for V2X services. With V2XAPPs there is a chance of sending packets to groups of cars that belong to the same platoon during the driving (see Figure 1.4), by identifying all the cars within the platoon so as the leader. Furthermore, given the strict requirements by means of latency and reliability, as the 10 ms E2E latency and 99.99% reliability of platooning services, the V2XAPP concept considers that the application can monitor the network performance to meet the service requirements, and additionally it can request for resources for the underlying 3GPP network. More specifically, V2X applications can ask for QoS analytics regarding the provided service, and adapt their execution to the current network conditions, such as network congestion due to traffic jams. This latter example is possible, since the 3GPP Release 16 [3GP20h] considers the possibility of a V2XAPP asking for QoS notifications regarding a specific geographic area, i.e., the location where a traffic jam occurs on a daily basis. In case of changing network conditions, not only the V2XAPP can be conscious of them so as to adapt itself to the existing conditions, but as well the 5G NR will manage the resources so as to meet the delay and reliability requirements asked by the V2XAPP. This is possible thanks to the per flow QoS differentiation that 5G NR offers.

Apart from smart factory and V2X services, there exist other 5G use cases that are representative by means of usage of the technology capabilities, e.g., the streaming of 4K quality in crowded events, VR video streaming/rendering, mission critical communications, or communications with Unmaned Aerial Vehicle (UAV) over 5G networks. This section has paid attention to V2X and smart factory because they convey applications that require of URLLC communications, which are studied throughout the experiments present in this thesis. This is due to the importance of assessing an adequate orchestration of resources to ensure the latency and reliability of such 5G services. Because even if the 5G network, briefly introduced in this section, achieves faster communications and data-rates because of its technologies (e.g., mmWv), it is not enough to rely on them to meet the promised KPIs. In the deployments of 5G services it is crucial to ensure that traffic routing in between the access network and the servers (e.g., in the cloud) is fast enough to meet the delay requirements, not to mention the election of the proper target servers to process the service traffic. Even more, for location-constrained services, the election of near-user resources is crucial to meet the constrains, and to shrink the problem size of both steering traffic, and selecting the servers to process the demanded service. Hence, the focus of this thesis is to solve the resource allocation and traffic steering of 5G services, using reference 5G infrastructures, and their promised performance.

1.2 Network Function Virtualization (NFV)

5G networks aim at being more autonomous, and to reduce the human-based maintenance. That is, it should be able to address changes required as the network demand varies over time, e.g., to scale a service to accommodate the new incoming users. Traditionally, such changes required that the network operator manually increase the network capacity by addition of supplementary hardware, or instantiation of the used application – as a Content Delivery Network (CDN) server – within the new supplementary hardware. This obliged the operator to have a dedicated technician to assess the required tasks each time network demand changes, or even worse, to over-provision the required hardware resources to meet the peaks of demand foreseen in the traffic history. But the dimensioning of hardware resources was not the only manually performed task by means of network management. Despite the robust and already working distributed routing protocols, like Border Gateway Protocol (BGP), localized traffic routing management had to be assessed manually by the system administration of networks providing any service. For example, the additional video streaming demand needed to be redirected towards the CDN server installed inside the new operating server, which translates into a reconfiguration of traffic rules towards the new machine.

Static traffic routing, and infrastructure dimensioning leads to large Operational Expense (OPEX) for a network operator, and its minimization is tightly coupled with the deployments flexibility. The more flexible a deployment is, the less investment for maintenance, as the deployment is capable of adapting to more various situations such as drops/increase of demand. Additionally, a flexible service deployment allows to modify aspects such as CPU used, or allocated bandwidth over a link. NFV aims to bring flexibility to network deployments thanks to the virtualization technologies such as containers, and virtual machines. The paradigm can additionally make use of virtualized traffic routing layers as Virtual Extensible Local Area Network (VXLAN) to bring abstraction and isolation to the network infrastructure. The idea of NFV is to come up with an architecture in which resources are abstracted as virtual resources later managed by entities that allocate services over the abstracted pool of resources. For example, in a data center of x10 servers with x4 CPUs each one, one would have a pool of x40 vCPUs corresponding to the aggregation of all the CPUs of the data center servers. Moreover, the interconnection among servers might be abstracted into multiple VXLANs for dedicated purposes such as data-plane and control-plane traffic. Given the mentioned abstractions, a NFV-based network would be able to manage both the computational and network resources as an aggregated pool easier to manage than by taking into account the association in between hardware and resources.

On top of the network infrastructure abstraction, the European Telecommunications Standards Institute (ETSI) abstracts the composing elements of a service offered by the network. More specifically, ETSI conceives a functional element of the network as a Virtual Network Function (VNF), and it is nothing but a part of a service formed by multiple VNFs that are connected as a graph. For example, a Network Service (NS) providing video streaming would be formed of x2 VNFs: one acting as a pool of video files; and the other doing the video encoding before it is downstreamed towards the end user. Therefore, [ETS14] defines a NS as a chain of VNFs interconnected with link abstractions that are called Virtual Link (VL). Intuitively, a NS would have one VL per each consecutive VNF, that is, there would be one VL interconnecting the video pool VNF and the video encoder VNF. But for some NSs the interconnection in between VNFs might be more elaborated and require more than one link for each VNF. To model this, ETSI proposes the VNF Forwarding Graph (VNFFG); a graph that represents the traffic flow in between the VNFs of a given NS, in which every node represents a VNF, and every edge is associated to a VL. An example of a not so straight forward VNFFG would be a load balancer NS composed of a firewall VNF that filters and redirects the traffic among several web servers – see Figure 1.5. Such NS would have one VL for every link interconnecting the firewall with all the subsequent web servers. Additionally, the VNFFG associated to a NS may have a functional element that cannot be virtualized, e.g., a camera recording a video that is streamed to the internet. In a video

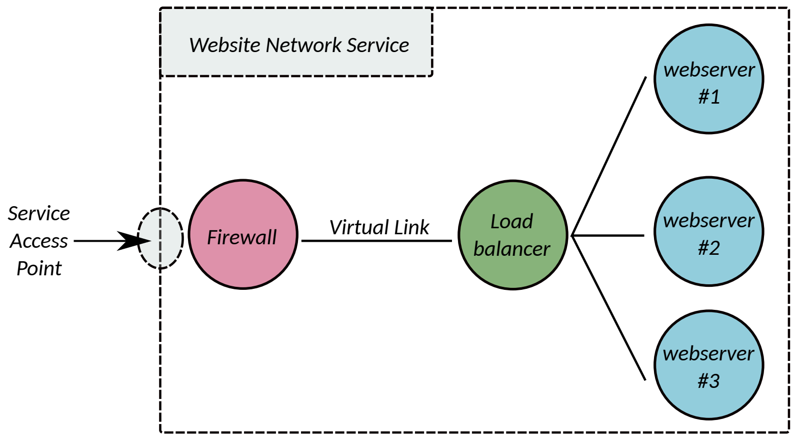


Figure 1.5: A website NS with firewall, load balancer, and web server VNFs

streaming NS such camera is known as a Physical Network Function (PNF); and it belongs to the corresponding VNFFG so as the VNFs.

To manage and orchestrate virtualized services over a pool of computing and network resources, ETSI proposes a reference architecture [ETS14] known as NFV Management and Orchestration (MANO) architecture – see Figure 1.6. The idea is to have a common reference architecture that deals with tasks as (*i*) scaling of VNFs; (*ii*) instantiation of NSs; (*iii*) management of physical and virtual resources; (*iv*) providing connectivity in between VNFs or PNFs; (*v*) providing fault tolerance support; (*vi*) update of VNFs; or (*vii*) healing procedures. All these tasks are delegated to different functional blocks that are within the NFV MANO architectural framework – see Figure 1.6. As an overview, the NFV Infrastructure (NFVI) functional block represents the virtualization infrastructure on top of which the NSs will execute, the Virtualised Infrastructure Manager (VIM) functional block assesses the management of underlying virtual resources, the VNF Manager (VNFM) is responsible of the lifecycle management of the deployed VNFs, and the NFV Orchestrator (NFVO) does both the lifecycle management of NSs, and resource management via its interaction with the VIM. The next paragraphs provide a more detailed overview of the interconnections and functionalities of each of the functional blocks. We proceed explaining them in a bottom-up approach, i.e., first the functional blocks dealing with resources, and later those dealing with the management and orchestration duties.

In the context of the NFV paradigm, each administrative domain – whether it is the domain of a tenant, or an infrastructure domain – is responsible of the management of its own resources, virtual of physical. It does not have to be aware of what is being executed by each of the VNFs that run within its infrastructure (i.e., it is application agnostic), but it needs to keep track of its own resources availability over time. Thus, an administrative domain benefits from having a VIM to have a control of its resources. Inside the NFV MANO architecture, the VIM is aware of the available physical and virtual resources, and it keeps track of the association of VNFs and VLs to the underlying resources. For example, the VIM knows that the VL that connects the firewall VNF with the web server VNF of a NS, is sending the packets across an specific physical link like an Ethernet cable of the correspondent administrative domain. This is, the VIM is aware of how resources are allocated for the NSs already running inside the administrative domain. There exist several solutions that implement the VIM functionality like OpenStack [RB14], or vshepere [GLC13]; that provide not only VIM functionalities, but other ones that correspond to different functional blocks of the NFV MANO stack.

Given that the VIM keeps track of the pool of resources, either virtual or physical; the administrative domain still needs to assess the corresponding management of the NSs and VNFs deployed in the underlying resources like CPU, memory, network, or bandwidth. ETSI NFV delegates the VNF lifecycle management to the VNFM, a functional block inside the NFV MANO architecture that communicates with the VIM via the Vi-Vnfm interface. The VNFM performs

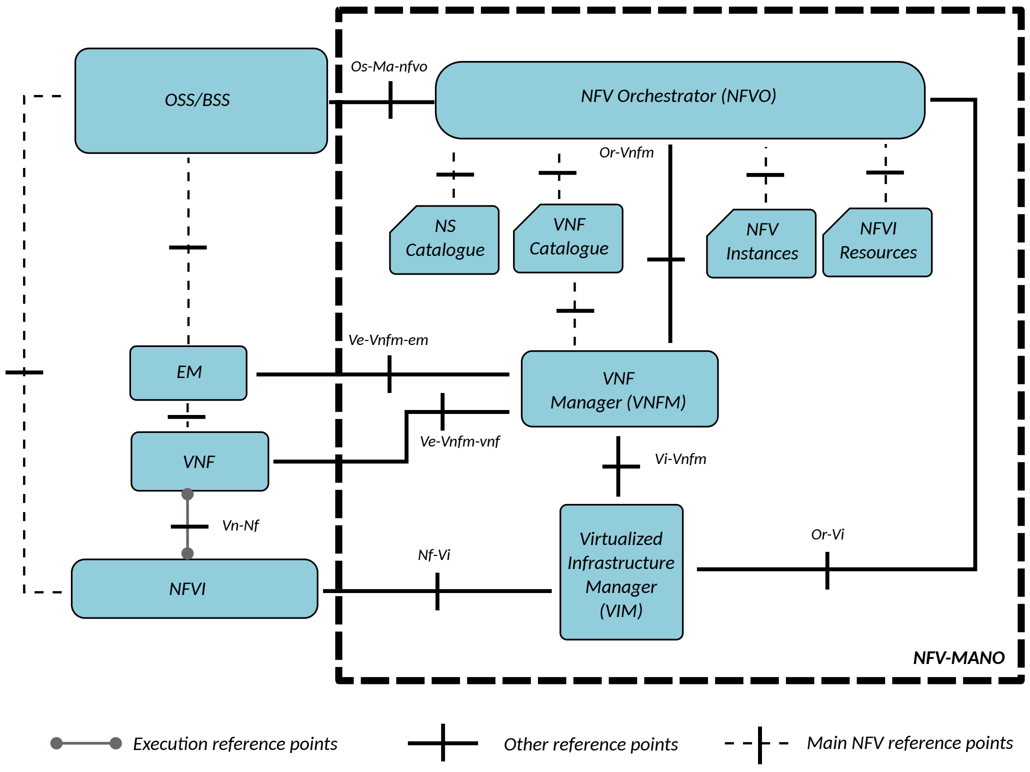


Figure 1.6: ETSI MANO reference architecture [ETS14]

the lifecyle tasks of the instantiated VNFs, that is, it deals with the (*i*) instantiation; (*ii*) healing; (*iii*) scaling; (*iv*) termination; (*v*) software update; (*vi*) modifications; and (*vii*) event reporting of the corresponding VNF. [ETS14] specifies that each instance of a VNF should have associated a VNFM that acts as a watchdog to perform the required lifecycle management actions just mentioned. However, there is the chance of having a single VNFM dealing with the multiple existing instances of a VNF; or even a VNFM for all the VNFs deployed by the NFV MANO of the administrative domain. As explained later in this section, the Virtual Network Function Descriptor (VNFD)s contain information regarding scaling procedures, so as resources and monitoring-related tasks. These information present in the VNFs is used by the VNFM to follow the associated scaling, healing, or instantiation indications. For example, if a VNFs specifies in its flavour that it requires of x3 CPUs, the VNFM will ask the VIM for such amount of CPUs during the deployment. Similarly, if the scaling details of the VNFD specify to increase x1 CPU whenever the user demand increase by a 10%, the VNFM will trigger a scaling procedure towards the VIM. Note that the described scaling is reactive, i.e., the increase of an additional CPU is triggered by the decrease of demand. And the scaling procedures one specifies in the description of a VNF are not preemptive. To fill this gap, Chapter 6 proposes the forecasting of future demand to preemptively trigger the required scaling of resources. In particular, it studies such a preemptive scaling for V2N services.

On top of the VNFM there is the NFVO, the upper-most functional block of the ETSI MANO framework architecture, which similar to the VNFM, manages the lifecycle of NSs. The reader might notice that still none of the mentioned functional blocks perform the decisions of mapping NSs to resources. This is exactly the other task that a NFVO performs within the ETSI MANO, that is, the NFVO is responsible of taking the decisions of where the NSs should run among the different NFVIs present in one or multiple administrative domains. This implies that the NFVO should be aware of both the pool of resources, and the lifecycle of the VNFs already deployed; tasks that are assessed by the VIM and VNFM, respectively. Hence, the NFVO coordinates with the VIM via the Or-Vi interface, and with the VNFM via the Or-Vnfm interface.

The first main functionality of an NFVO is the orchestration of NSs. This means that the NFVO is responsible of (*i*) deploying onboarded Network Service Descriptor (NSD)s (later described in this section); (*ii*) manage the instantiation of VNFs coordinating with the VNFM underneath for operations as VNF scaling; (*iii*) update an already instantiated NS; (*iv*) ask for measurements to a NS; (*v*) scale a NS and trigger the associated VNF scaling operations to the corresponding VNFMs; and (*vi*) do policy management for every NS regarding aspects such as scaling, affinity or location constraints. All these orchestration functionalities turn the NFVO into a global entity responsible of coordinating all the lifecycle management of the deployed NSs. As an example, lets imagine a video streaming NS that is broadcasting the final of a sport competition. The NFVO must be aware of the current status of the virtual CDN VNF, and of the network bandwidth congestion. Lets say that by the end of the event both teams tie and there is a sudden peak of demand in the area where the tying local team is. The NFVO will be monitoring the VLs traffic and detect that there is a congestion in such an area, however, the allocated links are saturated, and the streamed video is losing quality. To overcome the situation, following the monitoring and QoS requirements specified in the NSD, the NFVO will decide to scale the NS and allocate for more bandwidth in the underlying NFVI, so the service has enough room for the increase of bandwidth demand. Once the match is finished, the sport season may have come to an end, and it is the duty of the NFVO to perfom the termination of the service. That is, it will inform the VIM that the associated resources should be freed, in particular, it will contact the VNFM to terminate the virtual CDN VNF, and ask the VIM to free the bandwidth allocated in the associated NFVI.

The second functionality of the NFVO is to perform the resource orchestration. This functionality conveys the tasks associated with the (*i*) validation of allocation requests that the VIM issues to the NFVI; (*ii*) manage and optimization of resources' usage across the different managed administrative domains; or (*iii*) management of NFVI resources used by the different VNF instances. Among the enumerated tasks, the most related to this thesis is the (*ii*) one, that is, the optimization of resource allocation, which is discussed in detail in Chapter 5. Note that whenever the ETSI MANO architecture decides the assignment of NSs to resources, there is an algorithm running inside the NFVO's resource orchestrator, that finds a solution for such assignment. The mapping of NSs to NFVI resources must meet the objectives or policies of the administrative domain managed by the NFVO. For example, it might be the case that the administrative domain wants to minimize the energy consumption of its infrastructure. Thus, the NFVO should account for it when deploying all the NSs held by the architecture, and find a mapping that satisfies the deployment flavour requirements of every NS, without exceeding a threshold of energy consumption. Note that the objective does not necessarily have to be the energy consumption minimization. For instance, it is a common practice to try to maximize the resource usage upon the arrival of NS instantiation requests, since in that way the administration domain typically achieve higher revenues. These optimization problem, with the various NS requirements is the object of interest of this thesis, and chapters 4 and 5 treat it more in detail.

Another important element considered in the NFV MANO architecture is the Operation Support System (OSS) / Business Support System (BSS) functional block. This block holds the operational and business logic that is not present in the other functional blocks, and triggers actions such as instantiation or termination requests based on its inner logic.

Lets now get into the details on how to describe both the NSs and VNFs. To formally define a NS, ETSI proposes a set of descriptors to define the components and details regarding the specified NS [ETS19a]. More specifically, a NSD provides information regarding (*i*) the VNFFG; (*ii*) the Service Access Point (SAP); (*iii*) the deployment flavour; and (*iv*) the Physical Network Function Descriptor (PNFD). A SAP specifies the entry point for a service, or in other words, what is the first VNF to be traversed by the user's traffic flow. Such information is useful to allocate the physical resource that is going to provide access to the service, for example, the destination router towards which the service users will forward their traffic. Additionally, it is required to mention the size or dimensioning required by the associated NS; and this is done with the information present in the deployment flavour specified inside the NSD. In particular, the deployment flavour gives details regarding how to perform scaling for the described service, the instantiation level (e.g., number of instances of the associated NS), and what is the profiles of the VNFs, PNFs, and VLs inside the service. And last of all, it is the description of the PNFs inside the described NS, which is specially useful to plug functionalities that cannot be virtualized, like hardware components as antennae. And as a PNF must be placed at a given location, the descriptor associated to the PNF offers the possibility of specifying the geographical location where it should be, as it might be require for specific use cases such as giving a video streaming service in a crowded event [Fra+17a], where an physical antenna must be present to provide radio coverage.

Similar to NSs, ETSI defines descriptors for VNFs, the so called VNFDs [ETS19b]. This descriptor provides deployment details about the VNF of consideration, e.g., the already mentioned web server. The elements specified are the (*i*) Virtualization Deployment Unit (VDU); (*ii*) the VNF deployment flavour; (*iii*) the VL descriptor; and (*iv*) information related to the connection points – the so called Connection Point Descriptor (CPD). First of all, when talking about a VNF, ETSI refers to a VDU as a construct to specify the operational behaviour of the container or Virtual Machine (VM) running the VNF logic. The latter is referred as a Virtual Network Function Component (VNFC) and is nothing but an instantiation of the description provided by the VDU, which gives details regarding how to monitor the performance of the VNFC, what is the software image associated to the VM or container, what is the virtual CPU and disk that is associated to the VNFC, so as the pointers to the VLs that are connected to the VNF. Second of all, it is the deployment flavour of the VNF, which is similar to the NSD deployment flavour by means of specifying how to scale the VNF, tackle the lifecycle management, whether the VNF is small or large by means of resources, how to monitor the VNF performance, and affinity constraints to be considered within the deployment phase. Third of all there are the pointers towards the associated VLs descriptors. There, the VNF has information regarding the flavour of the VL and its associated QoS, what is the metrics to be monitoring, e.g., jitter; and what is the type of connectivity of the VL – IPv4, IPv7, Multiprotocol Label Switching (MPLS), a tree flow pattern, etc. Last of all it is the CPD, that is the logical representation of a port, and it specifies the connectivity between compute resources and VLs. In particular, a CPD provides information of the bit rates, the VL associated to a VDU, its security role, and other details as whether multiple Virtual LAN (VLAN)s can be carried by the CPD.

Both the NSD and VNFD, provide a reference point for NS developers, and infrastructure owners. Any NS following the descriptor template would be understood by the MANO stack, which will just have to follow the described indications to assess tasks such as the lifecyle management, and assignment of resources for the deployed service. However, the level of detail and expertise required by NSDs may be too much for verticals. Vertical Service Blueprint (VSB)s fill such a lack of expertise, and serve as a simplified version of a NSD that verticals can fill according to the needs of the service they want to provide. Once verticals fill the details required by the VSB, it is translated into a NSD so the MANO stack deploys it following the requisites. For example, if a sport streaming company wants to deploy a virtual CDN to emit tonight's match, it could fill a VSB that is passed to the infrastructure owner, which will produce a NSD of a generic virtual CDN NS. Afterwards it can just fill the default values of the descriptors to specify instantiation level details such as the required resources like CPUs, or VLs bandwidth to support the match retransmission. European research projects as [Sga+17] or [Man+19] have prototyped solutions implementing the VSB, NSD and VNF concept in coexistence with the NSD catalogue concept, so any incoming vertical can deploy on demand a NS if the underlying infrastructure has enough resources.

An advantage of NFV descriptors is the reusability of existing functionalities, as a NSD is just a pointer to VNFDs, and to descriptors of VLs connecting them. Thus, several NSs can build on top of common VNFDs. For example, an online store NS might use the Deep Packet Inspector (DPI) VNF also used by an online chat NS; as both may require to perform packet inspection to prevent malicious traffic getting inside the application. Other example of a common VNF is a firewall. Nevertheless, it is worth stating that NSDs allow the reusability of VNFDs, not the reuse of VNF instances.

In the context of 5G networks, the 2014 specification [ETS14] supposed a paramount, since the document served as a reference point in the networking community during the developing of the 5G technology. Deliverable documents of European projects like [Sga+17] or [Man+19] relied on it as the fundamental architecture upon which the network management should be envisioned in 5G networks. The decomposition of traditional network services into their most elemental functional elements – the VNFs – brought the required deployment flexibility demanded by 5G. And not only that, but even the 3GPP TSs on the 5G technology have relied on the NFV paradigm to build up conceptual ideas such as the functional split of certain parts of the 5G networks. For example Release 15 [3GP19c] talks about the possibility of splitting a gNB into a Central Unit (CU) and Distributed Unit (DU) that hold the L1 and L2 functionalities, and L2 and L3; respectively. This functional split of a Base Station (BS) functionality enhances the deployment of a VNF running CU tasks in a remote computing entity that does not have to be co-located within the DU functionality that lives inside the gNB. As a result, the CUs can be hosted in a server not necessarily next to the gNB. For example, a cloud server may be performing the processing of the CU tasks. Moreover, this enhances a coordinated management of multiple DUs from a single CU, and an all-at-once scaling/update of the RAN resource allocation. Motivated by the possible decoupling of RAN functionalities that the NFV paradigm brings in, the O-RAN [ORA20] alliance was created. Its objective is to converge towards a virtualized RAN of future networks, in which operators can ease and improve the RAN management and operability.

To conclude this section, it is worth mentioning that the NFV paradigm is not only tightened to the 5G networks that are a matter of study in the present thesis. The paradigm seems to have came to the networking community to remain within the upcoming years and 5G-beyond network technologies, and it is already a reality in the recent networking deployments. Indeed, the trend of the networks is to go towards a fully virtualization of functionalities, except from those ones that cannot be detached from the hardware elements of the network, e.g., antennae. However, and as foreseen in this section, even the "non-virtualizable" components are considered as PNFs that can operate in consonance with the VNF composing the NSs. Chapters 4 and 5 of this thesis will show how to exploit the flexibility of NS split into VNFs to maximize revenues, and meet 5G KPI.

1.3 Multi-access Edge Computing (MEC)

With the arise of latency sensitive applications, the networking community popped up the idea of bringing the computing resources nearer the end user. Typically this is known as the edge of the infrastructure, i.e., the resources that are closer to the access network – the BSs. Note that a server co-located with a BS prevents the traffic from traversing the access ring, aggregation ring, and core ring. Such bypass translates in a significant reduction of E2E delay, that even if it was in the order of milliseconds, would be critical for URLLC services, as it is the difference in between violating the latency requirements or not. There is not a strict definition of what is the edge in an operator network infrastructure, indeed the edge is frequently divided into (*i*) far edge; and (*ii*) near edge. With the latter referring to those network resources nearer to the BS, and the former corresponding to resources that are pushed deeper in the network infrastructure, i.e., towards the core of the network. It is up to the infrastructure owner to decide whether they want to go for far or near edge deployments, and such a decision is governed by the requirements of the services provided to the end-users. For example, V2X services sometimes have to offer one way delays below 5 ms (see [ETS18a] service requirements for remote driving), and this means that upon a ~ 4 ms transmission delay there is only 1 ms left to tackle both the packet transmission towards the server, and its processing. This might be unfeasible for some services, that would require a higher delay budget for the packet processing. Under these scenarios, Multi-access Edge Computing (MEC) stands up as a paradigm to ease the delay burden in time-sensitive applications, which are specially present in 5G networks, e.g., car platooning, collision avoidance, or hazard warning services. And operator might decide to go for the business market of V2X services and install servers near the BSs along the country most transited highways, so the vehicular traffic is fully supported by means of latency and reliability requirements. Furthermore, these services running in the edge can cooperate with a more localized set of neighboring servers that process cars' traffic that might be about to handover towards a near BS, which is of special interest in vehicular services in which the localization constraints are key to provide a proper functionality.

Motivated by the aforementioned services, and delay requirements in the 5G services, ETSI MEC defines a reference architecture [ETS19c] to provide MEC functionalities in a network deployment. The idea is to set up a set of functional blocks handling the different tasks required in the MEC paradigm, i.e., it does not just state that an application server must be nearer the end-user. Note that MEC initially stood for "Mobile Edge Computing", however, and due to the heterogeneity of the Radio Access Technology (RAT), the paradigm acronym shifted the name to embrace the possibility of supporting multiple radio technologies. Thus, the architecture definition pays attention to the connectivity of the RAT where the user establishes an E2E connection with the MEC service. The reference MEC architecture considers as well the orchestration of multiple MEC hosts across the network infrastructure. MEC hosts are envisioned entities with computational and network resources to run an application on top. ETSI MEC reference architecture decides if an MEC host (a server near the edge of the network) is more adequate than another one to execute the functionality of a given service, e.g., the already mentioned collision avoidance service. Moreover, ETSI considered the chance of cooperating among several MEC hosts for the sake of sharing critical information that might be of importance in certain applications, such as location of end-users or even radio signal strength received by the end-users that are being attended by an MEC host. On top of these, MEC reference architecture gives mobility support by definition, as one of the first main goals of the specification group was to provide of full support high mobility services. This means that handover and traffic rules to control the flows is supported by the architecture, and there is a management aspect that controls the incoming requests, so as the lifecycle management of the running applications. Last of all, MEC reference architecture offers a support of a OSS that may interact with the architecture to ask for the instantiation and management of services deployed upon its requests. However, it might be even and end user the one requiring the instantiation of a given service. And last of all, given the arise of NFV, the ETSI MEC considered the possibility of using virtualization for both the computing resources and network resources, and it envisions a

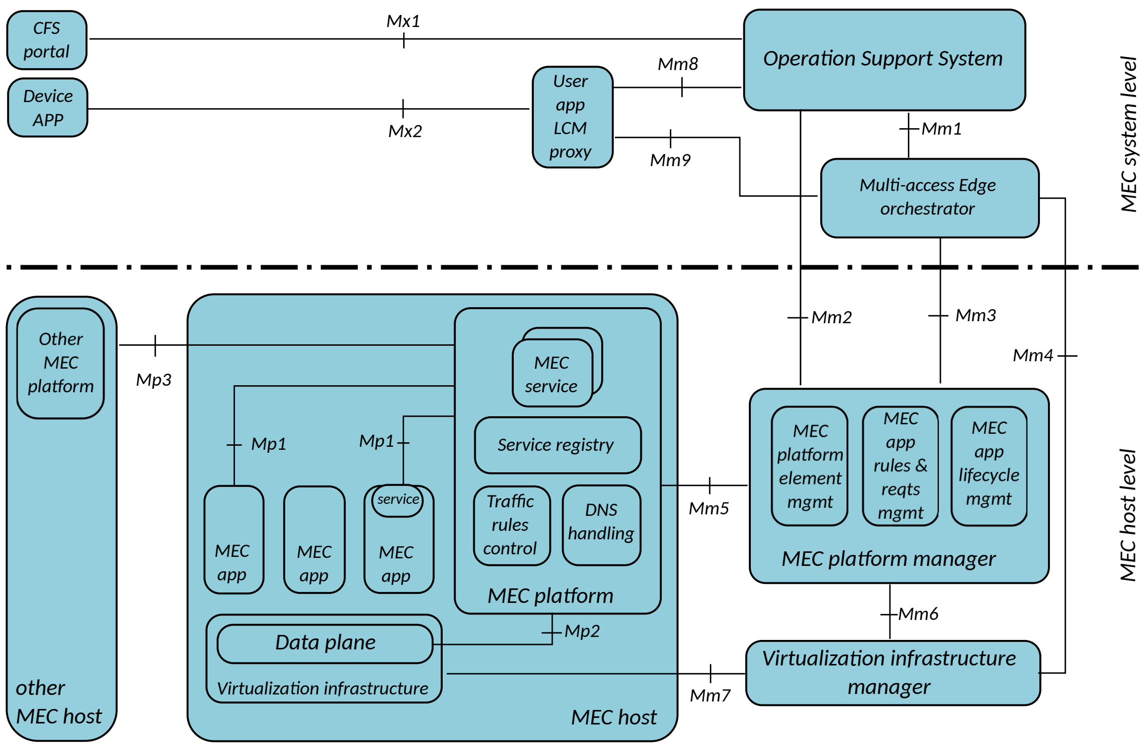


Figure 1.7: ETSI MEC reference architecture [ETS19c]

management entity responsible of dealing with it.

In the following, this thesis section overviews the key components of the ETSI MEC reference architecture (see Figure 1.7), and enters a bit more into details on their functionality. Before doing that, there are some concepts to be enumerated. The first one is a MEC app, which is an instance of a network functionality provided by the MEC infrastructure, e.g., a MEC app might be a data filter to process raw data from users before storing it in a database. The second one is a MEC host, the entity that runs a MEC app. Given these two key terms, lets proceed and go for the overview of the functional elements of the architecture in the next two paragraphs.

First of all it is the already mentioned term, i.e., the MEC host. This functional element might be envisioned as a server holding the successive functional elements as the (*i*) virtualization infrastructure; (*ii*) MEC platform; and (*iii*) MEC apps. That is, the MEC host is the server holding all these functional blocks; so it really is a server in the edge of the network to run a considerable amount of tasks done by a MEC architecture. As in the NFV paradigm, there is a functional element – the VIM – responsible of keeping track of the availability of virtualized resources as virtual CPUs, so as the data plane resources offered by the network portioned managed by the MEC host holding this functional element. On top of it, there is a MEC platform that interacts with the virtualization infrastructure to provide the necessary traffic rules for traffic control in the data plane. Additionally, it holds a registry of the applications of the services provided by the different MEC apps running inside the MEC host. At the same level of the MEC host, there are both the MEC platform manager, and the virtualization infrastructure manager, i.e., the VIM already mentioned in section 1.2, which performs the management of the virtual infrastructure resources offered by the virtualization infrastructure inside a MEC host. On top of the VIM there is the MEC platform manager, which does the management of MEC platforms, and handles the lifecycle management of the various MEC apps running in a MEC host, for example, to scale them up/down to meet the requirements upon peaks of traffic demand of the services offered by each of the existing MEC apps. As final remark about the MEC host level functional blocks, it is worth mentioning that all the MEC platforms running inside hosts, can communicate among each other to coordinate on tasks such as the migration or power-on of neighboring apps that may need to be activated upon users' handovers in between RUs associated to different MEC platforms.

On top of the MEC host level, ETSI defines the MEC system level, where the OSS and multi-access edge orchestrator reside. The OSS receives the requests of instantiating apps from the Customer Facing Service (CFS), and then approves whether such operation is accepted to be processed by the MEC architecture. But for this thesis, the component of highest importance is by far the MEC application orchestrator, which deals with the monitoring of the overall resources available in the underneath virtual infrastructures that are managed by the MEC reference architecture. This implies to be aware of the resources across the different MEC hosts deployed in the network infrastructure, a quantity that varies depending on the seize of the edge deployment of a network operator. Moreover, the MEC orchestrator is responsible of triggering the instantiation and termination of the different MEC apps in the system, and to validate the integrity of those apps that where asked to be deployed. Its last, but most important duty – for the sake of the thesis, as mentioned – is the orchestration of the resources and incoming MEC apps, that is, taking the decision of which MEC host must run each of the MEC apps to satisfy the latency, or even location constraints of the service to be provided. Note that such a decision must take into account numerous factors as energy consumption, deployment cost, and of course, the fulfillment of service requirements as reliability, or latency constraints. The election of the MEC hosts to hold the MEC apps, relates to the more generic election of servers to host the VNFs of a NS. Such a problem is the topic of discussion of this thesis, and chapters 4 and 5 discuss in detail possible solutions to such a problem, which is of special complexity, and well-known in the current literature; with the additive complexity of having to stick to strict service requirements that have not been of crucial interest up until the appearance of the 5G services, and their requirements. Thus, the orchestration algorithms present in the MEC orchestration functional block, relate to the ones discussed in chapter 5 of the present thesis.

Another relevant enhancement brought by the MEC paradigm, is the offloading of computational capabilities towards the edge of the network. With the increase of bandwidth in 5G technologies – as 4K video streaming – the network infrastructure faces a dimensioning challenge to support the promised traffic rates. Note that a bandwidth increase do not only suppose the need of increasing transmission rates in the access network, but as well an increase of fiber lines to assume the amounts of traffic that are later aggregated in higher rings as the aggregation, or core ring. In some cases as video streaming services, there is the possibility of locating a virtual CDN in a Point of Presence (PoP) nearer the end-user, or in other words, in a MEC host. This prevents the end-user traffic flow from traversing the whole network operator ring hierarchy, alleviating the bandwidth burden of the network. However, the computational offloading does not only finishes in a MEC host, indeed, it goes down to the end-user. For instance, there are numerous applications in which the user device might ask for the offloading of computational tasks on the edge. That is, the end-device might not have enough computational capabilities to perform a given task that a MEC host can handle. This is the case of Augmented Reality (AR)/VR applications that might be quite demanding to perform physic simulations in the device holding the application. In such a case, a MEC host might run these physic simulations on demand, and then forward the results back to the end-user of the AR/VR application. [ETS18c] gives a list of use cases in which the computational offloading is a must to ensure the performance of the applications. Among the examples there is the already mentioned AR/VR applications, IPTV offloading, Artificial Intelligence (AI) task offloading, or multi-RAT offloading.

Nevertheless, not all examples provided in [ETS18c] correspond to offloading use cases of the MEC paradigm. There is a wide variety of use cases that benefit from it, ranging from smart industries, to localized services provided in events like sport events in stadiums. In the latter example, the stadium might have operated drones doing video transmission of different angles recorded during the match. The video transmission might go over different RAT technologies – the reason of "multi-access" as first word of the MEC acronym – towards a video processing server, which is deployed in a MEC host near the stadium where the event is taking place. This MEC host

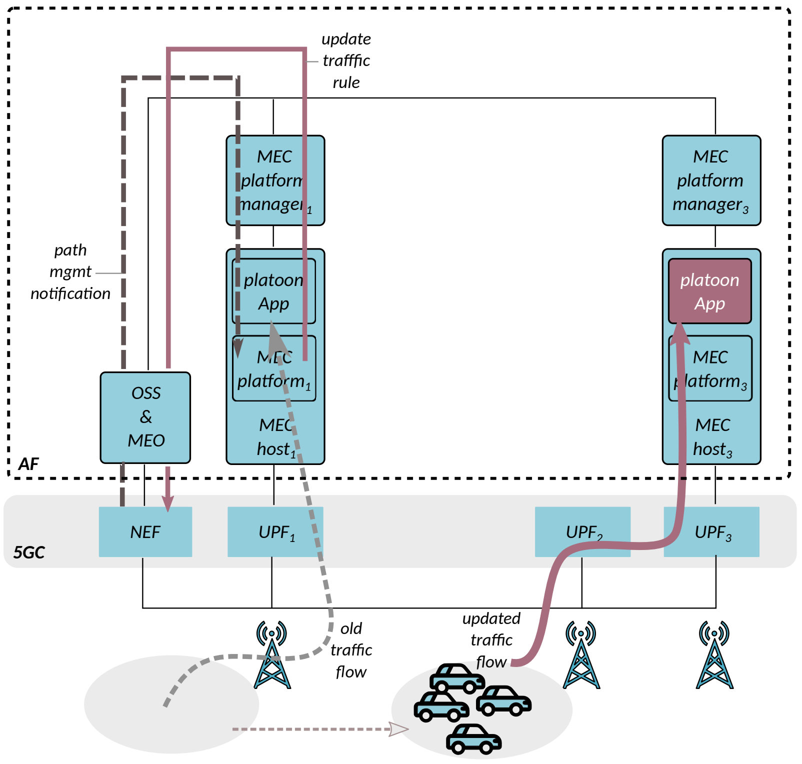


Figure 1.8: MEC traffic rule update in a V2X platooning service over 5G

could handle video encoding/decoding operations to later redistribute content across the match assistants inside the stadium, that could check on their mobile phones the different angles recorded by the drones of a polemic moment during the match. This example embraces the requirement of localization constrained services (the MEC host must be present nearby the stadium), and low-latency requirements (by deploying the service in a MEC host, it is ensured the low latency towards the end users). The reader might notice that in a sport event it is of high importance for the user experience, that a replay does not arrive with a considerable time delay with respect to the polemic moment. If the video was processed in a cloud server, for later redistribution of the processed video to the stadium spectators, then the video transmission would suffer from such delay. Works as [Fra+17a] consider this example for research in the context of the 2020 Japan Olympic games[2].

[ETS18c] also mention not so service-oriented benefits that the MEC paradigm brings to the 5G networks. Particularly, it considers the category of network performance and Quality of Experience (QoE) improvements, a set of scenarios in which the network leaves out the burden of having to steer vast amounts of traffic to the cloud even when it could be processed in the cloud; and scenarios in which the multi-RAT knowledge helps the network management. A concrete example is the usage of RAT metrics to improve the coordination in between the backhaul and access network of a network operator. [ETS18c] states that in given situations, the backhaul network might present a considerable degradation (specially if we think of the IAB technology mentioned in [3GP18c]), and the network edge has to react accordingly. Upon such a degradation, the MEC architecture can proceed in several ways as redirecting non delay-sensitive traffic towards non deteriorated backhaul flows, or even asking a gNB to decrease its microwave capacity due to the congestion existing in the associated backhaul link. In the latter case, energy efficiency is achieved without worsening the already damaged QoE due the backhaul deterioration.

To explain the interaction between the MEC architecture and the 5G network, ETSI specifies in [ETS20b] a description on how the MEC is envisioned as an AF of the 5G network described

[2] Due to the pandemic of COVID-19, the proposed solutions could not be tested.

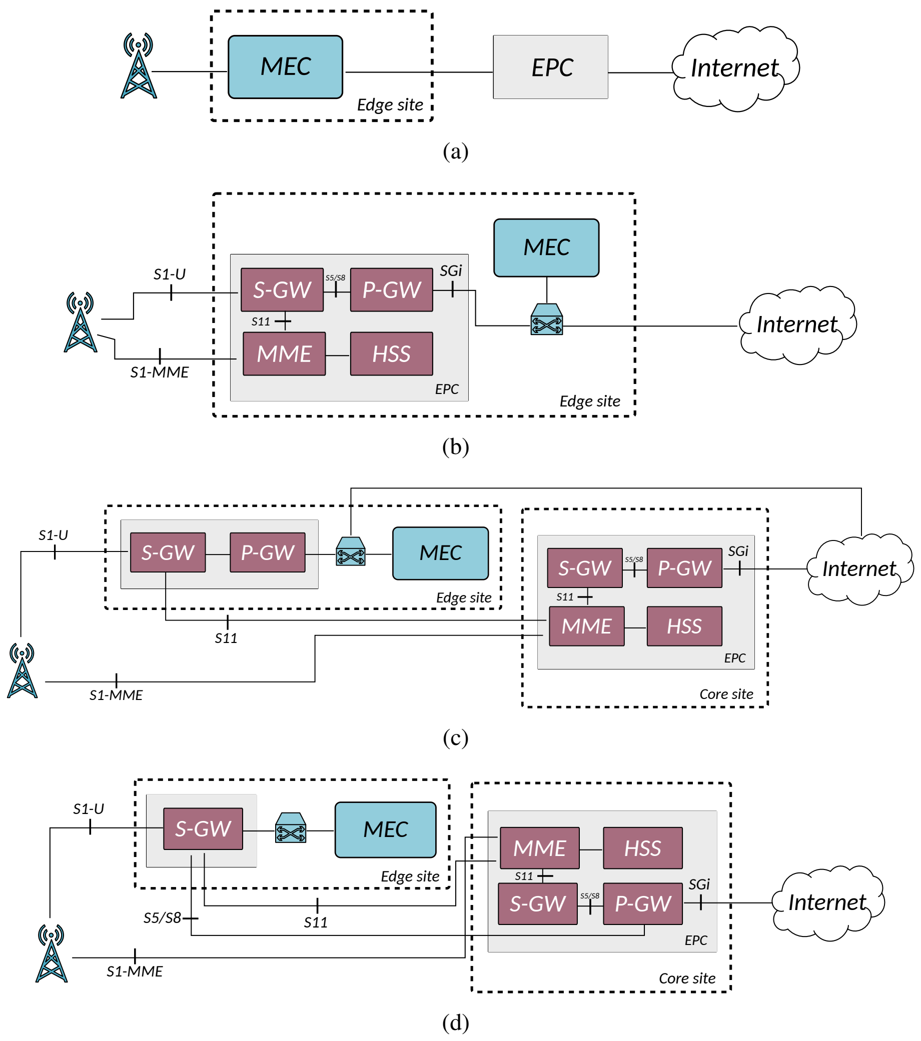


Figure 1.9: Possible MEC deployments in 4G: (a) bump in the wire; (b) distributed EPC; (c) distributed SGW & Packet Data Network Gateway (PGW); and (d) distributed SGW with local breakout – taken from [ETS18b]

in 3GPP standardization documents. A common interaction in between 5G and MEC will be the update in traffic rules of existing flows in the 5GC. Lets imagine that vehicles traveling along a highway use a car platooning service offered by the 5G network – see Figure 1.8. The platoon App will run inside the MEC platform, which is envisioned as an AF for the 5GC. Upon the handover of vehicles among RUs, the NEF will trigger a path management notification that will reach the MEC platform of the first MEC host. The latter will realize that it has to trigger a traffic rule update to maintain the strict delay requirements of V2X services. Consequently, it will ask the MEC Orchestrator (MEO) to select which is the nearest MEC host with a running platoon App, and it will ask the NEF to update the traffic routing so the flow travels towards the selected MEC host (number 3 in Figure 1.8) when the handover is completed.

In the above mentioned example, the MEC is envisioned as an AF requesting the UPF re selection to the 5GC. However, more details should be provided regarding the implementation, and even interaction in between components of the described workflow. To this extent, 3GPP talks about NEF supporting a Common API Framework (CAPIF), a common point to ask for services using an unified Application Programming Interface (API) where there are consumers and producers that may ask or publish content, or even interact among them to change the system status (as in the example of the UPF re selection). The CAPIF context has immediate applications in the MEC architecture, as services offered by MEC apps have to exchange their information with the external

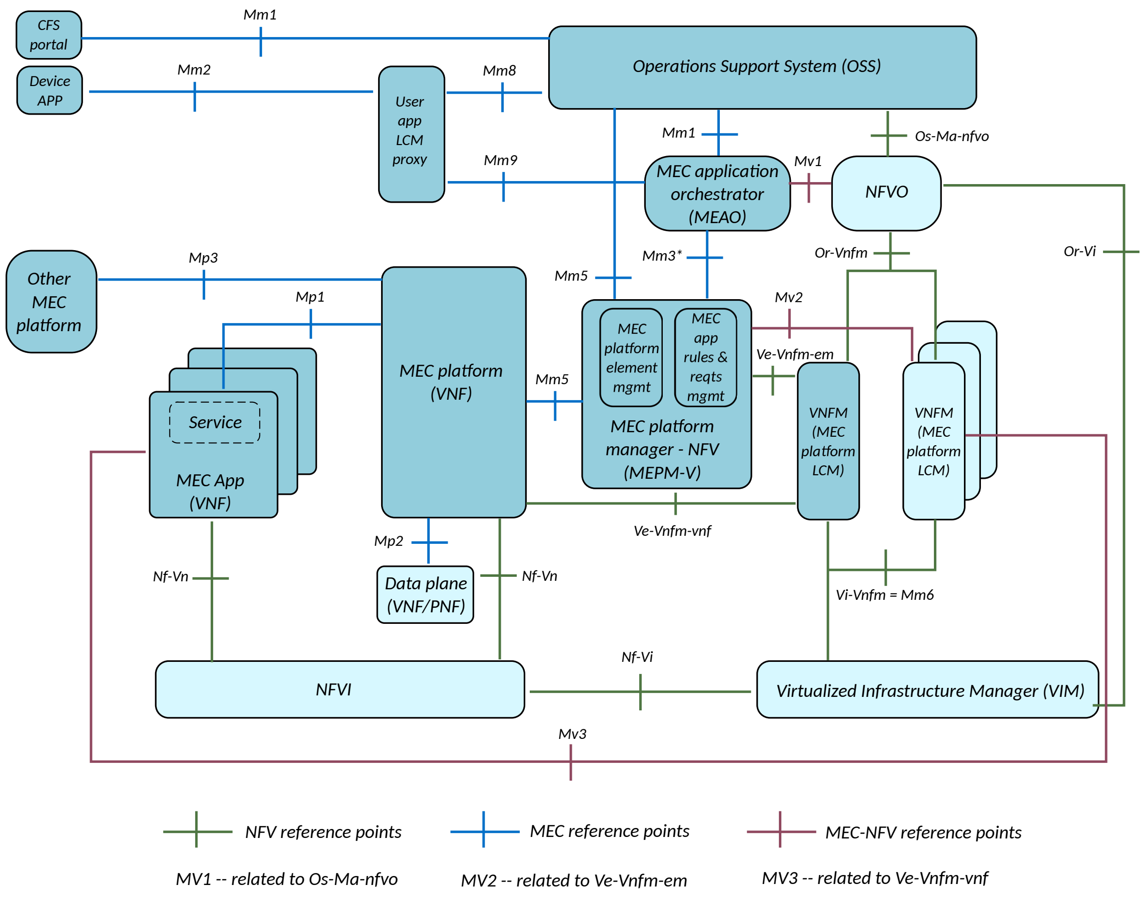


Figure 1.10: MEC architecture in NFV – taken from [ETS19c]

entities making use of them, e.g., a localization service has to report periodically where a specific user is, and a car platooning app would ask for such location information via the CAPIF. Moreover, the MEC platform can expose the registry of available MEC services using the 3GPP CAPIF. Thanks to it, MEC has an unified way of not only exposing its capabilities, but as well, a unified approach to operate in the 5G network.

However, it is worth mentioning that MEC is not only restricted to be deployed in 5G networks, as the paradigm apply to 4G networks, as discussed in [ETS18b]. The white paper describes the different deployments of a MEC architecture that can coexist with the 4G EPC. It mentions how the edge site – that is, where the MEC host is deployed – can be either in site at the access network, or somewhere in between it and the core network, i.e., at some point in the aggregation network. Obviously, the nearer it is of the access network, the lower latency, but as well the more expensive deployments. But this still does not mention how to coexist with the EPC of 4G networks. [ETS18b] specifies the (a) bump in the wire; (b) distributed EPC; (c) distributed SGW & PGW; and (d) distributed SGW with local break out – see Figure 1.9. Without entering into details, these are four different manners of deciding which elements of the EPC will coexist with the MEC host in the access network. In these deployments the EPC and MEC platform can run under a same virtualization infrastructure, and the local break out options prevent the traffic to traverse the backhaul to reach the central EPC entity responsible of session management tasks. This way, strict delay constraints are met, and it might be feasible to achieve them without having to deploy NR RUs that speed up the transmission in the access network. Additionally, there is a packet filtering enhancement that the MEC brings to the 4G networks, and that is in the (*iv*) distributed SGW with local breakout. Moreover, it brings the support of MEC host mobility, or pushing applications requiring paging functionalities for ultra low latency services.

In the above paragraph it was mentioned how the MEC architecture can coexist with the EPC.

This means that both elements may run over a virtualization infrastructure, so the EPC and MEC host elements are running as VNFs. This requires to blend the components defined in the ETSI specification for the NFV paradigm [ETS14], with the functional elements of the ETSI MEC definition in [ETS19c] – see Figure 1.10. The first clear common point in both architectures is the NFVI, that is present in the MEC reference architecture as the compound of virtual computational and network resources. Note that this maps exactly with the NFVI building block mentioned in the NFV paradigm presented in section 1.2, thus, the possible exchange of these components. Consequently, the VIM is a necessary component in both a generic NFV and MEC architecture, as both require the management of the virtual resources. Such virtual resources would be occupied by a VNF holding a MEC app, or part of a MEC app functionality. In particular, [ETS19c] says that a MEC app would map to a NS as the one depicted in Figure 1.5 of section 1.2. In a plain MEC architecture, the management of the MEC apps is performed by the MEC manager, which is virtualized in one or several VNFs to assess the traffic rules and MEC apps management. Additionally, the MEC platform, which directly interacts with the MEC apps, is deployed as a VNF in the described NFV deployment of the MEC architecture. And the lifecycle management of the MEC apps is performed by one or several VNFMs of the NFV architecture. Last of all, the mapping in between a MEC architecture and a NFV infrastructure would conclude with the coexistence of both the MEC app orchestrator and NFVO to perform the decisions of allocating and mapping resources for incoming VNFs.

With the end of this section, the present thesis concludes the overview of the MEC paradigm, which is a fundamental element in the 5G networks. Thanks to the MEC standards, the network community has a unified approach to deploy computing resources nearer the edge, having a well defined coordination with the already deployed network architecture, whether it is a 4G, or 5G deployment. The result of bringing closer to the edge the computing resources, is the possibility of offering lower latency to delay-sensitive services that belong to the 5G use cases, as the already cited V2X services. Last but not least, is the network offloading that MEC offers to the network deployments. As having resources closer to the end users allow to assess tasks that alleviate both the computing, and flows burden to the core network. And as final remark, given the key role that virtualization has in modern network deployments, the MEC reference architecture has been designed so it can be integrated with the NFV paradigm. In what concerns this thesis, MEC is a key enabler to map VNFs in those computing resources closer to the edge, so as to satisfy strict latency and reliability requirements.

1.4 Fog

The previous section of this thesis reviews how the 5G networks have the chance of benefiting from the MEC paradigm to meet the services requirements. The core idea is that both computing and network resources will go closer to the end-user, hence, a latency reduction. Servers will move towards the edge in the network infrastructure, and traffic of end-users is leveraged there whether it is to meet latency requirements, or to offload the computing tasks that go up to the cloud. But it would be a mistake to only assume that computing capabilities exist as a matter of servers, as we traditionally know them. Indeed, the cell phone that we hold in our hands have computing capabilities, and their computational power is increasing as years pass. Not only that, but single-board computers like Raspberry Pis, or connected robots with computing capacity, also belong to the networking infrastructure. Such devices have less computing resources than commodity servers, however, they typically have similar capabilities and architectures, as it is the case of the aforementioned Raspberry Pi single-board computers.

All these devices that are beyond the edge of the network, at the very last hops, if not the last, of the network infrastructure; are known as fog devices. The term fog refers to a blurry continuum of resources where everything in the end of the network with computing capacity, is envisioned as a serving entity of the infrastructure. The fog paradigm considers that both cloud racks of servers, and a single-board computer with internet connectivity, belong to a pool of computational resources in which there is no distinction of where the device is within the network operator hierarchy. It is only required to have information on what the networking and computing capacities of devices are, so as its level within the network hierarchy. Thanks to this, the pool of resources is widely increased with the addition of small, but numerous low-capacity devices connected to the network. And that gives the chance of splitting tasks among numerous devices, having computational presence in more locations than in traditional deployments, and the chance of measuring a wide variety of metrics that range from pedestrian presence, to humidity conditions in farming areas.

The Industrial Internet Consorcium (IIC) has put effort in defining what is the fog, so as the use cases, and a reference architecture [Con17] where the consortium explains the advantages and functionalities brought by the fog concept. The whole idea proposed by IIC relies on envisioning each computing element as a fog node, that is an entity holding storage, computing, and networking capabilities; and even hardware acceleration capabilities as for the case of FPGAs present in single-board computers. An example of a fog node would be a Raspberry Pi single-board computer controlling a moving robot that has luminance sensors, connected with a NR RU, and holding a log of movements in a memory card. Note that this fog node does not only require to manage whether its computational resources are enough or not to perform a task as reporting the luminance across a street to a cloud server. If the robot is responsible of such a task, then what is required is that the fog node manages (*i*) its storage resources; (*ii*) the networking considerations, including whether it is wired, wireless, the coverage, an channel status; (*iii*) the security of the communications it handles, to prevent malicious traffic intrusion; (*iv*) its accelerator resources as GPUs, FPGAS, etc.; and (*v*) its computational resources, how the node capacities are already at their maximum, and whether it can handle the computational operations without running out of battery. Although the previous example of a moving robot managed by a Raspberry Pi is envisioned as a fog node, it might be the case that a fog node has associated numerous actuators, or sensors that send it their data. This is of great interest because many actuators or sensors implement very basic functionality that do not go beyond the task execution, e.g., report a humidity measurement. If that is the case, then the fog node holding the multiple connections across actuators or sensors, implements an abstraction layer that allows the management of all devices underneath, or just the single device associated to the fog node.

Fog devices are just entities with enough computing resources to hold other network nodes' functionalities. Typically the CPU and memory of fog devices is limited, as they assess low demanding tasks like periodic reports of sensor data. Thus, they tend to be not that "smart" to take autonomously management decisions. This raises the necessity of doing a proper management of fog devices so they recover upon failures, and make a more efficient usage of power and resource consumption. To ease the management of fog devices, typically there is a Hardware Platform Management (HPM) device collecting all the device measurements, as peaks of temperature in the processor, to take actions accordingly and mitigate possible failures. For example, a HPM device would detect that the temperature of the single-board computer is increasing substantially, and it will trigger an energy-saving mode that would limit the processing capacities of the device. Such an energy-saving would even have counter-effects in the network transmission, as the device antenna Tx/Rx may decide to lower the transmission scheme to save battery, and hence, decrement the bandwidth capacity of the device. Both the computational and networking side-effects, are of special interest in this thesis – see section 5.2 – as it comes as a consequence of the devices limitations in the fog, something critical for the orchestration of services with very demanding constraints.

On top of computational and networking constraints, there exists the battery constraints that many fog devices present. Many of them might be powered by sun light and have a limited autonomy of few hours without the required charging, either by plugging the device to a power supply, or an external power source. Whenever a battery supplied fog device runs out of power, it is not only that it stops executing the task it has associated, e.g., an irrigation device; but it also will loose its connectivity towards the device controlling it, or simply collecting its information. Following the example, an agriculture NS with a irrigation control system, may loose connectivity with the fog device activating the irrigation, with the consequent service failure. Thus, it is of vital importance to keep track of the battery levels of devices to decide if they can perform a specific task. To do this the network should be aware of monitoring information of fog devices present in the network infrastructure. In the case of the reference architecture proposed in [Con17], IIC gives a high level reference that mentions a node management for all the underlying sensors, actuators that might increase in an infrastructure deployment. The consortium explicitly mentions that it is required to know the state of the devices (as battery levels), and the network they are connected to, so the network planning is performed correctly.

Another aspect of crucial importance is the mobility of some fog devices. Note that within the cloud-to-thing continuum that IIC envisions as “the fog”, not every node remains statically in the same location over time. Devices as moving robots may change their location, or even drones may act as fog devices going away from the coverage area of a RU providing internet connectivity. For those situations it is required to keep track of the robot mobility patterns, and such aspect is as well considered in the wireless connection considerations of the IIC reference architecture [Con17]. The mobility aspect involves the location information of fog devices across the network, as the mobility comes as a consequence of the change of location itself. Hence, the location reporting and tracking becomes as crucial as knowing or inferring mobility patterns of the fog devices. Services with location constraints, as a mobile robotics service for warehousing [3GP20l], require to be deployed in-place at the factory facilities, i.e., only robotic fog devices in that location can host the service functionalities. Indeed, section 5.2 of this thesis formulates the service orchestration problem so it meets the location constraints that may apply. However, it is not enough to ensure that the fog devices are in-place where the service happens, as it is of same importance the existence of communication in between the fog device – as the mobile robot – and the internet. To ensure the connectivity, the radio coverage is another aspect IIC points out in [Con17], as it is what determines whether a fog device is detached from the operator network infrastructure. Therefore, the coverage of fog devices must be considered in the orchestration of a NS to meet avoid loss of connection (see section 5.2 of the present thesis to check how to consider both mobility and coverage constraints in mobile robotics scenarios).

On top of the battery and mobility challenges introduced by fog devices, there is the wireless channel conditions of the medium fog devices are attached to. For instance, a non-reliable wireless channel should be neglected for usage of fog services as fire alert sensors, as a communication failure may lead to deaths. IIC also considers for the channel reliability as a requirement in the communications, and that may be expressed by means of packet loss, or channel downtime. Either it is one metric or the other which determines the channel reliability (packet loss is considered in section 5.2 of this thesis), there must be a monitoring of the channel/network status, as pointed out by the in band node management in IIC reference infrastructure [Con17]. Thanks to such information, the deployed fog services ensure a reliable communication towards the cloud, or neighboring fog nodes. In the fire alarm example mentioned before, it might be that a heat sensor decides to connect to an exterior LTE RU rather than a Wi-Fi access point, so the communication is more reliable upon a fire in a house.

Due to the device heterogeneity in the fog paradigm, the computational resources range from sensors with no computational capacity, up to cloud racks with considerable computing resources. Not only that, but the positioning of the resources within the network is different as well, since sensors or IoT devices are typically co-located next to the end user, before the jump towards the access ring, whilst cloud resources are pushed towards the core of the network. Hence the channel and medium heterogeneity mentioned in the previous paragraph, is not only concerning the nature of the wireless link, but as well if the connectivity is wired, and what kind of wired connectivity, which in most cases end-up being optical networks as we go up in the network hierarchy. Thus, a whole fog architecture comprising all the rings of the network operator, require the management of the vast heterogeneity of the network.

The management of a fog architecture is eased by using the NFV paradigm of section 1.2, as it allows having a high flexibility in the management of the services running on top of a network architecture. For instance, the concept of splitting a NS into VNFs is specially useful in the context of fog architectures. For example, the aforementioned irrigation service for a farm benefits from the VNF splitting, as the service is broken down into an irrigation control VNF to decide whether each humidity sensor VNF has to trigger the irrigation or not. With such a split, the NS deployment becomes more flexible, and the execution of some of the VNFs can take place among different computing nodes across the whole cloud-to-thing continuum of the fog architecture. In the example above, the irrigation control mechanism could be running either in a cloud server, or in a single-board computer with connectivity to the humidity sensors. It is up to the deploying agent to decide where it wants to deploy each of the VNFs of the service.

The deployment and assignment of VNFs to servers is of crucial importance, and the decision on how to steer the traffic of the fog services is not a trivial task. If both decisions are not done correctly, the deployment of the fog-based service may not meet the imposed latency, reliability and computational constraints; not to mention the possible battery, mobility, or even coverage constrains already discussed. If the NFV paradigm is used for fog deployments, the elements of the whole architecture overviewed in section 1.2 are reusable in any fog deployment. For instance, a VIM and NFV orchestrator are specially useful to tackle the assignment problem stated by the beginning of this paragraph. Inside the NFV orchestrator there would be an algorithm to decide the assignment, and it would be an algorithm as the ones solving the problem solved later in this thesis in section 5.2, i.e., the assignment of resources in fog environments, where both the volatility and fixed location of resources are considered. But note that any assignment algorithm in the fog needs to know the updated information of every parameter introduced by the fog services. Battery levels, channel status, type of device (e.g., a single-board computer), location and mobility are new metrics to be monitored in a VIM managing a fog architecture. [Con17] already mentions that all these metrics should be monitored, however, its high-level architecture does not get into the details on how to tackle it. The 5G-CORAL European project [Mou+18] describes precisely a whole ETSI NFV-alike architecture to assess the management and orchestration of resources in fog environments. The project proved that it is feasible to extend the NFV paradigm to tackle fog scenarios, and provide an end-to-end solution capable of managing and orchestrating services for the cloud-to-thing continuum. Indeed, the 5G-CORAL project supposed the inflexion point in the development of a publish/subscribe protocol for the exchange of information in fog network environments. This protocol is called zenoh [Ang19] and it serves as communication overlay

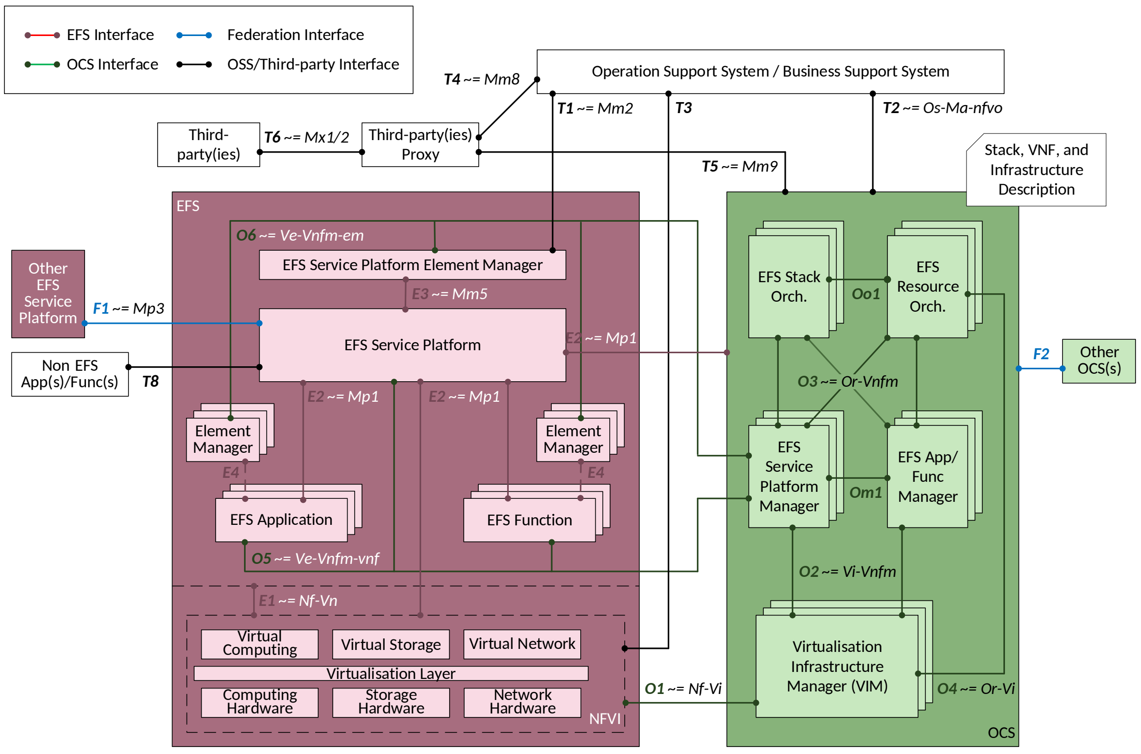


Figure 1.11: 5G-CORAL architecture – taken from [Mou+18]

for the [CB18] infrastructure, which envisions the whole network as a server-less cloud-to-thing continuum of networking and computing resources. The implementation of the infrastructure is a perfect blend of the NFV paradigm into the fog ecosystem, and it accounts for less heavy virtualization technologies as unikernels [Oli+19].

The architecture design of 5G-CORAL [Mou+18] is based on the ETSI NFV architecture [ETS14], and it accounts for the required VIM to handle the heterogeneity of computing devices by means of virtualization. Furthermore, the reference architecture of 5G-CORAL does not restrict its components to run in the cloud or edge, as it considers that any computational resource is capable of hosting architectural components as long as it has enough capacity. The 5G-CORAL envisions two main components: (*i*) the Edge and Fog computing System (EFS); and (*ii*) the Orchestration and Control System (OCS). With the first one being a logical block containing management of an administrative domain edge or fog resource, and a point to provide services. And the latter being acting as an orchestrator holding the actual VIM functionality of the ETSI NFV architecture. Additionally, the OCS orchestrates and manages the NSs deployed in the platform, which are referred as EFS stacks. As mentioned earlier, the 5G-CORAL has the advantage of allowing the EFS to run in any computing node, and it is an architecture design that considers the volatility, location constraints, mobility, and Device-to-Device (D2D) communications among fog devices as single-board computers. Moreover, the project has presented its applicability to use cases of fog-assisted robotics [Sam19], where 5G-CORAL managed the deployment and management of coordinating robots for tasks as warehousing.

As the reader might have noticed, the 5G-CORAL architecture – which is an actual proof-of concept of design principles stated in the OpenFog high-level reference architecture – considers the computational resources as the cloud-to-thing continuum, i.e., as the convergence of cloud, edge, and fog resources. It does not only focus at the very end of the network infrastructure. Indeed, another of the use cases proposed by the project was the usage of the 5G-CORAL architecture to provide an infotainment service for traffic jam situations, that is, cars/passengers act as fog devices that broadcast content they have cached among other end-users co-located in the traffic jam. Note this is an example in which multiple RAT technologies are considered, namely, the D2D

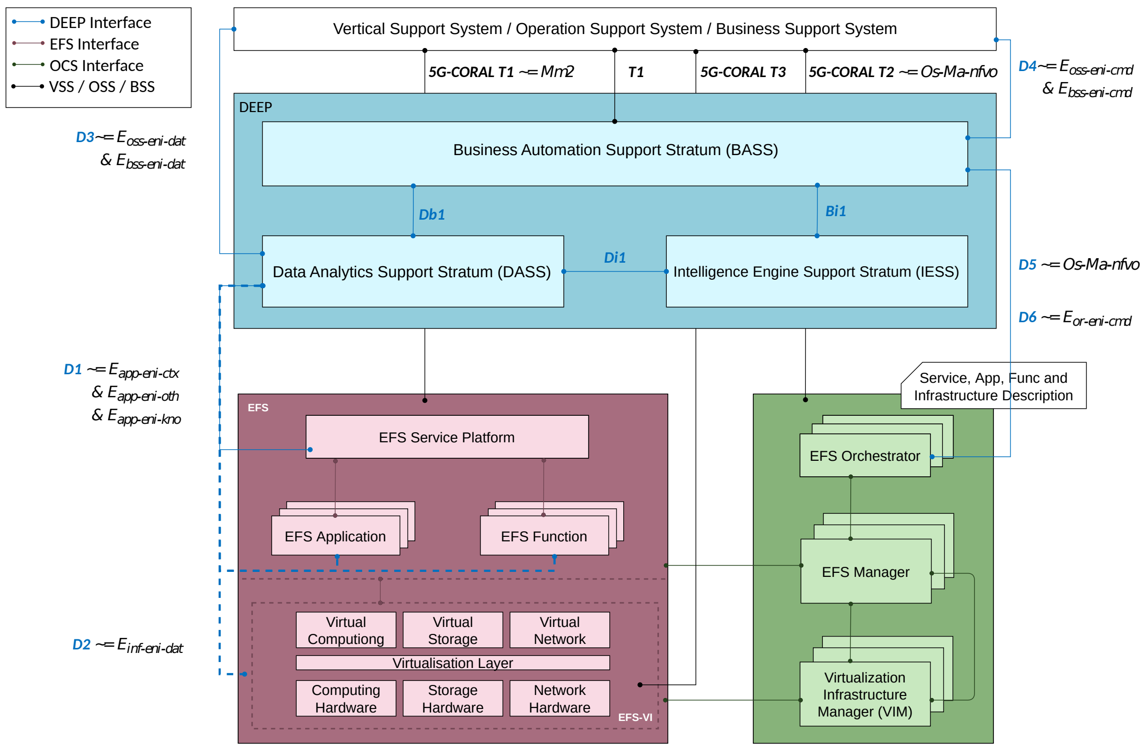

DEEP Interface
EFS Interface
OCS Interface
VSS / OSS / BSS
Vertical Support System / Operation Support System / Business Support System
5G-CORAL T1 ~= Mm2
T1
5G-CORAL T3
5G-CORAL T2 ~= Os-Ma-nfvo
D4 ~= $E_{oss\text{-}eni\text{-}cmd}$ & $E_{bss\text{-}eni\text{-}cmd}$
DEEP
D3 ~= $E_{oss\text{-}eni\text{-}dat}$ & $E_{bss\text{-}eni\text{-}dat}$
Business Automation Support Stratum (BASS)
Db1
Bi1
Data Analytics Support Stratum (DASS)
Di1
Intelligence Engine Support Stratum (IESS)
D5 ~= Os-Ma-nfvo
D6 ~= $E_{or\text{-}eni\text{-}cmd}$
D1 ~= $E_{app\text{-}eni\text{-}ctx}$ & $E_{app\text{-}eni\text{-}oth}$ & $E_{app\text{-}eni\text{-}kno}$
EFS
EFS Service Platform
Service, App, Func and Infrastructure Description
EFS Orchestrator
EFS Application
EFS Function
EFS Manager
Virtual Computiong
Virtual Storage
Virtual Network
Virtualisation Layer
Computing Hardware
Storage Hardware
Network Hardware
EFS-VI
Virtualization Infrastructure Manager (VIM)
D2 ~= $E_{inf\text{-}eni\text{-}dat}$

Figure 1.12: 5G-DIVE architecture – taken from [Sol+20]

communication to exchange content among cars, or the LTE/NR used to communicate with the upper layers of the network, which has a more centralised view of the traffic status in a city. With such example, the 5G-CORAL shows that edge and fog coexist in a network deployment, and the operator does not only have to restrict itself to an architecture in which either the MEC [ETS19c] or a fog architecture are used, both being independent to each other.

The edge and fog blurs then into the cloud-to-thing continuum envisioned in the document pointed out by the IIC OpenFog idea, and that was inherent in the design of the 5G-CORAL architecture, which focused more on robotics and fog-oriented Proof of Concept (PoC) to show the system capabilities. However, the architecture lacked the use of data sources and metrics monitoring to enhance the network management with the use of AI/ Machine Learning (ML) techniques. As a consequence, the 5G-DIVE European project [Sol+20] came as the natural evolution of the 5G-CORAL architecture. The new architecture tackled the cloud-to-thing continuum of the fog paradigm focusing on industrial automation and industry 4.0 use cases, and extended the 5G-CORAL architecture introducing 5G-DIVE Elastic Edge Platform (DEEP), a building block offering automation, data analytics and intelligence engines that improve the network management. Additionally, 5G-DIVE aims to tackle the distributed orchestration that appear in scenarios as UAVs, flying devices that fall in the spectrum of the fog computing, as they are autonomous devices with connectivity towards the internet, or even D2D connectivity among them. As UAVs typically have associated computing capabilities, the 5G-DIVE project considers that NFV-based building blocks, as the VIM inside their OCS element, may run not only inside the edge devices, but as well on top of UAVs. As a consequence, the whole infrastructure goes into a distributed fashion that shares data to the DEEP element in order to perform required analytics to assess tasks as mission control of the UAV fleet, which may be providing video streaming to a specific geographical location. Note that this supposes that the 5G-DIVE architecture must be resilient to the previously mentioned out-of-coverage problem that may come up with the mobility of the UAVs. Such events may be predicted by the DEEP functional block thanks to the trajectory reporting of UAVs to the 5G-DIVE architecture. This is where 5G-DIVE enhances the 5G-CORAL architecture, as the DEEP block allows to autonomously trigger preemptive actions that mitigate the volatility and mobility of the fog devices.

This section has presented the fog paradigm, and how the network edge presented in section 1.3 is extended to the IoT and volatile devices that are not wired to the network. The paradigm expands both the computational and network capabilities, and provides ubiquity with the side-effects of volatility and mobility that are inherent to fog devices. However, an adequate management and virtualization of the resources results into a benefit for administrative domains, as they can provide geographically constrained services, reduce latency, and make use of low-powered devices to tackle tasks that are perfectly feasible for basic devices as single-board computers and robots. European projects have proved that the fog paradigm is feasible, and that it is capable to make feasible use cases related to fog-robotics, and UAVs. The PoCs show that fog is a strong candidate to support robotics, and IoT devices with the help of a fog-aware network architecture.

1.5 Network Slicing

With the appearance of the 5G technology, network services are going to be even more diverse than in the predecessor networks of 4G and 3G. The Internet is no longer restricted only to streaming of video content, websites, or chatting and email applications. New verticals and use cases appear and they must be supported by the 5G infrastructures. Smart factories, mission critical services, vehicular services, or remote surgery are some of the use cases with requirements only met by 5G networks' capabilities. Typically, all the new services supported by 5G networks fall within one of these categories: (*i*) enhanced Mobile Broadband (eMBB); (*ii*) Ultra-Reliable and Low Latency Communications (URLLC); and (*iii*) Massive Machine-Type Communications (mMTC). The first one relates to those services asking for downlink peak data-rates above 20 Gbit/s in the downlink, and 10 Gbit/s in the uplink (according to [ITU17]); and services as AR/VR belong to these category, as the video stream requires high rates so the video corresponding to an user action is not delayed in time. The second one refers to services asking for E2E latency between 1 ms and 10 ms, and vehicular services are great examples, since the notification of a crash in a platoon of vehicles must be fast enough to guarantee brakes activation in order to prevent accidents. The third one serves as a categorical umbrella covering all services that require a connection of a massive number of devices, and it considers scenarios of IoT communications, or communications among a vast amount of automatas in the production lane of factories; which typically require low latency to have high level of synchronization among the factory/IoT devices connected to the 5G network. Use cases belonging to the latter category demand a high density of connected devices, indeed, [ITU17] points out that the 5G networks should satisfy the connectivity of up to 1,000,000 devices per km^2. And not only that, but mission critical services would require a wide coverage for those use cases related for rescuing citizens in far areas.

The presented use cases give an idea of the different services that 5G networks should support. Being brief, some might say that in order to deploy a 5G network infrastructure, it should be enough to just install more gNBs, and additional fiber/paths to handle the increasing network demand. Furthermore, strict latency requirements would be met by using the MEC paradigm (i.e., computing resources closer to the edge). However, it would be inefficient, for example, to assume that all users are going to be using enhanced Mobile Broadband (eMBB) services, and therefore, to have a vast network deployment with uplink data rates of 10 Gbit/s for every user in the network. Rather, network operators have to make a case study on the expected demand of those services that 5G offer, to scale and manage accordingly the network resources. Network slicing arises as a paradigm to assess the management of a network infrastructure in which there is a heterogeneity of service requirements, as the mentioned eMBB, URLLC, and mMTC. As the paradigm name suggests, the idea is to slice the network resources, such that each slice is dedicated to satisfy the requirements of the different services. For example, a sliced 5G network infrastructure would have a slice that steers and processes the network traffic of vehicular services, and such slice would ensure that the packets follow paths such that latency is below 1 ms.

In order to implement network slicing in a network operator infrastructure, it is required that all the available resources are managed in a coordinated manner, namely, the (*i*) radio network resources; (*ii*) computing resources; and (*iii*) wired network resources. With computing resources being the servers or devices processing application requests, and wired network resources being the switches, fiber links, and routing policies of the network. In a network slice, the UE forwards its traffic to the gNB that it is attached to, and it is up to the 5GC to decide the transmission slots granted for the UE traffic. For example, traffic of a eMBB slice would have more transmission slots to satisfy the high peak-rates, whilst traffic of URLLC services would have shorter, but prioritized transmission slots, or even slots dedicated to transmit redundant packets in order to achieve reliability. It is up to the network how the spectrum is managed (see [Aya+20]). Once the RAN spectrum has been managed, the packets are steered across the network to reach the server where the traffic is finally processed. The traffic steering of the slice must be such that it satisfies bandwidth, and latency requirements; and even more, the traffic should be queued/prioritized to not

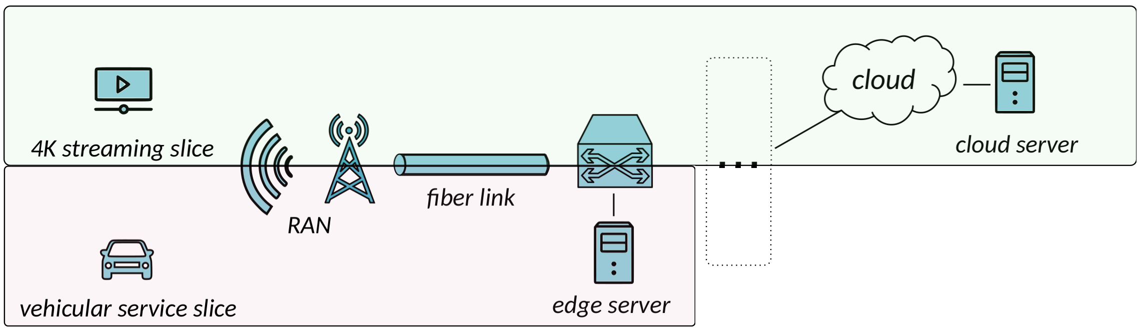


Figure 1.13: Network slicing concept illustration

deteriorate the QoS of existing slices (see [Com+18a]). For example, mission critical traffic should be prioritized over eMBB traffic, as the former services are more sensitive to delay violations than the latter, or at least, the QoS violations may have critic consequences in rescue-alike services. Finally, the traffic steering of the network slice should be such that the correct server is selected to process the traffic of a specific slice. As an example, a network slice for AR/VR should consider servers with GPUs to perform video encoding, processing, and rendering tasks.

By reading the previous paragraph, it seems like network slicing is a blurry approach on how a network operator should "divide" its network. However, the 3GPP details in [3GP20e] the operations required to perform network slicing in a network infrastructure. The document explains that a slice passes through different stages during its lifespan, i.e, since the moment it is created, up until when it terminates. In particular, the 3GPP talks about the (*i*) preparation; (*ii*) commissioning; (*iii*) operation; and (*iv*) decommissioning of a network slice. Before explaining each stage, it is worth mentioning that the 3GPP envisions a network slice as the compound of not only network and computing resources, but as well the set of NFs that are present within the slice, e.g., a video encoding NF within a slice dedicated for AR/VR. With such a definition in mind, the first stage of a network slice lifespan is its preparation, that is, the design of the slice and which NFs are going to belong to such a network slice. In this first stage, the creator of the network slice will design it so as to satisfy the requirements the slice should offer, e.g., to use mobility management NFs for vehicular services. Once the slice is prepared, the network operator has to tackle its commissioning, that is, the creation of the network slice by doing an adequate allocation and configuration of resources. To do so, the created network slice must have enough resources allocated to meet requirements such as low latency, which would imply the usage of servers nearer the edge of the network. The allocation of resources to meet requirements of services belonging to different network slices, is the matter of study of the present thesis, and chapter 5 present solutions to allocate resources for a network slice. Once the network slice is commissioned, it is mandatory to perform a KPI monitoring to check if the network slice services' performance meet the requirements, and the Service Level Agreement (SLA)s. In case there is a violation of a KPI, the network slice should be modified, and a capacity re-planing would be required to mitigate the KPI violation. For example, if a mobile robotics service asks for 3 ms of end to end latency, and the service experiences a latency of 5 ms, then the corresponding mMTC slice should add servers nearer the robots, or increase the RAN resources to meet the 3 ms of latency. How to assess the mentioned capacity re-planing is out of scope of [3GP20e], however, chapter 5 present solutions in which the planning of resources is dynamic depending on the services' lifespan, and thus, it covers such a gap. Finally, the last stage in a network slice lifespan would be the decommissioning of resources, and how to free the network slice resources for future usage of a new lifespan of the same slice, or for the usage of other network slices.

The aforementioned stages of a network slice do not come out of nowhere, indeed, the lifespan and definition of a network slice comes as a Network Slice Template (NEST), a set of attributes and operations that are available within a network slice. The NEST concept is elaborated by

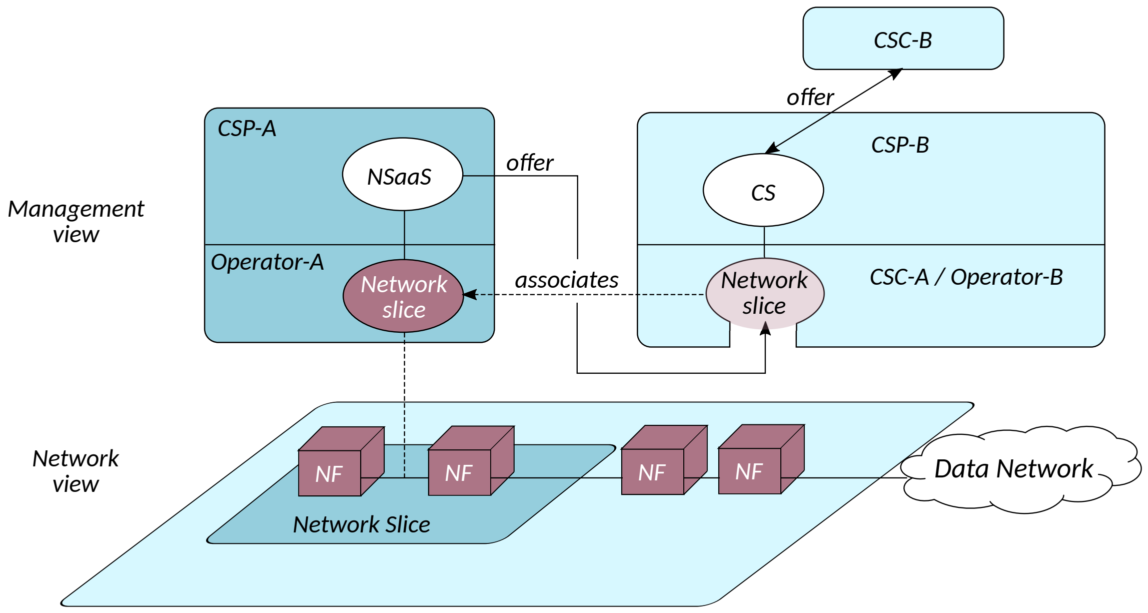


Figure 1.14: Example of Network Slice as a Service (NSaaS) – taken from [3GP20e]

the Global System for Mobile Communications (GSMA) in [GSM20], where the association proposes the Generic network Slice Template (GST), which is nothing but a specification on how a NEST should be like. In the document, the GSMA aligns the template definition with the requirements stated by 3GPP in [3GP16b] and [3GP20j] for 5G networks. Consequently, GSMA defines a GST as a set of attributes specifying how a network slice should work. [GSM20] provides a detailed set of attributes that fall within one of these categories: (*i*) character; (*ii*) scalability; (*iii*) performance; (*iv*) function; and (*v*) control and management related. Upon a definition of a GST, the customer must fill all the attributes and later can interact with the resulting network slice instance by using an API, and the KPI parameters that the network slice may report periodically. A minimalist example of a URLLC GST would be the specification of a 99.999% availability, the continuity support for 1 session, and a device velocity of 120 km/h. Such a GST would then be translated into a 5GC slice profile for the later configuration of the underlying infrastructure, e.g., the RAN resources, transport network, and computing capabilities of the network operator.

Thanks to the GST definition, a network operator proceeds and performs the commissioning of the network slice, and according to [3GP20e], it is able to provide support for services asking for these set of requirements: (*i*) area traffic capacity; (*ii*) charging; (*iii*) coverage area; (*iv*) degree of isolation; (*v*) end-to-end latency; (*vi*) mobility; (*vii*) overall user density; (*viii*) priority; (*ix*) service availability; (*x*) service reliability; and (*xi*) UE speed. Note that satisfying such requirements lets the network operator to provide a wide variety of services within a single infrastructure, if its network slices' resource provisioning is capable of meeting all the different requirements. Furthermore, the underlying slicing of the network might be provisioned as a service – which is known as NSaaS – so third parties can take benefit of the usage of a sliced portion of the infrastructure – see Figure 1.14. In this latter example, [3GP20e] uses the term Communication Service Provider (CSP) to refer to the business entity responsible of the maintenance of the network resources. A network operator, which owns the network resources for the RAN, the access network, switches and fiber links, so as computational resources for some services such as TV over IP; may be envisioned as a CSP. And [3GP20e] states it can either provide services or slices, and the consumers of both cases are known as Communication Service Consumer (CSC). A CSC acquiring a network slice instance from a CSP does not necessarily have to build services on top for its end-users, but it also may decide to create a network slice on top of the existing slice, and offer it for either consumption or management; so as the CSP did with it. Furthermore, a CSC has the chance of using several network service instances' resources – such resources are known as a network slice subnet – to build another network slice.

Although it may seem like the 3GPP just states a list of conceptual ideas regarding the network slicing, it is proven that the organization did not pitfall into only scratching the conceptual surface. A proof of it is that it accounted for the GSMA inputs in its ongoing technical documents. Additionally, the 3GPP has devoted effort to define in detail which are the parameters and information models that define a network slice [3GP20c]. Particularly, the 3GPP states that the information model of a network slice inherits the attributes of a subnet [3GP20m], with the addition of a list of service profiles it is able to support. Each service profile supported by the network slice is defined by the 3GPP as another information model object holding the already mentioned information about the requirements that the network slice should satisfy. Thus, its attributes refer to aspects as coverage, latency, mobility, maximum packet size, or reliability. It is important to remark that it is natural that the network slice information model follows a subnet model, as an instantiation of a network slice will result into the configuration of an associated subnet. In other words, the network slice is nothing but the definition of a subnet comprising computing and network resources that meet the requirements of services.

However, the conceptual idea of slicing the network has not only been investigated by standardization bodies as the 3GPP or the GSMA. The research community has also put effort into defining abstracted network views to offer verticals (or CSC) to deploy NSs without technical expertise in the resource management. 5G-TRANSFORMER [Oli+18] is an example of an implementation of the slicing concept. This European research project developed an end-to-end network architecture that did not only follow the ETSI NFV principles to provide an orchestration solution of NSs; but it also tackled the network slicing of the managed network. The project made a step further than the 3GPP for the accomplishment of network slicing, as it assumed that verticals as video content providers should be able to define their own network slices without that much technical expertise. To do so, 5G-TRANSFORMER proposed VSBs [Chi+19] that are templates in which a vertical specifies the behavior of its service. For example, a video content provider specifies inside a VSB that it wants to deliver live video streaming to a specific geographical area, to one thousand customers, and streaming quality of 4K. The 5G-TRANSFORMER platform then takes the VSB and translates it into a NSD that later is either added to an existing network slice, or put into a completely new network slice. The 5G-TRANSFORMER platform also assesses the orchestration of resources, and the assignment of VNFs and VLs to the underlying resources, which may be allocated to a specific network slice.

Network slicing does not only comprise the allocation and assignment of resources, as it also has to coordinate with the network communications flowing on top of each slice, i.e., the packet forwarding and traffic policies. This implies a coordination of either the EPC or 5GC with the existing network slices. 3GPP Release 16 [3GP20h] introduces features on how the network slices interact with the core of the network to ensure a coordinated behaviour of packet forwarding, and resource management. Since each UE might belong to several slices at the same time, e.g., it might receive/send traffic of a eMBB slice and a V2X slice at the same time; it is of crucial importance the prioritization, signaling, and RAN assignment to the different packets of each slice. 3GPP Release 16 [3GP20h] introduces the Network Slice-Specific Authentication and Authorization (NSSAA) to enhance the UE access to a network slice. In the specification, it is detailed how the AMF, the 5GC NF responsible of the access to the network, interacts with the UE to validate that it has access to network slice it is using. Thanks to this, it is possible to revoke the access to slices to some UEs that might have no authorization to use the slice resources. Note the vital importance of such control in services as remote-surgery operations, as the interference of non-authorized packets would lead to an arrival delay of packets that remotely control a surgery robotic arm, and the corresponding consequences. Another addition that [3GP20h] introduced, is the coordination in between the EPC and the 5GC when a UE moves from the former to the latter. In case the UE uses a network slice, 3GPP Release 16 specifies how to select a new AMF based on the network slice to which the PDU belongs. For example, if a vehicular service UE handovers from EPC to 5GC, the session traffic will be forwarded to an AMF that minimizes the signaling delay, so it does not impact the URLLC of the vehicular service.

Once the network slice is commissioned, and the UEs respect the slice access (thanks to the NSSAA); it is necessary to ensure that KPIs are met in every network slice. This requires a monitoring of metrics in the network, so the operation stage of the lifespan of a slice, handles the correspondent dimensioning operations upon KPIs violations. That is, if a metric as the average delay experienced on a network slice is not meeting the requirements, there must be a way of noticing such event, and that is possible thanks to the monitoring of metrics inside the 5GC. More specifically, the NEF of the 5GC is the responsible of monitoring the required information, that is later used to manage the network slices. However, it is not only the NEF which provides useful measurements for the network slice management, indeed here there are a couple of metrics that one can obtain according to [3GP20d]: (*i*) the Unified Data Management (UDM) provides the number of registered users; (*ii*) the gNB tells the packet drop, IP latency, interface delay, etc.; and (*iii*) the UPF can give the end-to-end delay for a service. According to [3GP20g], the monitoring of network slice metrics is accomplished using Network Slice Subnet Instance (NSSI) monitoring jobs. Remember that previously in this chapter it was explained that a network slice subnet is nothing but the set of underlying networking resources, computing resources, and NFs that are used to run the services held by a network slice. Thus, a NSSI monitoring job performs the collection of metrics to assess the later network slice performance assurance. Given the collected metrics, it is possible to have cumulative counters for metrics such as the amount of active sessions – see [3GP20g] – that are later useful to trigger the corresponding actions. For instance, given that the users reported in the monitoring jog of a network slice exceeds the threshold, it is possible to increase the number of servers in the network slice subnet to meet the users' demand. To perform such an action, [3GP20m] specifies how to assess threshold monitoring, to later perform the corresponding changes in the slice subnet associated to a Managed Object Instance (MOI) via modification, creation, or even deletion operations (see [3GP20f]).

The network slicing concept requires a very specific set of definitions and operational support to achieve network isolation, and flexible management of the different slices. The above paragraphs have given some insights and pointers of the specification documents stating the required operations and interactions for network slicing. Unlike the network paradigms discussed in the previous sections of this chapter, the network slicing idea came afterwards the NFV, MEC, and fog paradigms. As a consequence, its adaption in the 5G ecosystem requires to elaborate how the mentioned paradigms should behave to accomplish the network slicing. It comes natural that one or more VIMs manages the resources of a network slice subnet, or that a MEC app and its constituent VNFs belong to a specific slice. For example, a MEC app for vehicular services will naturally belong to the corresponding network slice. The integration of network slicing in MEC and NFV is treated in detail in [ETS19d], where ETSI details the interactions among both NFV and MEC platforms to assess the realization of a network slice(s). The document presents as well different set ups as holding multiple tenants in a single network slice, or how each slice can have dedicated MEC components running on top of it. The document also reviews how Software Defined Networking (SDN) controllers should handle the traffic for different slices, bringing in the chance of also solving the routing and packet forwarding challenges related to network slicing.

To sum up, network slicing brings is the possibility of sharing the network among various services with different requirements, ensuring all of them meet the QoS without interfering into the others' QoS. Standardization bodies have tackled the definition of the corresponding network slicing operations, and how to actually make it happen in the current and future 5G networks. However, it is out of the scope of the standardization process how to do the orchestration and assignment of network resources to each slice, so delay, computing, bandwidth, and reliability requirements are met in 5G services. To fill such gap, the research community has recently studied how to assess an adequate assignment and management of slice resources – see section 2.2. This thesis contributes to the state of the art in the assignment of slice resources in section 5.1, where OKpi is presented as an orchestration algorithm that meets KPIs of coexisting slices in a network infrastructure.

1.6 Thesis overview

The remainder of this thesis is organized as follows. Chapter 2 provides background on NFV orchestration algorithms, i.e., algorithms that decide the assignment of network and computing resources to the NSs to be deployed. In particular, the chapter starts providing an exemplary problem statement of the Virtual Network Embedding (VNE) problem, and later in section 2.1 and section 2.2, it goes over the state of the art of NFV orchestration algorithms before and after 5G, respectively. With the latter algorithms solving the VNE for 5G networks and services.

Since the present thesis proposes NFV orchestration algorithms in Edge and Fog scenarios (chapter 5), it is necessary to have a representation of the underlying network infrastructure to evaluate the performance of the proposed solutions. Hence, chapter 3 presents how to generate 5G infrastructure graphs. Sections 3.1 to 3.4 present a methodology to derive BSs locations, and MEC PoPs in the network planing stage. And later section 3.5 presents `5GEN`, an R package to derive 5G reference infrastructure graphs following the proposed methodology to derive both the BSs and MEC PoPs locations. To finish chapter 3, section 3.6 presents the generation of multi-domain graphs, in this case only accounting for the data centers and their topology, as such graphs are later considered in the NFV orchestration algorithms over federated networks (section 4.2).

After proposing the generation of graphs that represent 5G networks, chapter 4 proposes solutions to deal with the decisions regarding the deployment of NSs in federated networks. Section 4.1 motivates the use of network federation for the deployment of NSs, in particular, to consider the federation of NSs to other administrative domains, i.e., to use another domain resources, or delegate the NS deployment completely to another domain. Section 4.2 proposes modifications of well known algorithms that solve the shortest path problem to derive NFV orchestration algorithms that tackle the VNE problem in networks where multiple domains federate their resources. Later, and still in the context of network federation, section 4.3 studies when to delegate the deployment of NSs to a federated domain, given that the federation prices might vary over time. In particular, it studies the performance of Deep Q-Network (DQN) and Q-learning solutions that decide whether to delegate or not the deployment of NSs.

Then, in Chapter 5, this thesis goes over two different NFV orchestration algorithms that tackle the VNE problem in 5G networks, in particular in sliced Edge/Fog networks (section 5.1), and scenarios with mobile and volatile computing nodes (section 5.2). The former solution, named OKpi, is based on a multi-constrained shortest path algorithm that meets both latency and reliability constraints of vehicular and robotic NSs. And the latter solution is based on a randomized rounding algorithm to solve the optimization formulation of the VNE problem. Such solution takes into consideration the mobility, coverage, and battery constraints of the mobile fog devices of a warehousing NSs.

Chapter 6 presents the last problem studied in the present thesis. This is the scaling of V2N services, as remote driving, based on the changes of vehicular traffic flow. The chapter proposes a solution to decide the scaling of the number of CPUs required to meet the latency constraints of V2N services. In particular, it compares state of the art forecasting solutions to predict the number of vehicles in Torino city, and later decide the number of CPUs using a $M/M/c$ queuing model. The analysis is carried out using a real dataset of Torino traffic ranging from January to March of 2020 (including the COVID-19 lock-down period).

Finally, chapter 7 concludes the thesis briefing up the contributions of this thesis in the research field of NFV orchestration algorithms, and chapter 8 points out future work to further improve the proposed solutions.

2. Background on NFV orchestration algorithms

As discussed in the previous chapter of this thesis, recent innovations of the networks research community have lied substantially on the Network Function Virtualization (NFV) paradigm. Network operators and administrative domains have nowadays a flexible manner of assessing the management of its network services. If the European Telecommunications Standards Institute (ETSI) NFV stack [ETS14] is implemented by the operator, it means that it can handle the deployment of virtualized services as virtual Content Delivery Network (CDN)s, or other 5G-related services that fall within the category of slices as Ultra-Reliable and Low Latency Communications (URLLC), e.g., vehicular services as car platooning. Upon the deployment of a network service, there are three main decisions that have to be taken: (*i*) the allocation of resources for Virtual Network Function (VNF)s in the servers; (*ii*) the traffic steering of the Virtual Link (VL)s across the network links; and (*iii*) the allocation of bandwidth resources on the links that steer the Network Service (NS)s traffic. These decisions are taken by a NFV orchestration algorithm, i.e., an algorithm that maps the VLs and VNFs in the underlying physical infrastructure, or even in an abstraction of the physical infrastructure. Figure 2.1 illustrates how a solution of a NFV algorithm looks like, the traffic steering of the VLs, so as the mapping of VNFs to servers.

It is important to remark the difference between an orchestrator, which assesses the management and assignment of services to resources, and a NFV orchestration algorithm, which solves the Virtual Network Embedding (VNE) problem. That is, to embed a virtualized service in a substrate network. To this extent, this thesis refers to these algorithms as either NFV orchestration algorithms, VNE algorithms, or even VNF placement algorithms.

The target of a NFV orchestration algorithm is to deploy the requested NSs in the underlying network, taking into account the limited resources, so as the traffic steering. In the decision making, the research community typically refer to two problems that are tackled in the VNE problem, namely, (*i*) the resource allocation, i.e., the allocation of computational resources to host the composing VNFs; and (*ii*) the traffic steering, i.e., to decide which links are traversed to route the traffic in between VNFs, or in other words, to decide the traffic steering of the NSs' VLs. A NFV orchestration algorithm could decide either to solve both problems at the same time, or first to solve the resource allocation problem, to later decide the traffic steering among the servers where the VNFs were allocated. Each approach has its pros and cons. If the VNE problem is solved in two steps, it is easier to find a solution, that is, the problem complexity is reduced, and algorithms are more prone to find faster a solution. However, the solution is typically suboptimal and this leads to an not proper usage of the available resources. In case both problems are tackled at the same time – i.e., the resource allocation, and traffic steering decisions are taken together or concurrently – the solution is more likely to be "less suboptimal" than the solution obtained when tackling the problem in two steps. On top of that, the problem complexity is higher than just solving first the

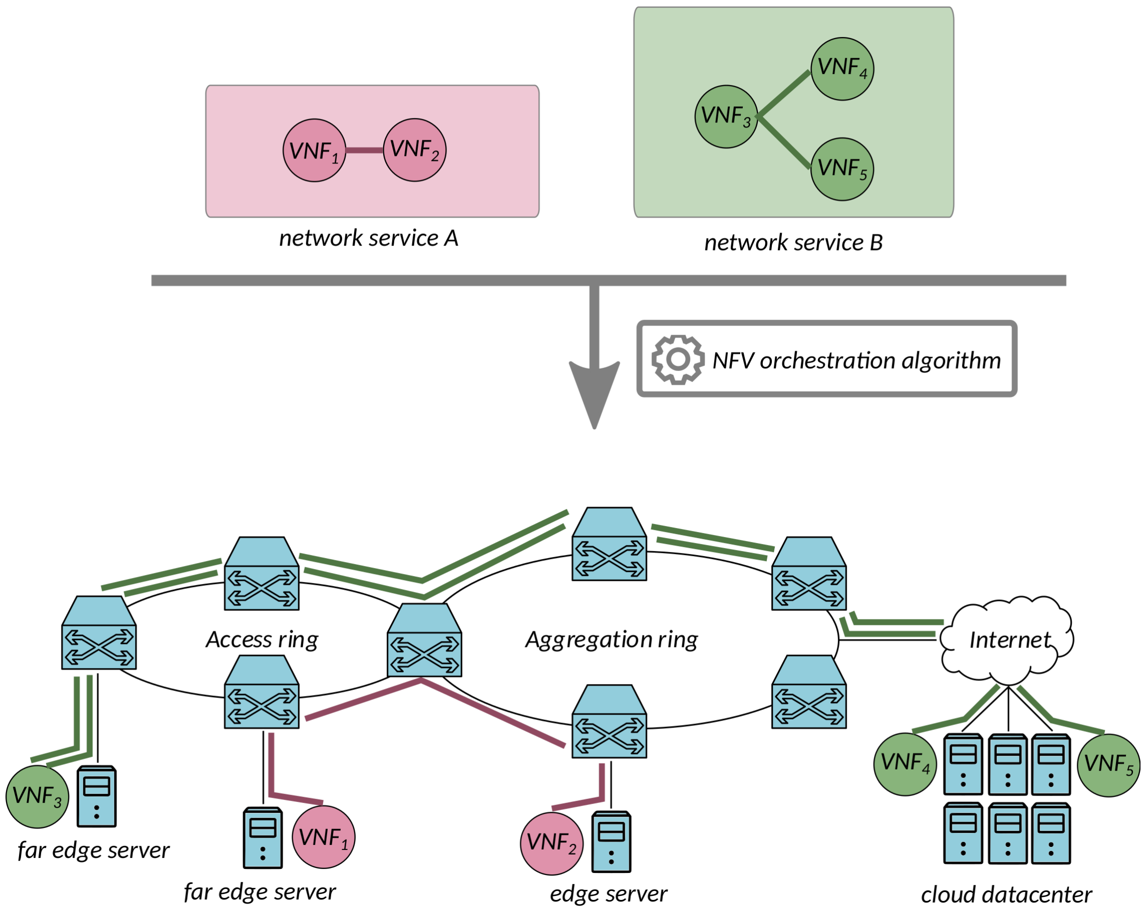


Figure 2.1: Solution of a NFV orchestration algorithm

resource allocation, and then the traffic steering (later on this chapter the thesis provides a detailed explanation about this statement). By saying that solving both problems at once the solution is "less suboptimal", we mean that the solutions would result in doing a more efficient use of resources, or achieving a higher monetary benefit with respect to the solutions obtained in the two-step approach.

Traditionally, it is common practice to formulate the VNE problem as an optimization problem, which may be either an Integer Linear Programming (ILP), or Mixed Integer Linear Programming (MILP). Given the optimization formulation, authors propose heuristic solutions to elaborate their NFV orchestration algorithms, and typically provide statements about their optimality if possible. If optimality bounds are given, the theoretical behaviour is proved, and any platform implementing the solution would feel safer about its performance. Hence, it is always preferred to provide optimality boundaries of the proposed heuristic solutions.

But how do we measure how optimal a solution is? Well, this is done using the objective function of the stated optimization problem. The objective function expresses what the algorithm wants to maximize or minimize. For example, one NFV algorithm could be designed so its objective is to minimize the resource consumption in the network servers, others could decide to maximize the revenue obtained trying to deploy as much NSs as possible, and others may decide to minimize link congestion and do the traffic steering accordingly. Thus, given an optimization formulation with the aforementioned objective functions, the designer of the NFV orchestration algorithm is capable of measuring how good is the solution by means of the chosen objective function. This does not mean that other metrics could be measured to check the algorithm behaviour, since given an embedding solution, one is able to measure other metric as latency, CPU usage, or experienced jitter. In any case, to provide theoretical statements on any metric, they should be present in the problem formulation. And it might even be the case that the goal of a NFV orchestration algorithm is to maximize/minimize multiple objectives at the same time. For example, it could maximize the monetary benefit, and minimize the link congestion.

The VNE problem formulations typically impose both (*i*) resource constraints; and (*ii*) flow constraints. Lets give a baseline formulation to provide an example on how a VNE problem looks like. Lets assume we have a set of servers S and each one has $C_s,\ s \in S$ CPUs available to be used. Then we consider also that the network has a set of links L with bandwidth capacities $B_l,\ l \in L$. For simplicity, the routers would be considered as servers with $C_s = 0$. Thus, we can say that a network link will connect two servers/routers, i.e., $l = (s_1, s_2),\ s_1, s_2 \in S$. Now we have to introduce a set of network services N, each one containing $V_n,\ n \in N$ VNFs. We refer to these VNFs as $v_{i,n},\ i \leq V_n$, and each one asks for $c_{i,n}$ CPUs in whatever server $s \in S$ hosts it. Similarly, a network service n contains $VL_n,\ n \in N$ virtual links, and we refer to them as $vl_{i,n},\ i \leq VL_n$. Each virtual link will ask for $b_{i,n},\ i \leq VL_n$ bandwidth over the link where its traffic is steered. In case a network service n is successfully deployed in the network infrastructure, it gives a revenue of r_n units.

Given the above problem description, the job of the NFV orchestration algorithm is to decide (*i*) to which server it assigns each VNF; and (*ii*) which links are used to steer the traffic of the virtual links. For the former, we define $x(v_{i,n}, s) \in \{0,1\}$ to denote whether a VNF $v_{i,n}$ is deployed in a server s. And for the latter, we define $x(vl_{i,n}, l) \in \{0,1\}$ to state whether VL $vl_{i,n}$ passes through the link l or not. The $x(\cdot)$ are known as decision variables. The objective of the NFV orchestrator would then be to maximize the reward obtained after assigning VNFs and VLs to the servers and network links.

But still there is another thing to be taken into account in problems of this nature, and that is to express the constraints that every solution must satisfy. In particular there are (*i*) resource constrains; and (*ii*) flow constraints that must be encoded as optimization constraints. The resource constraints correspond to both the CPU and bandwidth limitations in the discussed example problem. Specifically, all the VNFs deployed in a server should not exceed its capacity, which is denoted as $\sum_{n\in N, i\in V_n} x(v_{n,i}, s)\cdot c_{n,i} \leq C_s$. Similarly, the bandwidth of the VLs allocated within a network link l should not exceed its capacity, in other words, $\sum_{n\in N, i\in VL_n} x(vl_{i,n}, l)\cdot b_{i,n} \leq B_l$ must be satisfied. The two constraints we have just mentioned refer to the capacity constraints, and the last one to encode is the flow constraints. Typically flow constraints refer to "flow conservation", a rule that is summarized into this sentence "all traffic coming in, should come out". Given a switch inside the network, this means that any VL traffic entering itself should exit as well. In the discussed problem example such statements are translated into $x(vl_{i,n}, l) = x(vl_{i,n}, l_{+1}),\ l = (s_{-1}, s_0), l_{+1} = (s_0, s_1), C_{s_0} = 0$.

To formulate the optimization problem we just have to put together the decision variables, objective functions, so as the constraints discussed in the previous paragraph. As a result we have:

$$\max_{x(n)} \quad \sum_n x(n) r_n \tag{2.1}$$

$$\text{s.t.} \quad \sum_{n\in N, i\in V_n} x(v_{n,i}, s)\cdot c_{n,i} \leq C_s, \qquad \forall s \in S \tag{2.2}$$

$$\sum_{n\in N, i\in VL_n} x(vl_{i,n}, l)\cdot b_{i,n} \leq B_l, \qquad \forall l \in L \tag{2.3}$$

$$x(vl_{i,n}, l) = x(vl_{i,n}, l_{+1}), \qquad \forall vl_{i,n}, l = (s_{-1}, s_0), l_{+1} = (s_0, s_1), C_{s_0} = 0 \tag{2.4}$$

with

$$x(n) = \begin{cases} 1, & \text{if} \quad \sum_{i\leq V_n, s\in S} x(v_{i,n}, s) = V_n \quad \wedge \quad \sum_{i\leq VL_n, l\in L} x(vl_{i,n}, l) = VL_n \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

that is, $x(n) = 1$ only if all VNFs and VLs are deployed for the NS $n \in N$.

The above example is a simplistic, yet representative formulation of a typical typical optimization problem that solves the VNE problem. It conveys the resource constraints with (2.2) and (2.3) to limit the CPU and bandwidth usage, respectively. So as it also accounts for the widely used conservation constraint (2.4) to impose that ingress and egress traffic match. The presented problem is an ILP that is usually solved via existing optimization solvers.

The nature of the VNE problem makes it being of high complexity. Moreover, the optimization formulations lie in the category of Non-deterministic polynomial time (NP) problems [AB09]. Basically, if a problem lies in the NP category, that means that there is no algorithm capable of "deterministically" reaching the optimal solution in polynomial time. In other words, given an algorithm that solves a NP problem, we cannot ensure it will find the optimal solution in polynomial time. To express the run-time of an algorithm is common practice to write it down using big-O notation $\mathscr{O}$, and the number of decision variables. This notation is used in chapter 4 and chapter 5.

Lets give an example on how one expresses the problem size of the above VNE problem statement. First of all we need to count how many decision variables are in the stated problem, and that is the number of $x(v_{i,n},s)$ and $x(vl_{i,n},l)$ variables. For the server assignments we would have $S\cdot\sum_{n\in N}V_n$ variables, and for the links' assignments $L\cdot\sum_{n\in N}VL_n$ variables. If we assume that there are more servers than links $S\geq L$, that the number of service requests are larger than the number of VNFs in all the existing NSs, and the number of NS request is larger than the number of servers; then $S\cdot\sum_{n\in N}V_n\leq N^3$. And as the number of servers dominate the number of links, we can state that the number of decision variables is $\leq 2\cdot N^3$. In big-O notation we then say that the problem size is $\mathscr{O}(N^3)$, meaning that the number of variables is $\leq k\cdot N^3$ with $k\in\mathbb{N}$ a constant. In the computational theory field, the run-time complexity expresses the amount of operations required to solve a problem. Hence, in our example we would say that the run-time complexity is going to be proportional to $\mathscr{O}(N^3)$. If we said that the amount of operations is quadratic in terms of the problem size, then we have that it is required $\mathscr{O}\left((N^3)^2\right)$ operations to run the algorithm.

Going back to the aforementioned explanation about NP problems, we say that the presented optimization problem lies in the NP category if it cannot be guaranteed that the optimal solution is reached in polynomial time. In other words, if the number of operations – which we express in terms of the problem size $\mathscr{O}(N^3)$ – is not proportional to a polynomial function over the problem size. Remember that a polynomial function is an expression of the form $f(x)=\sum_{i\leq M}a_ix^i$. Hence, a polynomial over our problem size would take the form $f(N^2)=\sum_{i\leq M}a_iN^{3i}$ and satisfy $f(N^2)\leq a_MN^{3M}$ with $M\in\mathbb{N}$ being a constant. As a result we say that the running time complexity of our problem is polynomial if the number of operations is $\mathscr{O}(N^{3M})$.

In case our VNE problem is NP complex, the search of optimal solutions becomes harder. To give an idea, lets assume that in the proposed VNE problem we have came up with a NFV orchestration algorithm that should decide the embedding of $N=100$ NSs. First of all, the problem would have $\mathscr{O}(100^3)$ decision variables, which is quite a large number of variables to decide on. And second of all, no NFV orchestration algorithm would be able to find the optimal solution after running $\mathscr{O}(100^{3M})$ operations, no matter the chosen constant $M\in\mathbb{N}$. The statement is severe, and here it is the reason why. Imagine we choose $M=7$, then the number of operations to do is proportional to 100^{21}, and a CPU with the capacity of performing $10^6\,\frac{\text{op}}{\text{sec}}$ would take $\geq 31,709,791$ years to finish the operations. But, since the problem is NP, such a solution may not be optimal.

The statement is quite tough, and it might seem discouraging. And right now the reader might be thinking "ok, but is the VNE problem complexity NP?". Unluckily, the answer is yes. And such a statement makes the research community keep on searching for new solutions improving the state of the art of the embedding problem. Note that the implications of the VNE problem being NP come by means of monetary side-effects. If the VNE embedding problem complexity was not NP, then a network operator would have the chance to take the best deployment decisions to maximize it monetary reward, see (2.1). However, there is still an unproved statement of whether $P=NP$ or not. That is, it is not proven that the complexity of problems that manage to find optimal solutions in polynomial time, share complexity with those that cannot find it in polynomial time. For the time being the research community embraces the $P\neq NP$ conjecture, as the running time of already designed NFV orchestration algorithms have given evidence to defend such conjecture. However, in case there was a mathematical prove stating $P=NP$, then the VNE problem solving community would benefit from it, and as a consequence, the telecommunication industry; with an increase and better usage of its available resources.

The previous paragraphs have stated that the VNE problem formulated in this chapter has NP complexity. But still no proof was shown to support this statement, neither mathematically, nor by means of references. As the problem is just an example conveying the main constraints present in the wide majority of VNE problems, the next paragraphs will try to give a mathematical proof that follows the same strategy as the ones in the literature.

The strategy to follow is to proof that an instance of the optimization problem relates to another optimization problem that is proven to have NP complexity. And by an instance of a problem here we talk about a particularization of the stated optimization problem itself. In the case of this chapter problem statement, the proof is going to be about mapping it to the knapsack problem [Lew83]. Given a knapsack with capacity C, we want to pack items $i \in I$ of weight w_i into it, so we maximize the reward r_i obtained by packing them inside the knapsack. Its optimization problem formulation looks as follows:

$$\max_{x_i} \quad \sum_{i \in I} x_i r_i \tag{2.6}$$

$$\text{s.t.} \quad \sum_{i \in I} x_i \cdot w_i \leq C, \tag{2.7}$$

with $x_i \in \{0,1\}$ denoting whether the item i is packed inside or not. Having the knapsack problem in mind, the following paragraph details the complexity proof about our problem statement.

Lemma 2.1: The VNE problem (2.1)-(2.4) is NP-complete.

Proof. Lets take the following instance of our problem. In particular, one in which all network services $n \in N$ only have a single VNF $V_n = 1,\ \forall n \in N$, hence, no VLs $VL_n = 0,\ \forall n \in N$. Then, constraints (2.3) and (2.4) are not required, as there is no flow to link matching in the given instance. Moreover, the auxiliary variable $x(n)$ which denotes if a NS n is deployed or not, now is expressed as $x(n) = x_{0,n}$. Or in other words, if the only VNF $v_{0,n}$ of service n is deployed, that means that the whole NS is deployed. On top of that, now we will assume that there is only one server $s_0 \in S$ present in the network infrastructure, that is, that $|S| = 1$. As a result, the described instance of the problem statement looks as follows

$$\max_{x(v_{0,n},s_0)} \quad \sum_{n \in N} x(v_{0,n}, s_0) r_n \tag{2.8}$$

$$\text{s.t.} \quad \sum_{n \in N} x(v_{0,n}, s_0) \cdot c_{0,n} \leq C_{s_0}, \tag{2.9}$$

Note that this is the knapsack problem stated with server s_0 being a knapsack with capacity C_{s_0}, the VNFs being the items with weight equal to their CPU requirements $c_{0,n}$, and the NS deployment reward r_n corresponding to the item reward.

Since the described instance of problem (2.1)-(2.4) is the knapsack problem, and the knapsack problem is NP-complete; we conclude that the optimization problem (2.1)-(2.4) is NP-complete. ■

Lemma 2.1 states that the problem complexity is NP-complete. A problem of this nature is a NP problem one can solve using a brute force algorithm, i.e., trying all the different combinations of the $x(v_{i,n}, s)$ and $x(vl_{i,n}, l)$ decision variables. On top of that, a NP problem has the particularity of having the chance of simulating another problem with similar solvability, e.g., we could use the knapsack problem to solve the VNE problem proposed in this chapter. For instance, Lemma 2.1 shows it is possible to use the knapsack to solve the instance of the VNE problem with only one server, and NSs consisting of only a single VNF. About the brute-force solvability of a NP-complete problem, note that a brute force approach might be intractable in many cases as the example provided in previous paragraphs about the problem size. Recall that under the assumption of the number of NSs N dominating both the servers S and links L, the problem size is $\mathcal{O}(N^3)$. Knowing that the decision variables are binary, this implies that there are $\mathcal{O}(2^{N^3})$ possible solutions. That is, the number of solutions grows potentially with the number of NSs to embed. Thus, even the VNE problem set as basis in this chapter is NP-complete, it is unfeasible to try out all the possible solutions, not to mention the devoted time to check the correctness of every solution visited by a brute force algorithm.

In the particular VNE problem of this chapter we have proceeded to proof its complexity using the knapsack problem. However, this is not the only well-known problem used to proof the complexity of VNE optimization problems. Other examples as the bin packing problem, the set coverage, or the minimum-cut problem [AB09] have been used in the literature to show the problems' complexity.

On top of the common resource and flow constraints, the literature adds other elements to the VNE problem formulation. And it is not always that the solution aims to maximize monetary benefits. Instead, some solutions focus on minimizing energy consumption, or other metrics such as jitter (in case the problem formulation is capable of capturing latency aspects of the served packets). It is up to the NFV orchestration algorithm designer, to choose what problem it wants to tackle, and which aspects are going to be treated. Usually, VNE problem formulations and solutions have shifted towards the same direction of the research community at the moment. When data centers started to become a reality, and their management and resource utilization started to become a real problem to account for, NFV orchestration algorithms started to focus on how to deploy NSs in infrastructures with high and concentrated computational capacities, and tackled aspects as job distribution and load balancing. Latter on, as the energy consumption started to become a serious problem in the data center facilities, VNE problem focused on how to distribute the NSs deployments so as to minimize the energy consumption. In such a way, companies could label their cloud facilities as "green" and sustainable thanks to the reduction of the carbon footprint derived from the consumed energy of their facilities. And additionally, they could reduce the bill of the monthly power supply.

The latter changes in the networking community have required a change in the VNE problems, and how they are tackled. Recent paradigms discussed in chapter 1 brought in the possibility of breaking down monolithic NSs entities into their essential components, the VNFs (see section 1.2). Thanks to that, NFV orchestration algorithms started to take advantage of the split of a service, and distributed the deployment of a single NS into multiple servers. However, 5G networks and new networking paradigms also brought in tougher constraints regarding aspects such as latency, geographical constraints, or even reliability. With them, a shift was mandatory in the design of new NFV orchestration algorithms, as they started to account for the new constraints to satisfy the promises of 5G use cases.

In the next sections of the chapter, we provide an overview of existing works in the literature. Section 2.1 reviews NFV orchestration algorithms that tackled the VNE problem in non-5G scenarios, i.e., in deployments without constraints that 5G use cases typically have, and without 5G infrastructures. And section 2.2 discusses a set of NFV orchestration algorithms that tackle the embedding of NSs in 5G-scenarios. Thus, accounting for the strict latency, reliability, and availability requirements; using the infrastructure that new 5G technologies bring in, so as the paradigms discussed in chapter 1. The differentiation on both kind of solutions aims to show the reader what where the aspects that the VNE problem formulations tried to account for before, and after the 5G technologies.

2.1 NFV orchestration algorithms before 5G

The VNE problem started to fire up even before the 5G network services' requirements came up in the standardization community. Early works focused on the deployment of NSs within data center infrastructures, without accounting for the embedding of traffic flows in the substrate network. These works focused on energy consumption, minimization of resource waste, or even factors as the temperature that servers could achieve. On top of that, the monetary revenue of solutions was most of the times taken into account int the problem formulations.

Latter on, the research community started to consider the split of NSs into different servers, each of them running a constituent VNF, as specified in the NFV paradigm (see section 1.2). The constituent VNFs of the NSs could be spread across the substrate network infrastructure, and the flexibility of the deployments substantially increased. That came with an obvious growth of the problem complexity, as the solution space size increased with the split of the NSs. Additionally, the split of a NS into VNFs spread across different servers, resulted into the traffic steering in between such VNFs, which in previous years were all co-located in the same server. Hence, the VNE problem required the NFV orchestration algorithm to deal with both the VNF, and flow embedding. Not all solutions investigated approaches solving both problems at once, indeed, most of them only focused on the VNF embedding, and either neglected the flow embedding, or assumed that that was known on forehand.

The following paragraphs revisit existing solutions that propose different NFV orchestration algorithms solving the VNE problem that did not account for the latency, nor reliability requirements of 5G services. Most of the following works neglected such constraints probably because of the year when the research was done – by that time the 5G constraints were not still under definition standardisation documents – or because the URLLC and high data rates were still not a big thing the research community.

[LQ15] gives an overview of what is the VNE problem, and formulates a typical optimization problem, as the one specified in the current chapter. The NP problem complexity is proofed by reducing the problem to the set cover problem [PS98]. Solutions of the proposed problem decide the embedding of the service flows, assuming that the network functions (the paper does not account for function virtualization) are deployed on forehand. No heuristic solution is proposed, however authors evaluate the optimal solution in different scenarios.

[GMZ16] uses matrices and eigen-values to discover deployments that minimize the costs. The VNFs mappings are stored in matrices, and the Service Function Chain (SFC)s are embedded gradually. The proposed VNF orchestration algorithms solves sub problems to achieve a general solution, using in dynamic programming. In the SFC embedding, the solution accounts for cost of mapping of both VNFs and flows. Authors give insights about the polynomial time complexity, however, they do not give theoretical proofs on the optimality. The evaluations are carried out to measure the nodes and links' usage in low/high load scenarios (so as the provider revenue). As the problem is not formulated using an optimization problem, there is no comparison in between the proposed solutions and optimal ones.

[HK16] is more SDN oriented, and already accounts for virtualization of the services to be deployed. The proposed solution "lightchain", which aims to minimize the hop-count, that is, the number of links traversed in the virtual links mapped. On top of it, it tries to minimize the amount of flow rules that are present in the SDN switches that route the traffic among the deployed VNFs. Their proposed NFV orchestration algorithm is based on a two-step process. First, the SFCs are broken down into an acyclic graph by randomly deleting the cycles in the service graph. Second, an ordering of the VNFs is performed to respect the dependencies among VNFs, and finally perform a VNF embedding based on a shortest path algorithm. This work does not provide an optimization formulation, hence, does not give insights about the theoretical optimality of the proposed solution, neither does it compare the experimental evaluation against an optimal solution. Nevertheless, the experimental validation shows that in a matter of milliseconds, lightchain finds VNE solutions that impose less flow rules than a First Come First Placed approach.

[MGZ16] tackles the VNE problem formulating it as a weighted graph mapping problem, rather than an optimization problem, as stated in the example problem stated in this chapter. It models as graphs both the SFC, and the substrate network infrastructure. Given that, the proposed VNE orchestration algorithm takes the bandwidth as weight of both the SFCs and infrastructure graphs, neglecting then the delay. Nevertheless, the solution is based on performing an eigen-decomposition of the SFC and infrastructure adjacency matrices using a modified version of Umeyama's approach [Ume88], an spectral solution that uses the Hungarian algorithm [PS98] to find the minimum distance mapping in between two graphs. Hence, the matching or VNE is essentially relying on the Hungarian algorithm. The evaluation does not compare their VNE orchestration algorithm performance against an optimal solution, since the problem is not formulated as such. However, authors show that their solution clearly scales adequately to both the NSs and infrastructure sizes. And the experimental validation an acceptance ratio that reaches the 100% in their proposed evaluation scenarios.

[CDJ17] provides a NFV orchestration algorithm that uses replication of VNFs to minimize the link utilization in the network used to deploy the different NSs. Unlike the two previous works already discussed, [CDJ17] formulates the VNE problem as an optimization problem, and compares the proposed solutions against the stated ILP problem. The strategy followed by the authors, is to use a link cost function that exponentially grows after the utilization exceeds the 60% of the link capacity. Therefore, the obtained solutions rarely exceed such a value in the link usage. The optimal solution is compared against a genetic algorithm that first solves the traffic steering problem, and later performs the VNF-to-server embedding accounting for possible replications that might offload the links' congestion. Authors' results show that the genetic algorithm achieves near optimal results in terms of cost, and link usage.

[Kuo+16] investigates how to solve the VNE problem focusing on the link and server usage ratio. Authors suggest to reuse already deployed VNFs if there is still enough computational capacity in the server where the VNF is running. The idea is to assume larger paths if it allows to reuse residual resources, so as to maximize the resource usage. Otherwise, the shortest paths towards the VNFs would be congested, and the servers located further in the infrastructure would remain unused. To have an adequate link and server usage ratio, authors define a relationship between both parameters, and formulate the corresponding optimization problem to maximize the demand to be accommodated. The proposed NFV orchestration algorithm is based on dynamic programming, and is compared against shortest path solutions, and algorithms that greedily reuse the existing VNFs. Results show that the proposed solution is capable of maximizing the accommodated demands, as it captures properly the server and link usage ratio, no matter the NSs and network topologies. Thus, it adapts to a variety of scenarios, outperforming the greedy and shortest path approaches.

[XF10] aims to solve the VNE problem in a data center infrastructure. Indeed, it only tackles the VNF embedding into servers, thus, it does not assess the traffic routing. The proposed algorithm minimizes the waste of resources, power consumption, and peak temperature reached by the data center servers. The problem is formulated as a multi-objective optimization problem, and the proposed heuristic solution is based on a genetic algorithm that uses a fuzzy evaluation to evaluate the goodness of the produced mutations. That is, the genetic algorithm produces initial combinations of solutions, and it modifies (mutate) them using a weighted function over the multi-objective function terms, i.e., the resource waste, energy consumption, and peak temperature. The genetic algorithm outperforms existing bin packing heuristics as the first fit [PS98] in all the metrics used to evaluate the multi-objective function. However, the solution does not provide optimality guarantees, nor theoretical boundaries.

[San+17a] formulates a simplified version of the VNE problem in which there is a single NS instance composed of multiple VNFs that have to process a set of fixed flows in the substrate network. The problem to be solved is to determine the number of VNF that each server should hold to process completely, or partially the incoming flow demand. Authors formulate the corresponding optimization problem, and their objective is to minimize the number of VNFs deployed in the network. After formulating the corresponding ILP problem, two greedy heuristics are provided, both with optimality guarantees. The proposed NFV orchestration algorithms rely on the idea of choosing those servers that receive a higher number of flows, or a higher data rate. Such greedy solutions approximate to the optimal solution by a factor[1] of $(1-o(1))\log m$, with m denoting the number of flows to be processed in the substrate network. Additionally, authors state that given fat-tree [Lei85] topologies of data centers, there is an algorithm that finds an optimal solution of their problem statement. The proof on the optimality of such NFV orchestration algorithm is provided, and evaluations in the experimental phase show that such algorithm, so as the two other generic greedy heuristics, satisfy the optimality boundaries theoretically stated. A comparison against an optimal solution is carried out in large network scenarios, and results show the near-optimality performance of the proposed solutions. However, the stated problem is not quite realistic, as it does not account for the flow routing, neither for the deployment of more than just a single NS on top of the substrate network.

[Coh+15a] solves the VNE problem tackling only the VNF to server embedding. It provides solutions telling how many VNF instances should be deployed at each server to process the clients' requests. The optimization formulation is an ILP problem that minimizes the distance[2] in between users, and the cost of deploying the VNF instances across servers. The paper does not account for the flow routing. Authors propose NFV orchestration algorithms based on a linear relaxation of the ILP problem, and provide different versions which relax or not the capacity constraint on the number of clients' request that each VNF instance can process. The achieved solutions are later rounded up to match the ILP problem formulation, and authors state theoretical boundaries on the solution optimality. In particular, the proposed solutions ensure that the objective functions is at maximum 16 times away from the optimal solutions. The solutions are evaluated and compared against a greedy solution in scenarios with increasing number of functions, increasing VNF capacity, and number of clients to be attended. Results show that the solutions outperform substantially greedy algorithm, and meet the stated optimality boundaries.

All the solutions overviewed in this section focus in the accommodation of incoming demand, such that resource capacity is not exceeded. Solutions either try to minimize number of flows in Software Defined Networking (SDN)-based solutions, the link utilization, or energy consumption. However, they do not consider the Quality of Service (QoS) of the allocated NS. Whether it satisfies the desired latency, or transmission reliability, both of paramount importance in 5G NSs.

2.2 NFV orchestration algorithms after 5G

The previous section has overviewed a list of NFV orchestration algorithms that solved the VNE problem for services that did not impose the strict communication constraints of 5G services. Indeed, some of them even neglected the existence of the edge computing in their analysis.

However, as the first releases of 5G standards started to appear, i.e., Release 15 [3GP19c] and Release 16 [3GP20h]; conferences and journals started to publish work of researches tackling the VNE problem in the 5G scenarios that were tackled in the standardization bodies. The published VNE algorithms accounted for URLLC requirements, enhanced Mobile Broadband (eMBB) use cases, and even proposed solutions to offer network slicing. On top of that, the networking community started to take into consideration the network infrastructure benefits of 5G deployments, such as Multi-access Edge Computing (MEC) facilities, Millimeter Wave (mmWv) New Radio (NR) technologies, and even fog devices with their respective computational and communication constrains.

Hence, the section is entitled *NFV orchestration algorithms after 5G* since the solutions overviewed in the following paragraphs were designed to solve the VNE problem after 5G requirements and infrastructures started to be considered in works published in the literature.

[1] $f(x)=o(g(x))$ is known as *little o* notation, and it means that function $g(x)$ grows faster than $f(x)$. Formally one says such a statement if for every ε there exists a constant N such that $f(x)\leq \varepsilon g(x)$ with $x\geq N$.

[2] by distance here, authors refer to any metric to be chosen.

[ZD18] solves the VNE problem in scenarios in which it is required to deploy process control functions related to industrial use cases. The work considers the presence of MEC servers co-located with some Base Station (BS)s taken from a data set in the city of Milano. Authors propose an ILP problem formulation to assess the embedding of VNFs that perform the control functions into the MEC servers. Such an embedding is done to ensure that the control operation would still work under failure scenarios, i.e., the MEC server would still have running its VNF in case there is a failure. In such a way, the problem solutions are only feasible in case the reliability of the service is ensured. The proposed NFV orchestration does a Bender decomposition [Geo72] over the constrain matrix, and iteratively finds two solutions, one for an upper bound on the cost function, and another to find a lower bound solution. Both solutions are found on a relaxation of the proposed problem, and authors show via experimentation and theoretically, the finite convergence time of the found solutions. Moreover, experimental results via simulations in the city of Milano show that the algorithm does not suffer from increasing the number of VNFs to be deployed.

[MC19] gives a solution to decide the assignment of VNFs to Virtual Machine (VM)s, and not only that but the possibility of sharing VNFs. On top of that, authors propose to specify the priorities among the services that might share a VNF. Following the standard approach, the paper formulates as an ILP optimization problem which tries to minimize the cost of the deployment. In particular, it accounts for both the amount of resources used in a VM, and the cost of having it turned on. The problem statement assumes that there is a limiting service delay, and authors impose a constraint related to the maximum average delay experienced by a service. To derive the delay computation, the work assumes a M/M/1 queuing model [Kle75]. Once the problem is formulated, and its complexity is analyzed, the paper presents the FlexShare NFV orchestration algorithm, which solves their stated problem in four different steps, namely (*i*) the creation of a bipartite graph with edges associating the VNFs to VMs; (*ii*) the execution of the Hungarian algorithm [Kuh55] to decide the assignment; (*iii*) taking the scaling and priorities decision; and (*iv*) pruning out those solutions that violate the constraints. The run-time complexity of FlexShare is proven to be polynomial on the number of VMs and VNFs, and authors show its close-to optimal performance in scenarios that allow to find the best solution by brute force.

[Aga+19] proposes a slightly different modeling of the VNE problem. The work envisions each VNF as a queue system, thus, all NSs consist in a chain of queues dispatching requests using a Processor Sharing (PS) policy [Kle67]. The queuing theory is used to derive the processing delays of the incoming requests, which do not only pass through a single VNF, but across a set of VNFs with a given transition probability. Authors even account for the possibility of traversing a VNF multiple times, and the derived propagation latency that is induced in the communication in between the servers where the VNFs are embedded. The solutions then, have to select the servers' embedding, so as the amount of CPU that is assigned to a specific VNF. And the objective is to minimize the delay so as to satisfy the QoS constraint of the considered NSs. Given the problem formulation, which is an ILP problem with NP-hard complexity, authors propose the maxZ heuristic as a NFV orchestration algorithm to tackle the VNE problem. maxZ is a multi-step algorithm that overcomes the non-convexity and binary variables by creating a relaxed version of the problem that is later solved using a metric Z that authors used to know the goodness of embedding a VNF into a specific server. In the work, it is shown that maxZ has a cubic/quadratic run-time complexity on the number of servers and VNFs involved in the problem. Additionally, throughout extensive experimentation on topologies of increasing complexity, authors show the close-to-optimal performance of maxZ by means of service time.

[Ma+17] does not explicitly account for 5G requirements, however, it is a solution that could perfectly serve as a NFV orchestration algorithm for 5G use cases. This is because it considers the problem of assigning VNFs to middleboxes along traffic flow paths, so as to minimize the link cost after the embedding. Authors mention that such a link cost function maps to algorithms like the extended and well know routing protocol OSPF [Moy28], which use the link cost to compute shortest paths. Such a cost function could be used to either measure latency or reliability required in 5G use cases, so as to meet their strict requirements. In what concerns the article matter of study, authors formulate an associated ILP optimization problem consisting in selecting which hop along a flow path, i.e., which middlebox; should hold each of the constituent VNFs of the service to be processed. Analytical discussions show how the different variations of the problem affect to the complexity. For instance, the analyzed variations relate to flows whose packet processing depend on the VNF processing order or not. Some of the problem variations are have a NP-hard complexity (by reducing some of the problems to the clique [PS98], or the problem of telling whether a graph has a Hamiltonian cycle), and authors compare their proposed NFV orchestration algorithm against firs-fit alike heuristics. Their solution is based on a two step algorithm that starts looking for the path that minimizes the link cost, and later choose which middle-boxes would hold the VNFs. Experimental evaluation show that under increasing flow demand, authors manage to outperform other heuristics by means of End-to-End (E2E) delay and packet loss; which relate to metrics to be considered in URLLC.

[SYC18] gives an even more complete solution for the VNE problem. Rather than only envisioning an algorithm that tackles where each VNF should be deployed, it also takes into account the surveillance and monitoring of the NSs Key Performance Indicator (KPI)s to ensure them. Authors propose to associate the VNFs to affinity groups with similar characteristics. That is, if a VNF has similar CPU requirements and memory usage as other, both should be in the same affinity group. Authors mention a gravity center, that is nothing but how the aggregated requirements and resource usage of the VNFs inside an affinity group vary over time. The paper proposes z-TORCH a whole solution designed to (*i*) monitor the VNFs KPIs; (*ii*) decide the sampling frequency to assess the monitoring; (*iii*) decide the affinity group that each VNF belongs to; and (*iv*) solve the VNE problem using the affinity groups. Before the last stage is reached (the one concerning this thesis), authors cluster VNFs using k-means [Mac+67] to generate the different affinity groups. Once the affinity groups are decided, and the monitoring sampling frequency is set/updated, z-TORCH proceeds and tackles the VNE problem formulating an ILP. The formulation tries to maximize the quality of the taken decision without exceeding the available resources. Rather than tackling the embedding of the every VNF, z-TORCH uses the gravity center of each affinity group, and finds where to deploy the VNFs belonging to it. The problem is proven to be NP-hard in the embedding stage, and authors show via experiments in a real testbed, how z-TORCH is capable of adjusting the monitoring frequency upon usage peaks, so as to tackle the VNE reaching optimal quality of decisions. The work is a proof of how NFV orchestration algorithms can be implemented in concordance with monitoring and virtualized environments to ensure the 5G services' KPIs.

[Jin+20] focuses on solving the VNE problem not only accounting for edge and cloud servers. The work considers even the first hop connection in between the user and the BSs to take such communication delay into consideration. In particular, authors consider a scenario in which MEC servers are co-located with BSs, thus, being used to minimize communication latency with the end users of the deployed services. The paper formulates the associated optimization problem as an ILP problem in which each user has an associated set of BSs that are accessible. The VNE problem consists then into finding those servers and paths where the VNFs are embedded, and their traffic is steered, respectively. The problem formulation is proven to be NP-hard by reducing it to a bin-packing problem instance, and authors propose a two step NFV orchestration algorithm to solve the problem and minimize the resources' consumption thanks to the consideration of VNF sharing. The proposed solution first tries to find the paths that satisfy the bandwidth and latency constraints of a NS, and then it embeds the constituent VNF over the servers present in such paths. To find the feasible paths where each NS steers its traffic, authors use a constrained depth first traversal of the infrastructure graph using the delay as cost. In such traversal, the delay computation accounts for the BSs queuing delay using an M/M/1 model. Once the paths are computed, the embedding problem uses a path based greedy approach to do the embedding. The two step procedure have optimality upper and lower bounds, and a high running time complexity of $\mathcal{O}(N!2^M)$, with N being the number of nodes in the network infrastructure graph, and M the number of VNFs in a given NS. The results' section shows that the proposed solution outperforms other heuristics based on shortest paths computations along with the First Fit strategy [PS98]. The comparison is performed using delay metrics, and resource consumption. Author's heuristic outperform the others except by means of latency in some specific scenarios.

[ZLZ19] shifts the point of view of traditional VNE problems, and proposes a NFV orchestration algorithm to solve the problem of allocating NSs over a sliced 5G network. In the paper it is studied how to accommodate the requests incoming to a specific slice. For example, given the set of requests arriving to a 4K video slice, the problem is to decide where to deploy the VNFs, either in the edge or the cloud, so as to how to steer the traffic up to the servers holding the running VNFs. In such a process, authors pay attention to the degradation of throughput derived from the co-location of several VNFs at the same server. The more VNFs running on a server, the more likely it is that the throughput of each VNF degrades because of the resource usage interference among the coexisting VNFs. The proposed algorithm, called AIA, solves the VNE problem over a network slice to meet delay and throughput requirements of 5G services, whilst minimizing the mentioned interference. The idea of the algorithm is to (*i*) choose first those VNFs producing more throughput and requiring few computational resources; (*ii*) then choose those NSs with the higher number of VNFs, hoping they will be reused; (*iii*) decide to either deploy in the cloud or edge based on less/more strict latency constraints; and (*iv*) in the embedding of an accepted NS, only put different VNFs in the same server if it is required to meet the latency constraint, otherwise avoid it to not degrade the throughput interference. The AIA algorithm has a theoretical proof regarding its optimality, and a running-time complexity that is $\sim \mathcal{O}(R \cdot P^3)$; with R being the number of requests in a slice, and P the maximum number of VNFs among the deployed NSs. Results show that AIA outperforms other heuristics by means of throughput, thanks to its "VNF interference-awareness". On top of it, although it does not achieve as good latency as the other solutions, the deployed services meet the 5G latency constraints of the analyzed vehicular and 4K video services.

The above works show that the current literature has shifted towards the latency and increased bandwidth requirements in the VNE problem. The overviewed solutions consider not only the 5G services' constraints, but as well the capabilities of 5G infrastructures, either by considering the paradigms presented in chapter 1, or by accounting for the promised data rates and transmission delays of 5G technologies.

With the appearance of 5G, older approaches to solve the VNE are no longer valid, and the research community is focusing on solutions to handle communication requirements of a plethora of new services and use cases that arise as the standardization bodies progress in their definition of the 5G technologies. Although this section gave an overview of some solutions of NFV orchestration algorithms, still there is work pending to cover the wide diversity of services held by 5G infrastructures. For example, none of the presented solutions tackled the communication reliability of the deployed NSs, which is a parameter of paramount importance in the URLLC use cases. Not to mention the complexity of handling the monitoring and scaling of the multiple NSs in a substrate network, so as to meet the QoS imposed by the different slices.

The presented solutions of this section are strong candidates to solve VNE problem in 5G networks, and they clearly set the basis of how NFV orchestration algorithms should look after the incorporation of 5G technologies. The present thesis will discuss in chapter 5 some proposed

NFV orchestration algorithms that also tackle the VNE problem in 5G edge and fog scenarios.

3. Generation of 5G infrastructure graphs

5G networks come with the promise of new and enhanced services, through the introduction of an improved radio interface, plus a network core that allows to dynamically deploy services closer to the location of the users. 5G networks will need to accommodate on top of the same physical infrastructure multiple kinds of services with very distinct requirements, spanning from ultra-low latency to high bandwidth and high reliability. These services are grouped in three main categories by 3rd Generation Partnership Project (3GPP): enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (URLLC), and Massive Internet of Things (MIoT) [3GP18d].

Among the several use cases that may be supported by 5G are Augmented Reality (AR) and Virtual Reality (VR), which can be included into glsembb and URLLC categories. In particular, AR/VR impose a Motion to Photon (MTP) latency that does not exceed 20 ms, requiring a network Round Trip Time (RTT) below 2 ms [HAS17]. Moreover, a response within 1 ms is desired in case of visual-haptic interaction [Shi+10]. Although the new 5G radio interface promises ultra low latency enhancements, to truly fulfill the AR/VR ultra-low latency requirements it is also necessary to reduce the communication distance by bringing the multimedia applications close to the end users. This is achieved by Multi-access Edge Computing (MEC) [Hu+15] [GCR17].

As discussed in section 1.3, MEC is a key enabler for 5G technology and its main principle is to host computation and storage at edge hosts, close to end users. Typically, these edge hosts are highly distributed in the network, located close to the radio access network nodes (e.g., gNodeBs in 5G). As a result, MEC enables two types of services: (*i*) low-latency services, requiring a very low and bounded delay between the end user device and the server hosting the application; and, (*ii*) context-aware services, which need to access end-user contexts, such as the user channel quality conditions, in order to adapt the delivered service. MEC is being standardized within European Telecommunications Standards Institute (ETSI), via the MEC ISG group. The main components of the architecture include: the MEC host, the MEC application and the MEC orchestrator. The MEC host is the key element. It provides the environment to run MEC applications, while it interacts with the mobile network entities, via the MEC platform, to provide MEC services and deliver mobile traffic to MEC applications.

MEC hosts are expected to be deployed by mobile operators in their 5G network infrastructure. To enable the pervasive service offering of AR/VR multimedia services, it is hence necessary to study how a MEC deployment should look like to support URLLC. Specifically, it is important to understand what are the suitable locations of future MEC Point of Presence (PoP)s (see Sec. 3.3) within the mobile network infrastructure.

This chapter proposes how to generate 5G infrastructure graphs to satisfy the strict service requirements of 5G use cases. In particular, section 3.1 describes how to generate the locations of

Base Station (BS)s, section 3.3 specifies where the MEC PoPs should be located, and section 3.4 shows how to generate a 5G network infrastructure in rural, urban, and industrial scenarios. Later in the chapter, in particular in section 3.5, `5GEN` is presented as an R package to generate 5G infrastructure graphs that leveraging on both the generation of BSs and MEC PoPs. And by the end of the chapter, section 3.6 describes the generation of multi-domain graphs that are present in network federation scenarios. Finally, the chapter gives some conclusions on the proposed methods and generated graphs.

3.1 SoA on BSs' generation

To create a 5G network infrastructure, the nearest elements to the end user are the BSs. However, that information is not publicly available. On top, whenever the network infrastructure planing is done, the operator needs to evaluate where to locate its BSs. Thus, it is required to derive a methodology to generate the locations of BSs in a network infrastructure. The current subsection gives an overview of the SoA on the generation of BSs.

Existing work in the literature, such as [SRF15], studies feasible network infrastructure deployments using (Point Process (PP)s) that randomly scatter points on some space (e.g., a line, a Cartesian plane, etc.). In particular [SRF15] uses Poisson Point Process (PPP), a family of PPs used to generate the location of base stations (BSs) that are distributed following Poisson counting processes. PPPs can impose a minimum distance between BSs to increase coverage area and reduce the interference (see *hard-core* PPPs in [BeW07]), and control if every region in the space has the same or different probability to host a BS (see *homogeneous* and *inhomogeneous* PPPs, respectively, in [BeW07]).

Different types of PPPs are used in the State of the Art (SoA). For instance, the authors of [SMF15a] use Neyman-Scott [NS58] PPPs to generate small base stations (BSs) clustered around macro BSs with the objective of modeling the coverage and interference in heterogeneous networks. Similarly, [IEE13] shows that homogeneous PPPs can be used in realistic deployment scenarios to reduce interference and increase the coverage area by properly configuring the distance between the BS sites and the intensity parameter.

Works like [SRF15] and [AD18] analyze potential infrastructure deployments with the help of homogeneous PPPs. The former focuses on studying the cost of a Cloud Radio Access Network (Cloud-RAN) infrastructure using PPPs, while the latter analyzes the throughput and coverage of end users, using PPPs to determine hotspot locations in heterogeneous cellular networks.

Regarding future MEC deployments in real scenarios, [SBD18] studies the existing BS deployment in several USA locations (including highways and rural areas). Likewise, [Fra+17b] characterizes AR/VR deployments in stadium scenarios in the perspective of forthcoming Tokyo 2020 Olympics. The authors conclude that such use case requires a single media room dedicated to the video processing and proposes two distinct BS deployments: one powerful BS in the stadium (e.g., on the roof) or multiple small BSs (e.g., located close to the stadium *vomitoria*[1]).

Unlike [SBD18], the current section accounts for 5G New Radio (NR) technologies rather than legacy radio technologies. Section 3.2 shows how to generate feasible locations for gNodeBs, and derive deployments of MEC PoPs guaranteeing low latency requirements that deployments in [SBD18] do not ensure. In particular, inhomogeneous Matérn II PPs (see Proposition 3.2 are used in Sec. 3.2 for more details) to obtain feasible locations for the BSs. The inhomogeneity of these processes improves the Matérn PPs in the SoA by locating the BSs based on the density of population. That is, more BSs are generated where the population density[2] is higher. In other words, the inhomogeneity allows to concentrate the BSs where they are really needed and to minimize the generation in those areas with little traffic demand.

[1] A vomitorium is a passage situated below a tier of seats in a stadium.

[2] In our scenarios we refer to human population. Nevertheless, inhomogeneous Matérn PPs can be applied to other types of populations.

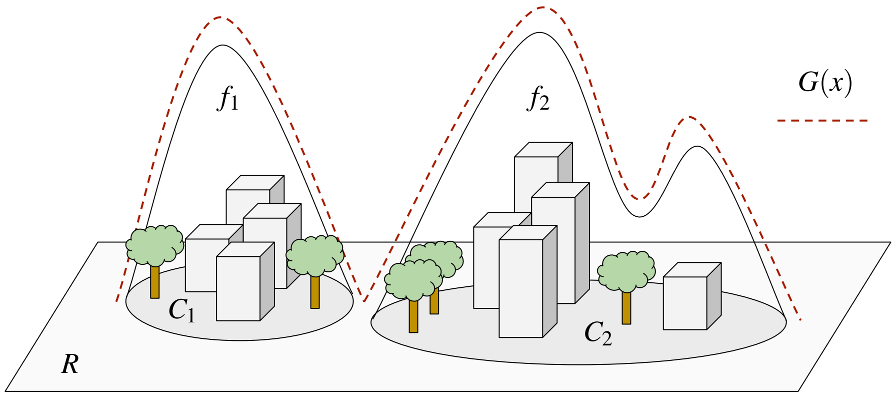


Figure 3.1: *Revolution functions* of a region with two building areas.

The generated BS locations are used later in section 3.3 to derive the required number of PoPs and their potential location.

3.2 Macro cells generation

With the model we aim to generate more BSs in regions where the population is higher. Here we refer to a region R as a subset of a complete separable metric space $\mathscr{R}$, for example a map of Spain. Hence R can represent a city like Madrid and the model locates BSs in city areas where there are more people.

To achieve this we consider that every region R contains population circles $C_i \subset R$ defined by a center c_i and a revolution function $f_i(x)$ (with $i \in [1, R_C] \cap \mathbb{N}$, and R_C the number of population circles defined inside the region R).

The *revolution function* $f_i(x)$ with $x \in C_i$ determines where it is more likely to have a person in C_i. This function leads to a surface of revolution defined at C_i that can be expressed as $f_i(\|x - c_i\|_d)$, and which height expresses the amount of people at a given location $x \in R$. Therefore, the presence of people at $x \in R$ is expressed as:

$$G(x) := \sum_i f_i(\|x - c_i\|_d), \quad \forall C_i \subset R, \ \forall d \in \mathbb{N} \tag{3.1}$$

where $G(x)$ is referred as *gentrification* in the following paragraphs.

Figure 3.1 illustrates an example of a region R with two population circles (i.e., C_1, C_2) corresponding to two distinct areas. Moreover, the *revolution functions* f_i considered in this example are Gaussian-like surfaces of revolution that can be multi lobed as the case of f_2, and the *gentrification* function $G(x)$ is just the dashed line merging both of them.

If an *inhomogeneous* PPP uses the *gentrification* function $G(x)$, the process generates BSs in concordance with the population, but it is still necessary to know how many we have to generate. Based on [Com+18a] we grid the region R in cells to satisfy the imposed number of BSs per km^2 in future 5G deployments, and we take $\mathscr{R}$ as a two dimensional space where the region to be gridded $R \subset \mathscr{R}$ (for example Madrid) is expressed as a rectangle $R = [x_{1,l}, x_{1,r}] \times [x_{2,b}, x_{2,t}]$ divided into cells $R_i \subset R$ of width $x_{1,s}$ and height $x_{2,s}$. The resulting grid has w_i rows and u_i columns that are completely determined by the cell index i (which increases from left to right, and upside down as shown in Fig. 3.2):

$$w_i := \left\lfloor \frac{i \cdot x_{1,s}}{u_n} \right\rfloor, \quad u_i = i \cdot x_{1,s} \mod u_n \tag{3.2}$$

$$u_n = \left\lceil \frac{x_{1,s} - x_{1,l}}{x_{1,s}} \right\rceil \tag{3.3}$$

with u_n being the number of columns of the gridded region R. Finally, we set each cell R_i as the product of two intervals $R_i = [x^i_{1,l}, x^i_{1,r}] \times [x^i_{2,b}, x^i_{2,t}]$ whose limits are:

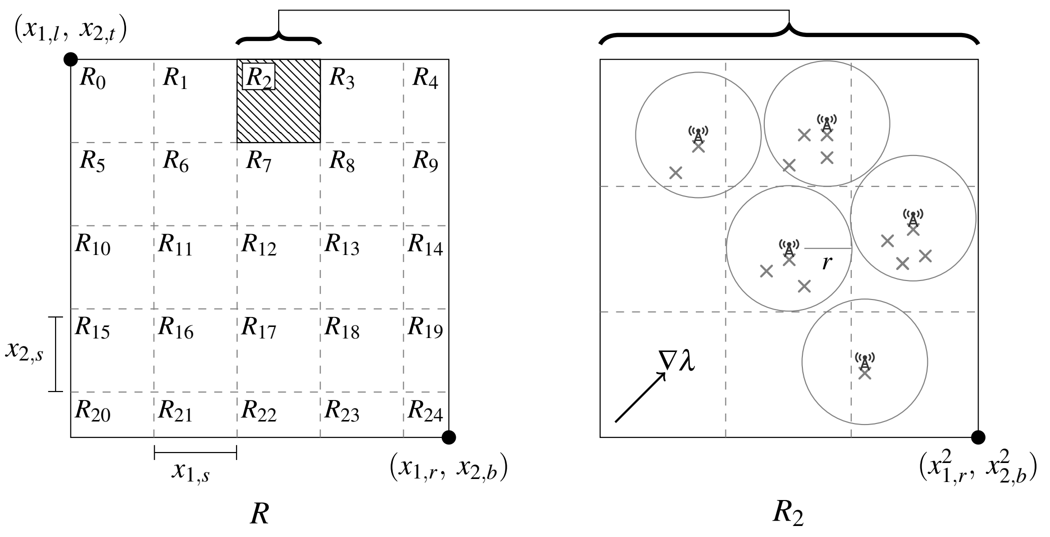
$$x^i_{1,l} = x_{1,l} + x_{1,s}u_i \tag{3.4}$$

$$x^i_{1,r} = \min\{x_{1,r},\ x_{1,l} + (1+u_i)x_{1,s}\} \tag{3.5}$$

$$x^i_{2,b} = \max\{x_{2,b},\ x_{2,t} - (1+w_i)x_{2,s}\} \tag{3.6}$$

$$x^i_{2,t} = x_{2,t} - x_{2,s}w_i \tag{3.7}$$

Fig. 3.2 illustrates on the left-hand side (lhs) the limiting coordinates of a region R that is gridded



$(x_{1,l},\ x_{2,t})$
R_0 R_1 R_2 R_3 R_4
R_5 R_6 R_7 R_8 R_9
R_{10} R_{11} R_{12} R_{13} R_{14}
R_{15} R_{16} R_{17} R_{18} R_{19}
R_{20} R_{21} R_{22} R_{23} R_{24}
$x_{2,s}$
$x_{1,s}$
$(x_{1,r},\ x_{2,b})$
R
r
$\nabla\lambda$
$(x^2_{1,r},\ x^2_{2,b})$
R_2

Figure 3.2: Gridded region on the left side, and macro cell generation inside a cell on the right side. Gray crosses without a cell tower represent BS points that did not survive after the I_2 *thinning*. The rhs shows how the region cell R_2 is gridded when the repulsion radius is chosen as specified in Eq. (3.25) with $N(R_2) = 5$.

into cells R_i according to the limits above, i.e., (3.4)-(3.7).

In every single cell an *inhomogeneous* PPP generates a specific average number of BSs using an *intensity function* defined as $\lambda(x) = k \cdot G(x)$ (where $k \in \mathbb{N}$ is a constant), so BSs are located with higher probability where there are more people. But since we want to have the BSs as sparse as possible, it is necessary to impose a minimum distance between them. This work leverages the Matérn *hard-core* processes [Mat86] to model the BSs' generation. The first process under consideration for our model is the Matérn I process.

Definition 3.1: Matérn I point process: is the point process obtained after applying a *thinning* with index function:

$$I_1(x,X) := \begin{cases} 0 & \text{if } N(B(x,r)) > 1 \\ 1 & \text{if } N(B(x,r)) = 1 \end{cases} \tag{3.8}$$

to a *stationary* PPP X, where $N(B(x,r))$ denotes the *number of points* of the point process X falling in the ball centered at x with radius r.

In other words, a point $x \in X$ is removed if it has a neighbor $x' \in X$ with distance $\|x - x'\|_d < r$. This property suits the random generation of the BSs' locations because only one BS can be in a neighborhood. However, these point processes are *stationary* (see [BeW07]), and therefore *homogeneous*, before the *thinning* (see Definition 3.1), but this model uses *inhomogeneous* PPPs to generate BSs (based on $G(x)$) at each cell $R_i \subset R$. Thus we modify the original definition of a

Matérn I point process to use an *inhomogeneous* PPP before the *thinning*. We call these processes *inhomogeneous Matérn I* PPs.

After *inhomogeneity* is introduced, we need to know an expression for the average *number of points* $\mathbb{E}[N(C)]$ that can appear in a certain set $C \subset \mathscr{R}$. The reason, as shown in Proposition 3.1, is that this expression is in terms of the repulsion radius r and the *intensity function* $\lambda(x)$, and we must select these parameters accordingly to generate the desired average number of BSs in a cell R_i. Since there is no expression in the literature for the average *number of points* of a "*inhomogeneous Matérn I* PPs", we have obtained it with the help of the Campbell-Mecke formula [SW08].

Lemma 3.1: Given an *inhomogeneous* PPP X with *intensity function* λ, and the *thinning* function I_1, the resulting thinned point process, called *inhomogeneous Matérn I* PP, has the following average *number of points* at C:

$$\mathbb{E}[N(C)] := \int_C e^{-\int_{B(x,r)} \lambda(u)du} \lambda(x)\, dx \tag{3.9}$$

where r is the *thinning* radius of I_1.

Proof. First we define the auxiliary function g:

$$g: \quad \mathscr{R} \times \Omega \to \{0,1\} \tag{3.10}$$
$$(x,A) \mapsto \mathbb{1}_C(x)\; \mathbb{1}\,(dist(x, X \setminus x) > r) \tag{3.11}$$

where Ω is the space of events (see [BeW07]). We can then rewrite the average number of points at C as:

$$\mathbb{E}[N(C)] := \mathbb{E} \sum_{x \in X} g(x,X) \tag{3.12}$$

Next, we use the Campbell-Mecke formula to obtain:

$$\mathbb{E} \sum_{x \in X} g(x,X) := \int_{\mathscr{R}} \mathbb{E}^x[g(x,X)]\lambda(x)\, dx = \tag{3.13}$$
$$= \int_{\mathscr{R}} \int_{\Omega} g(x,X)\mathbb{P}^x(A)\, dA\; \lambda(x)\, dx = \tag{3.14}$$
$$= \int_C \int_{\Omega} \mathbb{1}\,(dist(x, X \setminus x) > r)\mathbb{P}^x(A)\, dA\; \lambda(x)\, dx = \tag{3.15}$$
$$= \int_C \mathbb{P}^x\,(dist(x, X \setminus x) > r)\;\; \lambda(x)\, dx = \tag{3.16}$$
$$= \int_C \mathbb{P}\,(N\,(B(x,r)) = 0)\;\; \lambda(x)\, dx \tag{3.17}$$

where $\mathbb{P}^x$ denotes the Palm probability. Finally, we apply the capacity functional (see [BeW07]) of a PPP to obtain the stated equality. ■

One drawback of these "*inhomogeneous Matérn I*" processes is their very restrictive *dependent thinning* procedure, which might reach the case where all the points are removed in certain neighborhoods. To overcome such limitation, our model considers a second type of processes, known as *Matérn II* point processes, that rely on *marked point processes* (see [BeW07]). *Matérn II* processes assign a *mark* to every point generated so as to allow the *dependent thinning* processes (see [BeW07]) distinguish which point is retained in a neighborhood.

Definition 3.2: Matérn II point process: is the point process obtained after applying a *thinning* with *index function*:

$$I_2(x,m,X,M_X) := \begin{cases} 1 & \text{if } m = \min_{m' \in M_X} \{(x',m') : \\ & \qquad\qquad x' \in B(x,r)\} \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

to a *stationary marked* PPP X, where M_X denote the marks associated to the point process X.

In other words, among all the points falling in the ball of radius r, only the one with the lowest mark m survives. These kind of processes present the advantage that even when the *intensity function* takes high values, at least one point remains in every neighborhood. In our model this means that in a certain neighborhood there is no more than one BS.

If rather than using a *stationary* PPP before applying the *dependent thinning* I_2, we use an *inhomogeneous* PPP (something novel in the SoA); then it is possible that the retained BS in a neighborhood is the one with higher $\lambda(x)$ by choosing the mark (x,m) as $m \sim \frac{1}{\lambda(x)}$. But still is missing how we can control that a correct number of BSs is generated at each region $R_i \subset R$ based on the repulsion radius r and the *intensity function* $\lambda(x)$. Thus we proceed as with the "*inhomogeneous Matérn I*" PPs to obtain an expression for the average *number of points* (something novel in the SoA).

Lemma 3.2: Given an *inhomogeneous* marked PPP X with *intensity function* λ, the *thinning* function I_2, and marks $m \sim \frac{1}{\lambda(x)}$, the resulting thinned point process, called *inhomogeneous Matérn II* PP, has the following average *number of points* at C:

$$\mathbb{E}[N(C)] := \int_C e^{-\int_{B(x,r)} \mathbb{1}(\lambda(u)>\lambda(x))\lambda(u)du} \lambda(x)\, dx \tag{3.19}$$

where r is the *thinning* radius of I_2.

Proof. As in the proof of Proposition 3.1, we proceed defining the function g:

$$g: \quad \mathscr{R} \times \Omega \to \{0,1\} \tag{3.20}$$

$$(x,A) \mapsto \mathbb{1}_C(x)\, \mathbb{1}\left(\lambda(x) = \max_{x' \in X \cap B(x,r)} \{\lambda(x')\}\right) \tag{3.21}$$

Then, the average number of points in a subset C can be expressed as in Eq. (3.12). Next, we apply the Campbell-Mecke formula as in the previous proof and we obtain the average number of points as:

$$\mathbb{E}\sum_{x\in X} g(x,X) := \int_C \mathbb{P}^x\left(\lambda(x) = \max_{x' \in X \cap B(x,r)}\right) \lambda(x)\, dx = \tag{3.22}$$

$$= \int_C \mathbb{P}\left(N\left(B_{>\lambda(x)}(x,r)\right) = 0\right) \lambda(x)\, dx = \tag{3.23}$$

$$= \int_C e^{-\int_{B(x,r)} \mathbb{1}(\lambda(u)>\lambda(x))\lambda(u)du} \lambda(x)\, dx \tag{3.24}$$

where $B_{>\lambda(x)}(x,r) = \{u : x \in B(x,r) \wedge \lambda(u) > \lambda(x)\}$. ■

The right-hand side (rhs) of Fig. 3.2 depicts how to obtain BSs' locations using an *inhomogeneous Matérn II* PP (whose average number of points is derived in Proposition 3.2). First, it generates the gray crosses using an *inhomogeneous PPP* with intensity function $\lambda(x)$ (note that more points are generated on the upper right corner because of the direction of $\nabla\lambda(x)$). Then, a I_2 thinning is applied using repulsion radius r with marks $m \sim \frac{1}{\lambda(x)}$, so only the gray cross with the most top-right coordinate falling within a ball of radius r survive. This surviving cross is hence illustrated as a cell tower in Fig. 3.2.

As both Propositions 3.1 and 3.2 state, the average *number of points* of "*inhomogeneous* Matérn I and II" PPs depend on the repulsion radius r. Then for a cell $R_i \subset R$, we need to know which r allows the generation of the required number of BSs. To generate $N(R_i)$ points we set the cell repulsion radius as:

$$r := \frac{2\sqrt{x_{1,s} \cdot x_{2,s}}}{\lceil \sqrt{N(R_i)} \rceil} \tag{3.25}$$

which grids R_i in cells that can contain the repulsion balls $B(x,r)$ of the *Matérn* PPs. These new cells are squares of side $2r$ and area $4r^2 > \pi r^2 = |B(x,r)|$, and they divide the parent cell R_i in $\lceil \sqrt{N(R_i)} \rceil^2$ smaller cells that can host a whole macro cell repulsion area.

3.3 MEC PoPs generation

We assign the traffic of every BS generated in Sec. 3.2 to a MEC PoP that is deployed somewhere within the operators' network infrastructure, and at a specific geographic location (i.e., to minimize RTT). The further the distance between the MEC PoP at location m, and a BS at location x, the higher the propagation delay $l(\cdot)$ of the link connecting them. The other two contributions for the packet RTT are the radio transmission delay t_r between a BS and the final user, and the packet processing delay $p(\cdot)$:

$$RTT := 2l\left(\|x-m\|_d\right) + 2p(M) + t_r \tag{3.26}$$

with $p(M)$ denoting the processing delay introduced by the network hops to be traversed to reach the network ring M. Hence, our model envisions the operator infrastructure as a hierarchy of network rings $\mathscr{M}$, in which traffic traverses more hops to reach network rings that are higher in the hierarchy (see Fig. 3.3). That is, taking $\prec$ as a relationship expressing which network ring is higher in the network hierarchy $\mathscr{M}$, if $M_a, M_b \in \mathscr{M}$ and $M_a \prec M_b$, we can say that a MEC PoP deployed at M_a is reached in less hops than one at M_b. Thus, $p(M_a) < p(M_b)$, and network ring M_a aggregates less traffic than M_b.

So if we place a MEC PoP at location m and assign it to network ring M, after fixing the maximum RTT and radio technology, from Eq. (3.26) we know the maximum distance to those BSs whose traffic is assigned to the new MEC PoP

$$\|x-m\|_d \leq l^{-1}\left(\frac{RTT - 2p(M) - t_r}{2}\right) = m_M \tag{3.27}$$

Algorithm 1 determines the MEC PoPs' deployment in three stages:

1. **Initialization**: the first stage (lines 1-4) creates the set of MEC PoPs locations, the set of their network rings, the set of BSs and one matrix of locations $x \in R$ per network ring. Each entry x in the matrix $matrices[M]$ represents the number of BSs that can be assigned to a MEC PoP deployed at a given location x and associated to the ring $M \in \mathscr{M}$
2. **Candidates search**: this second stage (lines 5-10) determines how many BSs can be assigned to a MEC PoP depending on its location and associated network ring $M \in \mathscr{M}$. For every M it loops through each BS and increases by one those location entries of $matrices[M]$ satisfying that if a MEC PoP is deployed there, it could satisfy the RTT and hence, the BS could be assigned to that MEC PoP. If $matrices[M_a][x_0] = 4$ after this stage, it would mean that a MEC PoP associated to M_a and deployed at x_0 would have 4 BSs assigned to itself.
3. **MEC PoPs selection**: this third and last stage (lines 11-35) is the main loop of Algorithm 1, and each iteration decides a new MEC PoP location and network ring. It comprises two phases:
   (a) **MEC PoP location**: this phase (lines 15-24) obtains the best location where a MEC PoP can be located. It iterates through all the possible network rings in $\mathscr{M}$ searching for the location where the maximum amount of BSs can be assigned to a MEC PoP. In case of

having multiple locations assigning the same amount of BSs at different rings, let's call them M_a, M_b, then the algorithm selects the location related to the ring with the minimum propagation delay, i.e., $\min\{p(M_a), p(M_b)\}$.

(b) **Assignment update**: the MEC PoP obtained in the previous step handles the traffic of up to maximum number of BSs *ringMaxBSs()*, depending on the network ring where it is associated. Taking into account that consideration, this phase (lines 28-34) iterates through every BS assigned to the MEC PoP and updates every $matrices[M]$ to reflect the assignment by decreasing in one unit the neighborhood of each BS. That is, if the new MEC PoP has an assigned BS at location x, the neighboring locations of x at every ring, i.e., $matrices[M][B(x, m_M)]$, must be decreased by one. Thus if another MEC PoP is later located inside $B(x, m_M)$, it knows that it will cover one less BS. Finally all the assigned BSs are removed from the unassigned list.

Hence, Algorithm 1 iterates through every MEC PoP candidate location m, and chooses the one maximizing the number of BSs falling inside the ball $B(m, m_M)$. This process minimizes the number of MEC PoPs, and is done for every possible network ring M to find the best (m, M) combination.

3.4 An example of macro cells and MEC PoPs generation

This subsection provides an example of how to apply the macro cell, and PoP generation procedures described in section 3.2 and section 3.3 to derive macro cells and PoPs locations urban, industrial, and rural scenarios. This subsection first characterizes the network traffic infrastructure to derive realistic values for the average number of BSs/km^2 and network RTT. Second, it characterizes three deployment areas in Spain, namely Madrid city center (104 km^2), Cobo Calleja industrial estate (8 km^2), and Hoces del Cabriel valley (2193 km^2), And third, it applies the generation of macro cells and PoPs in these three deployment areas.

3.4.1 Characterization of network traffic and infrastructure

According to the Next Generation Mobile Network Alliance (NGMN) [NGM16], 5G services are expected to be provided via ad-hoc network slices over the same physical infrastructure. In order to enable the desired traffic treatment in the network infrastructure, the 3GPP has defined a set of flows with the corresponding traffic requirements for the eMBB and URLLC slices [3GP18a], while NGMN in [NGM15] defined the traffic requirements for the MIoT slice. eMBB services are characterized by high bandwidth and span from classical mobile traffic (e.g., mobile terminals), to broadcast-like services (e.g., IPTV), high-speed vehicles (e.g., in-vehicle infotainment), indoor hotspot (e.g., fiber-like access), and dense urban (e.g., crows in a stadium, square). On the contrary, URLLC services are characterized by low latency and span from discrete automation (i.e., remote-controlled robots), to intelligent transport systems (e.g., autonomous cars), and tactile interaction (e.g., augmented reality). Finally, MIoT services are characterized by a high number of intermittent and low-power communications (e.g., sensors). Table 3.1 reports a selected number of the above traffic requirements as reported in [3GP18a, NGM15].

Based on the different slices and traffic flows introduced above, the authors in [Com+18a] first identify three reference deployment scenarios, namely urban, industrial, and rural. Next, they characterize the average number of BSs/km^2 for each scenario. Specifically, they report for the urban scenario an average number of 72 BS/km^2 in case of supporting the indoor hotspot traffic flow, which is characteristic of business districts and office areas that require 4 BSs per building floor. In residential/commercial areas instead, the average number of BS/km^2 is 12 in urban scenarios. Similarly, 12 BSs are also required in the industrial scenario to satisfy the traffic demand of 1 km^2. Finally, the rural scenario considers a 4-lane road (e.g., highway) supporting the intelligent transport system flow (e.g., V2X) and requires 1 BS per kilometer of road.

The characterization of the network traffic and infrastructure was carried out by Dr. Luca Cominardi in the collaboration that led to the content of the current chapter 3.

Algorithm 1: MEC PoPs placement.

```
Data: BSs, $R$, RTT
Result: MECPoPLocations, MECPoPRings
$matrices$ = array(int.matrix($R$), length = $|\mathscr{M}|$);
unassigned = Set(BSs);
MECPoPLocations = Set();
MECPoPRings = Set();
foreach $x$ in BSs do
    foreach $M$ in $\mathscr{M}$ do
        $m_M = l^{-1}\left(\frac{RTT-2p(M)-t_r}{2}\right)$;
        $matrices[M][B(x,m_M)]$ += 1;
    end
end
while not empty unassigned do
    covBSs = $-1$;
    MECPoP = NULL;
    ring = NULL;
    foreach $M \in \mathscr{M}$ do
        maxCov = $\max_{x'}\{matrices[M][x']\}$;
        moreBSsCovered = $maxCov < covBSs$;
        eQ = $(maxCov = covBSs \wedge\ p(M) < p(ring))$;
        if moreBSsCovered OR eQ then
            covBSs = maxCov;
            MECPoP = $x : matrices[M][x] = maxCov$;
            ring = $M$;
        end
    end
    MECPoPLocations.add(MECPoP);
    MECPoPRings.add(ring);
    ringMaxDis = $l^{-1}\left(\frac{RTT-2p(ring)-t_r}{2}\right)$;
    assignBSs = BSs $\cap$ $B(MECPoP,\ ringMaxDis)$;
    foreach $x \in$ assignBSs.subset(ringMaxBSs(ring)) do
        foreach $M \in \mathscr{M}$ do
            $matrices[M][B(x,m_M)]$ -=1;
        end
        unassigned.pop($x$);
    end
end
```

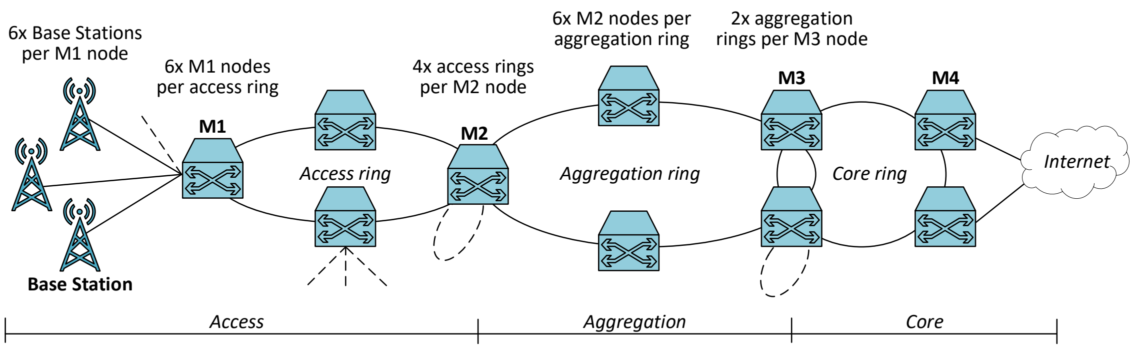

6x Base Stations per M1 node
6x M1 nodes per access ring
M1
Access ring
4x access rings per M2 node
M2
6x M2 nodes per aggregation ring
Aggregation ring
2x aggregation rings per M3 node
M3
Core ring
M4
Internet
Base Station
Access
Aggregation
Core

Figure 3.3: Reference network infrastructure as illustrated in [Com+18a] and based on [ITU18]

Table 3.1: Exemplary 5G traffic requirements.

SLICE	FLOW	REQUIREMENTS
eMBB	Indoor Hotspot	Up to 1Gbps/user
	Broadcast services	Up to 200 Mbps/TV channel
	High-speed vehicle	Up to 100 Gbps/Km^2
URLLC	Discrete automation	Maximum jitter of 1 μs
	Intelligent transport	Reliability of 99.9999%
	Tactile interaction	Maximum latency of 0.5 ms[1]
MIoT	Sensors data	Up to 200 Mbps/Km^2

[1] *Note*: end-to-end maximum delay as defined in [3GP18a].

The next step is to characterize the network infrastructure. To that end, we leverage the reference network infrastructure illustrated in [Com+18a] and based on [ITU18]. The network infrastructure comprises three segments: (*i*) access, (*ii*) aggregation, and (*iii*) core. The access comprises 6 BSs for each node M1 connected via a point-to-point link, and 6 nodes M1 connected in a ring topology. Thus, each access ring hence connects a total of 36 BSs. It is worth highlighting that from a network topology point of view, there is no difference whether the BSs are macro, micro, pico, and any variation thereof. For the sake of validating our model, what really matters is the number of BSs and how they are connected to the transport network.

Next, each aggregation ring comprises 6 M2 nodes, each of which serves 4 access rings. In turn, each aggregation ring is served by two M3 nodes for redundancy reasons, while each M3 node provides gateway capabilities to 2 aggregation rings. Finally, M4 nodes are connected to the core ring and serve as gateway to M3 nodes. According to the reference network infrastructure, any of these BSs and M nodes (e.g., M1, M2, M3, and M4) is a good candidate for placing a MEC PoP.

To better understanding which location is most suitable, we need to characterize the RTT. To that end, all the network links are assumed to be fiber optic and are characterized by a propagation delay of 5 μs/km [Cav+17]. We also consider a processing delay of 50 μs on each of the M nodes [EAN18, EAN16]. Therefore, the Eq. (3.26) becomes:

$$RTT = 2d \cdot 5\frac{\mu s}{km} + 2M \cdot 50\mu s + UL + DL \tag{3.28}$$

where d stands for the distance in kilometers between the MEC PoP and the BS, and M is the number of M nodes (i.e., number of hops) being traversed. For instance, $M = 0$ in case of collocating the MEC PoP with the BSs, $M = 1$ in case of collocating the MEC PoP with M1 nodes, $M = 2$ in case of collocating the MEC PoP with M2 nodes, etc. Therefore, the first two terms in the right hand side of Eq. (3.28) correspond to the propagation delay RTT and the packet processing delay

Table 3.2: NR profiles satisfying the tactile interaction latency

PROFILE	DL	UL	M1 DISTANCE	M2 DISTANCE
FDD 30 kHz 2s	0.39 ms	0.39 ms	12 km	2 km
FDD 120 kHZ 7s	0.33 ms	0.33 ms	24 km	14 km
TDD 120 kHz 7s	0.39 ms	0.39 ms	12 km	2 km

Note: FDD 30 kHz 2s stands for Frequency Division Duplex scheme with a subcarrier of 30 kHz and 2 symbols.
Note: DL and UL values are the worst case transmission latency presented in [Sac+18].

RTT in the transport network, respectively. The last two terms (i.e., Uplink (UL) and Downlink (DL)) correspond instead to the uplink and downlink delay over the radio link.

The 3GPP defines multiple profiles for the radio interface (i.e., NR) and each of these profiles is characterized by distinct UL and DL delay values [Sac+18]. Bound to the most stringent one-way latency of 0.5 ms for the tactile interaction URLLC traffic flow (see Table 3.1), the BSs used for the results exposed in Sec.3 are 5G gNodeBs with the suitable radio profiles that satisfy $UL+DL < RTT = 1ms$; hence "BS" refers to a 5G gNodeB from now on. Table 3.2 reports the NR profiles used, which all adopt an uplink semi-persistent scheduling (SPS), and the maximum distances d from a BS to a MEC PoP.

3.4.2 Characterization of the deployment areas

Based on the identified urban, industrial, and rural scenarios, we select three reference areas in Spain to apply our model (see Sec. 3.1 and Sec. 3.3) and obtain the MEC PoP locations. Specifically, we select *Madrid city center* for the urban scenario, *Cobo Calleja* area for the industrial scenario, and *Hoces del Cabriel* valley for the rural scenario; then we consider the following characterization aspects.

1. **Characterization of $G(x)$**: before applying the model we characterize each scenario's *gentrification* function $G(x)$, *revolution function* f_i, and population circles C_i. Particularly f_i is based on the smooth step function, which is derived from Hermite interpolation polynomials [BF11] and has the following expression:

$$S_N(x) = \begin{cases} 0 & x \leq 0 \\ x^{N+1} \sum_{n=0}^{N} \binom{N+n}{n} \binom{2N+1}{N-n} (-x)^n & \text{if } 0 \leq x \leq 1 \\ 1 & \text{if } 1 \leq x \end{cases} \tag{3.29}$$

more specifically we define f_i as:

$$f_i(x) = \begin{cases} 0 & if\ \|x-c_i\|_V > \frac{b}{2}+a \\ P_i & if\ \|x-c_i\|_V \leq \frac{b}{2} \\ S_N\left(\frac{b}{2}+a-\|x-c_i\|_V\right) & if\ \frac{b}{2} < \|x-c_i\|_V < \frac{b}{2}+a \end{cases} \tag{3.30}$$

where P_i is the population present in the population circle C_i with center c_i. The *revolution function* f_i takes the value P_i in the circle $B\left(x, \frac{b}{2}\right)$ and transitions from P_i to 0 in the outer disk, $D\left(x, \frac{b}{2}, \frac{b}{2}+a\right)$, where $\|\cdot\|_V$ denotes Vincenty's distance [Vin75] using the WGS-84 datum [W3C06].

For Madrid city center we consider as population circles the different districts inside the M-30 ring highway, which administratively identifies the city center and its outskirts. Then, for each of these districts we obtain the P_i, a, b values from the city center population census [Pad18]. For what concerns the Cobo Calleja industrial area, around 100000 people[3] work daily in the area, and one population circle is enough to cover the 8 km^2 region. In Madrid city center and Cobo Calleja, each population circle center c_i lies over a LTE tower retrieved from OpenCellID [18]. Finally, regarding the Hoces del Cabriel rural area, rather than using the population to determine the BSs' location, we limit our analysis to the location of 1 BS/km [Com+18a] along the A-3 highway crossing the rural area.

2. **Characterization of BSs' generation**: to generate the BSs' locations, this work uses the *inhomogeneous Matérn II* point process described in Proposition 3.2 together with the average number of BSs described in Sec. 3.4.1. During our experimentation with such process and the "*inhomogeneous Matérn I*", we have realized that the latter has a more restrictive *thinning* I_1 (shown in (3.8)), that did not allow us achieving the desired *average number* of macro-cells. The reason is that an increase in the *intensity function* λ does not correspond to an increase in the resulting number of points. And this is because an *inhomogeneous PPP* satisfies that $\mathbb{P}(N(A) > 1) \propto \lambda$, hence the $\mathbb{E}[N(A)]$ of equation (3.9) decreases with λ after I_1 is applied. Graphically one can understand this behavior by realizing that the more points we generate (i.e., the higher λ), the less probability we have that a point is alone in its neighborhood (i.e., the less likely it will survive to I_1 *thinning*).
   Unlikely, the described behavior was not experimented with the "*inhomogeneous Matérn II*" point process when providing proper values of the repulsion radius r for the *thinning* I_2.
   In the case of Madrid city center and Cobo Calleja areas, we require an average of $\mathbb{E}(N(R_i)) = 12$ BSs/km^2 [Com+18a][4], which is obtained by taking $x_{1,s} = x_{2,s} = 1$ km, $r = 2 \cdot 1\text{km}/\lceil\sqrt(12)\rceil$ (see Eq. (3.25)) and $\lambda(x) = k \cdot G(x)$, where $k = 16$ for Madrid city center and $k = 13$ for Cobo Calleja. These values have been obtained using the average *number of points* expression that we derive in Proposition 3.2.
   The urban scenario necessitates additional considerations on the indoor hotspot traffic flow, which is not ubiquitous but rather present at few and specific location in Madrid city center. Following the same approach of [Com+18a], we consider the indoor hotspot traffic flow to be present only in office buildings[5] which require 4 femtocells on each floor, see Table 3.3.
   On the other hand for Hoces del Cabriel Valley, as mentioned in section 3.4.1, we locate 1 BS/km along the highway to support the intelligent transport system flow. That is, the location and number of BSs follows the route of the highway rather than the population.
3. **Characterization of MEC PoPs maximum distances**: Algorithm 1 uses Eq. (3.27) to determine the maximum separation between a MEC PoP and a BS. To do so it needs to know the used distance for the propagation delay between a MEC PoP and a BS (i.e., what is d in $l(\|x-m\|_d)$), and the used BSs' NR profile of Table 3.2. Since Sec. 3.4.1 assumes that a BS is connected to a MEC PoP with a fiber link, which are usually installed along the road lanes, Algorithm 1 uses the Manhattan distance, so we have $l(\|x-m\|_d) = l(\|x-m\|_1)$. Regarding the NR profile, it assumes that all the generated BSs have the same radio technology to have a fixed value of $t_r = UL + DL$ in Eq. (3.27). Hence we need a dedicated execution per NR profile to know how it affects the MEC PoPs' deployment.
   For sake of clarity, when applying our model to the urban and rural scenarios, it might happen that some of the generated points are not be physically suitable for hosting a Next Generation NodeB (gNB) (e.g., they fall in the middle of a road). Nevertheless, given the propagation delay of 5 μs/km for fiber optics, a slight misplacement of gNodeBs is negligible since it would only vary the end-to-end delay of few microseconds, which is an order of magnitude smaller when considering an overall latency of milliseconds.

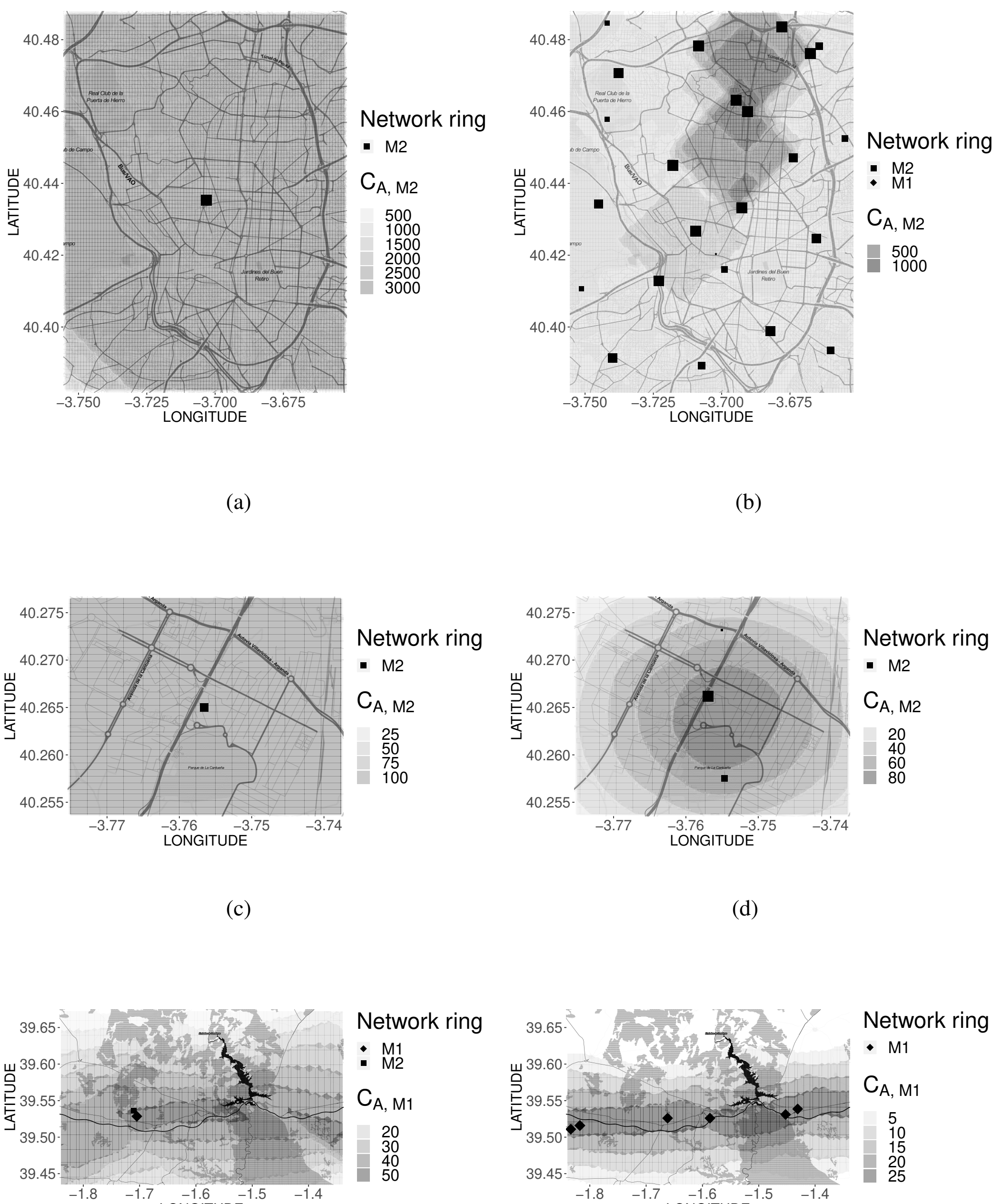


Figure 3.4: MEC PoP locations for FDD 120 kHz 7s on the left. Locations for FDD 30 kHz 2s and TDD 120 kHz 7s on the right. Top maps correspond to the urban scenario (Madrid city), middle to the industrial scenario (Cobo Calleja), and the bottom to the rural area scenario (Hoces del Cabriel). The markers shape indicates the network ring a MEC PoP is associated to. For each coordinate, heat map $C_{A,M1}$ denotes the number of BSs that can be assigned to a MEC PoP deployed in the target location and assigned to network ring M1. Similarly, heat map $C_{A,M2}$ denotes the number of BSs that can be assigned to a MEC PoP deployed in the target location and assigned to network ring M2.

Table 3.3: Madrid city skyscrapers and femtocell requirements

NAME	LOCATION	FLOORS	FEMTOCELLS
Torre espacio	(40.479, -3.686)	56	224
Torre Madrid	(40.424, -3.714)	37	148
Castellana 81	(40.447, -3.694)	28	112
Torres de Colón	(40.425, 40.425)	23	92
Torre titania	(40.446, -3.696)	23	92
Torre Mahou	(40.448, -3.696)	29	116
Puerta de Europa 1	(40.466, -3.692)	26	104
Puerta de Europa 2	(40.466, -3.690)	26	104

Table 3.4: Number of MEC PoPs necessary per NR profile

R	NR	M1 MEC PoPs[1]	M2 MEC PoPs[1]
Urban	FDD 120 kHz 7s	0	21
	FDD 30 kHz 2s TDD 120 kHz 7s	3	30
Industrial	FDD 120 kHz 7s	0	1
	FDD 30 kHz 2s TDD 120 kHz 7s	0	3
Rural	FDD 120 kHz 7s	2	1
	FDD 30 kHz 2s TDD 120 kHz 7s	9	0

[1] For Urban and Industrial, average number of MEC PoPs across the 100 simulations.

Simulation results

To obtain the MEC PoP locations we run 100 simulations using R and the `spatstat` package [BRT15]. Each simulation[6] consists of two steps:

(a) Generation of 12 BSs/km^2 using "*inhomogeneous Matérn II*" PPs with the parameters derived from Sec. 2 (in Madrid city center the indoor hotspot BSs are included as well);

(b) Generation of MEC PoPs using the NR profiles of Table 3.2, and Algorithm 1;

For the rural scenario of Hoces del Cabriel valley, we skip step 3a) and manually generate 1 BS/km along the A-3 highway that crosses the region. Therefore, we only run step 3b) of the simulation to obtain the MEC PoPs needed for the BSs generated across the highway.

For each scenario only one of the 100 simulations is depicted in Fig. 3.4, where we represent the geographical locations of the MEC PoPs as squares or rhomboids depending on whether they are associated to network ring M2 or M1, respectively. Urban, industrial and rural scenarios are illustrated in Fig. 3.4 top, middle, and bottom, respectively; with left and right

[3] Estimation based in the number of employees working in Cobo Calleja companies [Reg18].

[4] Here we are not considering the indoor hotspot traffic flow for the urban scenario.

[5] In this work we consider office buildings with more than 15 floors [Min18].

[6] Code available at: `https://github.com/MartinPJorge/mec-generator/tree/32513cbb7fa2ec3c22567a944d234dc6dd051a36`

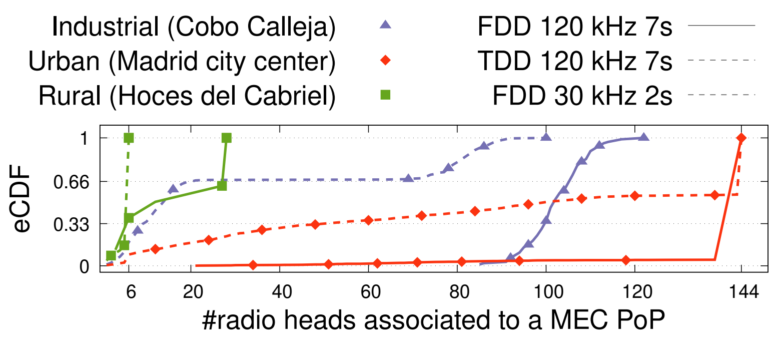


Figure 3.5: eCDF of the number of BSs assigned to a MEC PoP in the studied scenarios.

figures representing how MEC PoP locations vary depending on the used radio technology. A heat map is then used to show the number of BSs $C_{A,M}$ that can be assigned to a MEC PoP at level M. The darkest area in the rhs of Fig. 3.4(d) means that any MEC PoP associated to M2 and deployed inside that area has $C_{A,M2}$=80 BSs whose traffic can be assigned to itself. The average number of MEC PoPs for each scenario is reported in Table 3.4. Results show that collocating the MEC PoPs with the BSs doesn't provide enough advantages in terms of BSs aggregation and target traffic delay requirement. This is because the network delay (see Eq. (3.28)) can be satisfactorily fulfilled by aggregating more BSs in fewer MEC PoPs at higher network rings (e.g., M1, M2, etc.). In fact, Algorithm 1 minimizes the number of MEC PoPs whilst fulfilling the traffic requirements. Such traffic requirements (e.g., 1 ms RTT constraint for the URLLC slice) are never satisfied when the MEC PoPs are located at the M3 and M4 network rings, yielding to empty $matrices[M3]$ and $matrices[M4]$. The reason of such behavior is that packet processing delay (see Eq. (3.28)) increases linearly with the number of network rings to be traversed. As a result, the MEC PoPs have been always associated to M1 or M2 network rings in all our simulations, as it can be appreciated in one of the simulation realizations shown in Fig. 3.4.

The lower the packet processing time, the higher the maximum distance between a BS and a MEC PoP (see Eq. (3.28)). Thus MEC PoPs associated to M1 have more candidate BSs to be assigned than MEC PoPs associated to M2. But among all the candidate BSs it can only have 6 BSs assigned, while a MEC PoP associated to M2 can have up to 144 BSs assigned. For both the urban and industrial scenarios, the results of our 100 simulations (see Table 3.4) show that most of the MEC PoPs are associated to M2. Since both scenarios have short distances and propagation delays because of the high density of BSs/km^2, the addition of M2 packet processing delay does not exceed the 1 ms RTT of URLLC. Therefore, Algorithm 1 associates the MEC PoPs to the M2 network ring, and assigns them as many BSs as possible to reduce the number of MEC PoPs. Conversely, looking to Figure 3.4(e)-(f), more MEC PoPs are associated to M1 in the rural scenario because distances and propagation delay to BSs are high enough to exceed the 1 ms RTT when MEC PoPs are associated to M2.

Regarding the different NR profiles (see Table 3.2), low UL and DL delays of FDD 120 kHz 7s permit to increase the maximum distance between a BS and a MEC PoP. Therefore, a MEC PoP can serve a larger number of BSs, as shown in the darker heat maps at the lhs of Fig. 3.4. As a result, using FDD 120 kHz 7s as NR profile necessitates the deployment of fewer MEC PoPs compared to all the other NR profiles (see Table 3.4). Indeed, any MEC PoP location in the urban and industrial scenario can serve any FDD 120 kHz 7s BSs in the region as shown in Fig. 3.4(a)-(b) and Fig. 3.4(c)-(d). Instead, FDD 30 kHz 2s and TDD 120 kHz 7s impose a higher UL and DL delay, thus requiring a shorter distance between a BS and a MEC PoP. This results in a larger number of MEC PoPs sparsely located in the region (see rhs of Fig. 3.4).

Fig. 3.5 illustrates the experimental Cumulative Density Function (eCDF) for the number of

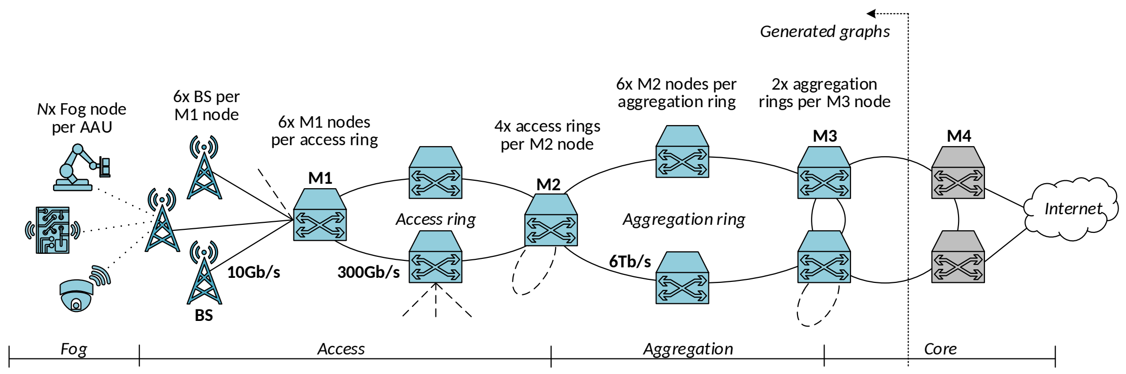


Figure 3.6: Network topology of the generated infrastructure graphs

BSs assigned to a MEC PoP. Results show that the FDD 120 kHz 7s NR technology increases the number of BSs associated to a MEC PoP (less than the 33% of them have less than 100 BSs assigned in the industrial and urban scenario), while the other NR technologies lead to a higher percentage of MEC PoPs with fewer BSs assigned. For example, 68% of the MEC PoPs of the industrial scenario have less than 70 BSs assigned when TDD 120 kHz 7s or FDD 30 kHz 2s are used, which is less than half the BSs that can be assigned to a MEC PoP associated to M2.

Summarizing, there is a trade-off between the performance of the NR profile and the number of MEC PoPs. Higher performance radio profiles, which can be more expensive, allow to associate a larger number of BSs to a MEC PoP, resulting in fewer MEC PoPs. On the contrary, a less performant radio profile, which is cheaper, requires a large number of MEC PoPs for satisfying URLLC traffic. This trade-off should be taken into consideration by the network operators to optimize costs when building their network.

3.5 Operator graph generation with 5GEN

Given the macro cells generation described in Sec. 3.2, and MEC PoP generation of Sec. 3.3, it is still left to generate a graph to represent the network infrastructure of a 5G operator. To that extent, this section introduces 5GEN, an R package that creates graphs as the one depicted in Fig. 3.6. The package includes a set of functions that allow the creation of the operator graph following clustering strategy. Paragraphs below detail the steps and functions invoked for the generation the operator graph. The idea is to create clusters of 6 BSs that can be derived as described in Sec. 3.2, and connect each one to a M1 switch. Then, access and aggregation rings are created as groups of M1 and M2 switches, respectively. Hence, 5GEN package takes care of the assignment of BSs' traffic to switches, which misses in the previous sections of the present chapter. Since, Sec.3.2-3.4 covered the BS generation, and the assignment of them to PoPs at different rings in the network infrastructure, but not the linkage of BSs towards the operator switches.

The *build5GScenario* function (see Algorithm 2) generates 5G infrastructure graphs based on a set of BSs. First, it creates a dendrogram[7] of BSs based on the distances between them (line 5). Then, line 6 cuts the dendrogram in clusters where BSs have a maximum distance of 10km among them, and such clusters are split in line 7 into groups of no more than 6 BSs. Every group of 6 BSs is later connected to the nearest $M1_n$ switch based on Vicenty's distance [TVi75].

To generate the access and aggregation rings, line 14 connects each group of 6 M1/M2 switches (respectively) in a ring fashion, and creates groups of $g = 4$ access rings, and $g = 2$ aggregation rings in line 16. As last step, line 19 links M2 switches to each of the 4 access rings R inside a group G_i. Same process is done to link groups of 2 aggregation rings to M3 switches.

[7] *dendrogram*: tree diagram representing a hierarchical grouping of elements.

Algorithm 2: build5GScenario

```
Data: BSs
Result: access and aggregation graph
levels = [BSs, access, aggregation];
distances = [10km, 20km, 40km];
foreach (l, d) in (levels, distance) do
    $$N = \begin{cases} BSs, & \text{if } l = \text{BSs} \\ M1, & \text{if } l = access \\ M2, & \text{if } l = aggregation \end{cases}$$
    $H$ = hierarchical_clustering($N$);
    $C$ = cut_dendrogram($H$, $d$);
    $R = \cup_{C' \in C} \left\{ \left\{ C'_{i+6j} \right\}_{i=0}^{\min\{C'-1-6j,5\}} \right\}_{j=0}^{\lfloor C'/6 \rfloor}$;
    if $l = BSs$ then
        for $R_j \in R$ do
            $\text{M1}_n = \min_{\text{M1}} \sum_{r \in R_j} \text{Vicenty}(\text{M1}, r)$;
            $\text{connect}(r, \text{M1}_n)$, $\forall r \in R_j$;
        end
    else
        $\text{connect}(r_i, r_{i+1 \mod 6})$, $\forall r_i \in R_j$, $\forall R_j \in R$;
        $g = 4$ if $l = \text{M1}$ else 2;
        $G = \left\{ \left\{ R_{i+gj} \right\}_{i=0}^{\min\{R-1-gj, g-1\}} \right\}_{j=0}^{\lfloor R/g \rfloor}$;
        for $G_i \in G$ do
            upper=M2 if $l$ =M1 else M3;
            connect'$(R, \text{upper})$, $\forall R \in G_i$;
        end
    end
end
```

Once Algorithm 2 returns a graph with the access and aggregation rings, the next step is to attach computing resources to the generated infrastructure invoking 5GEN *attachServers*, and *attachFogNodes*. Note that *attachServers* would be fed with the MEC PoP generation described in Sec. 3.3. Our tool allows to specify where to attach resources in the graph (fog nodes are always attached to the nearest BS), and how many CPU, memory and disk they have. Both *attachServers* and *attachFogNodes* collapse all the information in two R data frames that contain nodes and edges of the infrastructure graph. One data frame contains the switches, BSs, servers and fog devices. The other contains edges representing fiber links among switches, links between them and the generated servers, and the wireless connectivity of BSs and fog devices. 5GEN allows further customization, thanks to the *addNodeProps* and *addLinkProps* functions, which can be used to add or edit properties of the nodes and edges of the generated graphs.

To sum up, the generation of a infrastructure with 5GEN comprises the invocation of the following functions[8]:

1. *build5GScenario(BSs)*;

[8] For further details check code snippet examples at: `https://github.com/MartinPJorge/mec-generator/tree/f7e0aa3b7db2b24eb910e623e4ad2d8d9ada9718`

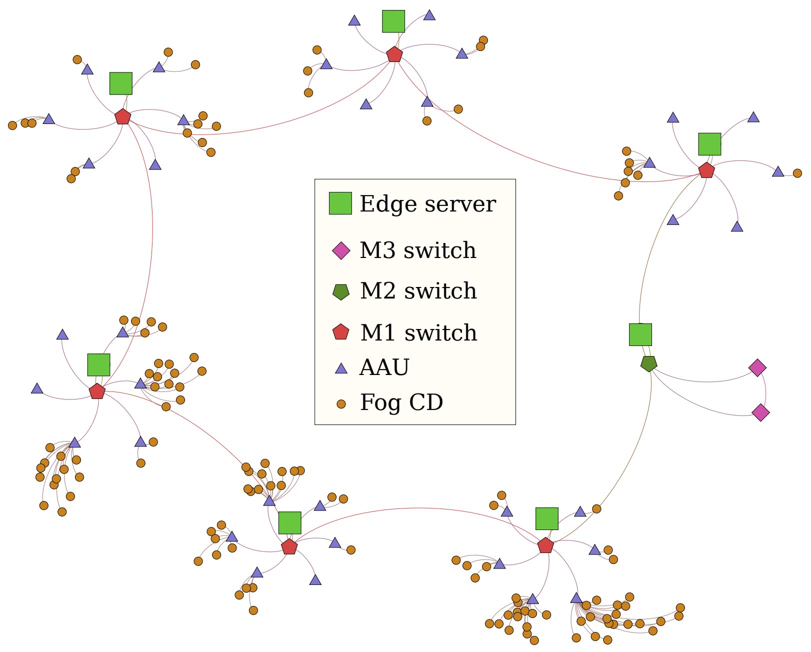


Figure 3.7: 5G infrastructure graph with fog and MEC entities

2. *attachServers(number, properties)*;
3. *attachFogNodes(number, properties)*; and
4. (*addNodeProps* or *addLinksProps*).

After that, the user generates a GML graph file calling *igraph::graph_from_data_frame* [CN06] using 5GEN data frames of nodes and edges.

Fig. 3.7 shows an instance of Fig. 3.6 reference 5G graph, generated invoking the functions above using the BSs generated in Sec. 3.4 for Cobo Calleja, Madrid. Such set of BSs are included in the *cobo* dataset of 5GEN R package.

3.6 Multi-domain graphs

Given the arise of Network Function Virtualization (NFV), the management of the network infrastructure has became more flexible. The network operator has the chance of deploying a Network Service (NS) on the fly with the respective Network Service Descriptor (NSD)s, and an adequate infrastructure supporting the ETSI Management and Orchestration (MANO) [ETS14] architecture. As a consequence, European projects as 5GEx [Ger+17], 5G-TRANSFORMER [Oli+18], and 5GROWTH [Saa+]; started to raise as solutions to cope with the management and orchestration of NFV architectures. Furthermore, and what is of interest in the current section, such projects considered the possibility of assessing the management and orchestration in multi-domain environments, i.e., on scenarios in which multiple network operators put their administrative domains under a common orchestration architecture.

When multiple domains are being managed by a central entity, it is natural to consider that it is capable of delegating NSs deployments across domains. For example, if a network operator wants to deploy a virtual Content Delivery Network (CDN) NS in an area where it has no administrative domain, such service could be deployed in such area on other network operator administrative domain. To do so, it is necessary an agreement in between operators, so they can make use of others' pool of resources. The 5GEx architecture assumed that network operators shared and abstracted information on the pool of resources that they shared with other operators administrative domains. In particular, such abstracted view of available resources was exchanged relying on top of the Big Switch with Big Software (BiS-BiS) virtualization concept, i.e., administrative domains shared among them BiS-BiS nodes [Son+15] with information regarding networking and computing

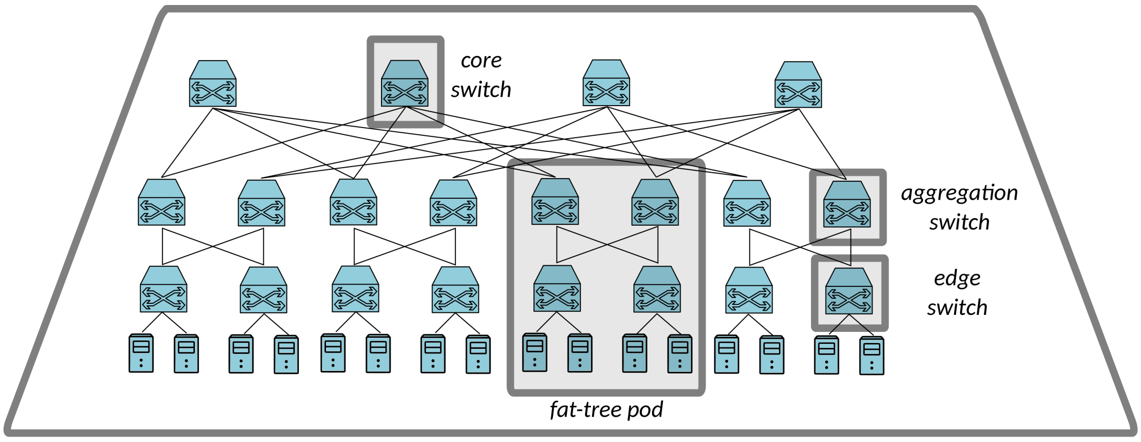


Figure 3.8: Illustration of a fat-tree of size $k = 4$.

capabilities.

When network operators share a pool of resources among them, it is said that they share a pool of federated resources, i.e., resources that they can access and even manage, to deploy their own NSs. We refer to scenarios in which operators deploy NSs on others' administrative domains, as federation scenarios. If a NFV orchestration algorithm has to solve the Virtual Network Embedding (VNE) problem in federation scenarios, it will account for multiple domains with their corresponding resources. To evaluate the performance of such algorithms, the present thesis models the multiple domains using graphs, which are referred to as the title of the present section, i.e., *multi-domain graphs*.

Each domain is assumed to have a data center with a fat-tree topology [Lei85], see Fig. 3.8. Every domain shares resources of its own data center, and neighboring domains can access them using a meshed connection of gateways that steer the inter-domain traffic.

The multi-domain graph consists of a set of nodes that either represent a gateway or a switch, or a server present in a pod of the fat-trees. Edges represent the links inside the fat-tree, or even the links connecting the gateways that steer the traffic in between the different domains' fat-trees. Each domain has its own graph representing its network resources, so as the resources that other domains share with itself in the federation. As a result, each domain knows a graph $G_d = (N_d, E_d)$ with all the resources it can use. Every node $n \in N_d$ has its computational capabilities $c_{n,d}, m_{n,d}, h_{n,d}$, which correspond to the CPU, memory and hard disk available; respectively. Similarly, every edge $e \in E_d$ representing a link has its associated $b_{e,d}, d_{e,d}$ bandwidth and delay, respectively.

The multi-domain graph $G = (N, E)$ is just the graph resulting of all the domain graphs, and the addition of their respective properties. That is, $N = \cup_d N_d$ and $E = \cup_d E_d$, with[9] $c_n = \sum_{d:n\in N_d} c_{n,d}$ and $b_e = \sum_{e:e\in E_d} b_{e,d}$.

In the generation of the multi-domain graph, it is required to split the resources across the different domains. Steps below specify the followed procedure:

1. First, it is to **generate the gateways** that interconnect the fat-trees of the different domains. Then, a wheel graph [Tru13] is generated with every node representing the gateway that each domain's fat-tree connects to. The result is a wheel graph W_D with D denoting the number of available domains. Once W_D is created, it is possible to select if the gateways should be connected on a mesh fashion or not. That is controlled with the mesh degree $m \in [0, 1]$ parameter, which denotes whether the gateways are all interconnected in a mesh fashion $m = 1$, or as a wheel $m = 0$. Values in between correspond to additional links in between domains' gateways. In particular, the number of additional links is computed as $m \cdot (\binom{D}{2} - D)$.
2. Second, it is to **generate the fat-trees** of every domain d. In particular, (i) to generate the fat-

[9] the same properties apply for the memory m_n, hard disk h_n, and delay d_e properties of servers and links.

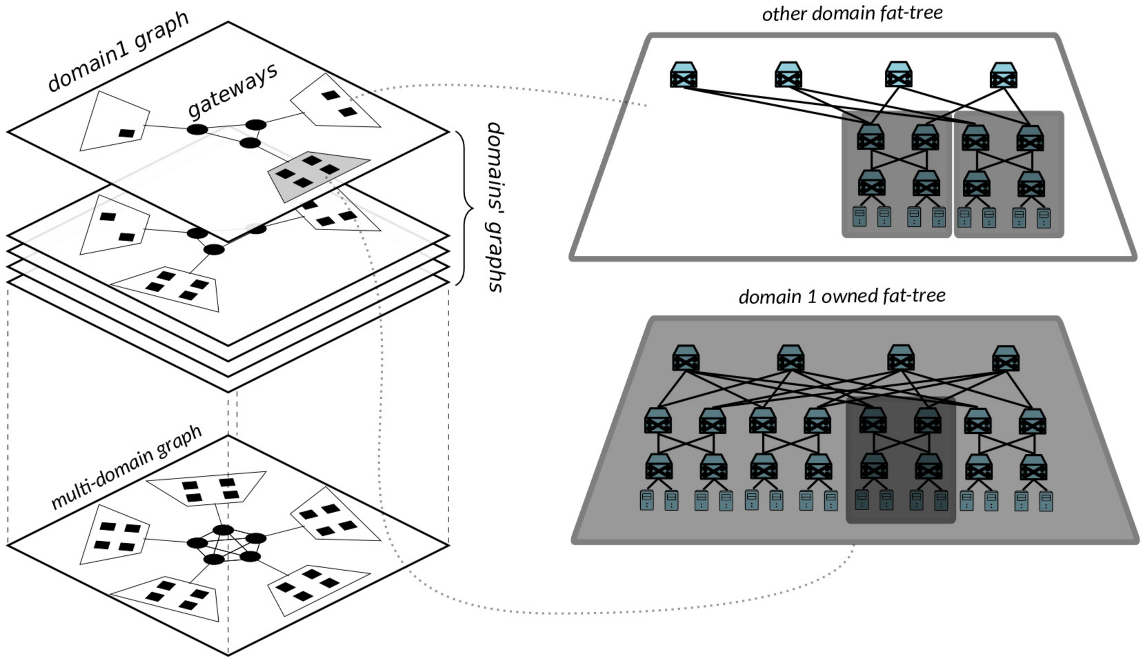


Figure 3.9: Illustration of a multi-domain graph.

tree of size $k = 2n$, $n \in \mathbb{N}$ [Lei85]; and (*ii*) validate the correctness of the generated graph. The first stage consists in generating a graph having as nodes the servers and switches presented in Fig. 3.8, so as the links connecting servers as links; as nodes and edges of the domain graph G_d, respectively. The second stage consists in validating the fat-tree topology checking (*i*) there are $P_2 = \frac{k^2}{2}\binom{k/2}{2}$ paths of 2 hops in between servers; (*ii*) $P_4 = k[\binom{k^2/4}{2} - \frac{k}{2}\binom{k/2}{2}]$ paths of 4 hops; and (*iii*) $\binom{k^3/4}{2} - P_4 - P_2$ paths of 6 hops – see [Lei85].

3. Third, after the fat-trees of every domain are generated, it is necessary to specify how many **resources** each of them will **share** with other. In particular, each domain shares with a foreign domain the resources of one or several pods of the fat-tree, in addition to the core, aggregation, and edge switches required to access the pod (links connecting the mentioned elements are also added to the domain graph G_d). Moreover, to access other domains' pods, the domain graph incorporates the required gateways and links to reach its facilities. As a result, the domain graph has a partial view of the multi-domain graph G, see Fig. 3.9.
4. Fourth, the available **bandwidth** has to be **shared** among the different domains that hold access to a specific link $e \in E$. Each domain will have $b_e \cdot p_d$ Mbps with $p_d \in [0,1]$ being the portion of held by each operator, and $\sum_d p_d = 1$.
5. Fifth, and similarly to the bandwidth sharing, it is the **split of computational resources** in the fat-trees. It is taken as assumption that an administrative domain shares a half of its resources to the other domains, e.g., domain $d = 1$ only shares $\frac{c_{n,d}}{2}$ of the CPU resources of server n_d. Hence, the CPU resources of the pod servers' shared are computed as $\frac{c_n}{2} \cdot p_d$. Same applies to the memory, and hard disk shared resources.

Fig. 3.9 illustrates how a each domain graph looks like after steps 1-5 are performed. The illustration shows how domain $d = 1$ only has a full view of its own fat-tree data center, and how it only has a partial view of the fat-tree resources of another domain fat-tree. Thanks to this, any NFV orchestration algorithm developed for domain 1, will be able of using the resources that the federated multi-domain environment offers, and it is completely agnostic of the resources' property, as it is legit to use the federated pool of resources. Section 4.2 evaluates the performance of different NFV orchestration algorithms for federation scenarios, using the multi-domain graphs explained in this section.

3.7 Conclusions

This chapter presents how to derive 5G infrastructures, and it discusses not only the generation of the associated graphs that represent such infrastructures, but the methods to derive the deployment and locations of BSs and MEC PoPs to satisfy 5G service requirements.

The chapter proposes the use of PPPs to generate BSs in order to ensure the radio coverage requirements of 5G. Moreover, the election of the PoPs' geographical location, is done so as to satisfy the latency constraints of 5G services. Results show that the BSs' technology is a key factor that impacts on the spread of PoPs across the network operator area.

Given the procedures to derive the BSs and MEC PoPs, the chapter presents the `5GEN` R package, which generate graphs that represent 5G operator infrastructures. Additionally, the chapter presents how to generate multi-domain graphs for federated environments with shared resources.

Future research directions would be to derive optimal network operator deployments using customers demand. That is, given the geographical location of users over an area, to derive the location of BSs and PoPs to satisfy current or future service demands. Note that this chapter had to use the random generation of BSs assuming that the amount of operator subscribers is proportional to the population of an area. If rather, one knows the location of the subscribers and their mobility patterns, it is possible to derive an optimal 5G network deployment to meet the requirements for a real demand.

Another future research direction would be to extend the multi-domain graph generation, so it accounts not only for fat-tree data center topologies. Additionally, it would be possible to extend the idea of having multiple graphs, one per domain, for graphs considering the Radio Access Network (RAN), and traffic steering infrastructure. That is, to generate multiple infrastructure graphs generated by `5GEN`, one per operator, and create graph views of the shared resources among them; as done in the multi-domain graphs of the present chapter. Such graphs would be useful for the evaluation of network slicing solutions for federated environments.

4. NFV Orchestration in federated environments

Thanks to the NFV paradigm discussed in section 1.2, the deployment 5G NSs is split all across the network resources. Each constituent Virtual Network Function (VNF) of the NS has the chance of running at whatever server in the network infrastructure as long as the service requirements are satisfied. NFV orchestration algorithms have to find the best allocation of both the VNFs and NSs in the underlying substrate network. The solutions have to take the advantage of the flexibility of the NFV paradigm to maximize the utilization of the available infrastructure to increase benefits. On top of that, it is necessary to account for federated scenarios in which administrative domains may introduce pricing changes with the peering domains, so as the limited resource capacity offered in the federation pool of resources.

This chapter focuses on NFV orchestration algorithms to tackle NS deployments in federated environments. In particular, section 4.1 motivates the study of federation scenarios showing that the deployment of federated services is fast enough to be introduced in production environments, thus, worth study as a realistic scenario. Given the short deployment times in federated networks, section 4.2 presents the analysis of an NFV orchestration algorithm that solves the VNE problem in federated networks represented by the multi-domain graphs discussed in section 3.6. Following the study of federated networks, section 4.3 focuses on the delegation of NSs deployment to federated domains, when federation prices change over time. And section 4.4 closes the chapter summarizing the contributions, and open future work and challenges in the orchestration of NSs in federated environments.

4.1 Is federation fast enough?

Network federation gives to administrative domains the possibility of deploying services there where they do not have facilities, or to deploy services in moments where there are no resources available under their facilities. The 5GEx [5GE18] European project proposed a multi-domain infrastructure to handle the federation of services across multiple domains, after they establish a federation agreement. The project validated the developed infrastructure, showing that it was capable of deploying services in less than 115 seconds, which demonstrates the feasibility of integrating network federation in real production environments. In particular, [5GE18] and [Con20] show how a roaming service is deployed in less than 115 seconds, which is and assumable amount of time to consider the federation process in the deployment of NSs.

Given the speed of deploying a service in a federated domain, it is worth studying agents to take decisions on either the embedding of NSs in a pool of federated resources, or to delegate to another administrative domain the deployment of a NS. This triggered the study of the orchestration algorithms presented in both section 4.2 and section 4.3. The former tackling the VNE problem using resources of other administrative domains that belong to a network federation. And the latter studying the delegation of NS deployments under dynamic pricing of peering domains in a network federation.

4.2 Federated multi-domain graphs

Motivated by the fast deployment times of NSs in network federation scenarios, this section presents some NFV orchestration algorithms that solve the VNE problem in federated multi-domain graphs. In particular, the presented algorithms solve the embedding of NSs in federated networks represented with the graphs of section 3.6 of this thesis.

4.2.1 Mapping algorithms

All the algorithms are variations of a greedy approach that tries to minimize the overall delay of the mapped NS (4.1), and to do so it iterates through the Service Function Chain (SFC)'s VNFs trying to find the closest server that can host them attending computational constraints (4.4). The algorithms ensure that delay and bandwidth constraints are satisfied as well for the mapped NS — equations (4.2) and (4.3), respectively.

$$minimize \sum_{NS\in requests} delay_{map}(V_1,V_l) \tag{4.1}$$

$$s.t.\quad delay_{map}(V_i,V_k) \leq delay_{req}(V_i,V_k) \tag{4.2}$$

$$\sum_{NS}\sum_{(V_A,V_B)\in NS} u_{l,(V_A,V_B)}\cdot bw(V_A,V_B) \leq bw(l), \forall l \tag{4.3}$$

$$\sum_{NS}\sum_{V\in NS} u_{s,V}\cdot comput_V \leq comput_s, \quad \forall s \tag{4.4}$$

In the optimization problem solved with this section algorithms, (4.1) to (4.4), V_1 and V_l are the first and last VNFs in the NS. The variables $u_{l,(V_A,V_B)}$ and $u_{s,V}$ are booleans used to know if the link l is used to connect VNFs V_A and V_B, and if V is mapped in server s. To describe the restrictions parameters are used to express the bandwidth available in a link $bw(l)$ and the required to connect two VNFs $bw(V_A,V_B)$; on the other hand $comput_V$ and $comput_s$ hold the computational requirements (CPU, memory and disk) required by a VNF V, and the ones present in a server s. And last but not less important, $delay_{map}$ and $delay_{req}$ are used for the delay between 2 mapped VNFs, and for the required delay between 2 VNFs in the NS mapping request.

The following subsections present all the algorithms.

Greedy algorithm

The implemented greedy algorithm (Algorithm 3) iterates in order through every VNF in the SFC of the NS request. It starts looking for the servers that have enough CPU, memory and disk to host a VNF (line 3). Then it traverses the graph starting from the server where the previous VNF was mapped until it finds another server capable of hosting the next VNF (line 4). Once the VNF is mapped, the algorithm extracts the following VNFs directly connected to the one already mapped — this is what is called the neighbor VNFs — and repeats the process of finding the appropriate servers to host them, and traversing the graph to find a path between them and the server where the previous VNF was mapped.

To keep track of the consumed resources in the placement, a resource watchdog is responsible of allocating the bandwidth used along the paths and the server resources that every VNF requires (Algorithm 3, line 9). The resource watchdog is not asked to allocate resources in case there are no capable servers of hosting the VNF, or there is no path with available bandwidth to connect it with the previous one. In case the algorithm fails, the watchdog frees resources previously allocated, and it exits with an error.

Algorithm 3: Greedy search

```
Data: NsReq
Result: NsMapping
foreach VNF in NSreq do
    foreach nextVNF in neighbors(VNF) do
        capServs = capableServers(opView, nextVNF);
        path = findPath(VNFserv, nextVNF, capServs);
        if no path then
            watchDog.unwatch();
            return ERROR;
        else
            watchDog.watch(VNFserv, nextVNF, path);
            NsMapping.setPath(VNF, nextVNF, path);
            NsMapping.setLnkDelay(VNF, nextVNF,path.delay);
        end
    end
end
```

During the mapping process a *NsMapping* object is created to keep track of the paths found between the servers selected to host the VNFs of the SFC.

The *findPath* method

This subsection presents the different implementations of the *findPath* method. This method is used to search a server capable of hosting a VNF ensuring that delay and bandwidth requirements are satisfied. Implementations as the random walk, Dijkstra and Breadth-First Search (BFS) have already been studied; but not the Depth-First Search (DFS) proposed.

Random walk: this algorithm traverses the Service Provider (SP) available resources graph choosing randomly the links to visit (all the links have same possibilities of being visited). The implementation includes a hash table of already visited nodes to avoid choosing them twice in the random walk.

Dijkstra: Dijkstra's shortest path algorithm is adapted to the studied scenario. The modified version discards links and paths that don't satisfy the bandwidth and link restrictions imposed by the NS. It uses link delay as edge cost to reach the server nodes that can host the VNF to be mapped.

BFS : another alternative is to search a server that can host a VNF using the BFS algorithm. This implementation creates a tree having as root the previous VNF and expands the tree in a BFS manner across the SP graph.

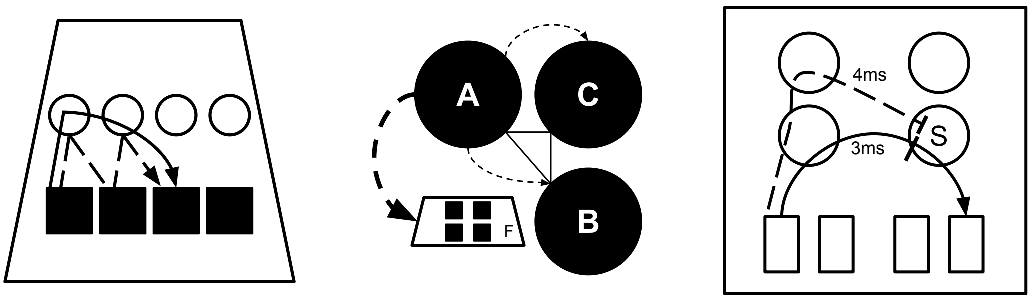
DFS: this implementation traverses the SP graph using a tree but following the DFS strategy. It has not been tried in the literature yet, but it reaches the servers of the SP graph faster than the previous implementations. This is because it goes directly to the leaf nodes of the generated tree.

BFS and DFS run-time complexity

If all nodes and paths in the SP graph are visited to map a VNF, BFS and DFS deal with their worst case scenario:

$$\mathcal{O}\left((k-1)^6 \cdot \left[\left(\frac{k}{2}\right)^2 - 1 + (p-1)\right]^2\right) \tag{4.5}$$

Equation (4.5) run-time complexity refers to the paths that DFS and BFS do in the worst case. k is the fat-tree degree, and p is the number of SP gateways in the meshed scenario (in Figure 3.9 there



(a) **Forbidden move** as a dashed line, shortest move across pods as continuous line. Circles are switches, boxes are pods.

(b) $P(A,F) > P(A,C) = P(A,B)$ thanks to **DFS priorities**. Circles are SP gateways and the trapezoid is a fat-tree.

(c) The dashed path reaches switch S with more delay than the continuous line **already visited** path.

Figure 4.1: Cutoffs representation

are $p = 5$ gateways in the federation graph). Every core, aggregate and edge switch of a fat tree has k links, and it is needed to walk through three switches (one of each type) to reach the gateways that connects the initial SP to the others. Each gateway has connection to the $\left(\frac{k}{2}\right)^2$ core switches underneath, but it also has links with another $p-1$ SPs within the federated scenario of this work.

The worst-case run-time (4.5) can be expressed in terms of the number of nodes N (switches, gateways and servers) of a fat-tree [Lei85].

$$\mathcal{O}\left(N^{\frac{1}{3}}N^3\right) \tag{4.6}$$

BFS and DFS cutoffs

Following lines describe the “cutoffs” introduced to improve the worst case run-time complexity expressed in Equation (4.6):

Forbidden moves: the algorithm can not go from one server to another of the fat-tree if it does not use the shortest path.

DFS priorities: when DFS is visiting a gateway node, it must visit fat-tree underneath before other gateway nodes.

Already visited: if the algorithm is in a node that was previously visited through a path that reached it traversing links with lower delays, the node is not visited.

The first two cutoffs are based in infrastructure knowledge. This is novel in the VNF mapping algorithms context and it is one of the contributions to the state of the art.

BFS and DFS run-time complexity with cutoffs

With the described cutoffs the search space is reduced considerably. With this improvement, BFS is close to be a kind of Dijkstra search, because it checks if nodes have already been visited, and if the new path to reach already visited ones improves the cost of a previous path. But it has the advantage of the forbidden moves, and that makes the algorithm going forward to server nodes using the shortest paths that fat-trees were designed for [Lei85]. This means that movements like going through an intermediate pod to reach a third one where the target server is located, are avoided. With DFS the speedup is even better, since in 9 comparisons it can reach a server from another SP. Then it goes straightforward to the servers, and in case it chooses properly the first server to visit, it goes faster than BFS with the implemented cutoffs. Having the cutoffs implies comparisons in every node the algorithm visits. In the switches this means having k comparisons to each neighbor, and these comparisons are $\mathcal{O}(1)$ since they rely on math operations or hash tables checks. Then, in the scenario where every already visited node has always been reached by a better path, BFS and

DFS will visit each node of the fat tree only once and make k comparisons in all of them. So the best scenario thanks to the cutoff improvements has a running time complexity of $\mathcal{O}(N)$.

Since is very reasonable that all links within a fat-tree have same delay characteristics, this will be the most common scenario, and the performance improvement is more than two orders of magnitude of the initial rudimentary solution that we exposed. The explanation is that first time a node is reached within a fat-tree, it will be in the shortest path that fat-trees have, because the forbidden moves enforces the algorithm to do so. Then it is not possible to reach in a second chance, a node in the same fat-tree having less delay in the new path (unless link characteristic are not the same in the fat-tree).

Tabu search algorithm

It is well known that a greedy strategy leads to suboptimal solutions in the mapping of VNFs, hence this section tries to improve the solutions using a meta heuristic called tabu search. This algorithm is based on an initial solution provided by the greedy algorithm of section 4.2.1, using all different implementations of the *findPath* method also mentioned in section 4.2.1.

When the tabu search receives an initial solution it repeats the following operations in order:

1. Select the next VNF inside the NS.
2. Mark the server where it is mapped as tabu for t turns.
3. Execute *findPath* to search a new server to host it that is not marked as tabu.
4. Repeat steps 1)-3) through all the VNFs inside the NS.
5. Check if the new mappings have decreased the NS end-to-end delay. If it is the case, store the solution.
6. Decrease counter t for all tabu servers and back to 1).

These steps force to find new solutions hoping that the end-to-end delay of the mapped NS will be reduced. The idea is trying to go out of local minimums.

4.2.2 Stress test

In this section carries out an experiment that consists in issuing NS mapping requests in the federation graph as the resources are reduced. Several algorithms are tried out to test their performance not only under circumstances when all resources are available, but when they deal with "stress", understanding it as lack of resources to do the mapping.

This section simulations were done using a Dell PowerEdge C6220 with 2 Intel Xeon E5-2670 @ 2.60GHz processors and 96GB of memory. To schedule the simulation jobs in parallel were managed by GNU parallel [Tan11].

Python was used to implement the algorithms of section 4.2.1, and the `networkx` software package [HSS08] to manipulate the graphs. All the code used to obtain the results presented in this section are published as a public repository[1].

Experiment setup

The graphs generated for this experiment are made up of 20 SPs, each one has a $k = 4$ fat-tree data center connected to the federation. The gateway nodes are connected as a full mesh. In terms of resources sharing, every SP has access to the computational resources of other 9 SPs, and the foreign SP can share up to 4 pods with it. Every server equally shares its resources with the SPs that can access itself.

The experiment issues 400 NS requests (each of them made up of 6 VNFs), having every VNF same computational resources requirements, an end-to-end SFC delay of 15 time units, and links requesting 1 bandwidth unit. Each request is performed by one of the 20 SPs, to decide which one requests the NS the experiment used a random variable that follows a uniform distribution $SP \sim \mathcal{U}\{1,20\}$.

For the initial step of the experiment all of the 400 NS requests must have enough resources to be allocated, that is a 100% acceptance ratio. To achieve that, the experiment starts with the

[1] `https://github.com/MartinPJorge/vnfs-mapping/commit/c8172327860443ac8abcc9f4a51d66abf5c26e19`

Table 4.1: Tabu search best iterations and blockings parameters settings to reach highest acceptance ratio.

algorithm	iterations	blockings	avg. time	acceptance (%)
DFS	6	4	3.24 sec.	61.5 %
BFS	6	2	5.04 sec.	63.5 %
Dijkstra	5	3	4.37 sec.	64%

Table 4.2: Acceptance ratios as resources are reduced

	0/10	2/10	4/10	6/10	8/10	10/10
DFS	100%	90.75%	61.25%	27.25%	3%	0%
BFS	100%	89%	59.75%	27%	3%	0%
Dijkstra	100%	89.75%	60.5%	27.75%	2.75%	0%
tabu	100%	89.75%	61.75%	28.25%	3.25%	0%

following conditions: 1 time unit of delay and 2400 bandwidth units in the links used to connect switches, gateways and servers in the generated infrastructure; and computational resources in each server (320 present in the multi-domain graph of this experiment) to host up to 1 VNF for every SP that can access it (remember every requested VNF has same computational requirements in the stress test).

Tabu search parameters

Before performing the stress test, it is checked which implementation of the *findPath* must be used in the initial greedy algorithm that tabu search uses as initial solution to perform modifications. The tabu search parameters were also tuned to get the best acceptance ratios. The incoming NS request requirements were modified so the link's delay and bandwidth, and VNF computational resources are not always the same. If the VNFs to be mapped require between 1/200 and 1/20 of a single server disk resources, between 1/200 and 1/50 of the CPU resources, and between 1/200 and 1/12 of the server memory resources; then the generated multi-domain is not able of hosting all the incoming NS requests.

With this in mind 400 NS requests were performed across the 20 SPs using Dijkstra, DFS, and BFS (the last two with the cutoffs) as the *findPath* algorithms to be used in the initial solution provided to the tabu algorithm. Then, it studied how many iterations the tabu meta heuristic must perform over the SFC trying to remap every VNF, and for how many iterations the performed mappings must be blocked (marked as tabu). Table 4.1 shows the best configurations and average mapping time per NS request among the 400 ones, and the acceptance ratio. According to the table, Dijkstra achieves the best acceptance ratio, but the DFS gets only a 2.5% lower acceptance ratio while obtaining the quickest average mapping times.

Resource reduction stress test

For this section's experiment the first step is to have a 100% acceptance ratio scenario as the described above, then computational resources are reduced in steps of tenths until every server in the infrastructure has no more CPU, memory and disk available.

After decreasing the computational resources of all the servers to a tenth, the simulator performs the 400 NS requests with the aforementioned requirements. It tries with 4 different algorithmic approaches. Three of them are just greedy search using Dijkstra, BFS and DFS with cutoffs for the *findPath* method. The tabu algorithm tested corresponds to that whose parameters retrieved the best acceptance ratio (the one based in Dijkstra). The random walk implementation of the *findPath* method is not included in the experiment because it does not have 100% acceptance ratio even when there are enough computational and bandwidth resources for the incoming 400 NS requests

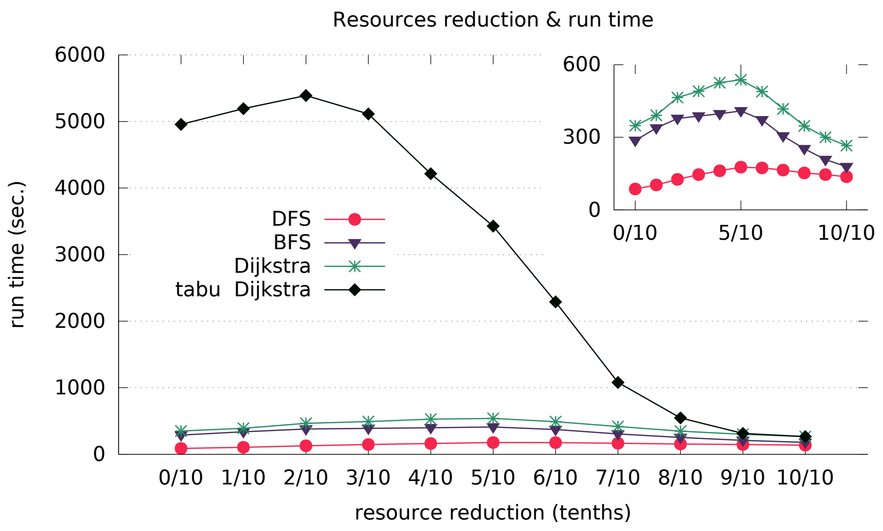


Figure 4.2: Running time of 400 NS mapping requests using different algorithms as server resources are reduced.

of the experiment setup.

All the tried algorithms yield acceptance ratios that differ from each others in $\leq 2.24\%$ in every step reducing the resources (see Table 4.2). That is, the experiment reduces a 10% the computational resources, and in each step all the four algorithms mentioned in the previous paragraph obtain very similar acceptance ratios in the 400 NS requests they are asked to map.

But although acceptance ratios are almost the same, the running times differ within the four used algorithms (see Figure 4.2). The highest execution time has been reached in the tabu search, while the greedy search using DFS to find server nodes is the one that has taken less time to perform the 400 NS request mappings. The reason why tabu decreases its execution times as resources are reduced, is because acceptance ratio is diminished, and if the initial greedy search fails, the tabu algorithm exits.

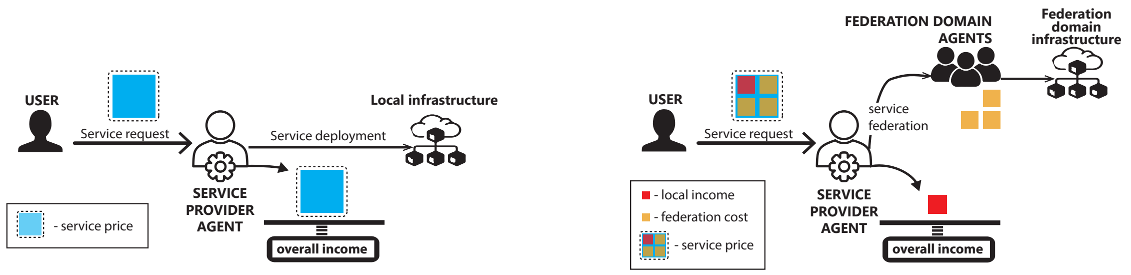
In the top-right corner of Figure 4.2, a zoom is applied to the graph to display the differences between the greedy algorithms using DFS, BFS and Dijkstra to reach shortest paths. The greedy DFS is the quickest performing the mapping of the 400 NS requests in every resource reduction scenario.

4.3 Delegation of NSs to federated domains

The stress test of the previous section shows how the resource consumption suppose an obvious decrease of the acceptance ratios of incoming NSs, due to the lack of available resources. Moreover, this will happen even if there is a pool of federated resources, or a fixed amount of shared resources among the peering domains in the network federation (this is the case of study of the previous section). However, precedent section assumed that there is a fixed price to use another domain's resources. But administrative domains may rather decide to change over time what do they charge to other domains in the federation using its resources. For example, upon an increase of demand to use its resources, it might decide to increase the price.

Given the possible change of prices of the federated domain, it is necessary to decide whether it is a good idea or not to federate, to prevent future peaks of pricing. This section focuses on developing an agent that takes such decision, aiming to maximize the revenue of a domain in a federated network. That is, the proposed agents of this section decide whether to delegate the deployment of a NS to a federated domain, or locally deploy it, accounting for the changes of deployment prices over time.

In summary, the contributions of this section are as follows:



(a) Service local deployment (b) Service federation

Figure 4.3: Business model

Table 4.3: Notation table

Symbol	Definition
σ	service
$p^{(t)}(\sigma)$	price rate [\$/hour] to deploy a service σ at time t
$f^{(t)}(\sigma)$	federation cost [\$/hour] at time t for service σ
$a(\sigma)$	arrival time of service σ
$d(\sigma)$	departure time of service σ
$x(\sigma)$	deployment action for service σ
$c(\sigma), m(\sigma), h(\sigma)$	CPU, memory, and disk asked by a service σ
C_l, M_l, H_l	local domain CPU, memory, and disk
C_f, M_f, H_f	federation pool of CPU, memory, and disk

- It analyzes the price dynamics of a public cloud provider, and take it as reference for the service pricing of a federated domain. By considering the prominent pricing fluctuations it is possible to explore higher revenues in the federation process.
- It derives a dynamic arrival process, which is impacted by the fluctuations in the service prices.
- It characterizes a federated multi-domain scenario as an online decision-making problem that aims at maximizing revenue;
- It design and implement two model-free mechanisms based on reinforcement learning: Q-table solutions, and a proposed Deep Q-Network (DQN) solution. Both require training phase to derive a policy for revenue enlargement.
- It performs a thorough data-driven performance evaluation of the proposed solution, and compared against state-of-the-art solutions. Results show that DQN achieved 90% optimal performance with no *apriori* knowledge of the future arrivals. Both Q-table and DQN solutions outperform a greedy strategy, and are suitable for generic environments.

The rest of the section is organized as follows. The basic business scenario which is formulated in section 4.3.1. Section 4.3.2, defines the dynamic pricing, and arrival process of incoming services. Latter, the related optimization problem is formulated in section 4.3.3, and the Markov Decision Problem in section 4.3.4; which also describes the Q-table algorithms, and the proposed DQN algorithm used to solve the problem. Following, section 4.3.5, presents the experimental evaluation of the algorithms.

4.3.1 Business model

First it is to analyze the business model of interest for this section. Inspired by the market of cloud services, this section considers a system where a SP offers cloud resources or services at a *service price* rate $p^{(t)}$ that may vary over time depending on the operator's pricing model. Correspondingly, each user wishing to deploy service σ at the offered price, makes the request (arrives to the system) at time $a(\sigma)$ and leaves at time $d(\sigma)$. A federation scenario is considered. Upon each request to deploy service σ, the SP can take an action $x(\sigma) := \{0,1,2\}$ indicating, respectively, whether the service is deployed locally, deployed in the federated domain, or rejected. See Table 4.3 for a summary of the notation.

The pricing model does have an impact on the arrival process of the service requests: intuitively, lower prices incentivize a higher user arrival rate. The pricing model and the related arrival process model is later discussed in section 4.3.2. Importantly, however, once there is an agreement between customer and provider, the customer pays the agreed rate $p^{a(\sigma)}$ for every time slot t during which the service is active, i.e., for every $t : a(\sigma) \leq t \leq d(\sigma)$. In contrast, however, should the service σ be deployed in the federated domain, the service provider has to pay the federated domain agents a time-varying fee $f^{(t)}(\sigma), \forall t : a(\sigma) \leq t \leq d(\sigma)$. If the service will not be deployed either locally or in the federated domain, then the service will be rejected and in this case the customer does not need to pay the service fee and thus there will be no income for the service provider. This business model allows to exploit opportunistically (uncertain) price fluctuations, which can provide substantial cost savings, yet provide certainty to the end users, which is essential for vertical sectors.

As a result, at every t, there are two concurrent cash flows:

(Figure 4.3(a)) The service provider uses local resources to grant the request, and therefore the agent's income is equal to $p^{a(\sigma)}(\sigma), \forall t : a(\sigma) \leq t \leq d(\sigma)$;

(Figure 4.3(b)) The service provider uses federated resources, and therefore the provider gets $p^{a(\sigma)}(\sigma) - f^{(t)}(\sigma), \forall t : a(\sigma) \leq t \leq d(\sigma)$, where $f^{(t)}(\sigma)$ is the *federation cost*, which fluctuates over time.

In this way, it is possible to derive the agent's income, which represents the instantaneous revenue of the SP, at time t as follows:

$$r^{(t)}(X_t) := \sum_{\substack{\sigma:\ x(\sigma)=0 \\ a(\sigma)\leq t \leq d(\sigma)}} p^{a(\sigma)}(\sigma) + \sum_{\substack{\sigma:\ x(\sigma)=1 \\ a(\sigma)\leq t \leq d(\sigma)}} \left[p^{a(\sigma)}(\sigma) - f^{(t)}(\sigma) \right] \tag{4.7}$$

where $X_t := \{x(\sigma)\}_{\sigma:a(\sigma)\leq t}$.

4.3.2 System Dynamics

There are three sources of uncertainty in the system: (*i*) the pricing model used in federated domains $f^{(t)}$, which may be highly volatile; (*ii*) the cost associated with local deployments (which ultimately drives the service pricing $p^{(t)}$); and (*iii*) the process that characterizes the arrival of customers into the system, which is certainly associated with the set fees ($p^{(t)}$) in a way that is unknown *a priori*. In the following it is discussed (*i*) and (*ii*) first; and later (*iii*).

Pricing

Dynamic pricing mechanisms have become very popular in cloud computing services because they have the ability to maximize the cloud provider's revenue while minimizing the price of the offered service. The literature presents abundant research on the topic, being [Tok+20] a remarkable example. However, although the pricing problem has been well studied and, intuitively, prices shall follow the offer-demand trade-off, it is very hard to model the underlying pricing mechanisms applied in practice today. For instance, works such as [Bau+19, Geo+19a, Geo+19b] present spatio-temporal analyses of the pricing method applied by a large cloud provider but have failed to model it. Others, such as [Lan+19, SLK18], have applied predictive methods to this phenomenon, but the proposed solutions pitfall on either predicting price peaks, or the tendency of price over time.

Let us analyze, in the following, the price dynamics of service instances from a major cloud provider[2]. To this end, it was collected price data of every service instance offered by the cloud provider between 29/02/2012 and 31/07/2020 for the "Paris, Europe" region. Figure 4.4 depicts

[2] `https://docs.aws.amazon.com/cli/latest/reference/ec2/describe-spot-price-history.html` [Accessed 30/11/2020]

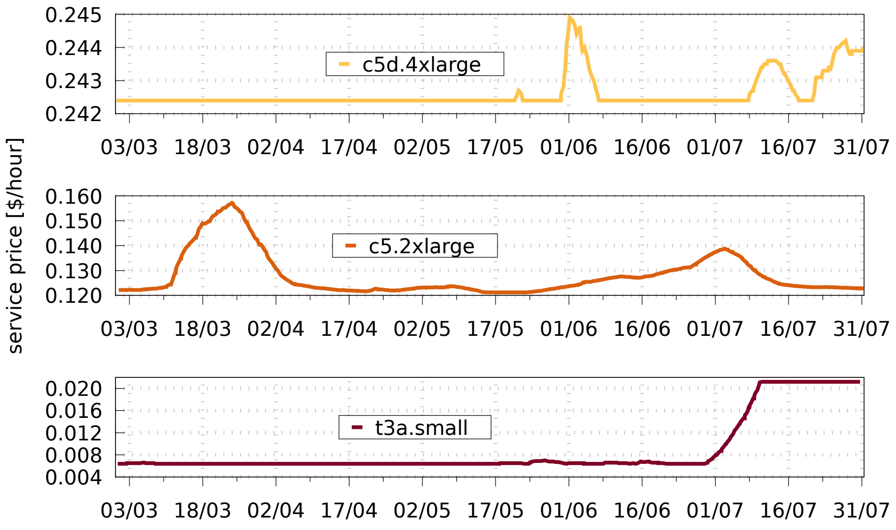


Figure 4.4: Service prices of a cloud provider during 2020.

the price evolution of three service instances over a time window of five months trivially chosen. Specifically, it was selected *t3a.small*, *c5.2xlarge* and *c5d.4xlarge* from AWS EC2 Spot instances because they are the closest, in term of resource requirements, to the services used in a multi-domain case study—a similar scenario as the studied in this section–from a main network operator in Spain [SC20a].

The figure illustrates the fact that, though prices are reasonably stable over medium-long time periods, there occur a large number of short-time, yet sporadic, fluctuations that may play havoc with standard price prediction mechanisms. These fluctuations may be due to sudden changes on the arrival rate of the users but also on unknown external phenomena, such as energy costs or system failures. This section argues that it is paramount to design a decision-making model that considers such random dynamic events to explore service federation (with unknown pricing model $f^{(t)}$) opportunistically such that it is maximized the agent's revenue.

Motivated by the above, *service price* rates are modelled as

$$p^{(t)}(\sigma) = (1+P)l^{(t)}(\sigma) \tag{4.8}$$

where $l^{(t)}(\sigma)$ is the local *deployment cost* (which depends on uncertain phenomena, as explained above), and P is the marginal profit over the local deployment cost as *a choice of the operator*.

User arrivals

Let $\mathscr{A}$ denote the stochastic process modeling the arrival of users at the system. Intuitively, this shall be a non-homogeneous price-dependant process, i.e., a lower price $p^{(t)}$ incentivize higher arrival rate $\lambda^{(t)}$. In the context of cloud services, this phenomenon has been studied in, for instance, [XL13], where $\lambda^{(t)} = f(p^{(t)})$ and f satisfying the following assumption.
Assumption 4.1: The arrival rate function is a non-negative $f(\underline{p}^{(t)}) \geq 0$, decreasing $f'(\underline{p}^{(t)}) < 0$, and concave function $f''(\underline{p}^{(t)}) < 0$; with no slope when the price is at its minimum $f'(0) = 0$, and taking zero values when the price is at its maximum $f(1) = 0$. Additionally the arrival rate function should drastically drop to zero as the price reaches its maximum $f'(\underline{p}^{(t)}) \xrightarrow[\underline{p}^{(t)}\to 1]{} -\infty$

with $\underline{p}^{(t)} \in [0,1]$ being the normalized price.
Then, given Assumption 4.1,

$$f(\underline{p}^{(t)}) := k\left(1-(\underline{p}^{(t)})^a\right)^b \tag{4.9}$$

where k, a and b are parameters that depend on the system and hence have to be estimated, e.g.,

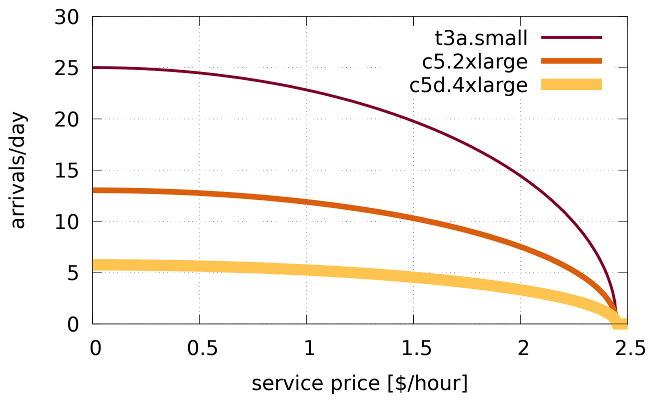


Figure 4.5: Impact of the *service price* $p^{(t)}(\sigma)$ on the arrival rate of the different cloud provider service instances. This illustration uses $P = 0.2, a = 2, b = \frac{1}{2}, K = 2$, and k fit to the data provider of a main network operator in Spanish [SC20a].

by a learning model. Note that the arrival rate function $f(\underline{p}^{(t)})$ is a positive, decreasing, concave function that is 0 when the dynamic price reaches its maximum.

To obtain the proper arrival rate based on $f(p^{(t)}(\sigma))$, the function must be properly defined, and the *service price* $p^{(t)}$ normalized accordingly. Thus, in this section f is redefined as follows:

$$f(p^{(t)}(\sigma)) := \begin{cases} k\left(1 - \left(\frac{p^{(t)}(\sigma)}{K \cdot M}\right)^a\right)^b, & p^{(t)}(\sigma) \leq K \cdot M \\ 0, & p^{(t)}(\sigma) > K \cdot M \end{cases} \tag{4.10}$$

where $M = \max_{\sigma,t}\{l^{(t)}(\sigma)\}$ is the maximum local deployment cost over time across all services σ (e.g., *t3a.small*), and K is a normalization constant to control the decay of the arrival rate. Note that equation (4.10) satisfies Assumption 4.1.

Using this model and the aforementioned service pricing model, Figure 4.5 shows arrival rate (described in equation (4.10)) associated to the services that are used in this section. As stated in equation (4.8), the *service price* (hence the arrival rate) depends on the local *deployment cost* $l^{(t)}(\sigma)$ and margin P. Therefore, the mean arrival rate will decrease with high marginal benefit P or high local deployment costs (see Figure 4.6).

4.3.3 Optimization problem

Given the above, hereafter this subsection formulates an optimization problem with the goal of maximizing the revenue of a service provider in a multi-domain federation scenario – see section 3.6 and section 4.2. As introduced earlier, in the studied scenario, a local SP may use a limited set of resources available locally, or resort to a federated resource provider (at a fee). Specifically, let $(C_l, M_l, H_l) \in \mathbb{N}$ denote, respectively, the total number of CPUs, memory, and disk resources available locally. Similarly, $(C_f, M_f, H_f) \in \mathbb{N}$ denote the respective resources at federated domain.

A service σ that arrives at time $a(\sigma)$ is characterized by a set of resource requirements $\big(c(\sigma), m(\sigma), h(\sigma)\big) \in \mathbb{N}$. Upon each request, the local agent makes a decision $x(\sigma)$, which shall guarantee that the resource capacity is not exhausted at any time. To this end, these are the constraints:

$$C_l \geq \sum_{\substack{\sigma:\, x(\sigma)=0 \\ a(\sigma)\leq t \\ d(\sigma)>t}} c(\sigma), \quad M_l \geq \sum_{\substack{\sigma:\, x(\sigma)=0 \\ a(\sigma)\leq t \\ d(\sigma)>t}} m(\sigma), \quad H_l \geq \sum_{\substack{\sigma:\, x(\sigma)=0 \\ a(\sigma)\leq t \\ d(\sigma)>t}} h(\sigma), \quad \forall t \tag{4.11}$$

$$C_f \geq \sum_{\substack{\sigma:\, x(\sigma)=1 \\ a(\sigma)\leq t \\ d(\sigma)>t}} c(\sigma), \quad M_f \geq \sum_{\substack{\sigma:\, x(\sigma)=1 \\ a(\sigma)\leq t \\ d(\sigma)>t}} m(\sigma), \quad H_f \geq \sum_{\substack{\sigma:\, x(\sigma)=1 \\ a(\sigma)\leq t \\ d(\sigma)>t}} h(\sigma), \quad \forall t \tag{4.12}$$

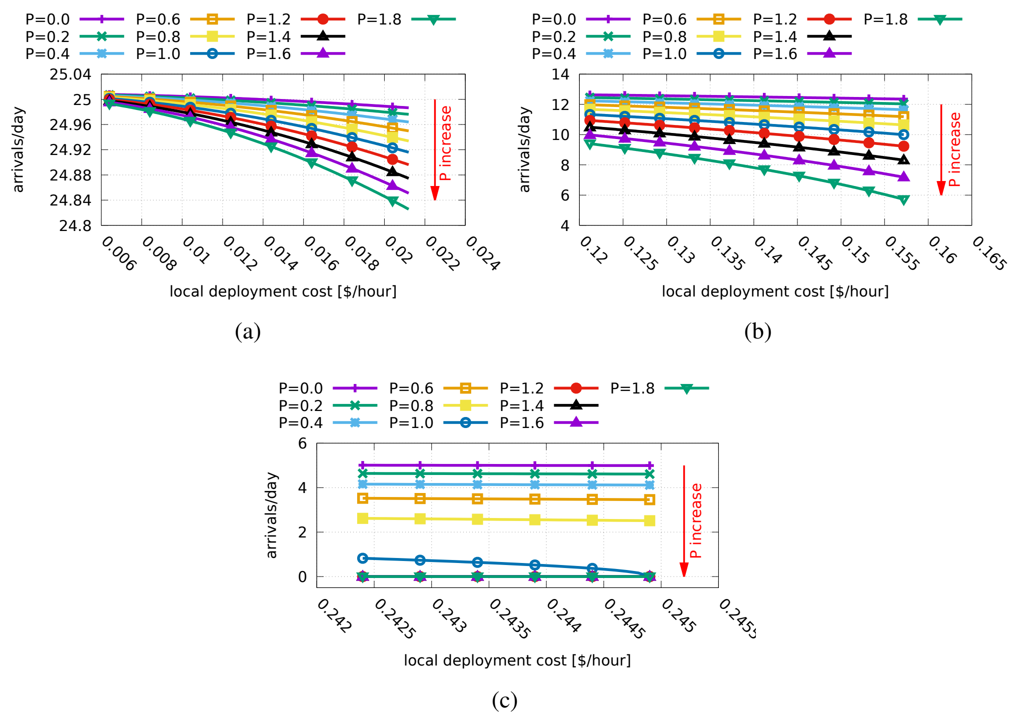


Figure 4.6: Impact of local *deployment cost* $l^{(t)}$ and marginal benefit P in (a) t3a.small; (b) c5.2xlarge; and (c) c5d.4xlarge arrival rates. Graphs derived using Figure 4.4 prices as $l^{(t)}$ local *deployment cost*.

Constraints (4.11) refer to the conservation of local domain resources, and constraints (4.12) to the conservation of the federated pool of resources.

Our objective is to choose the most appropriate action for every service request such that the obtained income (according to the business model in section 4.3.1) over the long-run is maximized. Hence, the optimization problem becomes:

Problem 4.1: Federation deployment problem.

$$\max_{X_T} \quad \lim_{T\to\infty} \frac{1}{T} \sum_{t}^{T} r^{(t)}\left(X_t\right) \tag{4.13}$$

$$\text{s.t.} \quad (4.11), (4.12)$$

$$x(\sigma) \in \{0,1,2\}, \quad \forall x(\sigma) \in X_T$$

with $r^{(t)}\left(X_t\right)$ being the instantaneous reward defined in eq. (4.7).

The complexity of Problem 4.1 is analyzed in Lemma 4.2.

Lemma 4.2: Problem 4.1 is NP-complete.

Proof. Problem 4.1 can be cast into the knapsack problem [Lew83], which is well-known to be NP-complete. To do the mapping, take a problem instance with $T = 1$, no federation resources $C_f = M_f = H_f = 0$, and assume that all services (i) do not leave the system, i.e., $d(\sigma) > T$, $\forall\sigma$; (ii) only ask for CPU resources, i.e., $c(\sigma) > 0 \wedge m(\sigma) = h(\sigma) = 0$, $\forall\sigma$; (iii) have a service price equal to the requested CPU $p^{a(\sigma)} = c(\sigma)$, $\forall\sigma$; and (iv) arrive at $T = 1$, that is, $a(\sigma) = 1$, $\forall\sigma$. The resulting instance of the problem becomes

$$\max_{x(\sigma)\in X_T} \quad \sum_{\sigma:x(\sigma)=0} c(\sigma) \tag{4.14}$$

$$s.t. \quad C_l \geq \sum_{\sigma:x(\sigma)=0} c(\sigma) \tag{4.15}$$

which is the knapsack problem with $c(\sigma)$ being the object weights, and C_l the sack capacity. ■

Apart from being NP-complete, the studied problem is not implementable as it has an overly large number of decision variables (more so as the time horizon T is increased towards infinity), and requires knowledge of future arrivals of services requests, and service and federation prices. For this reason, this section resorts to online decision.

4.3.4 Markov Decision Problem (MDP)

The optimization problem stated in section 4.3.3 is equivalent to a Markov Decision Process (MDP) [BEL57], which is solved using three different algorithms introduced in this section. Any MDP is well defined given the tuple $\left(\mathscr{S},\mathscr{A},P_{X_T},r^{(t)}\right)$, which describes, respectively, state set, action set, transition probabilities given X_T, and a reward function.

The state space $\mathscr{S}$ contains information related to (*i*) the local and *federation cost* of all services $\{\sigma\}$; (*ii*) the resources required for each arriving service σ; and (*iii*) the available resources at both the local domain, and federation domains. Specifically, the state has information concerning $\left(\{\delta_{k_l}^{(t)},\delta_{k_f}^{(t)}\}_{k\in\{C,M,H\}},\{\sigma:a(\sigma)=t\}\right)$, where $\{\sigma:a(\sigma)=t\}$ contains the service requests at time t, and $\delta_{C_l}^{(t)}$ and $\delta_{C_f}^{(t)}$ are the normalized residual resource capacities, e.g.,

$$\delta_{C_l}^{(t)}=\frac{1}{C_l}\sum_{\substack{\sigma:x(\sigma)=0\\a(\sigma)\leq t<d(\sigma)}}c(\sigma),\qquad \delta_{C_f}^{(t)}=\frac{1}{C_f}\sum_{\substack{\sigma:x(\sigma)=1\\a(\sigma)\leq t<d(\sigma)}}c(\sigma) \tag{4.16}$$

for the case of CPU resources. Note that the state space is redefined for each algorithm presented next, for notation convenience.

Conversely, the action space is $\mathscr{A}=\{0,1,2\}$ corresponding to the "accept at local domain", "accept at federated domain", and "reject" actions upon incoming service requests. That is, the action variable $x(\sigma)$ of the optimization problem in section 4.3.3, belongs to the action space $\mathscr{A}$ of the MDP.

The transition probabilities are given by the function $P_{x(\sigma_t)}(s^{(t+1)}|s^{(t)})\in[0,1]$, that is, how likely it is to end up in state $s^{(t+1)}$ after taking action $x(\sigma_t)$ in state $s^{(t)}$; with σ_t being the service arriving at time t. The transition probabilities function $P_{x(\sigma_t)}$ is known given (*i*) the arrivals of new services; (*ii*) if previous services were deployed locally, federated, or rejected; and (*iii*) the lifetime $d(\sigma)-a(\sigma)$ of each running service σ.

In the MDP, the rewards $r^{(t)}$ correspond to the instantaneous reward already defined in (4.7) for the optimization problem in section 4.3.3. The goal of the MDP is to derive a policy $\pi:\mathscr{S}\mapsto\mathscr{A}$ that in each state $s^{(t)}$ takes an action $x(\sigma_t)$ that maximizes the long-term reward. However, the state transitions $s^{(t+1)}|s^{(t)}$ are not deterministic, indeed they are governed by the aforementioned $P_{x(\sigma_t)}$ probabilities. Moreover, the instantaneous reward (4.7) depends also on the *service price* $p^{(t)}(\sigma)$, and *federation cost* $f^{(t)}(\sigma)$, with the latter being a random variable set by the federated agents.

The following three algorithms propose different policies π with the goal of maximizing the expected long-term reward

$$\mathbb{E}_{x(\sigma_t)\sim\pi}\left[\sum_t\gamma^t r^{(t)}(\pi)\right]$$

with $\gamma\in[0,1]$ being the discount factor for future rewards. This long-term reward refers to the income of the operator in the business scenario (section 4.3.1).

Greedy algorithm

In this subsection introduces a *greedy* approach, which renders a simple strategy suitable for comparison. This approach consists of a simple policy where each service request is locally deployed as long as there is availability of *local* resources, first, or otherwise *federated* resources. If there are no resources in the whole system, the service request is rejected.

Algorithm 4: Greedy algorithm

```
1  for $t \in [0,T]$ do
2      if $\frac{c(\sigma_t)}{C_l\delta_{C_l}^{(t)}} \leq 1 \wedge \frac{m(\sigma_t)}{M_l\delta_{M_l}^{(t)}} \leq 1 \wedge \frac{h(\sigma_t)}{H_l\delta_{H_l}^{(t)}} \leq 1$ then
3          $x(\sigma_t) = 0$;
4      else
5          if $\frac{c(\sigma_t)}{C_f\delta_{C_f}^{(t)}} \leq 1 \wedge \frac{m(\sigma_t)}{M_f\delta_{M_f}^{(t)}} \leq 1 \wedge \frac{h(\sigma_t)}{H_f\delta_{H_f}^{(t)}} \leq 1$ then
6              $x(\sigma_t) = 1$;
7          else
8              $x(\sigma_t) = 2$;
9          end
10     end
11     $r^{(t)}, s^{(t+1)}$ = environment.takeAction($x(\sigma_t)$);
12 end
```

In the context of our MDP (described in section 4.3.4), the state vector used by the greedy approach is:

$$s^{(t)} = \left(\delta_{C_l}^{(t)}, \delta_{M_l}^{(t)}, \delta_{H_l}^{(t)}, \delta_{C_f}^{(t)}, \delta_{M_f}^{(t)}, \delta_{H_f}^{(t)}, \frac{c(\sigma_t)}{C_l\delta_{C_l}^{(t)}}, \frac{m(\sigma_t)}{M_l\delta_{M_l}^{(t)}}, \frac{h(\sigma_t)}{H_l\delta_{H_l}^{(t)}}\right) \tag{4.17}$$

where the first six elements are the normalized amount of residual resources (available) (*cpu*, *memory*, *disk*) at time *t*; and the last three represent the normalized amount of requested resources. In this way, this greedy policy, presented in pseudocode fashion in Algorithm 4, is described as follows:

$$\begin{aligned}
\pi(0|s^{(t)}) = 1, \quad & \frac{c(\sigma_t)}{C_l\delta_{C_l}^{(t)}} \leq 1 \wedge \frac{m(\sigma_t)}{M_l\delta_{M_l}^{(t)}} \leq 1 \wedge \frac{h(\sigma_t)}{H_l\delta_{H_l}^{(t)}} \leq 1 \\
\pi(1|s^{(t)}) = 1, \quad & \frac{c(\sigma_t)}{C_l\delta_{C_l}^{(t)}} > 1 \wedge \frac{m(\sigma_t)}{M_l\delta_{M_l}^{(t)}} > 1 \wedge \frac{h(\sigma_t)}{H_l\delta_{H_l}^{(t)}} > 1 \\
& \wedge \frac{c(\sigma_t)}{C_f\delta_{C_f}^{(t)}} \leq 1 \wedge \frac{m(\sigma_t)}{M_f\delta_{M_f}^{(t)}} \leq 1 \wedge \frac{h(\sigma_t)}{H_f\delta_{H_f}^{(t)}} \leq 1 \\
\pi(2|s^{(t)}) = 1, \quad & \text{otherwise}
\end{aligned} \tag{4.18}$$

Q-table algorithm

This subsection adapts[3] the solution presented in [Ant+20], which is a Q-table-based reinforcement-learning solution to the MDP. Q-table is a simple reinforcement learning realization based on a lookup table to search over the $\mathscr{S} \times \mathscr{A}$ space. The rows of the table present the state space $\mathscr{S}$ of the MDP, and the columns present the set of actions that can be taken for each state, i.e., the set $\mathscr{A} = \{0,1,2\}$. Each state $s^{(t)}$ is represented by normalized values that represent the average residual resource availability for local and federated resources, the most demanding resource of the arriving service, the instantaneous reward if the service is deployed locally, $r_{x_0}^{(t)}$, and its *federation*

[3] the algorithm adaption was programmed by Kiril Antevski

cost:

$$s^{(t)} = \Bigg(\frac{\delta_{C_l}^{(t)} + \delta_{M_l}^{(t)} + \delta_{H_l}^{(t)}}{3}, \frac{\delta_{C_f}^{(t)} + \delta_{M_f}^{(t)} + \delta_{H_f}^{(t)}}{3},$$

$$\max \left\{ \frac{c(\sigma_t)}{C_l \delta_{C_l}^{(t)}}, \frac{m(\sigma_t)}{M_l \delta_{M_l}^{(t)}}, \frac{h(\sigma_t)}{H_l \delta_{H_l}^{(t)}} \right\}, r_{x_0}^{(t)}, f^{(t)}(\sigma_t) \Bigg) \tag{4.19}$$

As mentioned earlier, the state transitions are not deterministic. As a result, the Q-table requires a training period where its cells are populated to converge towards the expected future rewards. That is, $Q(s^{(t)}, x(\sigma_t))$ will be equal to the expected cumulative reward if action $x(\sigma_t)$ is taken at state $s^{(t)}$. To converge towards the expected future rewards, the Q-table values are filled using the Q-learning recurrence approach during the training stage, which approximates the Bellman equation [BEL57]:

$$Q(s^{(t)}, x(\sigma_t)) = (1-\alpha) Q(s^{(t)}, x(\sigma_t)) +$$

$$\alpha \left(r^{(t)} + \gamma \max_x Q(s^{(t+1)}, x) \right) \tag{4.20}$$

where $r^{(t)}$ is the instantaneous reward foreseen after taking action $x(\sigma_t)$ at state $s^{(t)}$. Note the instantaneous reward is aggregated with the discounted future reward $\gamma \max_x Q(s^{(t+1)}, x)$. The parameters α—the learning rate, and γ—the discounted factor, are fixed parameters.

In summary, in each time t, a new service request arrives at the operator agent, who takes action $x(\sigma_t)$, and then populates the Q-table for $Q(s^{(t)}, x(\sigma_t))$. However, it must be noted that the instantaneous reward $r^{(t)}$ is not always positive. For example, if the action $x(\sigma_t) = 0$ for local domain deployment at $s^{(t)}$ exceeds the local domain capacity, the service deployment is rejected by the local domain and the instantaneous reward is negative. In other words, the operator agent receives a *penalty* for taking the wrong action at time t.

At the start of the training ($t = 0$), all the Q-table values are initialized to zero. The training procedure consists of repetitive runs of a generated set of arrivals for a timing interval $[0, T]$. Each training repetition is a single *episode* and the training set of arrivals is consistent for E_p episodes. In order to train this model so as to *learn* a policy that maximizes long-term reward, two different policies are used in the training stage, namely (*i*) Q-table legacy, and (*ii*) Q-table exploration.

On the one hand, the Q-table legacy strategy chooses the action as:

$$x(\sigma_t) = \max_x \left\{ Q(s^{(t)}, x) + \frac{u}{1+e} \right\} \tag{4.21}$$

where $e \in \{1, \ldots, E_p\}$ is the current training episode, and $u \sim \mathscr{U}\{0,2\}$ is drawn from a discrete uniform distribution. The actions taken in the first episodes (e.g., $e = 1$) have a larger random component compared to the last training episodes (e.g., $e \to E_p$). This exploration strategy is used in [Ant+20].

On the other hand, the Q-table exploration strategy uses a standard ε-greedy policy [SB18]:

$$x(\sigma_t) = \begin{cases} \max_x \left\{ Q(s^{(t)}, x) \right\}, & \bar{u} \geq \varepsilon(e) \\ u \sim \mathscr{U}\{0,2\}, & \bar{u} < \varepsilon(e) \end{cases} \tag{4.22}$$

where e is the training episode, and $\bar{u} \sim \mathscr{U}(0,1)$ is a random number drawn from a uniform distribution. Moreover, $\varepsilon(e) = \frac{e}{E_p}(\varepsilon_{max} - \varepsilon_{min}) + \varepsilon_{min}$ is a linear interpolation between ε_{min} and ε_{max}. In this way $\varepsilon(e)$ (which defines random exploration) drops as episodes pass. Algorithm 5 describes both the training procedure of both the legacy and exploration strategies.

Once the training stage has finished, the followed policy is applied:

$$\pi(x(\sigma_t) | s^{(t)}) = \mathbf{1}_{x(\sigma_t)} \left[\arg\max_x Q(s^{(t)}, x) \right] \tag{4.23}$$

that is, the action with highest Q-value is selected.

Algorithm 5: Q-learning training algorithm

```
1  $Q_T \leftarrow \mathbf{0}$;
2  for $e \in E_p$ do
3      for $t \in [0,T]$ do
4          if legacy strategy then
5              $u \sim \mathcal{U}\{0,2\}$ ;
6              $x(\sigma_t) = \max_x \left\{ Q(s^{(t)},x) + \frac{u}{1+e} \right\}$ ;
7          end
8          if exploration strategy then
9              $\varepsilon(e) = \frac{e}{E_p}(\varepsilon_{max} - \varepsilon_{min}) + \varepsilon_{min}$;
10             $\bar{u} \sim \mathcal{U}\{0,1\}$;
11             $x(\sigma_t) = \begin{cases} \max_x \left\{ Q(s^{(t)},x) \right\}, & \bar{u} \geq \varepsilon(e) \\ \mathcal{U}\{0,2\}, & \bar{u} < \varepsilon(e) \end{cases}$;
12         end
13         $r^{(t)}, s^{(t+1)}$ = environment.takeAction($x(\sigma_t)$);
14         $Q_T[s^{(t)},x(\sigma_t)] \leftarrow (1-\alpha)Q_T[s^{(t)},x(\sigma_t)] + \alpha\left(r^{(t)} + \gamma \max_a Q_T[s^{(t+1)},a]\right)$;
15     end
16 end
```

Deep Q Network (DQN)

This subsection presents the Neural Network (NN) approach proposed for this section. Specifically, the proposed algorithm is based on the DQN solution of [Mni+13]. As stated in section 4.3.4, the goal is to maximize the expected long-term reward $\mathbb{E}_{x(\sigma_t)\sim\pi}\left[\sum_t \gamma^t r^{(t)}(\pi)\right]$. If $Q(s^{(t)},x(\sigma_t))$ denotes the action-value function, i.e., a function estimating how good action $x(\sigma_t) \in \mathscr{A}$ is, given state $s^{(t)} \in \mathscr{S}$. The authors of [Mni+13] presented a NN with weights $\vec{w}$ to approximate the action-value function $Q(s^{(t)},x(\sigma_t),\vec{w})$. Such a NN is referred as the Q-network (see the NN in Figure 4.7).

The output of the Q-network is a layer of 3 neurons that specify the action-value estimation for each action, namely the local deployment $x(\sigma_t) = 0$, federation $x(\sigma_t) = 1$, and rejection $x(\sigma_t) = 2$. The Q-network's input is the state representation $s^{(t)}$, which in this section correspond to a vector containing the normalized residual capacity of local and federated resources, the normalized amount of resources demanded by the arriving instance and the instantaneous reward if it is locally deployed, as well as its *federation cost*:

$$s^{(t)} = \Bigg(\delta_{C_l}^{(t)}, \delta_{M_l}^{(t)}, \delta_{H_l}^{(t)}, \delta_{C_f}^{(t)}, \delta_{M_f}^{(t)}, \delta_{H_f}^{(t)}, \\ \frac{c(\sigma_t)}{C_l \delta_{C_l}^{(t)}}, \frac{m(\sigma_t)}{M_l \delta_{M_l}^{(t)}}, \frac{h(\sigma_t)}{H_l \delta_{H_l}^{(t)}}, r_{x_0}^{(t)}, f^{(t)}(\sigma_t) \Bigg) \tag{4.24}$$

The Q-network training must update its weights $\vec{w}$ to achieve the best possible estimation of the action-value function. As in the Q-table solution presented previously, the training procedure is based on the Bellman equation [BEL57] to converge to the optimal action-value function. That is $Q(s,x,\vec{w}_i) \to Q(s,x,\vec{w}^*)$ as $i \to \infty$ with $\vec{w}_i$ denoting the Q-network weights at iteration i in the training process, and $\vec{w}^*$ denoting the weights with which the Q-network estimates the optimal action-value function. Rather than directly using the Bellman equation recurrence, the Q-network updates its action-value estimate by changing the weights $\vec{w}_i$. In more detail, gradient descend on the loss function is used[4]:

$$L_i(\vec{w}) = \left[\left(r^{(i)} + \gamma \max_x Q(s^{(i+1)},x,\vec{w}_i) \right) - Q(s^{(i)},x(\sigma_i),\vec{w}) \right]^2 \tag{4.25}$$

[4] $s^{(i)},x(\sigma_i)$ denote the state and actions taken at iteration i of the training phase.

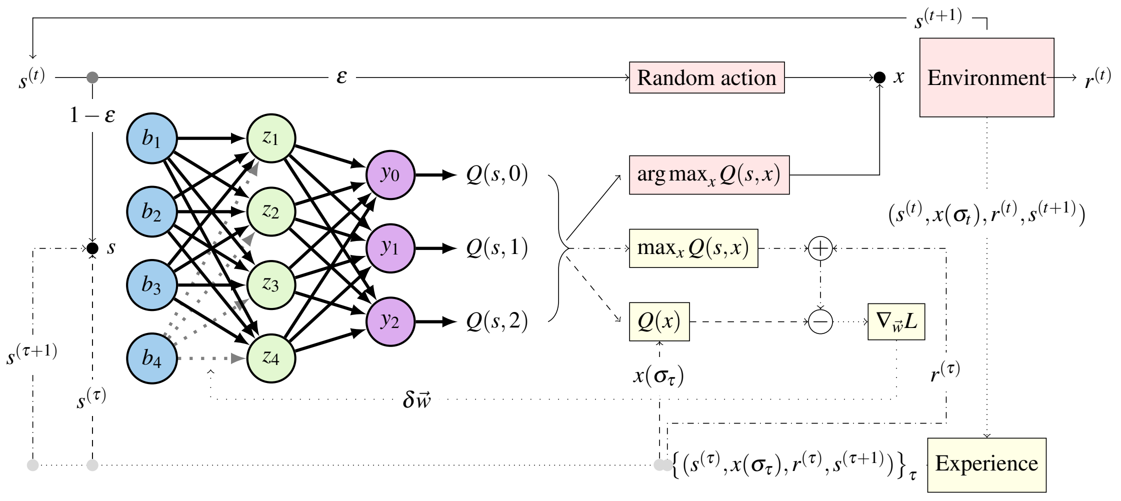


Figure 4.7: Deep Q-network with one hidden layer, $k=1$, and experience replay training. Non-continuous lines illustrate the training interactions, while continuous lines are used to show the execution interactions. In the illustration the state $s^{(t)}$ is a vector of only 4 components.

This corresponds to the squared difference between the discounted reward with the current weights $\vec{w}_i$, and the action-value estimate given weights $\vec{w}$.

An initial approach could be to compute the loss function $L_i(\vec{w})$ for every iteration i in the training stage, e.g., perform an update as $\vec{w}_{i+1} = \delta\vec{w} + \vec{w}_i$ with $\delta\vec{w} \propto \nabla L_i(\vec{w})$. However, the states of consequent iterations are very likely to be correlated, as the arrival rate will not drastically change in the next time instant (see Figure 4.4). To mitigate this phenomenon in the training stage, the Q-network computes the loss gradient over past state-action-reward-state transitions. These transitions are stored into an Experience memory $\mathscr{D}$ with capacity of up to M past transitions. As Figure 4.7 illustrates, the Environment stores in the Experience the current transition, and the Experience will make room for it, if necessary, by removing the oldest transition. At iteration i of the training stage, the Q-network grabs a mini-batch of random past transitions $\left\{(s^{(\tau)},x(\sigma_\tau),r^{(\tau)},s^{(\tau+1)})\right\}_\tau$, and computes the loss function (4.25) gradient by using the current weights $\vec{w}_i$. The random transitions τ are uniformly selected among the M transitions present in the Experience memory $\mathscr{D}$. Note that using past experience does not only reduce the weight updates variability[5], but it also boosts data efficiency as each transition is used in many iterations of the training stage. The Q-network, together with the Experience and the aforementioned training procedure, is referred as DQN. Algorithm 6 details the training steps of the DQN using the RMSprop [Geo12] as gradient descend method.

The state space is represented as in the Q-table algorithm of previously presented, and the Q-network input correspond to a concatenation of the last k transitions. During the training stage the Q-network follows an ε-greedy policy, that is later substituted by a greedy policy during the test stage

$$\pi(x(\sigma_t)|s^{(t)}) = \mathbf{1}_{x(\sigma_t)}\left[\arg\max_x Q(s^{(t)},x,\vec{w}_{E_p})\right] \tag{4.26}$$

with $\vec{w}_{E_p}$ being the weights after the last training episode E_p.

4.3.5 Performance Evaluation

This section presents the experimental evaluation of the algorithms introduced in section 4.3.4. It also uses an optimal oracle approach, which solves Problem 4.1 optimally for a sufficiently large time horizon and known future prices. Evidently, such an approach is unfeasible in practice because future prices are unknown; it is only used as a means to assess the optimality of our algorithms empirically.

[5] The weights' updates variability comes as a consequence of the correlation between samples.

Algorithm 6: DQN training algorithm

```
for $e \in E_p$ do
    environment.reset();
    initialize $\phi \in \mathbb{R}^{11k}$;
    for $i \in [0, T]$ do
        $s^{(i)}$ = env.getState();
        $x(\sigma_i) = \begin{cases} u \sim \mathscr{U}\{0,2\} & ,\bar{u} < \varepsilon \\ \arg\max_x Q(s^{(t)}, x, \vec{w}_i) & ,\bar{u} \geq \varepsilon \end{cases}$;
        $r^{(i)}, s^{(i+1)}$ =environment.takeAction($x(\sigma_t)$);
        $\mathscr{D}$.addExperience($(s^{(i)}, x(\sigma_i), r^{(i)}, s^{(i+1)})$);
        $\{(s^{(\tau)}, x(\sigma_\tau), r^{(\tau)}, s^{(\tau+1)})\}_\tau^M = \mathscr{D}$.sample($M$);
        $G_0 = 0$;
        for $\tau \in [0, M)$ do
            $G_\tau = \beta G_{\tau-1} + (1-\beta) L_{\tau-1}^2(\vec{w})$;
            $\vec{w}_{\tau+1} = \vec{w}_\tau - \frac{\alpha}{\sqrt{G_\tau}} L_{\tau-1}(\vec{w})$;
        end
        $\vec{w}_{i+1} = \vec{w}_{M_l}$;
    end
end
```

Table 4.4: Service requirements (from [SC20a])

	t3a.small	**c5.2xlarge**	**c5d.4xlarge**
$f\left(\overline{p^{(t)}(\sigma)}\right)$	$5\frac{inst.}{day}$	$12.5\frac{inst.}{day}$	$25\frac{inst.}{day}$
# CPUs	2	8	16
Memory	2 GB	16 GB	32 GB
Storage	100 GB	400 GB	800 GB
Life-time	$\frac{1}{192}L$	$\frac{1}{8}L$	$L = [96\ h, 240\ h]$
Marginal benefit		$P = 0.2$	

The scenario set up for evaluation is based on a mobile network operator (MNO) federation study case [SC20a] from a large Spanish provider. In addition, it is used the price evolution of a large cloud provider presented in section 4.3.2 as a reference of federation and local service fees. Specifically, both the *federation cost* $f^{(t)}(\sigma)$, and local *deployment cost* $l^{(t)}(\sigma)$ correspond to the service prices of our reference cloud provider in the *eu-west-3a* region (see Fig 4.4). Hence, $\sigma \in \{$*t3a.small, c5.2xlarge, c5d.4xlarge*$\}$.

Experimental setup & environment

As in the selected reference case study [SC20a], the assessed scenario is prone to encounter resource scarcity. The goal is to emulate the business scenario explained in section 4.3.1. The MDP algorithms are employed by the MNO to generate deployment decisions for the incoming arrivals of service requests. Once a service is deployed, it books the requested resources for the requested lifetime period. Upon reaching the lifetime period, the deployed service leaves the system. To this end, two data centers are considered (local and federated) with different capacities. For the local domain, it is selected a medium-size data center from [SC20a]; for the federated domain, [SC20a]'s large data center. The details are depicted in Table 4.5; the federation domains' data center has 6x the capacity of the local one, whereas the local's has capacity to host 5x *c5d.4xlarge* services from Table 4.4.

Table 4.5: Data centers' resource capacities (from [SC20a])

data center	CPU	memory	disk
local	80	2000 GB	160 GB
federation	480	12000 GB	960 GB

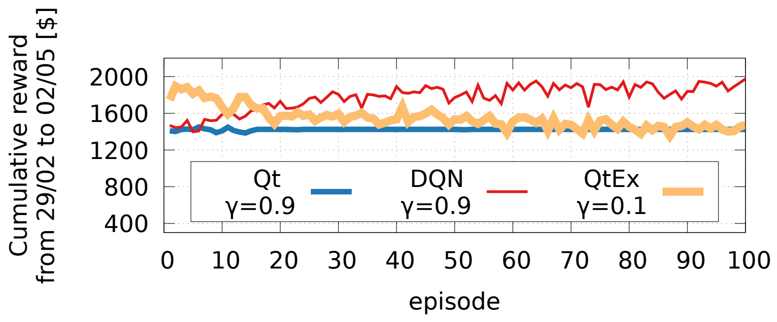


Figure 4.8: Convergence of DQN, QtEx, and Qt after 100 training episodes.

The experiment generates service requests following a Poisson process with arrival rate $f(p^{(t)}(\sigma))$, as described in section 4.3.2, using the price evolution data from a major cloud provider as mentioned before. In more detail, k from eq. (4.10) is set to fit [SC20a]'s data when the service price reaches its average value $\overline{p^{(t)}(\sigma)}$ (see Table 4.4); and parameters a, b from eq. (4.10) are set to $a = 2, b = \frac{1}{2}$ as in [XL13]. Moreover, the marginal benefit P in eq. (4.8) is set to $P = 0.2$ unless otherwise stated (the sensitivity of the approach to P later evaluated). Finally, the departure times are also obtained from [SC20a], i.e., the lifetime of each service is determined by a truncated normal distribution centered in the intervals $\frac{1}{192}L, \frac{1}{8}L, L$ specified in Table 4.4.

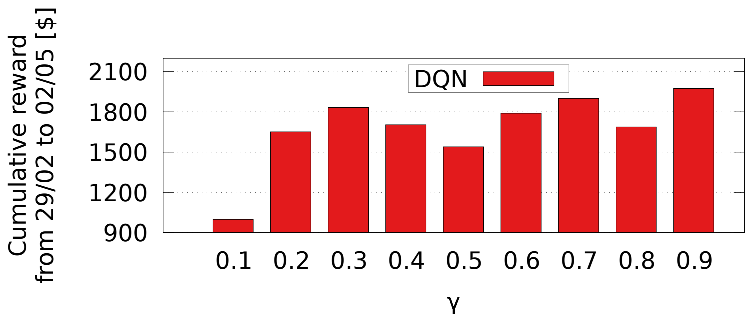
Training

Among the three algorithms presented in section 4.3.4, both the Q-table and the DQN approaches required a prior training phase, which let us fine tune the different hyper-parameters empirically to attain the best performance.

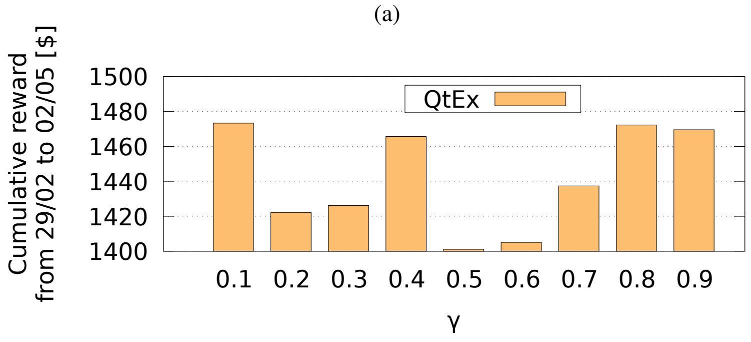
In detail, in the case of the Q-table algorithm, the learning rate and discount factor of the legacy strategy were selected as in [Ant+20], that is $\alpha = 0.95$ and $\gamma = 0.9$, respectively. For the Q-table explore strategy (QtEx), it was used the same learning rate as in the Q-table legacy strategy, and the epsilon values decreased from $\varepsilon_{max} = 0.9$ in episode 1, down to $\varepsilon_{min} = 0.1$ in episode 100. We also tested a variety of discount factors γ between 0.1 and 0.9.

For our DQN approach, RMSprop was used to implement gradient descend, $\alpha = 0.001$, and a moving average parameter $\beta = 0.9$. As in the Q-table explore strategy, different discount factors are tested (in the loss function (4.25)) between 0.1 and 0.9. The RMSprop gradient descend was computed using mini-batches of size $M = 30$ taken from the Experience (see line 11 of Algorithm 6). Figure 4.8 shows that convergence was achieved within 20 episodes for the legacy Q-table solution, whilst QtEx and DQN reached convergence within 60-70 episodes.

Both the Q-table, and DQN algorithms were trained over $E_p = 100$ episodes. Each episode spanned over the service arrivals generated between the 29/02/2020, and the 02/05/2020. Figure 4.9 shows how the discount factor impact the cumulative reward of the DQN and the QtEx approaches during the training stage. None of the algorithms show a monotonic increasing/decreasing tendency with respect to the discount factor. Although, in general, DQN achieved higher cumulative reward for higher values of γ, QtEx obtained the highest cumulative reward given $\gamma = 0.1$. This means that QtEx behaves better by just relying on the instantaneous rewards, whilst DQN did better when



(a)



(b)

Figure 4.9: Impact of discount γ parameter on DQN (a) and Q-table with the exploring strategy (b).

taking more into account the future rewards. The non-stationarity of the service arrivals as well as the transition probabilities make harder to meet a monotonic change in the cumulative reward over γ. Arrivals are more likely to be bursty, which means there is no rule of thumb on how much to take into account the future rewards.

Performance

This section shows the performance of the algorithms described in section 4.3.4, and trained as detailed in §4.3.5. Every algorithm was tested over the service arrivals generated in between 03/05/2020, and 31/07/2020.

1. **Cumulative reward and evolution of decisions:** Figure 4.10a illustrates the cumulative reward of each solution over the aforementioned time-span, Figure 4.10b depicts the price evolution of each service type in the federation domain, and Figs. 4.10c-g detail the decision-making evolution as a percentage of all service requests received at each time instant, for each of the algorithms under evaluation. Note that OPT refers to the optimal oracle, which is used as an ideal benchmark as introduced earlier, and Greedy is a simple policy that only federates when local resources are exhausted, and only rejects services when resources across all domains are exhausted.

   Figure 4.10a shows that OPT achieves a cumulative reward of \$3117.1, while DQN obtained \$2798.88, that is the DQN algorithm was 90% as good as an optimal oracle that knows future service arrivals and prices a priori. On the other hand, Qt and QtEx only attained a cumulative reward of \$1793.82 and \$1892.7, respectively. However, both resulted in a higher benefit than the baseline greedy approach, which barely obtained a cumulative reward equal to \$1418.22. The cumulative reward of each solution starts to diverge noticeably right after the *federation cost* for *c5d.4xlarge* reaches its peak, between 28/05 and 11/06 (see Figure 4.10b). This service is the most demanding in terms of resources, and accordingly the one with highest associated *federation cost*.

   Figure 4.10c depicts the evolution of the decisions made by our greedy baseline. The plot shows that a very small portion of service requests are rejected, as the system is properly dimensioned. It is also evident that this policy is not affected by price fluctuations

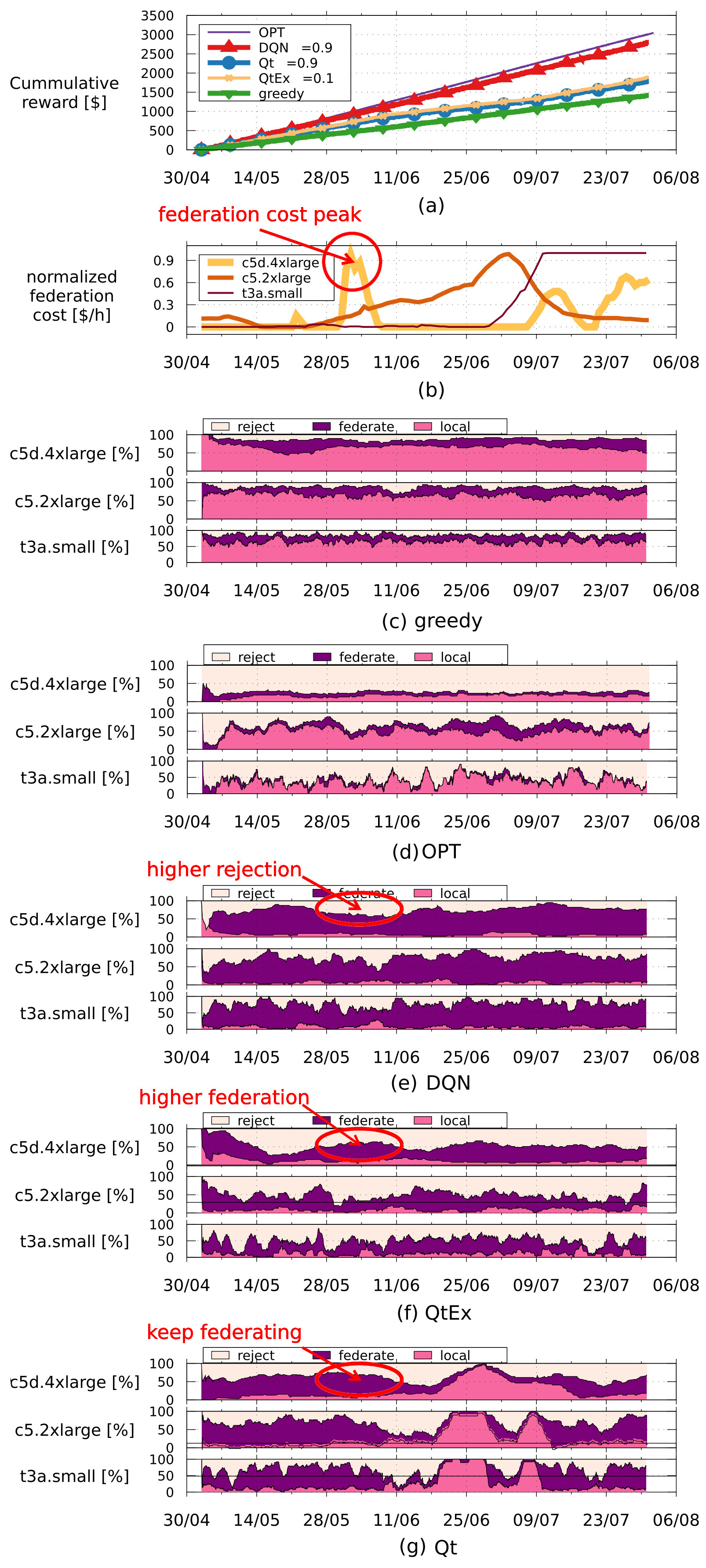


Figure 4.10: (a) cumulative reward of each solution during the May to July dataset; (b) the normalized *federation cost* $f^{(t)}$ over time; and the percentage of instances rejected, federated, or locally deployed by (c) greedy, (d) OPT, (e) DQN, (f) QtEx, and (g) Qt solution.

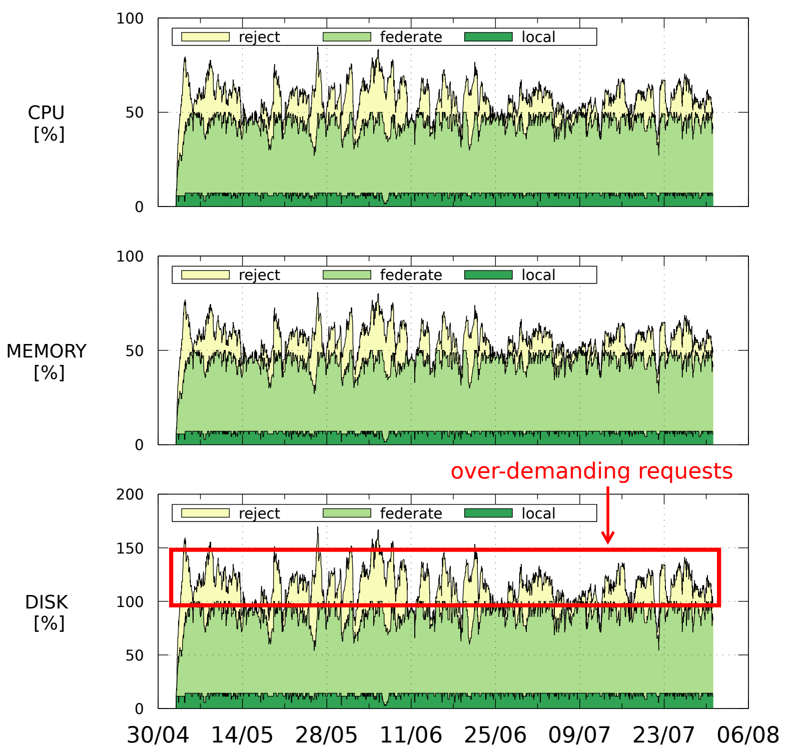


Figure 4.11: Resources consumption by DQN $\gamma = 0.9$.

(Figure 4.10d), which yields very poor performance in terms of revenue (Figure 4.10b). An optimal oracle (Figure 4.10c) is actually more conservative when granting service requests, with a much larger service rejection rate. This is specially true for the largest service (*c5d.4xlarge*). The reason is that is that this type of service incur into much larger net price fluctuations and, as a consequence, a better policy is to handle this type of service requests conservatively.

Lets explore now the evolution of the decisions made by the DQN, QtEX, and Qt methods under assessment (Figure 4.10e-g). The DQN algorithm follows a policy π_{DQN} that federates almost every service, except upon the prospect of price bumps when this policy rejects requests *even if resources are indeed available*. This allows this approach to preserve local resources available during periods when federated resources are expensive, and so this approach can also keep low number of rejections. One can observe that, after *c5d.4xlarge* federation cost reaches its peak, it increases the percentage of rejections, which helps preventing future losses. This is key to deal with the uncertainty a practical approach such as this one have to deal with (in contrast to OPT) without being penalized in terms of reward. Conversely, QtEx and Qt increase or at least keep the same ratio of *c5d.4xlarge* services being deployed at the federated domain, as shown in Figure 4.10f and Figure 4.10g, respectively. As a consequence, both Q-table based algorithms achieve a lower cumulative reward because of the high fees these methods have to pay during sudden price increases.

2. **Resource dynamics**:
Figure 4.11 plots the evolution of the resource consumption over time for our DQN approach between 03/05 and 31/07. This plot also shows, with different colors, whether the resource is consumed by a service deployed locally or federated, or whether the resource was requested by a service that was rejected. These results show that disk is the bottleneck resource in this case; note how this resource around 100% of usage at almost any time, both for the local domain and the federated domain. Even though, resources are dimensioned according to the case study of a large operator [SC20a], these results suggest that both the CPU and memory

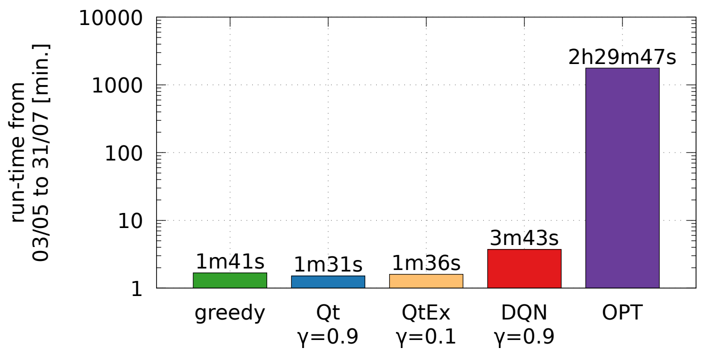


Figure 4.12: Run-time comparison.

resources should be shrunk by 50% without compromising reward, attaining substantial capital cost savings with no impact on revenue.

Computing complexity

Figure 4.12 presents the actual time it takes by each algorithm to expedite decisions over the 3-month period (over 3500 requests). This provides empirical evidence of their computing complexity. Unsurprisingly, OPT takes an order of magnitude longer time to run, as it has to explore a vast action space in a combinatorial problem (note the logarithmic scale of the y-axis). More importantly, both the DQN, QtEx and Qt approaches perform similarly, reaching well within 4 minutes with a slight increase in computing time for our DQN.

Resource dimensioning

Results above are based upon a specific deployment choice proposed by a large operator in Spain [SC20a]. Now it is evaluated the impact of our algorithms onto different deployments. Importantly, the learning approaches are not re-trained on the new setups, experiments simply use the same models trained on [SC20a]'s scenario. This should give us an insight on the portability of these approaches to generic environments.

To this end, two different deployments are set up: In Deployment A, [SC20a] is taken as a baseline and local resources are varied proportionally from 60% to 100%; In Deployment B, again [SC20a] is taken as a baseline and the federated domains' resources are varied from 0% to 600% (that is, 6x the resources available in our baseline's federated domain). The details of these two deployments can be found in Table 4.6 and 4.7, respectively. Figs. 4.13(a) and 4.13(b) depict the cumulative reward attained by each of our algorithms over the same 3-month time period used before, for Deployment A and B.

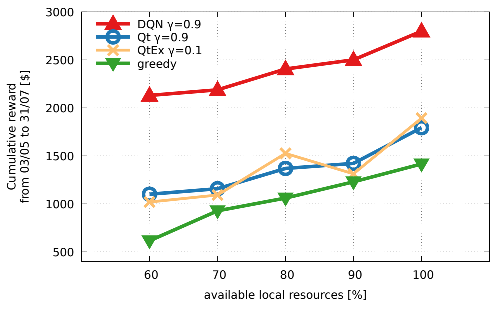
On the one hand, in Deployment A, DQN presents a substantial performance gain over the other

Table 4.6: Resource capacities for deployment A

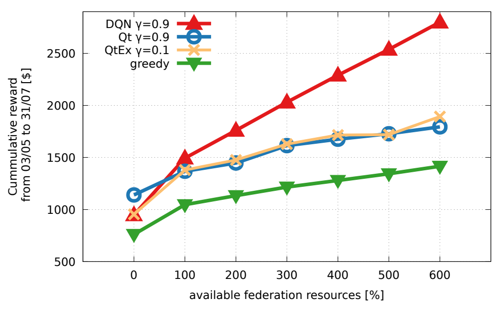
data center	# CPUs	memory	disk
local	[48,..., 80]	[1200,..., 2000] GB	[96,..., 160] GB
federation	480	12000 GB	960 GB

Table 4.7: Resource capacities for deployment B

data center	# CPUs	memory	disk
local	80	2000 GB	160 GB
federation	[0,..., 480]	[0,..., 12000] GB	[0,..., 960] GB



(a) Impact of available local resources in the cumulative reward achieved by each solution – deployment A



(b) Impact of the federation size in the cumulative reward achieved by each solution – deployment B.

Figure 4.13: Cumulative reward vs. available resources

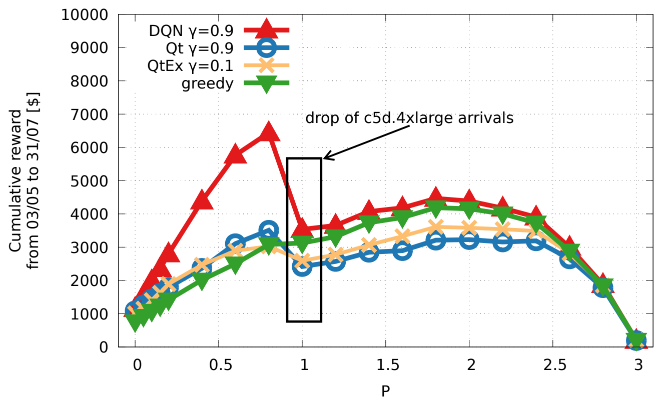


Figure 4.14: Impact of the marginal benefit P in the commutative reward achieved by each solution.

approaches, no matter the size of the local domain. And apart from the QtEx, which outperforms the greedy approach by roughly \$400, every other solution's gain has a similar, and monotonic growth with respect to the available local resources. On the other hand, in Deployment B, the gain achieved by our DQN approach grows much faster with the dimension of the federated domain than all other approaches, e.g., reaching an 80% revenue increase with 6x resources than [SC20a]'s. This is because the DQN federates more services than the greedy and Q-table solutions (see Figure 4.10), and a growth in the federation pool allows it to accommodate more services than the other solutions. Additionally, DQN takes advantage of the *federation cost* fluctuations to federate even more services when the *federation cost* is low, thus, increasing the benefits.

Marginal Profit:

Finally, a sensitivity analysis is performed on the marginal profit associated with the price of the services, P in eq. (4.8). Intuitively, P has an impact on reward via two phenomena: (*i*) higher P increases the net revenue associated with each incoming service, but (*ii*) higher P reduces incentives for customers to make requests into the system.

To this end, it is deployed the baseline scenario (Table 4.5), and all the algorithms are tested during the same 3-month period previously used for a variety of marginal profits between 0 (the *service price* equals that of the local *deployment cost*) and 3 (the *service price* is 4 times that of the local *deployment cost*). The results are depicted in Figure 4.14.

It is worth noting that, when $P \geq 1$, the greedy approach outperforms both Q-table based solutions. This is due to the fact that one gets a lower rate of service requests as P increases (see Figure 4.6), which allows the π_g policy to accommodate all incoming services locally. Conversely, the DQN algorithm reaches its maximum cumulative reward with $P = 0.8$, yielding a drastic drop in reward with $P > 0.8$. This is due to the fact that there is a drop in service requests of type *c5d.4xlarge* (see Figure 4.6(c) for $P \geq 1$). With $P > 1$, DQN obtains a profit mostly from *c5.2xlarge* and *t3a.small* services, and reaches its peak at $P = 1.8$. From that point on, all service requests decay prominently and so does the cumulative reward.

4.4 Conclusions

This chapter shows that the automated deployment of service in network federations is feasible by means of deployment time. Motivated by the experimentation carried out in the 5GEx European project, the chapter proposes heuristic algorithms to assess the embedding of NSs in a federation of multiple administrative domains. The performed stress tests showed the solutions performance under scarce of resources in the federation, and how the proposed set of cutoffs improved the running-times of the algorithms without impacting the acceptance ratios. Moreover, the proposed solutions have polynomial run-time complexity.

This chapter also studies the delegation of NS deployments in a network federation in which each administrative domain offers a time-changing price for using its resources. The proposed solution is built around a MDP formulation, and it is based on a DQN to take the decisions of whether to federate or locally deploy the services. Experiments show that the proposed solution achieves near optimal solutions by means of monetary reward. Moreover, the validity of the solution is strength out as it was tested using a real cloud provider pricing dynamics.

The work in this chapter shows that network federation is a powerful solution to increase the possibility of instantiating incoming NSs upon scarce of resources in an administrative domain. There is still future work in to be done regarding the solutions overviewed in this chapter. In particular, the orchestration algorithms studied in section 4.2 should be compared against the optimal solution to derive the optimality gap. Furthermore, the study has to be extended to consider richer federated network infrastructures not only consisting of interconnected data centers. By, for example, running them on the network operator graphs discussed in the beginning of chapter 3, accounting for a federation of resources, i.e., a blend of graphs generated by section 3.5 and section 3.6.

Another future direction in the contributions of the present chapter, would be the extension of the study on delegation of NSs. In particular, by considering more than two administrative domains in the federation environment. Furthermore, other techniques related to time-series to anticipate to future price peaks, will enhance the performance of not only the proposed DQN agent, but any other agent using price forecasting as input.

5. NFV orchestration for 5G networks

This chapter studies the Virtual Network Embedding (VNE) problem in 5G networks, and proposes solutions to assess the embedding of Network Service (NS)s in the substrate networks. Thanks to the Network Function Virtualization (NFV) paradigm, it is possible to break down the functionality of each 5G service in Virtual Network Function (VNF)s, which brings a deployment flexibility that eases meeting the service requirements of the various 5G use-cases, either by means of latency, bandwidth, or reliability requirements. In particular, the network slicing paradigm presented in section 1.5 proposes the allocation of network resources so as to isolate the network usage by the wide variety of NSs coexisting in 5G networks. As a result, the NFV orchestration algorithms must account for an adequate allocation of the network slices' resources so the services running on top of them satisfy 5G Key Performance Indicator (KPI)s as the aforementioned reliability or latency requirements.

In the following sections, this chapter studies solutions to allocate network resources for 5G NSs. Section 5.1 proposes a heuristic solution for the VNE problem considering network slicing so as to satisfy delay and reliability requirements. And section 5.2 focuses on volatile and mobile devices running 5G services as robotic warehousing, in which the resource allocation has to account for devices running out of battery, or even leaving coverage areas that provide the required wireless connectivity. The proposed NFV orchestration algorithm is evaluated in a haven warehousing scenario. Finally, section 5.3 concludes the chapter summarizing the contributions of the discussed solutions, and pointing out possible future directions for the NFV orchestration algorithms.

5.1 Sliced Edge/Fog networks

The Network slicing paradigm – see section 1.5 – brings the opportunity of meeting the requirements of a wide variety of heterogeneous services, in particular service availability (in both space and time) and service reliability, requirements, exemplified in Figure 5.1. The allocation and isolation of resources among network slices allows the coexistence of enhanced Mobile Broadband (eMBB) and Ultra-Reliable and Low Latency Communications (URLLC) traffic in the same substrate network. On the one hand, high speed and reliable links can be dedicated to the URLLC traffic; even computing resources closer to the edge may be assigned to run URLLC services. On the other hand, network links and Base Station (BS)s with high bit-rates would be assigned to eMBB services. It is important to do an adequate assignment of resources to meet the communication constraints of each network slice.

This section proposes a novel methodology to model the system, as well as the main features of network slicing. Exploiting such methodology, OKpi is developed as an efficient solution that can create high-quality, end-to-end network slices. Specifically, the main contributions of this section

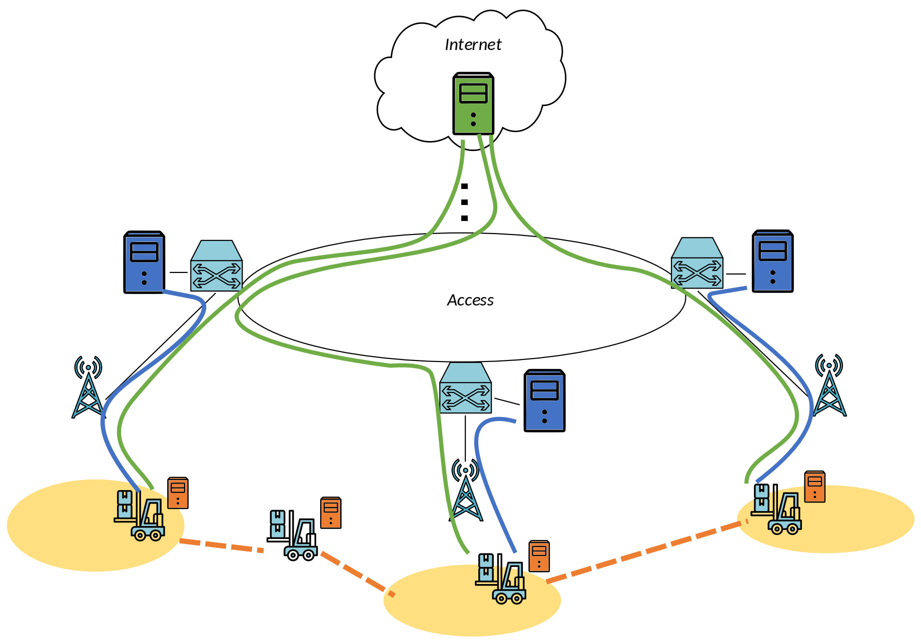


Figure 5.1: As per geographical availability requirements, a mobile robot, smart factory service must be provided within the yellow areas, with high reliability and low latency. This can be obtained by deploying: (a) service instances at the robots (fog resources), at lower cost but also lower reliability, hence, needing redundancy to meet reliability constraints (orange option); (b) three instances at the points of access (PoA), e.g., Access Point (AP)s, covering the target areas (edge resources, blue option); (c) deploying only one instance in the cloud, but with larger delay (green option).

are as follows:

(i) it develops a system model that captures the main aspects of NFV-based networks and can represent the availability of resources at different layers of the network topology, namely, cloud, edge, and fog, as well as the fact that existing VNF instances can be reused for newly-requested services[1];

(ii) it formulates an optimization problem that minimizes the resource cost, while meeting all target KPIs. This section proves that the problem is NP-hard, and proposes the OKpi solution, which has instead polynomial complexity. Leveraging a graph-based representation of the available resources, the possible decisions, and their impact on the KPIs, our scheme can make *joint* decisions on VNF placement and traffic routing that minimize the resources cost, by applying a shortest path algorithm over a multi-dimensional graph. Importantly, such a graph can be built with different levels of detail and size, which results into a tunable trade-off between computational complexity and decision quality;

(iii) it analyzes the properties of the proposed solution and, through numerical results derived under real-world automotive and robot scenarios, this section shows that OKpi closely matches the optimal performance. Furthermore, it show OKpi functionality by implementing it in a testbed supporting a mobile robot, smart factory service.

The rest of the section is organized as follows. Section 5.1.1 introduces the system model, and the problem is formulated in section 5.1.2. The OKpi solution and algorithm are described in section 5.1.3, where several properties of OKpi are proved, so as its computational complexity. Section 5.1.4 shows the performance of our solution through simulations in both small- and large-scale scenarios referring to 5G use-cases, while section 5.1.5 presents some field tests obtained through a real-world testbed.

[1] This is feasible if services share a common subset of VNFs and no service isolation constraints exist.

5.1.1 System model

Table 5.1: Notation table

Symbol	Type	Meaning
$\mathscr{T} = \{t\}$	Set	Set of time intervals
$\mathscr{S} = \{s\}$	Set	Set of vertical services
$\mathscr{V} = \{v\}$	Set	Set of VNFs
$\mathscr{A} = \{\alpha\}$	Set	Set of locations
$\mathscr{E} = \{\psi\}$	Set	Set of endpoints
$\mathscr{C} = \{c\}$	Set	Set of network nodes
$\mathscr{K} = \{\kappa\}$	Set	Set of resources
$\mathscr{I} = \{i\}$	Set	Set of radio interfaces
$\mathscr{L} = \{(i,j)\}$	Set	Set of physical links
$\mathscr{W} = \{w\}$	Set	Set of *strings*/paths
$k(\kappa,c)$	Parameter	Quantity of κ resources at node c
$r_\kappa(v)$	Parameter	Quantity of κ resources required by VNF v to process a unit of traffic
$R_i(c)$	Parameter	Whether node c is equipped with radio interface i
$D_{i,j}$	Parameter	Delay of link (i,j)
$C_{i,j}$	Parameter	Traffic capacity of link (i,j)
$\eta(c,t), \eta(i,j,t)$	Parameter	Reliability of node c and link (i,j) at time interval t
$l(\psi,v_1,v_2)$	Parameter	Traffic originated at π, processed last at v_1, before being processed at v_2
$\chi(v_1,v_2,v_3)$	Parameter	Fraction of traffic processed at v_1, currently processed at v_2, later processed at v_3
$\rho(v,c)$	Variable	Whether node c hosts VNF v
$a_c(\psi,v,\kappa)$	Variable	Quantity of κ resources assigned to VNF v at node c to process traffic from ψ
$\hat{f}_c(\psi,v_1,v_2)$	Variable	Fraction of flow $l(\psi,v_1,v_2)$ processed at VNF v in node c
$p_{i,j}(\psi,v_1,v_2)$	Variable	Traffic from ψ, traversing link (i,j), processed at v_1, later processed by v_2 at node j
$t_{i,j}(\psi,v_1,v_2)$	Variable	Traffic originated at ψ, last processed at v_1, just transiting link (i,j), and to be processed by v_2
$p_{i,j}(\psi,v_1,v_2,w)$	Variable	Processing traffic $p_{i,j}(\psi,v_1,v_2)$ traversing *string* w
$t_{i,j}(\psi,v_1,v_2,w)$	Variable	Transiting traffic $t_{i,j}(\psi,v_1,v_2)$ traversing *string* w
$f(\psi,v_1,v_2,w)$	Variable	Fraction of service flow $l(\psi,v_1,v_2)$ traversing *string* w
$\lambda_c(\psi,v)$	Auxiliary Variable	Quantity of traffic originated at ψ and processed by v at node c
$\tau_{i,j}(\psi,v_1,v_2)$	Auxiliary variable	Traffic originated at ψ, traversing link (i,j), last processed at v_1, and to be processed by v_2
$\tau_{i,j}(\psi,v_1,v_2,w)$	Auxiliary variable	Traversing traffic $\tau_{i,j}(\psi,v_1,v_2)$ traveling over *string* w

The model concisely describes the two main components of mobile, slicing-based networks: the services they support, and the computing and network resources they include. Each of them is modeled through a graph – the service graph and the physical graph, respectively. This section later describes how such graphs can be combined. Further, it is considered that a monitoring platform is in place, with the aim to periodically monitor both the service performance and the status of the system resources. Throughout this section, t denotes the generic time interval over which the system metrics are periodically monitored, and $\mathscr{T}$ the set of such intervals.

Services

A vertical service $s \in \mathscr{S}$ is described through a *service graph* where vertices are VNFs, $v \in \mathscr{V}$, and edges specify in which order the VNFs should process the related data traffic (i.e., how data shall be routed from a VNF instance running on a network node to the next). Note that VNFs can also represent database-related functionalities [KYM19], requiring storage resources: like other VNFs, they must be placed on a node and consume resources therein. An example of service graph for a mobile robot, smart factory use case[2] is depicted in Figure 5.2(left).

A service s is associated with one or more *KPIs*, namely,

– the required bandwidth, or expected traffic load l to be transferred and handled by the VNFs composing the service;
– the maximum allowed delay $D(s)$;
– the minimum level of reliability $H(s)$;
– the required geographical availability at a subset of locations, $A(s) \subseteq \mathscr{A}$, where $\mathscr{A} = \{\alpha\}$ represents the set of all possible locations in the considered region. As an example, $A(s)$

[2] http://wiki.ros.org

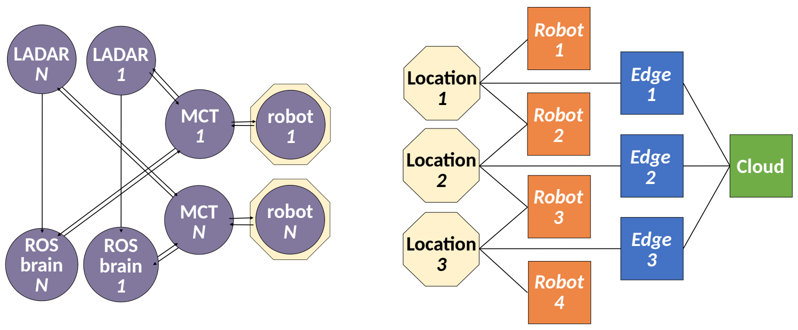


Figure 5.2: Service (left) and physical (right) graphs corresponding to the example in 5.1. In the mobile robot, smart factory graph, each robot transmits its sensors data to the LADAR and the the Robot OS (ROS) brain. The former provides a probabilistic localization of the robots, the latter leverages such a localization and the sensors data to control the robots. Messages are transferred through the Mobile Communication Transport (MCT), e.g., virtual AP. In the service graph, vertices are endpoints (yellow) or VNFs (purple), edges are directed and correspond to flows l. In the physical graph, vertices are endpoints in $\mathscr{E}$ or nodes in $\mathscr{C}$, and edges correspond to links in $\mathscr{L}$; colors correspond to those in 5.1 and refer to the different resource locations: fog (orange), edge (blue), cloud (green).

can represent the urban intersections where an automotive vertical wants to provide a safety service, or the areas where robots should move within a warehouse (5.1). We refer to the combination of a service and a location as an *endpoint* $\psi = (\alpha, s) \in \mathscr{E} \subseteq \mathscr{A} \times \mathscr{S}$;

– the lifetime (or temporal availability) $\varphi(\psi) \subseteq \mathscr{T}$, corresponding to a subset of all time intervals $\mathscr{T}$ during which the service must be available at endpoint ψ.

As foreseen by standards, services may be associated with one or more of these requirements, i.e., not all KPIs have to be specified for all services. Also, without loss of generality, it is considered that the traffic associated with a service is generated at endpoint ψ and has to be processed by the VNFs in the service graph; in Figure 5.2(left), this would correspond to uplink data transfers. Note however that, as discussed later, our model is general and can also capture downlink as well as bidirectional traffic patterns.

The quantity of traffic originated at endpoint $\psi \in \mathscr{E}$, that has been processed last at VNF v_1, and will be next processed at VNF v_2 is denoted by $l(\psi, v_1, v_2)$ (with $l(\psi, v, v)$ being the traffic that will be processed for the first time at v). After a traffic flow is processed at a VNF, the outgoing traffic can increase, decrease, or be split among several other VNFs, according to the service graph. Parameters $\chi(v_1, v_2, v_3)$ express the fraction of the traffic that was last processed (or originated) at $v_1 \in \mathscr{V} \cup \mathscr{E}$, that is currently processed at v_2, and that will next be processed at v_3. For instance, if v_2 is a deep packet inspector, $\chi(v_1, v_2, v_3) = 1$; but if v_2 is a firewall, then $\chi(v_1, v_2, v_3) \leq 1$.

Radio coverage and Fog/Edge/Cloud resources

Network nodes, with switching or computing capabilities, are denoted by $c \in \mathscr{C}$, while endpoints, which are origins or destinations of service traffic, are denoted by $\psi \in \mathscr{E}$. Nodes may be equipped with different resources, e.g., CPU or memory; the set of resources is identified by $\mathscr{K} = \{\kappa\}$. The quantity of resource type κ available at node c is specified through parameters $k(\kappa, c)$, hence, $k(\kappa, c) = 0 \; \forall \kappa$ for pure network equipment like traditional, non-software, switches. Also, binary parameters $R_i(c)$ express whether node c is equipped with radio interface $i \in \mathscr{I}$ or not. A radio interface available at node c determines which locations, hence endpoints, node c covers – an important feature of fog and edge nodes.

Radio coverage, fog, edge, and cloud resources can then be represented through a *physical graph* whose vertices are the network nodes and the endpoints, and the edges $(i, j) \in \mathscr{L} \subseteq (\mathscr{C} \cup \mathscr{E})^2$ represent the physical links connecting them, as per the network topology and the coverage provided by the radio interfaces. Each edge (i, j) is associated with delay $D_{i,j}$ and traffic capacity $C_{i,j}$. Also,

$\eta(c,t)$ and $\eta(i,j,t)$ denote the *reliability* level of any node c and link (i,j) monitored over a time interval $t \in \mathscr{T}$, respectively. Specifically, these quantities express the probability that a specific node or link works as intended by averaging their behavior over $t \in \mathscr{T}$, thus accounting for the time-varying quality of communication links involving fog nodes, e.g., robots or cars. It is worth mentioning that, in general, all the parameters introduced above may differ across fog, edge, and cloud resources.

Service support over the physical graph

To express whether a node c in the physical graph hosts VNF v, this section introduces a binary variable, $\rho(v,c) \in \{0,1\}$. Variables $a_c(\psi, v, \kappa)$, instead, express the quantity of resources of type κ assigned to that VNF v at node c and used to process traffic generated at endpoint ψ.

It is also introduced variables $\tau_{i,j}(\psi, v_1, v_2)$ representing the flows over the physical graph, or, more specifically, the traffic originated at $\psi \in \mathscr{E}$, traversing $(i,j) \in \mathscr{L}$, last processed at v_1, and to be next processed at v_2. Such traffic can be either processed at j, or just transiting through j; these two options are described by the two real variables $p_{i,j}(\psi, v_1, v_2)$ and $t_{i,j}(\psi, v_1, v_2)$, and by imposing: $\tau_{i,c}(\psi, v_1, v_2) = p_{i,c}(\psi, v_1, v_2) + t_{i,c}(\psi, v_1, v_2)$.

Further, to handle the service KPIs more easily, a *string* is defined, $w \in \mathscr{W}$, over the physical graph as a sequence of physical links traversed by a flow, with the first component of the string being an endpoint. Similarly to [Qaz+], the possible strings can be pre-computed and stored for later usage. Since a service flow can be split across different strings, $f(\psi, v_1, v_2, w)$ is defined as the fraction of service flow $l(\psi, v_1, v_2)$ traversing string w. Clearly, such fractions must sum to 1.

The string-wise equivalents to $\tau_{i,j}(\psi, v_1, v_2)$, $t_{i,j}(\psi, v_1, v_2)$, and $p_{i,j}(\psi, v_1, v_2)$ are then $t_{i,j}(\psi, v_1, v_2, w)$ and $p_{i,j}(\psi, v_1, v_2, w)$, respectively. Specifically, $\tau_{i,j}(\psi, v_1, v_2, w)$ represents the traffic of service flow $l(\psi, v_1, v_2)$ traversing link (i,j) on its journey through string $w \in \mathscr{W}$, and then it is imposed $\tau_{i,j}(\psi, v_1, v_2) = \sum_{w \in \mathscr{W}} \tau_{i,j}(\psi, v_1, v_2, w)$. Similar definitions and conditions hold for $t_{i,j}(\psi, v_1, v_2, w)$ and $p_{i,j}(\psi, v_1, v_2, w)$.

Furthermore, the fraction of service flow over a certain string w must match the physical traffic on the corresponding links, i.e., for all endpoints, VNFs v_1 and v_2, links, and strings: $f(\psi, v_1, v_2, w) l(\psi, v_1, v_2) = \tau_{i,j}(\psi, v_1, v_2) \mathbb{1}_{w(i,j)}$, where $\mathbb{1}_{w(i,j)}$ denotes that link $(i,j) \in \mathscr{L}$ belongs to w.

Now that *string*-related variables are defined, the following provides an expression for the latency and reliability KPIs of a service as set forth below.

The service latency comprises of network delay, due to traffic traversing links and switches, and processing times at the nodes hosting VNF instances. Given endpoint ψ, the average network delay can be computed as the weighted sum of the delays associated with the individual strings taken by the traffic originated at ψ:

$$d_{\text{net}}(\psi) = \sum_{w \in \mathscr{W}} \sum_{v_1, v_2 \in \mathscr{V}} f(\psi, v_1, v_2, w) \sum_{(i,j) \in w} D_{i,j}. \tag{5.1}$$

As for the processing time, let $\hat{f}_c(\psi, v_1, v_2)$ be the fraction of the service traffic flow $l(\psi, v_1, v_2)$ processed at the instance of VNF v_2 located at node c. Then the quantity of traffic $\lambda_c(\psi, v_2)$ originating at ψ and processed at the instance of v_2 in c is:

$$\lambda_c(\psi, v_2) = \sum_{v_1 \in \mathscr{V}} \hat{f}_c(\psi, v_1, v_2) l(\psi, v_1, v_2).$$

Note that such traffic may come from different physical links.

Next, VNF instances are modelled as M/M/1-PS queues (see, e.g., [Coh+15b, JPP16, Olj+17]); the choice of the processor sharing (PS) policy closely emulates the behavior of a multi-threaded application running on a virtual machine. The total processing time at the instance of v_2 deployed at node c can thus be written as: $1/(a_c(\psi, v_2, \text{cpu}) - r_{\text{cpu}}(v_2)\lambda_c(\psi, v_2))$, with $r_{\text{cpu}}(v_2)$ denoting the amount of CPU needed by VNF v_2 to process one unit of traffic. Summing over all flows, the total processing delay incurred by traffic originating at ψ is given by:

$$d_{\text{proc}}(\psi)=\sum_{v_1,v_2\in\mathscr{V},c\in\mathscr{C}} \hat{f}_c(\psi,v_1,v_2)\frac{1}{a_c(\psi,v_2,\text{cpu})-r_{\text{cpu}}(v_2)\lambda_c(\psi,v_2)}. \tag{5.2}$$

Finally, notice that the reliability of a string can be computed as the product between the reliability values of all links and nodes belonging to it.

5.1.2 Problem formulation

This section, formalizes the problem of creating end-to-end network slices that meet all the required KPI targets (5.1.2) while minimizing the total cost (5.1.2). First, it introduces the system constraints related to service processing, data routing, and KPI fulfillment. The problem complexity is later discussed.

Flow conservation on the service graph

First, as remarked by the example on the $\chi(\cdot)$ values, note that there is no flow conservation on the service graph. Instead, the following *generalized flow conservation* law holds:

$$\begin{aligned} l(\psi,v_2,v_3) = \sum_{v_1:v_1\neq v_2} & l(\psi,v_1,v_2)\chi(v_1,v_2,v_3) \\ & + l(\psi,v_2,v_2)\chi(\psi,v_2,v_3), \quad \forall v_2,v_3\in\mathscr{V}: v_2\neq v_3\,. \end{aligned} \tag{5.3}$$

The intuitive meaning of 5.3 is that either traffic traveling from VNF v_2 to VNF v_3 must come from another VNF v_1 and then it is transformed in v_2 according to the χ-coefficients (first term of the second member), or it has just originated at ψ and is processed for the first time at v_2 (second term).

Flow conservation and link capacity on the physical graph

The traffic going out of node c must be equal to the sum of that transiting through c and that just processed at c, i.e.,

$$\begin{aligned} \sum_{(c,h)\in\mathscr{L}} \tau_{c,h}(\psi,v_2,v_3)=\sum_{(i,c)\in\mathscr{L}} \Big[& t_{i,c}(\psi,v_2,v_3)+p_{i,c}(\psi,v_2,v_2)\cdot \\ & \chi(\psi,v_2,v_3)+\sum_{v_1\in\mathscr{V}} p_{i,c}(\psi,v_1,v_2)\chi(v_1,v_2,v_3)\Big]. \end{aligned} \tag{5.4}$$

Finally, each physical link (i,j) cannot carry more traffic than its capacity, i.e., $\sum_e\sum_{v_1,v_2}\tau_{i,j}(\psi,v_1,v_2)\leq C_{i,j}$.

Deploying VNFs and assigning resources

Given a set of VNFs, each consuming an amount of resources $a_c(\psi,v,\kappa)$ of type κ at node c, it is impose that the node capabilities are never exceeded, i.e., for any c and κ, $\sum_{\psi\in\mathscr{E}}\sum_{v\in\mathscr{V}} a_c(\psi,v,\kappa)\leq k(\kappa,c)$.

Importantly, for any $\kappa\in\mathscr{K}$, the quantity of traffic processed by v at node c cannot exceed the ratio between the quantity $a_c(\psi,v,\kappa)$ of resource type κ assigned to the VNF, and the quantity $r_\kappa(v)$ of resource type k needed by VNF v to process one unit of traffic, i.e.,

$$\sum_{(i,c)\in\mathscr{L}}\sum_{\psi\in\mathscr{E}}\sum_{v_1\in\mathscr{V}} p_{i,c}(\psi,v_1,v_2)\leq\frac{a_c(\psi,v_2,\kappa)}{r_\kappa(v_2)} \quad \forall\kappa\in\mathscr{K}\,. \tag{5.5}$$

Also, node c's resources can be assigned to a VNF v only if the latter is deployed therein: $a_c(\psi,v,\kappa)\leq\rho(v,c)k(\kappa,c)$, for any c, κ, and v. These conditions imply that no traffic is processed at a node where no instance of a VNF is deployed.

Last, it is ensured that VNFs are placed only at nodes where all the needed radio interface(s) are available, e.g., a MCT may work only at nodes equipped with specific radio interfaces. Thus, for any node c, interface i, and VNF v, it must be satisfied: $\rho(v,c)r_i(v)\leq R_i(c)$, where $r_i(v)\in\{0,1\}$ are parameters specifying whether interface i is needed by VNF v, and $R_i(c)$ specifies whether such an interface is available at c.

Matching service and physical flows

Since the system model includes two graphs, a service graph and a physical graph, it must ensure that service flows l and physical flows τ match. To this end, the flow entering the first VNF of a service graph must correspond to one or more traffic flows on the physical graph:

$$l(\psi,v,v) = \sum_{(\psi,c)\in\mathscr{L}} \tau_{e,c}(\psi,v,v), \forall\psi\in\mathscr{E}, v\in\mathscr{V}. \tag{5.6}$$

Once 5.6 is met, then 5.3 and 5.4 ensure that the traffic on subsequent links is processed as specified by the χ-parameters.

Meeting service KPIs

1. **Service latency:** the latency experienced by a service s is given by the sum of the network delay (as in 5.1) and the processing time (as in 5.2). Recalling that $D(s)$ is the maximum target delay for service s, the service latency constraint for its endpoints can be stated as:

$$d_{\text{net}}(\psi) + d_{\text{proc}}(\psi) \le D(s), \quad \forall\psi\in\mathscr{E}. \tag{5.7}$$

   Note that the relationship between assigned CPU and processing time in the expression of $d_{\text{proc}}(\psi)$ also means that the CPU has a different role from the other types of resources. Indeed, for resources other than CPU, one can assign to each VNF instance exactly the amount needed to honor 5.5, as a greater amount would yield no benefit. With CPU, instead, there is an additional degree of freedom one can play with: assigning more CPU results in shorter processing times, but higher costs.
2. **Service geographical availability:** by service availability requirements, all locations in $A(s)\subseteq\mathscr{A}$ must be covered by service s. In other words, for all endpoints $\psi=(\alpha,s)$: $\alpha\in A(s)$, there must be a link (ψ,c) on the physical graph to a node c that is equipped with a radio interface covering α and that runs (or it is connected to another node running) the first VNF of the service graph.
3. **Service reliability and temporal availability:** the solution ensures that at every monitoring slot the reliability $H(s)$ required for service s is honored by considering a weighted sum of the per-string reliability values. In symbols, $\forall\psi\in\mathscr{E}, t\in\varphi(\psi)$,

$$\prod_{v_1,v_2\in\mathscr{V}} \sum_{w\in\mathscr{W}} f(\psi,v_1,v_2,w) \prod_{(i,j)\in w} \eta(j,t)\eta(i,j,t) \ge H(s). \tag{5.8}$$

   Note that imposing the above constraint for every monitoring slot during the service lifetime also ensures that the service target temporal availability is met.

Objective

Cost is one of the main concerns related to service virtualization and network slicing. Such cost mainly comes from using network and computation resources. To model this issue, it is defined:

- a fixed cost $c_c(v)$, due to the creation at node c of a VNF instance v; this cost is null if an existing VNF instance can be reused;
- a cost $c_c(\kappa)$, incurred when using a unit resource κ at node c;
- a cost $c_{i,j}$, incurred when one unit traffic traverses link (i,j).

Then, upon receiving a request to deploy a service instance s, the following cost-minimization problem is formulated as follows:

$$\min \sum_c \sum_v \left[c_c(v) + \sum_e \sum_\kappa c_c(\kappa) a_c(\psi,v,\kappa) \right] + \sum_{(i,j)} \sum_e \sum_{v_1,v_2} c_{i,j}\tau_{i,j}(\psi,v_1,v_2) \tag{5.9}$$

subject to the constraints reported in Secs. 5.1.2–5.1.2.

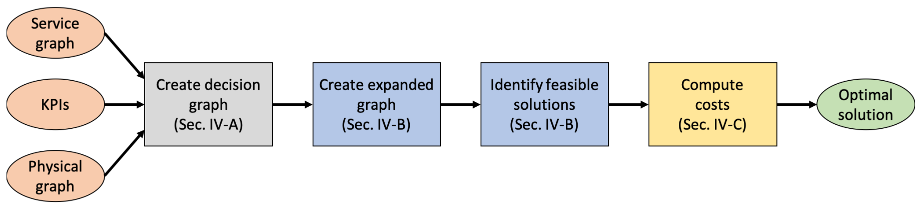

Service graph
KPIs
Physical graph
Create decision graph (Sec. IV-A)
Create expanded graph (Sec. IV-B)
Identify feasible solutions (Sec. IV-B)
Compute costs (Sec. IV-C)
Optimal solution

Figure 5.3: The main steps of the OKpi solution concept.

We recall that the endpoints ψ to consider depend on the service and on its geographic availability requirements, while the VNFs are those specified by the service graph. Furthermore, a solution to the above problem will always opt for reusing an existing instance of a VNF, whenever possible, as this would nullify the instantiation cost $c_c(v)$.

Nature and complexity of the problem

The problem of jointly making VNF placement and data routing decisions is notoriously hard, even when only one KPI is considered [Coh+15b, Fen+17, San+17b]. Te following theorem proves through a reduction from bin-packing the complexity of the problem described in section 5.1.1, showing that directly solving such a problem is impractical for all but very small instances.

Theorem 5.1: The VNF-placement and data routing problem described in 5.1.1 is NP-hard.

Proof. To prove the thesis, it is necessary reduce an NP-hard problem to VNF-placement in polynomial time. Les consider bin-packing, which is known to be NP hard [PS98]: given a set of *items* weighting ω_i each, they are placed throughout a set of *bins*, each having size σ_b, using as few bins as possible. A bin-packing instance is transformed into a corresponding VNF placement, by considering:

- one single VNF v and one single location;
- infinite-capacity, zero-cost, zero-delay, unitary-reliability links;
- as many nodes as there are bins, also with unitary reliability;
- the CPU available at each node is the same as the size of the corresponding bin;
- one single location and as many services (hence, endpoints) as there are items;
- all services include only one VNF, i.e., VNF v;
- the traffic $l(\psi,v,v)$ and the target delay $D(s)$ of each service are such that it requires ω_i CPU units to process the service traffic in time, i.e., $\frac{1}{\omega_i - l(\psi,v,v)r_{\text{cpu}}(v)} = D(s)$;
- all costs are set to zero, except for the VNF creation costs $c_c(v)$, which can be set to any positive value.

In this case, VNF placement and bin-packing decisions are equivalent: the former places VNFs in nodes, the latter places items in bins. The size of bins corresponds to the capacity needed by the VNF instances, and minimizing the cost is tantamount to minimizing the number of bins. The translation from bin-packing to the above simple VNF placement (with only one VNF, single-VNF services, and uniformly-priced nodes) takes polynomial (indeed, linear) time, hence, the VNF placement is (at least) as hard as bin-packing, i.e., NP-hard. Additionally, due to the infinite-capacity, zero-cost, zero-delay, unitary-reliability links, any data routing solution would be optimal. This suggests that, in practice, solving to optimality the problem described in section 5.1.1 would be substantially harder than bin-packing. ■

It also possible to observe that the problem can be seen as a more complex version of a Multi Constrained Path (MCP) problem, where the cost (hence, the weight of the edges in the MCP graph) *changes* at every hop. Although known solutions to the MCP problem, e.g., [Xue+07], are not applicable, such a similarity motivates us to propose an effective and efficient heuristic, called OKpi, that:

- provides high-quality VNF placement and data routing decisions, with guaranteed feasibility;

- such decisions are made in polynomial time;
- under mild homogeneity assumptions, decisions are optimal;
- in the general case, decisions can be arbitrarily close to the optimum.

5.1.3 The OKpi solution

The solution includes four main steps, as summarized in section 5.3. First, the physical graph, the service graph, and KPI targets are blended into a *decision graph* $\widetilde{G} = (\widetilde{N}, \widetilde{E})$, summarizing the service deployment decisions that can be made and their effect on the KPIs. Then this graph is translated into an *expanded graph*, to later use it to identify a set of feasible decisions as well as to select, among them, the lowest-cost one.

For clarity, OKpi is presented in the case where the service graph is a chain with uplink traffic starting from an endpoint ψ, and including N VNFs $v_1 \dots v_N$, each requiring only one instance. As later discussed in this subsection, all such limitations can be dropped: OKpi works with arbitrary service graphs requiring any number of instances for each VNF.

The decision graph

Given the physical graph modeling the service endpoints and the fog, edge, and cloud resources, the decision graph $\widetilde{G}$ is built with the aim to represent the possible service deployment decisions and their effects on the service KPIs.

As a preliminary step, the *computation-capable* nodes are considered in the physical graph (hence, a subset of $\mathscr{C}$), and $(|\mathscr{V}|-1)$ replicas are created for each of them. Consistently, auxiliary edges are created (i) connecting each node c and its replicas in a chain fashion, and assign them zero delay, infinite capacity, and reliability 1, and (ii) connecting any replica of c with any computing node d, for which a link $(c,d) \in \mathscr{L}$ exists. Crucially, introducing node replicas enables to account for the possibility to deploy multiple VNFs at the same computation-capable node without introducing self-loops in the decision graph. Indeed, as it will be more clear later, given that a VNF is placed in c, each replica thereof represents the possibility to deploy the next VNF again in c.

Let then $\widetilde{G} = (\widetilde{N}, \widetilde{E})$ be the decision graph where:
- $\widetilde{N}$ includes the endpoints in $\mathscr{E}$, and the computation-capable nodes in the physical graph as well as their replicas;
- $\widetilde{E}$ is the set of (i) the aforementioned auxiliary links, and (ii) the virtual links (i.e., single physical links or sequences thereof) connecting the vertices in $\widetilde{N}$.

Every edge $(\tilde{n}_1, \tilde{n}_2)$ in $\widetilde{E}$ representing a virtual link has the following properties:
- its capacity $\widetilde{C}_{\tilde{n}_1,\tilde{n}_2}$ is set to the minimum of the individual capacities of the physical links composing the virtual link;
- its delay $\widetilde{D}_{\tilde{n}_1,\tilde{n}_2}$ is set to the sum of the individual delays of the physical links composing the virtual link;
- its reliability $\widetilde{\eta}_{\tilde{n}_1,\tilde{n}_2}$ is set to the product of the reliability values of physical links and nodes (both computation and pure-routing capable) included in the virtual link.

Lets now consider the additive KPIs and, for simplicity, let us focus on two of them, e.g., delay and reliability. Every edge $(\tilde{n}_1, \tilde{n}_2)$ in the decision graph, has associated a *multi-dimensional weight* $\tilde{w}(\tilde{n}_1, \tilde{n}_2)$, defined as:

$$\tilde{w}(\tilde{n}_1, \tilde{n}_2) = \left(\frac{\widetilde{D}_{\tilde{n}_1,\tilde{n}_2}}{D(s)}, \frac{\log \widetilde{\eta}_{\tilde{n}_1,\tilde{n}_2}}{\log H(s)} \right). \tag{5.10}$$

The intuition behind 5.10 is that the weight of edge $(\tilde{n}_1, \tilde{n}_2)$ corresponds to the fraction of the target delay and reliability that will be “consumed” by taking that edge, i.e., by deploying a VNF at $\tilde{n}_1$ and the subsequent one at $\tilde{n}_2$. Using logarithms in the second term of the weight allows to translate a multiplicative performance index (namely, reliability) into an additive one[3].

[3] It is easy to see that $\widetilde{\eta}_{\tilde{n}_1,\tilde{n}_2} \widetilde{\eta}_{\tilde{n}_2,\tilde{n}_3} \geq H(s)$ translates into $\frac{\log \widetilde{\eta}_{\tilde{n}_1,\tilde{n}_2}}{\log H(s)} + \frac{\log \widetilde{\eta}_{\tilde{n}_2,\tilde{n}_3}}{\log H(s)} \leq 1$.

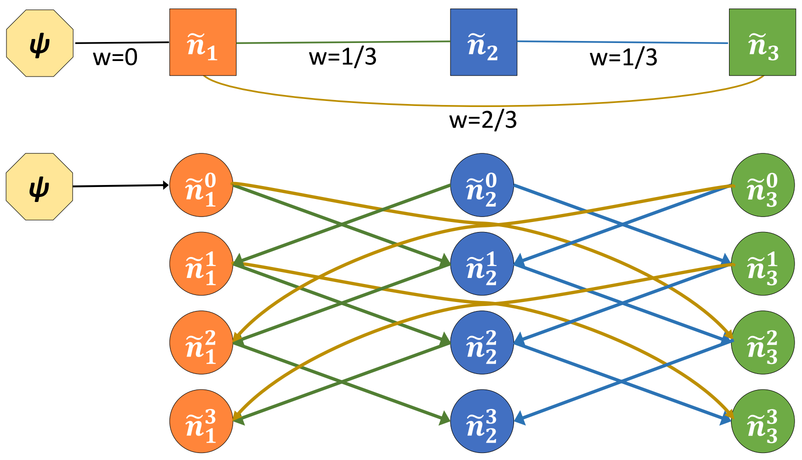


Figure 5.4: Decision graph (top) and expanded graph (bottom) when only delay is considered as a KPI and $\gamma = 3$. In the decision graph, edges $(\tilde{n}_1, \tilde{n}_2)$ and $(\tilde{n}_2, \tilde{n}_3)$ have delay of 1 ms, while $(\tilde{n}_1, \tilde{n}_3)$ has delay of 2 ms (vertices representing replica nodes are omitted for simplicity); the target delay is 3 ms.

It is worth mentioning that, when some services are already active in the network, the decision graph is built considering the *residual* capabilities of physical links and nodes, i.e., those not assigned to already-running services. Similarly, in case of virtual links sharing the same physical links, their capacity is updated as traffic is allocated to the physical links.

The expanded graph: finding decisions honoring availability and additive KPIs

Given the decision graph $\widetilde{G}$, the first purpose is to identify a set of *feasible* service deployment decisions that are consistent with the target KPIs. To this end, as a preliminary step, the service geographical and temporal availability requirements are met by pruning from $\widetilde{G}$ the vertices and edges that do not satisfy such constraints.

Next, the proposed solution follows an approach inspired by [Xue+07] and build a multi-dimensional, *expanded graph*, with as many dimensions as the number of additive KPIs. Specifically, given a positive integer value of resolution γ:

1. for each vertex $\tilde{n}$ in the decision graph, one creates as many corresponding vertices as $(\gamma+1)^2$, where the exponent 2 corresponds to the number of additive KPIs. Such vertices are denoted by $\tilde{n}^{\mathbf{d}}$ where $\mathbf{d}$ is a vector with as many integer elements as the number of additive KPIs and the value of such elements ranges between 0 and γ, i.e., $\tilde{n}^{\mathbf{d}} = \tilde{n}^{0,0}, \tilde{n}^{0,1} \dots, \tilde{n}^{0,\gamma} \dots, \tilde{n}^{\gamma,\gamma}$;
2. for every edge $(\tilde{n}_1, \tilde{n}_2) \in \widetilde{E}$ with capacity $\widetilde{C}_{\tilde{n}_1,\tilde{n}_2}$ greater or equal to the amount of traffic to process, create directed edges from each vertex $\tilde{n}_1^{i,j}$ to vertex $\tilde{n}_2^{i+\lceil \gamma w(\tilde{n}_1,\tilde{n}_2)[0] \rceil, j+\lceil \gamma w(\tilde{n}_1,\tilde{n}_2)[1] \rceil}$ (if such a vertex exists), where the two superscripts refer to delay and reliability, respectively. For example, with $\gamma = 1$, if edge $(\tilde{n}_1, \tilde{n}_2)$ has weight $\tilde{w}(\tilde{n}_1, \tilde{n}_2) = (0.1, 1.5)$ and enough traffic capacity, there will be a directed edge from $\tilde{n}_1^{0,0}$ to $\tilde{n}_2^{1,2}$, but not from $\tilde{n}_1^{0,0}$ to $\tilde{n}_2^{0,1}$.

The expanded graph has *no weights* on its edges: the delay and reliability information that is expressed by weights in the decision graph is now represented by the topology of the expanded graph. A one-dimensional (i.e., one-KPI) example of decision graph and corresponding expanded graph is depicted in Figure 5.4.

Finally, a set of possible service deployments are identified, i.e., VNF-to-compute node assignments and the corresponding data routing. That is done by looking for the shortest paths in the expanded graph that (a) begin at endpoints, and (b) contain as many edges as there are VNFs to place. The latter is trivially required by the need to deploy all VNFs on the service graph, and by the fact that placing more VNFs at a physical node is allowed thanks to the replica nodes and auxiliary edges.

In the following, several fundamental remarks are made about the expanded graph and on the paths, hence, the deployment decisions they correspond to. Given KPI, the *depth* of a vertex in the expanded graph is defined as the value of the element in the superscript corresponding to the KPI. Note that, by construction (see point 1 above), the maximum value of depth is γ. Also, let the *steepness* of an edge be the difference in depth between its target and source vertices. Considering the one-KPI example in Figure 5.4(bottom), vertex $\tilde{n}_1^0$ has depth 0, vertex $\tilde{n}_3^2$ has depth 2, and the edge between the two has steepness $2-0=2$, i.e., equal to $\lceil \gamma w(\tilde{n}_1,\tilde{n}_3)\rceil$.

By construction, for a given KPI, the ratio between the steepness of an edge and γ is greater or equal to the weight component on the corresponding edge of the decision graph, which in turn is the fraction of the KPI target values consumed by making that decision (see equation 5.10). As an example, considering edge $(\tilde{n}_1^0,\tilde{n}_3^2)$ in 5.4(bottom), one has:

$$\frac{\text{steepness}}{\gamma}=\frac{2}{3}\geq w(\tilde{n}_1^0,\tilde{n}_3^2)=\frac{D_{\tilde{n}_1,\tilde{n}_3}}{D(s)}=\frac{2}{3}. \tag{5.11}$$

The observations above allows to state a very relevant property of the decisions corresponding to the paths on the expanded graph.

Lemma 5.2: The decisions corresponding to any path on the expanded graph honor all additive KPIs.

Proof. By definition, the depth of a vertex corresponds to the total steepness of the path required to reach it from endpoint ψ. Given that the maximum depth in the expanded graph is γ, there is *no path* with total steepness[4] greater than γ. Thanks to the relation between weight and KPI targets (exemplified in equation 5.11), this implies that, given a path on the expanded graph, the sum of the weights of the corresponding edges in the decision graph cannot exceed 1, i.e., the corresponding decisions honor additive KPIs (including, thanks to the logarithmic weights, reliability). ■

Importantly, the smaller the resolution γ, the fewer the possible values of depth and steepness in the expanded graph, the fewer the levels of consumption of the KPI target values one is able to distinguish, which corresponds to introducing an error, akin to quantization. Indeed, $\gamma+1$ can be seen as the number of quantization levels[5] admitted: in the extreme case of $\gamma=1$, all edges would have a steepness of 1, which also corresponds to exhausting the whole KPI target in one hop. Such a quantization error may lead to discarding some feasible solutions, and thus, in the most general case, may jeopardize the optimality of OKpi. However, two important facts stand out: *(i)* even enumerating all feasible paths in the decision graph is NP-hard, as proven in [Xue+07], hence, quantization is necessary; *(ii)* by increasing γ, OKpi can get *arbitrarily close to the optimum* (at the price of higher complexity).

Last, it is remarked that all paths on the expanded graph honor additive KPIs constraints, *with the possible exception of delay*. Indeed, unlike other KPIs, whether or not the delay target is violated depends not only on the network latency, hence, the VNF placement, but also on the processing time, i.e., the quantity $a_c(\psi,v,\text{cpu})$ of CPU assigned to each VNF, which in turn impacts the deployment cost. It is possible to account for this important aspect thanks to the M/M/1-PS model used for the processing delay formula – see equation 5.2. In particular, below it is shown how to determine, given a possible deployment, whether there is a CPU assignment consistent with the target delay, and the cost thereof.

[4] The steepness of a path should be not confused with the length of a path.

[5] Using logarithms for reliability values, which are all typically very close to 1, is akin to performing adaptive quantization.

Minimizing the cost

Now it is necessary (*i*) for every path found, to identify the minimum-cost CPU assignment, i.e., the optimal values of the $a_c(\psi, v, \text{cpu})$ variables – if such an assignment exists –; and (*ii*) to determine the path that minimizes the overall cost.

To this end, for each path, hence, for a fixed ψ and for VNFs $v_1 \dots v_N$ to be deployed at computing nodes $\tilde{n}_1 \dots \tilde{n}_N$, respectively, the following problem is solved:

$$\min \sum_{\tilde{n},v} a_{\tilde{n}}(\psi, v, \text{cpu}) c_{\tilde{n}}(\text{cpu}) \qquad \text{s.t.} \tag{5.12}$$

$$\sum_{\tilde{n}_1,\tilde{n}_2} \left(D_{\tilde{n}_1,\tilde{n}_2} + \frac{1}{a_{\tilde{n}_2}(\psi, v_2, \text{cpu}) - r_{\text{cpu}}(v_2) \lambda_{\tilde{n}_2}(\psi, v_2)} \right) \leq D(s),$$

as well as to constraints concerning link capacity, node capability, and flow conservation, equivalent to those presented in section 5.1.2. If the problem above is infeasible for a given path, then that path (and the corresponding decisions) is incompatible with the target KPIs and must be discarded.

Once the problem in 5.12 is solved for all paths identified in the expanded graph, it is computed the total cost associated with each path (including all components defined in section 5.1.2) and select and enact the lowest-cost deployment, thus fulfilling OKpi's purpose. Importantly, the problem is *convex*, hence, it can be efficiently solved in polynomial time [BV04]. The proof simply follows from observing that (i) the objective in equation (5.12), as well as the flow conservation and capability constraints, are linear, and (ii) the second derivatives of the delay constraint, are positive in the decision variables, hence, the constraint itself is convex.

General scenarios

Now it is shown how OKpi tackles arbitrary scenarios.

- **Arbitrary service graphs:** if the service graph is more complex than a chain, it is possible to proceed by decomposing the graph into a set of chains (e.g., in Figure 5.2(left), one in uplink, from the MCT to the DB, and one in downlink, from the detector back to the MCT). OKpi is then applied subsequently to each chain, and the deployment decisions are cascaded. The case where multiple endpoints have to be covered, as in Figure 5.2(left), is handled in the same way.
- **Multiple VNF instances:** If the problem is infeasible for *all* possible paths found, a reason could be the need to split the processing burden across multiple instances of the same VNF. This case is handled by first identifying the bottleneck VNF, i.e., taking the longest to process the service traffic, and then increasing by one the number of instances of that VNF in the service graph. OKpi is then re-run on the modified service graph.

OKpi analysis

This subsection proves several properties about OKpi. We start with the most essential aspect related to its effectiveness, i.e., its ability to meet all service KPIs:

Property 5.3: OKpi's decisions honor all KPI targets.

Proof. By 5.2, all decisions honor the additive KPIs. Concerning delay, it is guaranteed that such a KPI target is met, thanks to the delay constraint imposed while performing the CPU assignment. As noted in "Minimizing the cost", decisions resulting in an infeasible problem are discarded, hence, the selected decision honors the delay target. Finally, the availability constraints are satisfied through the initial selection of the vertices of the decision graph. ■

The following addresses the computational complexity of OKpi:

Property 5.4: The worst-case computational complexity of OKpi (including the graph generation and the solution of the problem stated in (5.12)) is polynomial.

Proof. To prove the property, it is shown that each of the steps described in section 5.1.3 has a polynomial run-time. Specifically,

(i) creating the decision/expanded graph requires creating at most $\gamma^2(|\mathscr{V}||\mathscr{C}|+|\mathscr{E}|)$ nodes and at most $\gamma^2|\mathscr{V}||\mathscr{L}|$ edges, where $|\mathscr{V}|$ is the number of VNFs specifying the service and, given the service, is a constant.

(ii) Finding the possible decisions implies computing the shortest paths between any endpoint (i.e., vertex meeting the availability constraints) and any other node in the expanded graph, which, in the worst case, has complexity [Sei95] $o(n^{2.3})$ with n being the number of nodes in the expanded graph.

(iii) Computing the optimal CPU assignments requires solving a convex optimization problem, which has cubic complexity [BV04] in the problem size; indeed, convex problems are routinely solved in embedded computing scenarios.

Thus, the overall time complexity of the OKpi approach is polynomial. ■

Also, in the case where the physical graph is homogeneous, is possible to prove that OKpi can return the optimal solution:

Property 5.5: If all links and nodes have the same capabilities and cost, then the output of OKpi is optimal.

Proof. There is only one point in the procedure described where, in general scenarios, it may be the case that optimal solutions are overlooked. Finite γ values may cause a quantization-like error: solutions with different KPI consumption and/or cost can be associated with the same path over the extended graph; therefore, the extended graph may not consider all possible ways to move from one node of the decision graph to another. In the special case of homogeneous links and nodes, however, no such different possibilities exist: taking a finite value of γ is enough to consider all possible choices the system offers and, hence, to make an optimal decision. Note that restricting our attention to shortest paths on the expanded graph does not harm optimality, as adding hops implies consuming a higher (or equal at best) fraction of KPI targets and cannot decrease the cost. ■

Finally, the next property considers the expanded graph and show that it can be built in polynomial time:

Property 5.6: The worst-case computational complexity of building the expanded graph is $O\left((\gamma+1)^4 \cdot |\widetilde{N}|^2 \cdot K\right)$.

Proof. Given that two additive KPIs, the expanded graph has $(\gamma+1)^2$ nodes for each node in the decision graph, (with the number of nodes in the decision graph being $|\widetilde{N}|$). Therefore, the total number of pairs in the expanded graph is $O\left(\left[(\gamma+1)^2|\widetilde{N}|\right]^2\right)$, and it is necessary to evaluate whether or not an edge shall be created for each of these pairs. Doing so requires checking each KPI, for a total of $O\left((\gamma+1)^4|\widetilde{N}|^2K\right)$ checks, where K is the number of KPIs. It follows that the global, worst-case complexity of building the expanded graph is quadratic in the network topology and polynomial in γ. ■

5.1.4 Numerical results

This subsection first focuses on a small-scale scenario and an inter-robot communication service, and compare the performance of OKpi against the optimum obtained via brute force. Then it moves to a large-scale scenario and real-world automotive service, and characterizes how the quantity of traffic to serve and the maximum delay impact the decisions made by OKpi. Finally, this subsection considers a mobile robot, smart factory service and investigate the impact of the number of robots on the decisions made by OKpi and the resulting performance.

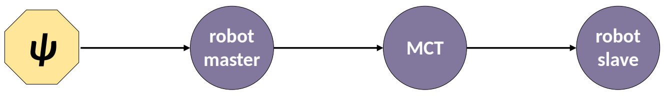


Figure 5.5: Service graph specifying the inter-robot communication service. Yellow and purple vertices denote endpoints and VNFs, respectively.

Table 5.2: Algorithm run times for the small-scale scenario, when the target reliability is 0.999, the maximum delay is 50 ms, and the traffic multiplier is 1

γ	Run time [s]
2	2.1
4	2.4
6	2.7
8	3.1
10	3.8
optimal	284.8

Small-scale scenario: comparison against the optimum

[6]The inter-robot communication service [Mou+18] is considered, whose graph is depicted in Figure 5.5. A room (hence, an endpoint) contains three robots, with different levels of reliability: $\eta(\text{robo1}) = 0.999999$, $\eta(\text{robo2}) = 0.99999$, and $\eta(\text{robo3}) = 0.9999$. Two of these three robots must be used to perform an operation, hence, run the robo-master and robo-slave VNFs. The communication between the two selected robots can take place through three types of PoAs, with different levels of reliability (micro-cell: 0.999999, pico-cell: 0.99999, femto-cell: 0.9994), and costs as reported in [NJ14b]. The offered traffic is 1 Mb/s per robot, as specified in [Mou+18].

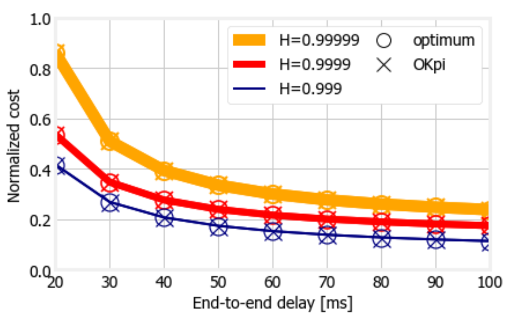
Figure 5.6 depicts the results when OKpi's resolution is set to $\gamma = 10$. The first aspect of interest is the relationship between the target KPIs and cost: as appreciated in Figure 5.6(a) and Figure 5.6(b), a longer allowable delay results in a lower cost; conversely, a higher traffic load or a higher target reliability both result in higher costs. Intuitively, this is due to the fact that cheaper resources (e.g., robot 3) tend to have lower reliability and/or capacity, hence, it is impossible to use them when the KPI targets become very strict.

Interestingly, in both Figure 5.6(a) and Figure 5.6(b), OKpi matches the optimum in *all* cases. Indeed, as previously discussed, OKpi always matches the optimum if the resolution γ is high enough; in the small-scale scenario considered for Figure 5.6, $\gamma = 10$ is sufficient to this end.

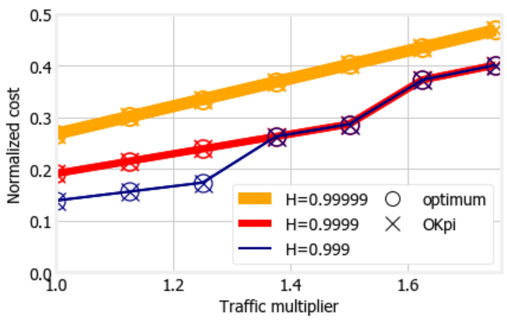
Figure 5.6(c) shows the effect of setting a lower resolution, namely, $\gamma = 3$. As appreciated by comparing the left and center bars, a lower value of γ results in suboptimal, higher-cost decisions. Specifically, the difference is due to the fact that, when $\gamma = 3$, a higher-cost PoA is selected, namely, the pico-cell *in lieu* of the femto-cell. This happens because, for $\gamma = 3$, the edges corresponding to the femto-cell in the expanded graph have steepness $\left\lceil \gamma \frac{\log 0.9994}{\log 0.999} \right\rceil = 2$. Considering that (*i*) all other edges have steepness 1 and (*ii*) OKpi seeks for paths composed of three edges (same as the number of VNFs to place) with a total steepness not exceeding $\gamma = 3$, the edges corresponding to the femto-cell will never be selected, hence, the corresponding decision is never considered. In summary, as discussed in the previous sections, using a too-low γ made us overlook a feasible – and, in this case, optimal – solution.

In the same settings as for Figure 5.6(right), Table 5.2 shows the time taken by the OKpi algorithm for different values of γ, as well as the time it takes to find the optimal solution. OKpi is implemented in Python, and all tests are run on a server with 40-core Intel Xeon E5-2690 v2 3.00GHz CPU and 64 GB of memory. One can observe that, as expect, OKpi run times are very short, much shorter than the time it takes to find the optimal solution. Even more interestingly,

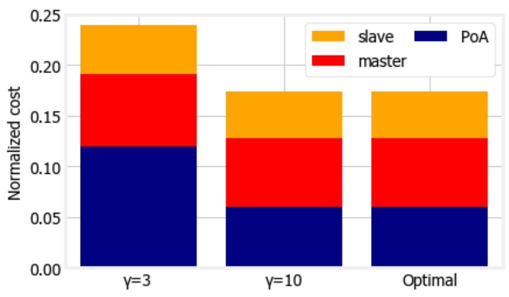
[6] Numerical simulations of the small-scale scenario where developed by the author of the present thesis, and later corrected by Dr. Francesco Malandrino.



(a)



(b)



(c)

Figure 5.6: Small-scale scenario (inter-robot communication service): cost as a function of the maximum delay (a) and of the traffic load (b), for different values of target reliability; cost breakdown (c) when the target reliability is 0.999, the maximum delay is 50 ms, the traffic multiplier is 1, and γ varies.

larger values of γ do result in longer run times, but the increase is substantially slower than the worst-case complexity derived in 5.6.

Large-scale scenario: impact of traffic and delay

Table 5.3: Large-scale scenario: points of access and computing nodes characteristics

Item	Reliability	Latency	Cost
Points of access			
macro-cell	0.99999999	6 ms	1.02 USD/Gbit
micro-cell	0.9999999	3 ms	2.31 USD/Gbit
pico-cell	0.999999	2 ms	3.80 USD/Gbit
Computing nodes			
cloud ring (Azure DataBox)	0.99999999	8 ms	2.23 USD/Gbit
aggregation ring (PowerEdge)	0.9999999	3 ms	5.23 USD/Gbit
local ring (small data center)	0.999999	1 ms	10.47 USD/Gbit

[7]Now OKpi is tested on a large-scale scenario to validate its performance in presence of more complex service graphs and under a larger, and more diverse, network infrastructure. Specifically, on an urban environment where a safety service, namely, vehicle collision avoidance [Avi+19, Mal+20], has to be provided at specific intersections. The service graph is depicted in Figure 5.7: messages sent by vehicles are collected through the MCT, e.g., virtual Evolved Node B (eNB) plus vEvolved Packet Core (EPC), then stored in a database and used for detecting vehicles on a collision course. The latter are warned by sending them an alert. Based on a real-world road topology (see Figure 5.8), a total of 9 intersections (hence, endpoints) are covered by a combination of PoAs, namely, macro-, micro- and pico-cells, whose coverage is shown in Figure 5.8.

[7] Numerical simulations of the large-scale scenario where developed by the author of the present thesis, and later corrected by Dr. Francesco Malandrino.

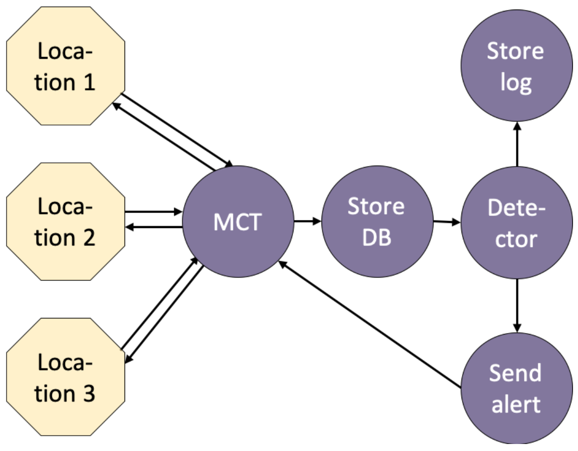


Figure 5.7: Service graph of the safety automotive service (vehicle collision avoidance). Yellow and purple vertices denote endpoints and VNFs, respectively.

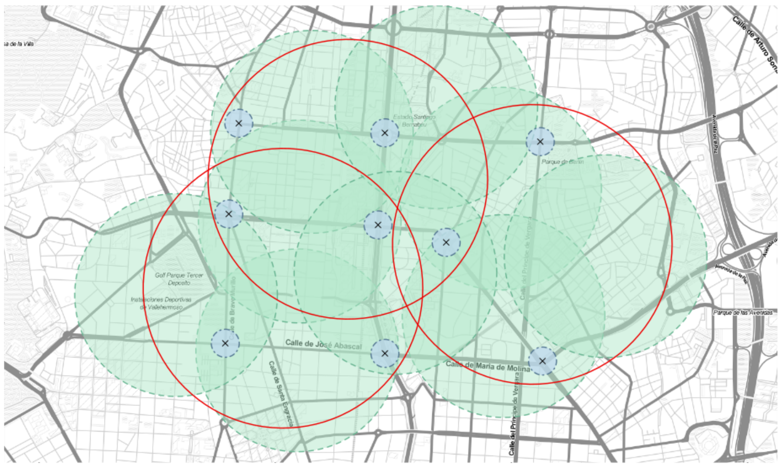


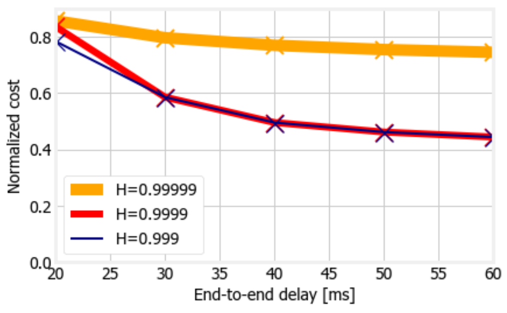
Figure 5.8: Road topology used in the large-scale scenario. The nine crossings correspond to endpoints; red, green, and blue circles represent the coverage areas of macro-, micro- and pico-cells, respectively.

Different PoAs have different reliability, latency, and cost values, as reported in Table 5.3. The front- and back-haul network topology is based on [Mar+19c] and ITU standard [ITU18], and includes core nodes, aggregation nodes, and local (i.e., close to the PoAs) nodes, with features as summarized in Table 5.3 [Car19]. The total service traffic is 1.5 Mb/s, and γ is set to 40 (higher values do not improve the performance). The generation of these graphs was done using the `5GEN` package mentioned in section 3.5.

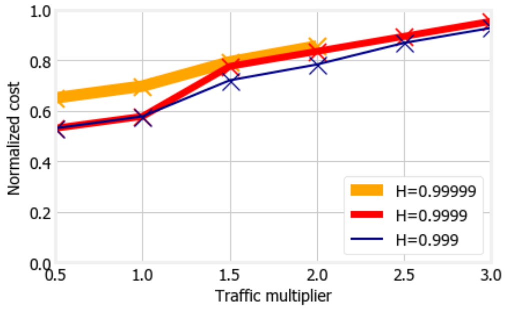
Figure 5.9(a) shows that, as one might expect, a shorter target delay results in higher costs. It is also interesting to observe the behavior of the intermediate curve, corresponding to $H(s) = 0.9999$: when the target delay is very short, its associated cost is almost the same as for $H(s) = 0.999999$ case; as the target delay increases, its cost drops to the same level as the $H(s) = 0.999$ case. This bespeaks the complexity of the decisions OKpi has to make, and their sometimes counter-intuitive effects.

In Figure 5.9(b), the traffic load is multiplied by a factor ranging between 0.5 and 3. Again one can observe that to a higher traffic corresponds a higher cost, even though the growth is less than linear, owing to the fixed costs described in the objective function of the associated optimization problem. Also notice how the yellow curve in Figure 5.9(b), corresponding to the highest reliability level, stops at a multiplier of 2: for higher traffic demands, the network capacity is insufficient to provide the service with the required reliability.

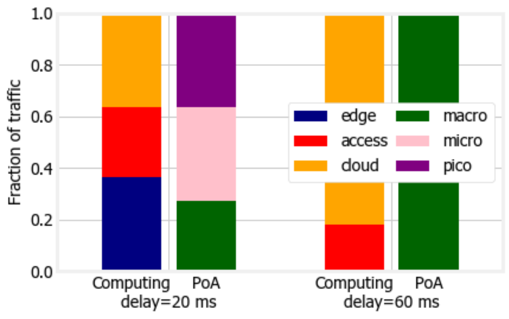
Figure 5.9(c) shows which PoAs and computing nodes are selected for the minimum and



(a)



(b)



(c)

Figure 5.9: Large-scale scenario (automotive safety service): cost as a function of the maximum delay (left) and of the traffic load (center), for different values of target reliability; fraction of traffic (right) traversing different PoAs and computing nodes when the target reliability is 0.999, the traffic multiplier is 1, and the target delay varies.

maximum target delay values. Interestingly, in the presence of tight delay constraints, different PoAs and resources are all used (left bars). On the contrary, for the largest target delay, the cheapest options – cloud and macro-cells – are preferred.

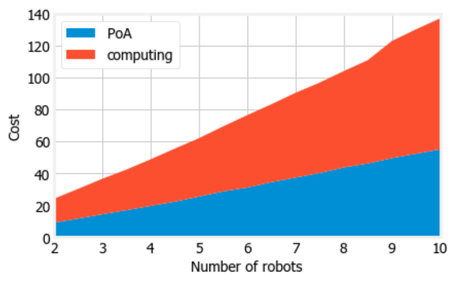
Smart factory scenario: impact of the number of robots

[8]To demonstrate the scalability of OKpi, the following analyzes the scenario depicted in Figure 5.1 and the mobile robot, smart factory service in Figure 5.2(left), including a set of N robots. This scenario is an enriched version of the small-case robotic service, as resource provisioning has to account for more robots, compared to the small-scale scenario, and the smart factory service graph now includes additional VNFs that can be placed on servers rather than on robots. The available computing nodes and PoAs are the same as in Table 5.3, and the goal is to study how the number of users (robots, in this case) impacts the decisions made by OKpi and the resulting performance. Figure 5.10(a) shows that, as one might expect, a larger number of robots always results in a higher cost; interestingly, computing nodes and PoAs account for comparable shares of the overall cost.

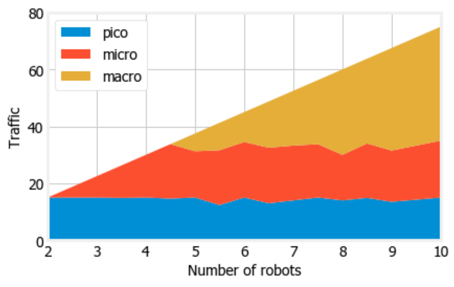
Figure 5.10(b) and Figure 5.10(c) summarize how much traffic is handled by different PoAs and computing nodes as the number of robots grows. Figure 5.10(b) is fairly straightforward: the cheapest options are always preferred; only after their capacity is exhausted, more expensive PoAs are exploited. Figure 5.10(c), concerning computing nodes, shows instead a different situation. The intermediate solution, namely, aggregation rings, is preferred in most scenarios; edge servers are used for a limited amount of traffic, thanks to their low latency that allows using cheaper (albeit slower) PoAs. Cloud servers, thanks to their low cost and high capacity, are the preferred option when the number of robots grows, provided the target latency can be met.

Figure 5.10(d) depicts the amount of computing resources consumed in the different sections of

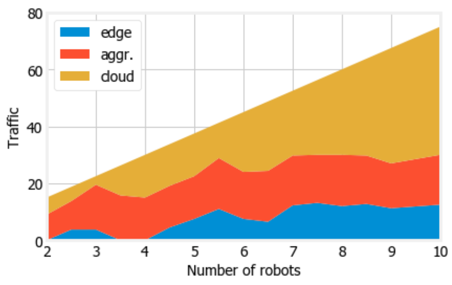
[8] Numerical simulations of the smart-factory scenario where conducted by Dr. Francesco Malandrino.



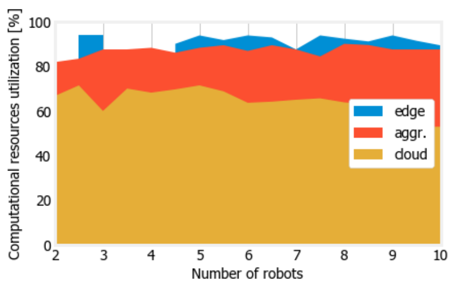
(a)



(b)



(c)



(d)

Figure 5.10: Smart factory service with a varying number of robots: cost breakdown (a), choice of PoAs (b) and computing nodes (c), usage of computing resources at different location in the network infrastructure (d).

the network infrastructure. Interestingly, Virtual Machine (VM)s closer to users (e.g., at the edge) are consistently used more than further-away ones (e.g., in the cloud). This is due to two reasons: first, more expensive computing nodes can be an optimal choice only if it is possible to fully utilize the VMs therein; second, faster processing times (hence, as specified in (5.2), more spare capacity) are required at farther-away nodes to make up for longer network delays.

5.1.5 Testbed and validation

[9]In the following it is presented the implementation of OKpi in an experimental testbed where the mobile robot, smart factory service (depicted in Figure 5.2(left)) is deployed in an indoor scenario.

The testbed consists of: (*i*) 5 ASUS WL500G Premium v1 APs running OpenWrt 18.06.2 [Fai08]; (*ii*) 2 MiniPC, with 4 vCPUs and 8GB of RAM each, one used as an AP and the latter as a local server (i.e., located close to the APs); (*iii*) 1 PowerEdge C6220 server with 94GB of RAM and 16 vCPUS, acting as edge server; and (*iv*) 1 PowerEdge R840 Rack Server7 with 94GB of RAM and 16 vCPUs, acting as cloud server. The six APs and the local server are deployed along two corridors of the Universidad Carlos III de Madrid building (see Figure 5.11), while the edge and cloud server are located in different buildings. To emulate different levels of link congestion, the experiments leveraged on `NetEm` [Hem+05] to artificially introduce some latency on the connection between the APs and the servers, as reported in Table 5.4. Note instead that the latency on the AP-robot link never exceeds 6 ms. The cost associated with the APs and servers match those presented in Table 5.3, considering the pico-cell value for the APs. Additionally, the experiments used ROS-compatible Kobuki Turtlebot S2 robot equipped with a laptop with 8-GB RAM and 2 vCPUs, and a RPLIDAR A2 lidar for 360-degree omnidirectional laser range scanning.

The laptop hosts the robot VNF, which, as mentioned in section 5.1.1, (a) probes the robot sensors (e.g., odometry, LIDAR), (b) transmits the sensors data to the ROS brain, and (c) executes

[9] In the testbed, Milan Groshev was encharged of the operation of the Kobuki robot, so as the software development related to RoS, the LADAR, and brain VNFs. The author of this thesis was responsible of the setup of the Asus OpenWrt routers with 802.11r, and the interaction of OKpi with the robot.

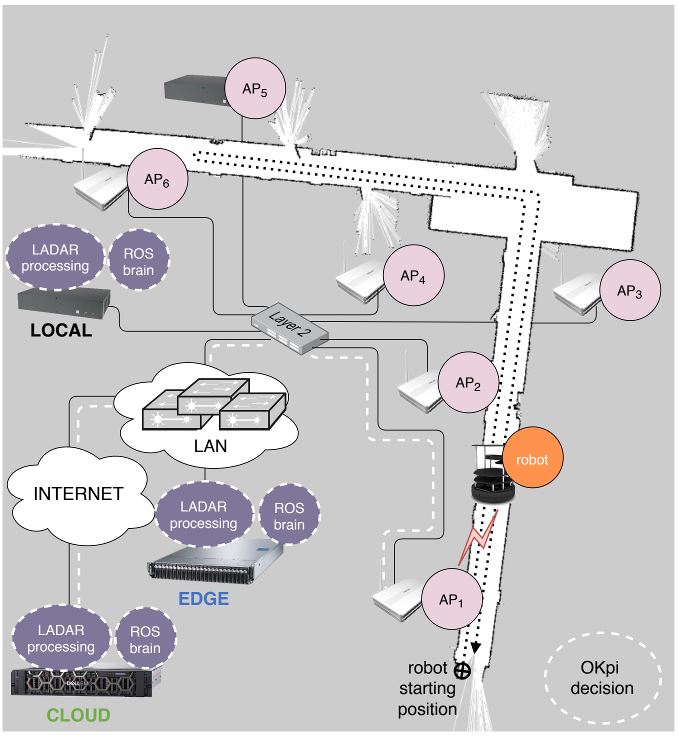


Figure 5.11: Illustration of the mobile robot, smart factory testbed: the robot starts at the bottom end of the corridor and comes back once it has reached the other end at the top left. Dashed circles highlight possible VNF deployments.

Table 5.4: AP-server latency

	cloud	**edge**	**local**
AP_1, AP_2	9 ms	4 ms	3 ms
AP_3, AP_4	18 ms	8 ms	9 ms
AP_5, AP_6	27 ms	12 ms	9 ms

the navigation instructions received from the ROS brain. The LADAR and ROS brain VNFs can be hosted at any of the available servers

The target of the experiment is to ensure that the one-way, End-to-End (E2E) latency of the service remains within 15 ms [3GP18b] during the robot's trip. The experiment starts with the robot positioned at the bottom end of the corridor and connected with AP_1 (see Figure 5.11). Also, the initial decision by OKpi is to deploy the ROS brain and LADAR VNFs in the cloud server. The ROS brain then navigates the robot along the trajectory shown in Figure 5.11 and, as the robot moves, OKpi determines which AP the robot should connect to[10] and which server (cloud, edge, local) should host the ROS brain and LADAR VNFs. Both the AP and server selection change depending on the robot position and latency of the AP, respectively. In particular, depending on which AP the robot is attached to, OKpi decides which server should host the ROS brain and LADAR VNFs by accounting for the latency values reported in Table 5.4, in such a way that the overall service latency remains below 15 ms. During the experiments, OKpi recomputed the AP selection, ROS brain, and LADAR VNFs embedding, in less than 1 s. The robot position is reported to OKpi through MQTT messages transmitted by the robot itself, while the APs coverage, which may vary over time, is acquired through 802.11r measurements.

Figure 5.12 compares the temporal behavior of the E2E latency obtained in the following cases:

– **SoA**, i.e., when the solution based on [Mal09] and [Keh+15] is used. In this case, the LADAR and ROS brain VNFs are always placed in the cloud and the robot connects to the

[10] As the robot moves, it roams to the selected AP using 802.11r Fast Transitioning [08].

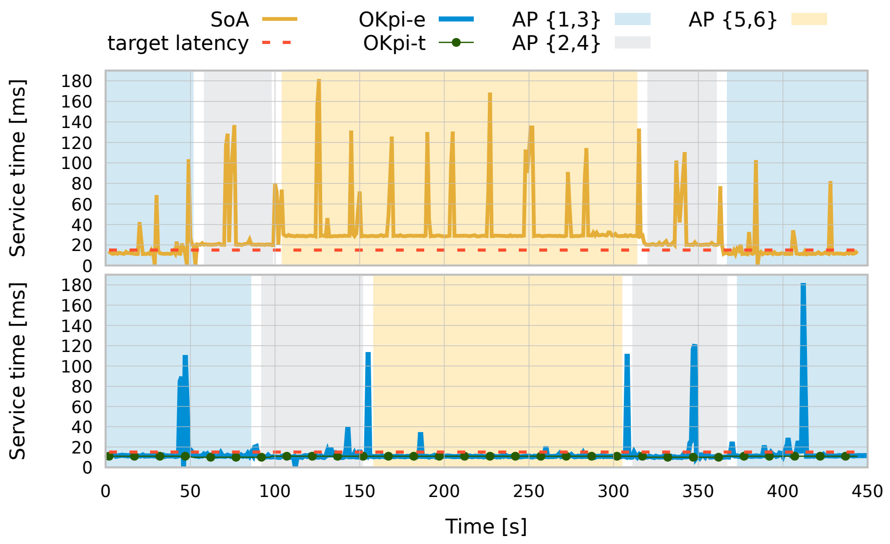


Figure 5.12: Service latency experienced during the robot's trip. Different background colors refer to time intervals in which the robot is connected to different APs.

Table 5.5: OKpi and SoA service latency

Solution	Average	Std. deviation	E2e violations
OKpi$_e$	13.63 ms	15.21 ms	7%
SoA	29.61 ms	26.18 ms.	74%

AP from which it receives the strongest signal (the latter information is acquired by the robot by performing Wi-Fi active scans every 20 s).

– **OKpi$_t$**, i.e., when OKpi makes decisions in a simulated environment that mimics the testbed, using the theoretical delays that each AP offers;

– **OKpi$_e$**, i.e., when OKpi operates in an experiment carried out through the developed testbed, using the robot's real time location and triggering 802.11r roam instructions when it leaves an AP's coverage area (note this prevents the robot from doing periodic scanning).

Figure 5.12 compares the service time measured during a single run execution of the SoA, OKpi$_e$, and OKpi$_t$ solutions.

At the beginning of the experiment, neither the SoA nor the OKpi$_e$ violate the target E2E latency of 15 ms, except for the peaks due to the robot Wi-Fi scans. While under SoA the robot performs a scan every 20 s, under OKpi$_e$ it does so only when it has to connect to a new AP, as per the OKpi decision. Under SoA, 50 s later the robot connects to AP$_3$, and the latency jumps above 20 ms because the LADAR and ROS brain VNFs are still running in the cloud server. In the case of OKpi$_e$, instead, when the robot connects to AP$_3$ at 89 s., the LADAR and ROS brain VNFs are moved to the edge server, so that the E2E latency remains below 15 ms. The same behavior is observed at time 102 s, when the robot connects to AP$_5$ in the case of SoA and the E2E latency increases up to 29 ms. On the contrary, OKpi$_e$ still meets the target e2e latency upon making the robot connecting with AP$_5$ (at time 156 s), since it now places the VNFs in the local server. In the rest of the time elapse, one can observe similar performance, as the robot returns to its initial position.

As a final remark, Figure 5.12 highlights that the performance of OKpi$_e$ is always close to the E2E latency exhibited by OKpi$_t$. Furthermore, the target E2E service latency (15 ms) was only violated the 7% of the times during the experiment by OKpi$_e$ (see Table 5.5), while it was violated the 74% of the times under SoA.

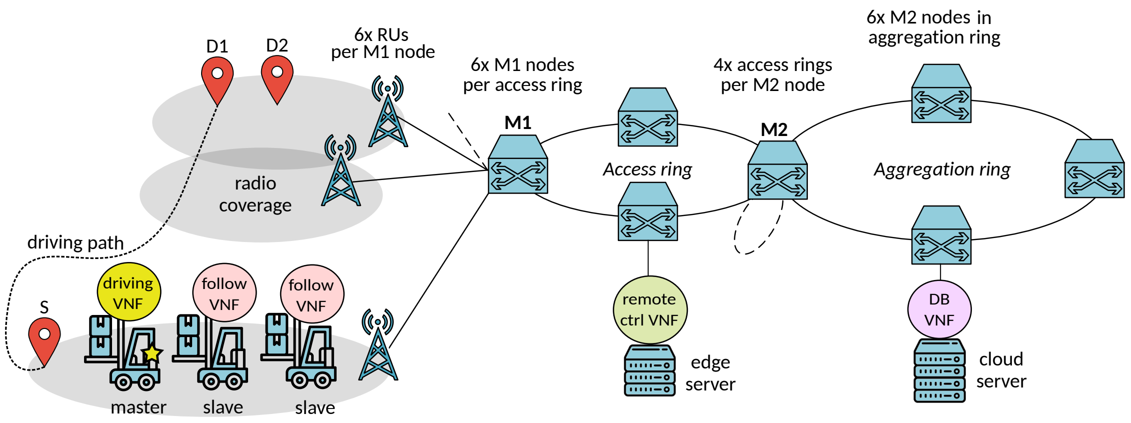


Figure 5.13: Deployment of a cloud robotics warehousing NS.

5.2 Mobile and volatile Fog networks

The previous section paid attention on meeting KPIs of the plethora of network services present in 5G networks. It put special emphasis on the traffic steering, and path selection to satisfy the related delay and reliability constraints. And although it tackles the coverage constraints of fog devices and end users, the approach is rather simple, as the system model does nothing but stating a binary variable to determine whether the end device lies within the coverage area or not. Moreover, mobility and battery constraints of fog devices where not considered.

This section studies how to solve the VNE problem for NSs that require from volatile fog devices. That is, devices that may belong or not to the infrastructure depending on whether they have battery. In particular, it focuses in a mobile robotics warehousing use case in Valencia city haven, where robots with wireless connectivity and computing capacity move the containers along the haven. Thus, the proposed solution and system model are designed for mobile devices within the fog, as robots or drones acting as an extension of the cloud and edge infrastructure.

The research contribution is threefold. First, the VNF placement problem is formulated as a cost-minimizing optimization problem. The section extends formulations in the state of the art imposing the radio coverage of mobile fog devices, and preventing that VNF deployments use fog devices that may run out of battery. Second, the optimization problem is solved by a novel heuristic algorithm that, to the best of the authors' knowledge, is the first one getting close to optimal results while tackling both radio coverage, and battery restrictions of fog environments. Finally, the proposed algorithms are evaluated via extensive simulations on a real-world scenario. The results confirm the beneficial properties of the heuristic and system model in terms of scalability, cost, and run-time.

The rest of the section is organized as follows. Section 5.2.1 introduces the warehousing use case motivating this section work. Section 5.2.2 is devoted to the detailed description of the associated model and optimization problem. Section 5.2.3 describes the proposed heuristic algorithm. And section 5.2.4 presents the algorithms' evaluation from different aspects based on extensive simulations.

5.2.1 Use Case: mobile robotics

This section tackles the mobile robotics use case [3GP18b, Table 5.3.1.1-1], as a warehousing solution for future factories [Mou+18, section 3.1.2]. In particular, it deals with the transport of goods from boats to specific locations of Valencia city haven.

The use case considers a cluster of robots, that move in a *master-slave* fashion to deliver goods arriving to the haven. Each of the robots carries containers from a pick up point (S in Figure 5.13) to a drop off point (D1 and D2). In particular, the *master* robot is followed by the other *slave* robots (represented in Figure 5.13) of the cluster along its way towards the drop off point. Robots communicate among themselves to report position status, or other context information useful for the *master-slave* coordination. Thus, robots have device-to-device communication between them, and computational capabilities so they can execute lightweight VNFs [Nog+18] as the *driving* and *follow* VNFs represented in Figure 5.13. The *driving VNF* runs in the *master* robot to drive it towards the drop off point, and the *follow VNFs* run in the *slave* robots to follow the master robot movements until it reaches a drop off point. The *driving VNF* receives driving instructions from the *remote ctrl VNF* running in the edge server in Figure 5.13, and reports sensor data like the speed to the *Database (DB) VNF* running in the cloud.

To enhance the robots' remote driving, the communication between the *remote ctrl VNF* and the *driving VNF* is crucial, indeed, resources proximity is needed as Mobile robotics demand communications with cycle times between 1ms and 100ms (for machine control, and video operated remote control cases) [3GP18b]. Thus, the placement of both *driving* and *remote ctrl VNFs* should satisfy latencies below 100ms.

While moving, robots may run out of battery or switch between Radio Unit (RU)s coverage area (see Figure 5.13). Whenever the *master* robot enters a new coverage area, it attaches to a new RU to keep the connectivity with the servers running the *remote driving VNF*, and *DB VNF* (edge, and cloud servers in Figure 5.13). Therefore, it is important to take into account that a robot is not selected for goods delivery if (*i*) it may run out of battery; or (*ii*) it may loose RU connectivity as it moves towards the drop off point.

To increase the RUs coverage and improve the E2E delay, the use case presented in this section considers that the haven is covered by Long Term Evolution (LTE) RUs managed by a network operator, and New Radio (NR) RUs belonging to its Non Public Network (NPN). This is called an NPN deployment in a public network [5GA19].

That is, Valencia city haven only owns the NR RUs, and its management (subscription, gateways, control plane) is done by the public network, i.e., a network operator.

For the public network infrastructure, a 5G transport network is assumed based on [ITU18] and [Com+18b]. As later specified in section 5.2.4, the considered infrastructures are generated as indicated in chapter 3, in particular, using the `5GEN` package – see section 3.5. All the RUs present in the use case transmit their traffic up to an access ring composed of several switches connected in a ring fashion. The traffic of the access rings is later gathered by the aggregation rings which forward traffic up to the core of the public infrastructure. The presented use case, assumes that cloud servers are in the core of the public network, edge servers are co-located next to the access ring and the aggregation ring switches. Regarding computational resources (i.e., CPU, memory and disk), edge servers in access rings are less powerful than edge servers in aggregation rings, and cloud servers are more powerful than edge servers.

It is worth highlighting that the problem formulation presented in this section will hold for public and private deployments, being the only consideration the cost of connection, that may vary depending on the type of management. Both deployments are mentioned for a better understanding on the real situation in the city haven.

5.2.2 Problem formulation

This section presents the formulation of the use case to tackle the VNF allocation as an optimization problem. The problem is solved using an integer program solver to gain optimality and scalability insights.

System model

The network infrastructure is represented by a graph G_I, where the nodes $V(G_I)$ contain NR and LTE RUs, generally referred to as APs $V_{AP}(G_I)$, server nodes (representing edge or cloud servers) $V_S(G_I)$, and mobile nodes $V_M(G_I)$. Hence, the vertex set of the graph is built up as $V(G_I) = V_{AP}(G_I) \cup V_S(G_I) \cup V_M(G_I)$. Host nodes N_i with computation capacities $\overline{C}_{N_i}$ are stored in $V_H(G_I) = V_S(G_I) \cup V_M(G_I)$, and their corresponding unitary price is represented by p_{N_i}. As a realistic generalization to the mobile robotics use case, the concurrent management of multiple robot clusters is assumed. The subsets of $V_M(G_I)$ define the clusters of robots $V_{RC_q}(G_I) \subseteq V_M(G_I), 1 \leq$

Table 5.6: Definition of variables and parameters

Notation		Definition
Parameters		
G_I		Network infrastructure graph
G_I	$V(G_I)$	All infrastructure nodes
	$V_*(G_I)$	Nodes of type $*$ in the infrastructure $* \in \{AP,\ S,\ M,\ H,\ RC_q\}$
	$E(G_I)$	Edges of the infrastructure
G_S		Network service graph
G_S	$V(G_S)$	All VNFs of the network service
	$\mathscr{P}(G_S)$	All paths of the service graph
	G_s	Graph of Service Function Chain (SFC) G_s
	$V(G_s)$	VNFs of SFC G_s
	$E(G_s)$	Edges of SFC path G_s
	Δ_{G_s}	Delay requirement of SFC G_s
	th^s_{bat}	Battery threshold for SFC G_s
	$\mathscr{C}_{SFC}$	Set of all SFCs
VNF v		Virtual Network Function v
VNF v	C_v	Capacity demand of VNF v
	L	Locality matrix $V(G_S) \times V(G_I)$
N_i	$\overline{C}_{N_i}$	Total resource capacity of node N_i
	p_{N_i}	Cost per resource unit used of node N_i
	$D_{AP,S}(N_i,N_j)$	Delay between N_i and $N_j \in V_{AP}(G_I) \cup V_S(G_I)$
	$D_{M_q}(N_i,N_j)$	Delay between N_i and $N_j \in V_{RC_q}(G_I)$
	$\mathbb{P}_{bat}(N_i,C_{N_i})$	Probability of having battery for the whole time interval using C_{N_i} resources
AP_k	d_{AP_k}	Delay for the coverage area of AP_k
	$\mathbb{P}_{AP^q_k}(t_u)$	Probability of cluster q to be in the coverage area of AP_k in time subinterval t_u
	p_{AP_k}	Cost of usage of AP_k
	κ_q	Coverage probability threshold for cluster q
Variables		
$d_{G_s}(t_u)$		Delay of SFC G_s in time t_u
$d(N_i,N_j,t_u)$		Delay between nodes N_i and N_j in time t_u
$x(v,N_i)$		Placement of VNF v in node N_i
C_{N_i}		Resource usage in node N_i
$AP^q_k(t_u)$		Usage of AP_k by cluster q time t_u
$\mu : V(G_S) \mapsto V_H(G_I)$		VNF to host node mapping structure
$\alpha : \{t_u\} \times \{q\} \mapsto V_{AP}(G_I)$		AP selection structure for all clusters

$q \leq Q$, where Q refers to the number of clusters. Moreover, graph edges $E(G_I)$ represent the connections between the infrastructure nodes, which are annotated by their transmission delays. Due to the mobile clusters' mobility, their connections to the static part of the infrastructure are not represented by edges in G_I.

The mobile nodes $V_M(G_I)$ are connected to access points $V_{AP}(G_I)$ in order to communicate with other nodes of the infrastructure. However, the nodes are moving and may encounter areas with overlapping access point coverage or areas where handover between different access points is needed to guarantee the connection to the servers deeper in the infrastructure. Thus, this section assumes that each AP has an associated coverage area AP_k, and the mobility pattern of robot cluster q is modeled by the probability distribution of being in the AP coverage areas $\mathbb{P}_{AP^q_k}(t)$, referred as *coverage probability* throughout the paper. Notice that a cluster can be in an area where several access points have coverage, with a different probability for each of them. Each value models the probability of a robot cluster q to fall inside the coverage area of each AP in each moment t. This model is able to compute the placement of NSs with guarantees of communication between the mobile and fixed parts of the infrastructure, while considering any model of coverage areas, such as [SMF15b] or a linear model, by using precomputed values of the coverage. The parameter t is a time instant within an interval (t_0, t_1) in which the network service will be running. For the sake of simplicity in the model, the time interval is discretized in subintervals, thus continuous time $t \in (t_0, t_1)$ becomes discrete time $t_u \in \{t_0, t_a, t_b \ldots, t_1\}$ with $t_0 \leq t_a \leq t_b \leq \ldots \leq t_1$. Subintervals help to identify the moments when handovers may occur during the service time. The time division guarantees the communication between robots and APs selected in each subinterval. Note that VNFs are deployed on the same servers during all the service time, thus, they must have communication with the APs selected in each subinterval.

The cost of using an AP for a single subinterval t_u by any single cluster is p_{AP_k}. The energy consumption of the mobile nodes is modeled by the distribution $\mathbb{P}_{bat}(N_i, C_{N_i})$ depending on the allocated load to node N_i, which represents the probability of having a not depleted battery for the whole interval (t_0, t_1). Both $\mathbb{P}_{AP_k^q}(t)$ and $\mathbb{P}_{bat}(N_i, C_{N_i})$, are used in the optimization problem to ensure robots' radio coverage, and battery needs are met during the interval (t_0, t_1).

The requested Network Services are represented with a NS graph G_S, with the nodes being VNFs $v \in V(G_S)$ and their capacity requirements C_v. Each SFC is a subgraph $G_s \subseteq G_S$ with its own set of VNFs and path, as the one depicted in the NS graph of Fig. 5.13, and expressed in Eq. (5.13).

$$\begin{aligned}\mathscr{C}_{SFC} = \big\{(G_s, \Delta_{G_s}) \mid\ & V(G_s) \subseteq V(G_S), \\ & E(G_s) \in \mathscr{P}(G_S), \Delta_{G_s} \in \mathbb{R}^+\big\}\end{aligned} \tag{5.13}$$

where $\mathscr{C}_{SFC}$ represents the set of SFCs in Network Service G_S, and $\mathscr{P}(G_S)$ represents the paths of the NS graph G_S. Each SFC has a corresponding delay requirement Δ_{G_s} which defines an upper bound of the total delay of the SFC path $E(G_s)$.

For a better understanding of the model, all the notations used for the mathematical formulation of the optimization problem are gathered in Table 5.6.

Optimization problem

$$d(N_i, N_j, t_u) = \begin{cases} \sum\limits_{AP_k \in V_{AP}(G_I)} AP_k^q(t_u)[D_{M_q}(N_i, N^{(r_q)}) + d_{AP_k} + D_{AP,S}(AP_k, N_j)], & \text{if } N_i \in V_{RC_q}(G_I) \wedge N_j \in V_S(G_I); \\ d(N_j, N_i, t_u), & \text{if } N_j \in V_{RC_q}(G_I) \wedge N_i \in V_S(G_I); \\ D_{AP,S}(N_i, N_j), & \text{if } N_i, N_j \in V_S(G_I) \cup V_{AP}(G_I); \\ D_{M_q}(N_i, N_j), & \text{if } N_i, N_j \in V_{RC_q}(G_I); \\ \sum\limits_{\substack{AP_{k_1}, AP_{k_2} \\ \in V_{AP}(G_I)}} AP_{k_1}^{q_i}(t_u) AP_{k_2}^{q_j}(t_u) \left(D_{AP,S}(AP_{k_1}, AP_{k_2}) + d_{AP_{k_1}} + d_{AP_{k_2}} + \sum\limits_{n \in \{i,j\}} D_{M_{q_n}}(N_n, N^{(r_{qn})}) \right), & \text{if } N_i \in V_{RC_{q_i}}(G_I) \wedge\ N_j \in V_{RC_{q_j}}(G_I) \end{cases} \tag{5.26}$$

Formulation 1 summarizes the associated optimization problem, and details are described below. The optimization must decide which infrastructure node $N_i \in V(G_I)$ should host which VNF $v \in V(G_S)$, this is represented by the binary decision variable $x(v, N_i)$ and constraints Eq. (5.14) and Eq. (5.15). The resource capacities $\overline{C}_{N_i}$ must be respected by the load allocation on each node N_i. This requirement is gathered in Eq. (5.16), where C_{N_i} stands for the allocated resources in infrastructure node N_i as presented in Eq. (5.17).

Furthermore, there may be a necessity of applying placement policies and VNF functional types. In order to include those policies in the model, the matrix $L(v, N_i)$ expresses locality constraints between the VNFs $v \in V(G_S)$ and infrastructure node $N_i \in V(G_I)$. Each element of the matrix is a binary constant, identifying whether the VNF can be located in an infrastructure node, as expressed in Eq. (5.18). In the use case presented in section 5.2.1, $L(v, N_i)$ enforces the deployment of the *driving* and *follow VNFs* in the robots (i.e., mobile nodes). This requirement may be useful for other use cases, such as Unmaned Aerial Vehicle (UAV)s running virtual access points that forward

Formulation 1 Optimization problem

$$x(v,N_i) \in \{0,1\} \qquad \forall v \in V(G_S), \forall N_i \in V(G_I) \tag{5.14}$$

$$\sum_{N_i \in V(G_I)} x(v,N_i) = 1 \quad \forall v \in V(G_S) \tag{5.15}$$

$$C_{N_i} \leq \overline{C}_{N_i}, \quad \forall N_i \in V(G_I) \tag{5.16}$$

$$C_{N_i} = \sum_{v \in V_S(G_I)} x(v,N_i) C_v, \quad \forall N_i \in V(G_I) \tag{5.17}$$

$$x(v,N_i) \leq L(v,N_i), \quad \forall v \in V(G_S), \forall N_i \in V(G_I) \tag{5.18}$$

$$\sum_{AP_k \in V_{AP}(G_I)} AP_k^q(t_u) = 1,\ \forall 1 \leq q \leq Q, \forall t_u \in (t_0,t_1) \tag{5.19}$$

$$\sum_{AP_k \in V_{AP}(G_I)} AP_k^q(t_u) \cdot \mathbb{P}_{AP_k^q}(t_u) \geq \kappa_q, \quad \substack{\forall 1 \leq q \leq Q \\ t_u \in (t_0,t_1)} \tag{5.20}$$

delay equation is presented in Eq. (5.26)

$$d_{G_s}(t_u) = \sum_{\substack{(v_i,v_j) \in E(G_s) \\ N_i,N_j \in V(G_I)}} x(v_i,N_i) x(v_j,N_j) d(N_i,N_j,t_u) \tag{5.21}$$

$$d_{G_s}(t_u) \leq \Delta_{G_s},\ \forall (G_s,\Delta_{G_s}) \in \mathscr{C}_{SFC}, \forall t_u \in (t_0,t_1) \tag{5.22}$$

$$\begin{aligned} \mathbb{P}_{bat}(N_i,C_{N_i}) &= \mathbb{P}_{bat}(N_i,0) - \\ -\frac{C_{N_i}}{\overline{C}_{N_i}} \left(\mathbb{P}_{bat}(N_i,0) - \mathbb{P}_{bat}(N_i,\overline{C}_{N_i})\right)&, \forall N_i \in V_M(G_I) \end{aligned} \tag{5.23}$$

$$\mathbb{P}_{bat}(N_i,C_{N_i}) \geq th_{bat}^s,\ \forall N_i \in V_M(G_I),\ \forall G_s \in \mathscr{C}_{SFC} \tag{5.24}$$

$$\min \sum_{N_i \in V(G_I)} C_{N_i} \cdot p_{N_i} + \sum_{t_u,q,k} AP_k^q(t_u) \cdot p_{AP_k} \tag{5.25}$$

traffic to the cloud (see [Nog+18, San+20]). Under such scenarios, $L(v,N_i)$ can be used to enforce virtual access points to run on top of UAVs.

1. **Radio coverage constraints:** The deployment must also decide at each time interval to which access point each cluster of robots is attached to, that is, $AP_k^q(t_u) = 1$ in case robot cluster RC_q is connected to access point AP_k at time t_u. (5.19) reflects the assumption that each cluster can only be attached to one AP at each interval. The deployment decision must also ensure that the coverage probability is above the imposed threshold κ_q for mobile cluster q, representing the requirements each cluster needs to guarantee connectivity during the time interval, as stated in (5.20). Notice that Optimization problem 1 only needs to know whether cluster q has radio coverage of AP_k at time t_u. Hence, Optimization problem 1 is agnostic about how $\mathbb{P}_{AP_k^q}(t_u)$ is obtained, and the values could be derived from any radio access model. For instance, Sec. 5.2.4 obtains $\mathbb{P}_{AP_k^q}(t_u)$ with a linear function directly proportional to the distance between q and AP_k.
2. **Delay constraints:**[11] in order to measure the distances between infrastructure nodes, the metric used is the delay, which in the case of the static nodes is given in a matrix containing the precomputed and the time-independent delays, $D_{AP,S}(N_i,N_i)\ \forall N_i,N_j \in V_S(G_I) \cup V_{AP}(G_I)$. Similarly, the distances inside each mobile cluster are time invariant, precalculated and stored

[11] The delay constraints, in particular (5.26), were proposed by Nuria Molner.

in matrix $D_{M_q}(M_i,M_j)\ \forall M_i,M_j \in V_{RC_q}(G_I), 1 \leq q \leq Q$.
Each access point $AP_k \in V_{AP}(G_I)$ provides a time- and distance-independent delay to its whole coverage area, its value is denoted by d_{AP_k}, while delay between APs is given with the value, $D_{AP,S}(AP_{k_1},AP_{k_2})$. The delay value between a mobile cluster and the static part of the infrastructure And between mobile nodes belonging to two different clusters Might vary according to the assigned APs during the time interval (t_0,t_1). A mobile cluster q has an appointed relay node $N^{(r_q)}$ (in our case the *master* robot), which is connected to the APs, and all the traffic of other mobile nodes of the same cluster towards the fixed part of the infrastructure goes through the corresponding relay mobile node. Thus, the orchestration system can execute the handover of the cluster by only connecting the relay node to a different AP. This way the delay of device-to-device communication is accounted in a different variable than the AP delays. Hence, the general delay function which covers any pair of infrastructure node types is expressed in (5.26). (5.26) is a piece-wise function that depends on the type of node hosting the different VNFs of the NS. The delay between two mobile nodes of the same cluster is accounted in $D_{M_q}(N_i,N_j)$. Delay between two nodes of the fixed infrastructure is $D_{AP,S}(N_i,N_j)$, while delay between a node of the fixed infrastructure and a mobile cluster depends on the AP that the cluster uses in that moment, which is gathered in
$\sum_{AP_k \in V_{AP}(G_I)} AP_k^q(t_u)[D_{M_q}(N_i,N^{(r_q)}) + d_{AP_k} + D_{AP,S}(AP_k,N_j)]$. Thus, the delay of a service is composed by the different delays between the nodes that host the different VNFs and the order in which they must be performed.
The overall delay of a SFC $G_s \in \mathscr{C}_{SFC}$ in time t_u is formulated in (5.21), where the delays between the hosts of each SFC edge are summed. The upper bound of the SFCs' total permitted delay Δ_{G_s} for the whole optimization interval is expressed in constraint (5.22).

3. **Battery constraints:** In order to place VNFs in mobile nodes it is necessary to ensure the mobile node will not run out of battery during the time interval (t_0,t_1). This is introduced in the problem formulation, in (5.23), as the probability of having battery for the whole time interval considered, based on the resources used in the node. C_{N_i} is the consumed capacity of mobile node N_i, and $\mathbb{P}_{bat}(N_i,C_{N_i})$ is the probability of having battery on N_i by the end of time interval (t_0,t_1) when using C_{N_i} resources as allocated capacity. Note that Optimization problem 1 is agnostic of the used battery consumption model, as $\mathbb{P}_{bat}(N_i,C_{N_i})$ values could be derived by any battery consumption model. For example, Sec. 5.2.4 derives $\mathbb{P}_{bat}(N_i,C_{N_i})$ as a linear function between the empty $C_{N_i}=0$ and the fully loaded states $C_{N_i}=\overline{C}_{N_i}$. To ensure the proper performance of the mobile nodes, the battery life is guaranteed in (5.24) by a threshold th_{bat}^s given per SFC G_s, for all nodes hosting VNFs. This threshold takes into account the battery of all the mobile nodes hosting the VNFs of the service and guarantees each of the nodes hosting a VNF of the service will have battery during the whole time interval with a probability higher than the threshold, for example a $th_{bat}^s = 0.9$.
4. **Cost minimization:** finally, the problem minimizes the total cost of allocating the whole service G_S demanded and AP usages by all of the mobile clusters. Hence, the objective function is shown in (5.25). The VNF mapping μ and AP selection structures α are defined by the variables $x(v,N_i)$ and $AP_k^q(t_u)$ of a solution to the optimization problem. This model is not linear in some equations as the one representing the delay in (5.21), but each product of two variables can be easily linearized due to the fact that all the variables involved are binary variables. Thus, the linearization is performed by substituting each product of two binary variables by one extra binary variable, as expressed in Property 5.7.

Property 5.7: Linearization of the product of two binary variables. Let $z = x \cdot y$, where z, x and y are binary, then the product can be linearized as follows:

$$\begin{aligned} llz &\leq x, \\ z &\leq y, \\ z &\geq x+y-1 \end{aligned}$$

Formulation 2 Bin Packing with Usage Cost [Cam+13]
Input: VNFs $V(G_S)$ as items with weight, host nodes $V_H(G_I)$ as bins with capacity
Output: VNF placement respecting only capacity constraints

$$ll \sum_{N_i \in V_H(G_I)} x(v,N_i) = 1 \quad \forall v \in V(G_S) \tag{5.27}$$

$$\sum_{v \in V(G_S)} x(v,N_i)C_v \leq \overline{C}_{N_i} \quad \forall N_i \in V_H(G_I) \tag{5.28}$$

$$x(v,N_i) \in \{0,1\} \quad \forall v \in V(G_S), N_i \in V_H(G_I) \tag{5.29}$$

$$min \sum_{N_i \in V_H(G_I)} C_{N_i} \cdot p_{N_i} \tag{5.30}$$

5.2.3 Heuristic

This section details[12] the design of the heuristic which exploits the peculiarities of the system model to design an efficient and practical algorithm.

Proposed heuristic

The core idea of our heuristic algorithm is to use the fractional optimal solution of a bin packing problem of the VNFs and host nodes, which is deterministically rounded to an invalid integer solution. Next, the algorithm iteratively resolves the capacity, delay, battery and coverage constraint violations by changing the mapping location of VNFs in the initial invalid integer solution until a feasible mapping is found.

First of all, lets introduce the bin packing problem variation with variable bin and item sizes supporting linear usage costs [Cam+13] in Formulation 2. Lemma 5.8 (taken from [Cam+13] and pasted down for readability) states how to construct a fractional optimal solution for this bin packing variant, relaxing the integrality constraint. The proof of Lemma 5.8 can be found in the original source [Cam+13].

Lemma 5.8: Fractional optimal solution of Formulation 2 [Cam+13]. Let $\{a_i\}$ be a permutation of all host infrastructure nodes $N_i \in V_H(G_I)$ in ascending order by their unit costs of computation capacity $p_{a_1} \leq p_{a_2} \leq \cdots \leq p_{a_{|V_H(G_I)|}}$. Let $W_C = \sum_{v \in V(G_S)} C_v$ be the sum of all VNF capacities. Let b be the minimum number of host nodes in order $\{a_i\}$ where $\sum_{i=1}^{b} \overline{C}_{N_{a_i}} \geq W_C$.

The fractional optimal solution (discarding the integrality constraint (5.29)) of Formulation 2 is

$$\tilde{x}(v,N_{a_i}) = \begin{cases} \frac{\overline{C}_{N_{a_i}}}{W_C} & \text{if} \quad i < b, \\ \frac{W_C - \sum_{i=1}^{b-1} \overline{C}_{N_{a_i}}}{W_C} & \text{if} \quad i = b, \\ 0 & \text{if} \quad i > b; \end{cases} \quad \forall v \in V(G_S).$$

The proposed heuristic's core pseudo-code is shown in Algorithm 7. Intuitively, the heuristic reallocates VNFs that violate any constraint, and measures the goodness of the reallocation with the *improvement score* (see Algorithm 9). The higher the *improvement score*, the better the VNF reallocation. Initially, the fractional optimal solution is retrieved and rounded to initial constraint-violating VNF placement, obeying only the locality constraints (5.18) as shown in lines 1-5. The cost increasing order $\{a_i\}$ of mobile and server nodes are used from Lemma 5.8 to involve

[12] The design of the algorithm proposed in this section was done by Dr. Balázs Németh

additional hosts to the VNF placement pool, starting only from the first b cheapest hosts. In each iteration a set of violating items, respecting all constraints is calculated based on the temporary decisions stored in the current VNF placement function μ. Next, the iteration in lines 11-19 collects improvement scores for moving a VNF which is involved in any constraint violation to any currently considered host node (i.e. until index b'). Line 14 heuristically filters only the VNF relocations whose improvement score is higher than a configured improvement score limit Υ. The improvement cost is calculated by the cost difference of VNF v on the current host $\mu(v)$ and the possible new host N_{a_i}. If any allowed VNF replacement is found, update actions are taken and a current AP selection α is retrieved as shown in lines 21-23. Otherwise, the algorithm exits the improvement operations, and the next cheapest mobile or server node is included in the search by increasing b'. If a feasible solution is found after any inner iteration (see line 30), the procedure returns the current VNF placement function μ and AP selection structure α. The presented algorithm could be easily extended to continue searching for better quality solutions at the price of increased running time.

All subsequently presented subroutines take the input of Algorithm 7, but these are omitted from the pseudo-codes for readability. VIOLATINGVNFMAPPINGS takes as input the current VNF placement function μ and returns a set of violating VNFs $\mathscr{V}$ and an information storage of the actual constraint violations $\mathscr{R}$. Based on the current VNF placement μ, the feasibility of AP selection for each robot cluster $q \in \{1 \dots Q\}$ is checked using the subroutine CHOOSEAPS. If the AP selection is not possible, all VNFs of the causing SFC G_s are added to $\mathscr{V}$ and the violation information is stored in constraint violation record $\mathscr{R}$.

Algorithm 8 shows the details of how the AP selection and its feasibility based on the placement function μ are derived for a given robot cluster $q \in \{1 \dots Q\}$ for all temporal subintervals. Line 2 chooses the affected SFCs G_s, which have any VNF mapped to the mobile nodes of the robot cluster q. Given the current VNF placement μ, the total delay used by the path of the whole SFC $E(G_s)$ can be calculated using the delay expression (5.26). Access points are chosen by discarding the ones which do not meet the coverage requirement κ_q and finding the one with minimal delay among the remaining ones:

$$AP_l = \mathrm{argmin}_{AP_k \in V_{AP}(G_I) \cap \{AP_\phi : \mathbb{P}_{AP_\phi^q}(t_u) \geq \kappa_q\}} (d_{AP_k}) \tag{5.31}$$

These operations are done by the function DELAYDISTWITHCOVERAGEANDAPSELECTION, which also ensures that the same AP is chosen for a given input robot cluster $q \in \{1 \dots Q\}$ in subinterval t_u, no matter which input SFC it gets. The algorithm discards the impractical option of placing the VNFs of a single SFC to distinct mobile clusters. This simplification is only applied for the delay bounded VNFs, not to the other VNFs of the network service G_S. If an access point AP_l is found for subinterval t_u with the given requirements, the selection is saved in AP selection function α, otherwise the structure is invalidated and the reason is saved in $\mathscr{R}^{AP}$, as shown by the logical structure starting at line 4. In case the computation capacities of a robot cluster are not used by any VNFs of any SFC, an access point still needs to be selected for the cluster, which is done by minimizing the cost instead of the unbounded delay and similarly filtering to the coverage probability (see line 10).

Finally, the improvement score calculation is shown in Algorithm 9, which takes the current VNF placement μ and a possible relocation of VNF v to N_{a_i} as input, and outputs an integer whose higher value represents a more significant improvement. The IMPROVESCORE procedure uses the previously presented VIOLATINGVNFMAPPINGS function to evaluate how the mapping would change by the VNF mapping modification. The mapping structure μ with less violating constraints is considered better, as shown in lines modifying the improvement score y. In case of capacity constraints, total improvement score y would decrease, keep unchanged or increase if the number of hosts with more than their max capacity allocated would increase, stay or decrease by the VNF movement, respectively (see line 3). A similar score modification is done for each SFC, using the change in the number of temporal subintervals t_u where the coverage or delay constraints are violated as shown by the iteration starting at line 4. In the case of the battery constraints, the number

Algorithm 7: PlaceVNFsSelectAPs(G_S, G_I, Υ)

```
Data: service graph $G_S$, infrastructure $G_I$, improvement score limit $\Upsilon$, and all constraints
      from Sec. 5.2.2
Result: VNF placement $\mu : V(G_S) \mapsto V_H(G_I)$ and AP selection
        $\alpha : \{t_u\} \times \{1 \dots Q\} \mapsto V_{AP}(G_I)$ satisfying all constraints
1  $\tilde{x}(v,N_i), b, \{a_i\} \leftarrow$ fractional solution based on Lemma 5.8 for host nodes $V_H(G_I)$ and
    VNFs $V(G_S)$;
2  foreach $v \in V(G_S)$; // Round initial solution
3   do
4      $\mu(v) \leftarrow argmax_{N_i \in V_H(G_I)} \tilde{x}(v,N_i)$ which obeys;
       locality constraints (5.18)
5  end
6  foreach $b' \in \{b \dots |V_H(G_I)|\}$ ; // In order of $\{a_i\}$
7   do
8      $\mathscr{V}, \mathscr{R} \leftarrow$ VIOLATINGVNFMAPPINS($\mu$);
9      while $\mathscr{V} \neq \emptyset$ do
10         $\mathscr{I} \leftarrow \emptyset$ ; // Allowed improving VNF moves
11         foreach $v \in \mathscr{V}$, $i \in \{1 \dots b'\}$ do
12             if $\mu(v) \neq N_{a_i}$ and $\mu(v) = N_{a_i}$ obeys
13                             locality constraints (5.18) then
14                 if $\Upsilon \leq$ IMPROVESCORE($\mu$, $v$, $N_{a_i}$) then
15                     $impr\_cost \leftarrow C_v(p_{N_{a_i}} - p_{\mu(v)})$;
16                     $\mathscr{I} \leftarrow \mathscr{I} \cup \{(v, N_{a_i}, impr\_cost)\}$;
17                 end
18             end
19         end
20         if $\mathscr{I} \neq \emptyset$ then
21             $\mu(v) \leftarrow N_{a_i} \mid (v, N_{a_i}, impr\_cost) \in \mathscr{I}$ and;
               $impr\_cost$ is minimal
22             $\mathscr{V}, \mathscr{R} \leftarrow$ VIOLATINGVNFMAPPINS($\mu$);
23             $\alpha \leftarrow$ retrieve AP selection from violation;
               record $\mathscr{R}$
24         else
25             break;
26         end
27     end
28     if AP selection $\alpha$ is valid and
29                     VNF placement $\mu$ is valid then
30         return $\mu$, $\alpha$ // Solution found
31     end
32 end
33 return $\emptyset, \emptyset$ // Solution not found
```

of VNFs mapped to mobile nodes with violated battery thresholds are used.

Complexity analysis

A brief analysis on the heuristic's complexity and its termination is presented in Theorem 5.9 and its corresponding proof.

Algorithm 8: chooseAPs(μ, α, q)

```
Data: Current VNF placemnet $\mu$, current (possibly incomplete or invalid) AP selection $\alpha$,
      robot cluster index $q$
Result: Extended and/or ivalidated AP selection $\alpha$, AP selection violation record $\mathscr{R}^{AP}$
1  foreach $t_u \in (t_0, t_1)$ do
2      if $\exists G_s \in \mathscr{C}_{SFC}, \exists v \in V(G_s)$, where $\mu(v) \in V_{RC_q}(G_I)$ then
3          $d^{G_s}, AP_l \leftarrow$ DELAYDISTWITHCOVERAGEANDAPSELECTION($E(G_s), \mu, t_u, q, \kappa_q$);
4          if $d^{G_s} \leq \Delta_{G_s}$ and $\exists AP_l$ then
5              Let $\alpha(t_u, q) = AP_l$ // Same AP for all SFCs
6          else
7              Let $\alpha(t_u, q) = \lceil$;
8              Add result $d^{G_s}$ and SFC $G_s$ to $\mathscr{R}^{AP}$;
9          end
10         Let $\alpha(t_u, q) = AP_l$, where $AP_l \in V_{AP}(G_I)$ and obeys coverage constraint (5.20) and
             $p_{AP_l}$ is minimal;
11     end
12 end
13 return $\alpha$, $\mathscr{R}^{AP}$;
```

Algorithm 9: improveScore(μ, v, N_{a_i})

```
Data: Current VNF placement $\mu$, movement of VNF $v$ to host $N_{a_i}$
Result: Integer in interval $[-|\mathscr{C}_{SFC}| - 2, |\mathscr{C}_{SFC}| + 2]$, the improvement score of the VNF
        movement
1  $y \leftarrow 0$ // Init. improvement score of moving $v$ to $N_{a_i}$
     $\mathscr{V}, \mathscr{R} \leftarrow$VIOLATINGVNFMAPPINGS($\mu$);
2  $\mathscr{V}', \mathscr{R}' \leftarrow$VIOLATINGVNFMAPPINGS($\mu \mid \mu(v) = N_{a_i}$);
3  $y \leftarrow y - 1/+0/+1$ if number of hosts $N_i$ with violated constraint (5.16)
     increases/stays/decreases in $\mathscr{R}'$ compared to $\mathscr{R}$;
4  foreach $G_s \in \mathscr{C}_{SFC}$ do
5      $y \leftarrow y - 1/+0/+1$ if number of subintervals $t_u$ with any invalid mappings (i.e. where
         $\exists t_u, q : \alpha(t_u, q) = \lceil$ ) increases/stays/decreases in $\mathscr{R}'$ compared to $\mathscr{R}$;
6  end
7  $y \leftarrow y - 1/+0/+1$ if number of VNFs $v$ which are mapped to any mobile node $V_M(G_I)$
     with violated battery constraint (5.24) increases/stays/decreases in $\mathscr{V}'$ compared to $\mathscr{V}$;
8  return $y$;
```

Theorem 5.9: Complexity of heuristic. The overall complexity of the heuristic with positive improvement score limit $\Upsilon > 0$ is:

$$\mathscr{O}\Big(|V(G_S)|^4 |V(G_I)|^3 |\mathscr{C}_{\mathscr{S}\mathscr{F}\mathscr{C}}| QT\Big) \tag{5.32}$$

where Q and T are the number of clusters and the number of subintervals t_u in the optimization time frame (t_0, t_1), respectively.

Proof. Looking at Algorithm 7, the fractional solution construction and its rounding are dominated by the iteration starting at line 7, which is executed at most $|V(G_I)|$ times. Assuming a positive improvement score limit Υ, the violating VNFs set $\mathscr{V} = \mathscr{O}(|V(G_S)|)$ decreases at least by one element in each iteration of the *while* cycle. At most every iteration runs VIOLATINGVNFMAPPINGS. Filtering for the allowed VNF movements in line 13 is done at most $\mathscr{O}(|V(G_S)||V(G_I)|)$ times,

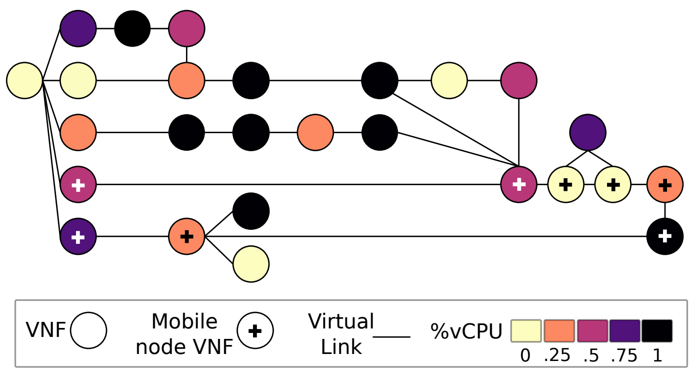


Figure 5.14: A service graph generated with a series-parallel graph. This instance contains x8 VNFs bounded to mobile nodes, and is used in the battery experiment, see Fig. 5.16

and in worst case for each of them we execute a IMPROVESCORE subroutine call. These observations make Algorithm 7's complexity to be $\mathcal{O}\Big(|V(G_S)||V(G_I)|\{|V(G_S)||V(G_I)|\text{IMPROVSCORE}$ $+\text{ VIOLATINGVNFMAPPINGS}\}\Big)$. The VIOLATINGVNFMAPPINGS's complexity is dominated by $Q\mathcal{O}(\text{CHOOSEAPS})$, because the other constraints can be checked in $\mathcal{O}(|V(G_I)||V(G_S)|)$ time. Access point filtering for sufficient coverage in a longest SFC can be done in $\mathcal{O}(|V(G_I)||V(G_S)|)$ time, which is done for all SFCs $|\mathcal{C}_{SFC}|$, all SFC edges $\mathcal{O}(|V(G_S)|)$ for all time subintervals T. Which gives $\mathcal{O}(\text{VIOLATINGVNFMAPPINGS}) = \mathcal{O}(QT|V(G_S)^2||V(G_I)||\mathcal{C}_{SFC}|)$. Similarly, IMPROVSCORE is dominated by VIOLATINGVNFMAPPINGS's complexity. Finally, a Floyd-Warshall algorithm is used to pre-calculate the all the delay matrices $D_{AP,S}$ and D_{M_q} with complexity $\mathcal{O}(|V(G_I)|^3)$, which is dominated by the previous operations. Substituting and ordering the $\mathcal{O}(\cdot)$ notations, the statement follows. ■

5.2.4 Evaluation and results

[13]This section compares the performance of Sec. 5.2.3 heuristic, with the optimal solution of Sec. 5.2.2 formulation from various aspects. As integer programs are generally impractical due to the hardness of the problem, the proposed heuristic is extensively evaluated to demonstrate its applicability. The heuristic solutions are compared to the optimal solution obtained with Gurobi which finds a solution within a gap optimality of 3%. Such comparison is done for the mobile robotics use case of Sec. 5.2.1, where scalability is a critical issue due to the size of the infrastructure and service graphs.

Additionally, this section compares Sec. 5.2.3 heuristic against "Follow Me Chain" (FMC) [CL19], a heuristic that tackles mobility by triggering VNF migrations upon AP handovers, but does not consider battery constraints. The implementation of FMC:

1. replaces [CL19, Algorithm 1] VNF-based Breadth-First Search (BFS) with a virtual-link-based BFS, so as to ensure the mapping of every virtual link;
2. uses a k-shortest paths in [CL19, Algorithm 2: line 1] to avoid getting stucked in the search of all paths between two nodes[14];
3. considers mobile compute nodes as well as edge servers; and
4. can map service graphs with unconnected components.

[13] Dr. Balázs Németh did the experiment implementation and design of the experiments, so as the scripts to automate the interaction with AMPL. The AMPL model encoding was done by Dr. Balázs Németh, Nuria Molner, and the author of this thesis. The generation of the network infrastructure graphs, so as the implementation and integration of the FMC, was done by the author of this thesis.

[14] FMC builds a full-mesh servers' graph, and even the proposed range-based Depth-Fist Search (DFS) incurs into a $\mathcal{O}(|V(G_i)|!)$ search space

Experiment setup

The presented evaluation scenario scales up the mobile robotics use case of Valencia city haven. A realistic 5G network infrastructure topology is considered with multiple types of wireless access points, while the service graph instances are random graphs. In particular, the infrastructure topology is generated using the 5GEN package presented in section 3.5, and following the 5G network infrastructures discussed in chapter 3. Many parameters of the experiment setting are examined during the presented simulations, varying the size of the input, SFC delay requirements, coverage probabilities and battery thresholds.

In order to generalize the service graphs and gain confidence in our simulations, series-parallel graphs are used to generate the network service topology G_S. Fig. 5.14 shows an example of such graph. This graph class covers the structure of many data streaming applications, such as map-reduce topologies, and have been used in other realistic, industrial case studies for fog application allocation [Sut+19]. The round-trip time experienced by every robot running a VNF must stay below the delay restriction, therefore, SFCs correspond to loops starting and ending in mobile node VNFs, and every SFC must satisfy the delay restriction. Among all the VNFs of the SFC, some of them are forced to run on top of the mobile robots' hardware $V_M(G_I)$ (denoted as *Mobile node VNF* in Fig. 5.14), and the rest can run on top of any server $V_S(G_I)$ or robot $V_M(G_I)$. It is up to the heuristic and the optimization formulation, to decide where to deploy them.

Every experiment uses the 5G infrastructure characterization of [Com+18b] and [ITU18], which considers Ultra Reliable Low Latency Communications. Table 5.7 shows every infrastructure element considered in the experiments, and Fig. 5.13 illustrates the interconnection of the network infrastructure. Each M1 switch is located in the access ring of the network, and it gathers the traffic of up to x6 LTE or NR RUs. Access rings have x6 M1 switches and x1 M2 switch, all of them interconnected in a ring fashion. Every M2 switch belongs to x4 access rings, and it steers the traffic up to the aggregation ring, where it is connected in a ring fashion with another x5 M2 switches. Experiments consider that edge and cloud servers are reachable using M1 and M2 switches, respectively.

Each point in the operation area of the robot cluster is covered at least by one LTE RU and at least by one NR RU. The coverage probabilities for each time instance are derived by a function which maps the distance of the RU and the cluster to the coverage probability. The probability slightly decreases until the end of the RU coverage area, and steeply drops to 0 at 120% of the RU reach. If a NR RU and the mobile cluster are not in LoS, the coverage probability is 0, independent of their distance. To achieve e2e delays demanded by the mobile robotics use case (between 1ms and 100ms), the experiment infrastructure assumes that aggregation and access ring switches introduce packet processing delays between 1ms and 10ms, under the same characterisation as performed in [Com+18b].

As previously stated, to derive this section's results a network infrastructure with just one cluster of robots has been generated with the 5GEN package presented in section 3.5. Then, a Python script generates series-parallel NS graphs G_S from which loop SFCs are selected. Robot cluster paths are encoded by coordinates which are used to calculate RUs coverage probabilities as robots move along the path. Next, the Python script runs section 5.2.3 heuristic to decide each VNF mapping on top of the infrastructure graph. Section 5.2.2 formulation is encoded in AMPL [FGK93], and the Python script invokes Gurobi 8.1 solver [Gur15] through the amplpy Application Programming Interface (API) to obtain the optimal mapping. All the experiments have been executed on two identical VMs with x4 vCPUs, 32GB of memory, and 132GB of disk.

Simulation results

This section presents the results of the extensive simulations performed with Algorithm 9, AMPL solver, and the state of the art FMC solution; denoted as *impr*-Υ, *AMPL*, and *FMC*; respectively. The details of the simulation parameters are shown in Table 5.8 for each of the experiments. The cluster paths in the Valencia haven are represented by their source and target locations.

All evaluation figures present boxplots, where the middle line shows the median (a.k.a. second

Table 5.7: Infrastructure used for experimentation

#	Element	Characteristics
x2	LTE RU [3GP16a]	8km radio coverage, 5ms one way delay [Stu12], 5.5 cost units OPEX [NJ14a]
x36	NR RU [Pat+18]	700m LoS coverage [Hal+18], 1ms one way delay, 11 cost units OPEX
x10	robots	x2 CPUs, 15.27 cost unit-s/CPU [Car19]
x6	edge server	x12 CPUs, 5.83 cost unit-s/CPU [Car19]
x2	cloud rack	200 CPUs, 2.46 cost unit-s/CPU [Car19]
x8	M1 switch	x8 dedicated CPUs
x6	M2 switch	x238 dedicated CPUs
x2	access rings	fiber ring connection, ≤6 M1 switches
x1	aggregation ring	fiber ring connection, x6 M2 switches

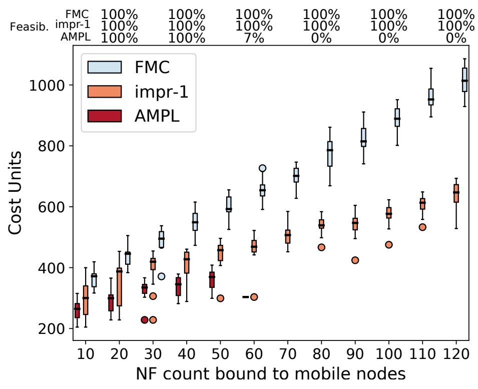
Table 5.8: Experiment parameters

Parameter name/explanation	**Value/range**			
Experiment name	Scalability	Coverage	Delay	Battery
Robot cluster path, see Fig. 5.13	$S \to D1$	$S \to D2$	$S \to D2$	$S \to D1$
Path total distance [meters]	678	488	488	678
Time interval count $t_u \in (t_0, t_1)$	24	24	24	24
Unloaded battery probability $\mathbb{P}_{bat}(N_i, 0), \forall N_i \in V_M(G_I)$	99%	99%	99%	99%
Full loaded battery probability $\mathbb{P}_{bat}(N_i, \overline{C}_{N_i}), \forall N_i \in V_M(G_I)$	50%	80%	80%	50%
Battery probability threshold th^s_{bat}	40%	70%	70%	72% & 75%
Infrastructure delay sample count	1	4	4	1
SFC delay [ms] Δ_{G_s}	1000	5	Varies	1000
Randomized VNF vCPU requirement $C_v \forall v \in V(G_S)$	0.5 x $\{0, \ldots 4\}$	0.5 x $\{0, \ldots 4\}$	0.5 x $\{0, \ldots 4\}$	0.25 x $\{0, \ldots 4\}$
VNF count $\lvert V(G_S) \rvert$	Varies	10	10	26
VNF count bound to robots	6	4	1	Varies
Coverage probability threshold κ_q	94%	Varies	94%	70%
Scenario repetition with different randomization seed	14	24	20	14

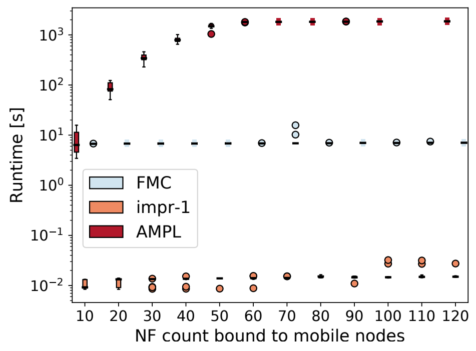
quartile) of the dataset, while the body of the boxplots show the first and third quartiles (a.k.a. the medians of the first half and the second half of the dataset separated by its median). The whiskers of the boxplots represent the datum which deviates from the boxplot body at most by 1.5 times the inter-quartile range, while outliers are individually plotted by circles which fall beyond the whiskers.

An input VNF placement problem with all previously presented constraints is deemed feasible, if the `AMPL` implementation finds a valid solution that respects all constraints in 30 minutes (measured in wall-clock time). In case of the heuristic, the timeout is reduced to 20 minutes. All experiments were executed with 3% optimality gap for `AMPL`, various improvement score limit values for the heuristic, and $k = 10$ for FMC.

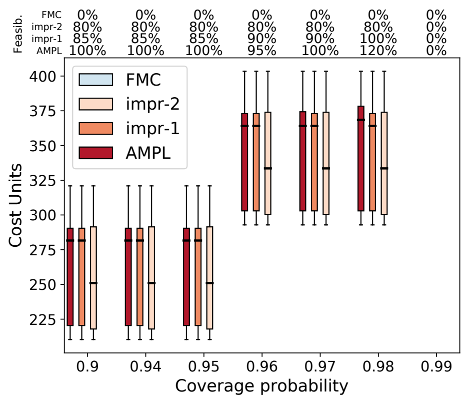
First of all, the scalability of the algorithms are compared depending on the number of VNFs to be placed; results in terms of cost and run-time are shown in Figure 5.15(a) and Figure 5.15(b), respectively. The time-bound feasibility is shown on top of the figures for each randomized scenario repetition corresponding to the dependant value on the horizontal axis. The scalability experiment is repeated multiple times for each input size, varying the distribution of VNF capacity requirements, the service graph's concrete topology, and the selection of the VNFs bound to the mobile cluster (see Table 5.8). The scenario parameters allow a solution to be found in any randomized generation,



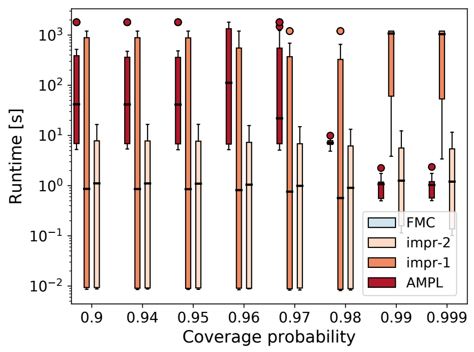
(a) Scalability test: costs



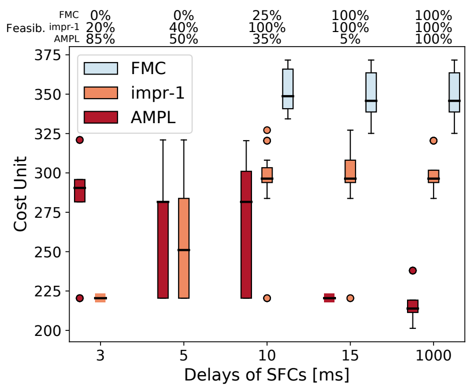
(b) Scalability test: run-times



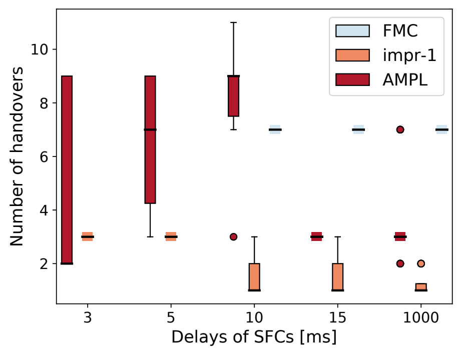
(c) Coverage threshold variation test: costs



(d) Coverage threshold variation test: run-times

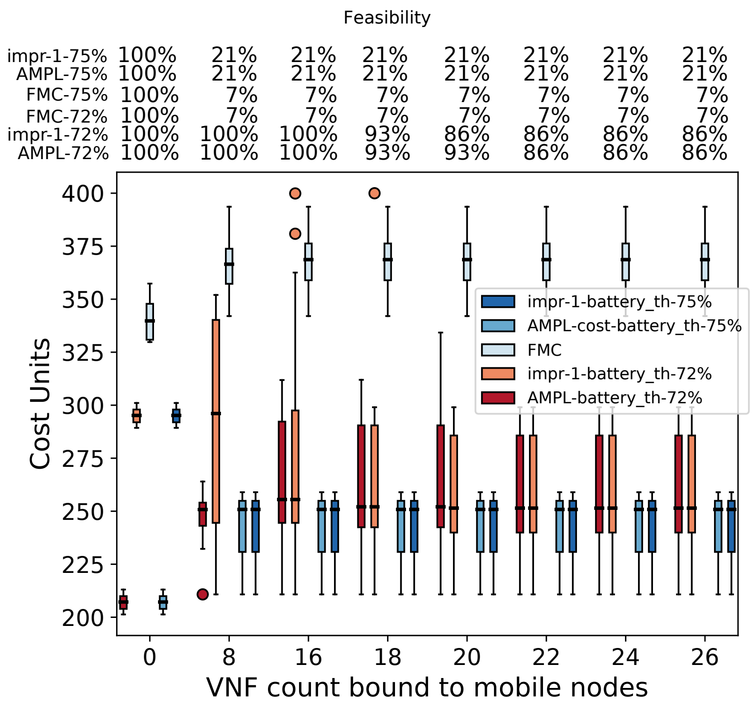


(e) SFC delay variation test: costs



(f) SFC delay variation test: handover counts

Figure 5.15: Results of scalability, coverage probability and SFC delay experiments



Feasibility

	0	8	16	18	20	22	24	26
impr-1-75%	100%	21%	21%	21%	21%	21%	21%	21%
AMPL-75%	100%	21%	21%	21%	21%	21%	21%	21%
FMC-75%	100%	7%	7%	7%	7%	7%	7%	7%
FMC-72%	100%	7%	7%	7%	7%	7%	7%	7%
impr-1-72%	100%	100%	100%	93%	86%	86%	86%	86%
AMPL-72%	100%	100%	100%	93%	93%	86%	86%	86%

Figure 5.16: Impact of battery probability threshold on cost and feasibility

due to loose SFC delay, coverage and battery probability thresholds; though the 30*mins* time limit may not be enough in all cases. Figure 5.15(a) shows the time-bound feasibility ratio calculated on the randomized repetitions. A steep drop of feasibility of the AMPL implementation occurs at the VNF count of 60, which is due to reaching the computation timeout in each case. The reason behind the timeouts is the exponential run-time of the AMPL solution, which is shown by Figure 5.15(b) in logarithmic time scale. On the contrary, both heuristics find feasible solutions in every possible setup. In terms of cost, our heuristic with $\Upsilon = 1$ outperforms FMC, with the former staying between 15% and 30% away of the optimal costs, and the latter increasing the cost gap with respect to our solution as the number of NFs bound to mobile nodes grows. Furthermore, our heuristic always find solutions below 100*ms* for all tests, whilst FMC takes around 10*s*.

Second, the effect of the coverage probability threshold κ_q is studied. Figure 5.15(c) shows how the cost varies by increasing the threshold, i.e. making the AP selection more strict. As the coverage probability requirement increases, deployment costs become more expensive, because the solutions impose the selection of the closer and more expensive NR antennae, rather than the cheap LTE antennae. Figure 5.15(c) depicts as well the feasibility, and shows that for $\kappa_q = 0.99$ all scenarios are infeasible, because there exists at least one subinterval in which the cluster is not covered by any antennae with such high probability. Regarding the impact of the improvement score Υ, Figure 5.15(c) and Figure 5.15(d) show that $\Upsilon = 2$ (*impr-2* time series) finds cheaper solutions faster. This is due to the heuristic design, which goes faster by shrinking the solution space and considering only VNF relocations with higher improvement score. The heuristic finds cheaper deployments faster, because they require less steps to make the rounded fractional solution feasible. Additionally, Figure 5.15(c) shows that FMC cannot find feasible solutions with the studied coverage thresholds $\kappa_q \geq 0.9$, since one or more migrations failed during the experienced handovers.

Next, the results of simulations varying the SFC delay are shown in Figure 5.15(e) and Figure 5.15(f). FMC cannot find feasible solutions for 3*ms* scenarios, as it is designed to try to map one VNF per compute node, and therefore, its mappings have to traverse more network links. The heuristic *impr-1* struggles with finding feasible solutions in the allocated time for the 3*ms* scenarios, while AMPL manages to prove the existence of valid solutions as shown by the feasibility percentages of Figure 5.15(e). This could be easily addressed by introducing a search space pruning step in addition to the locality constraints. In the 3*ms* scenarios the usage of the cheap and high capacity cloud nodes is not an option because their RTTs from all APs are above this value. Excluding these compute nodes from the allocation options for the VNFs contained in the strict SFCs would dramatically decrease the running time and thus increase the time-bound feasibility of the heuristic. Although, additional pruning steps decrease solution quality in the cases of more permissive delay requirements. On the other hand, the heuristic greatly outperforms the optimal solution search in the 10-15*ms* scenarios, where the `AMPL` algorithm fails to find any feasible solution before the 30 minutes timeout. This is due to the growth of the search space as the delay restriction is relaxed. Additionally, *impr-1* finds cheaper deployments than FMC, since the latter tries to use one compute node per VNF, and does not account for cloud nodes by design. Note that cloud nodes are cheap and strong candidates used by *impr-1* when the delay requirement relaxes (see the SFC delay case of 1000*ms*). Another interesting aspect of the solutions is the number of required handovers needed for the whole optimization time interval. A lower handover count requires less management operations and results in a more stable service. Handover comparison between the cost-optimal and the heuristic solutions are shown in Figure 5.15(f). The heuristic outperforms the optimal solution, which is especially relevant when the scenario could be solved by a few handovers as shown by the 10*ms* experiment scenarios with 100% heuristic feasibility. The `AMPL` algorithm could be modified to minimize the number of handovers, but it would further worsen its scalability, while the heuristic performs well by design. Furthermore, *impr-1* required less handovers than FMC in all the simulated scenarios.

Last, the results of the conducted experiments to examine the battery threshold parameter's effects are shown in Figure 5.16. The figure depicts cost values for both algorithms in cases of 72% and 75% battery alive probability requirements, as the number of VNFs to be placed on the mobile cluster increases. Note that FMC is agnostic of battery constraints and it reports the same solution, no matter the imposed battery alive probability. However, the feasibility of the FMC solution is depicted for battery alive cases. These scenarios challenge constraint (5.24), discovering the critical battery threshold to be around 72%-75%. In the 72% case the scenarios are vastly feasible with a slight decrease as the VNF bound to mobile nodes increase. The heuristic finds close to optimal allocations in almost all scenarios, except in the extreme case of much freedom. In the more strict case of 75%, besides the no location-bound VNF experiment which is essentially the same as the 72% case, the heuristic always finds all optimal solutions where it exists. Last of all, Figure 5.16 shows that FMC only finds solutions in the 7% of the simulations with more than 8 VNFs bound to mobile nodes. Indeed, it reports same feasibility ratios for both 72%, 75% battery thresholds, as it could only find deployments with $\mathbb{P}_{bat}(N_i, C_{N_i}) \geq 0.75$. As in previous results, FMC reports higher deployment costs because it tries to map each VNF to a different compute node.

The implementation of the algorithms, the simulation framework and all presented scenarios with raw data are available for further usage or result reproduction[15].

5.3 Conclusions

This chapter proposes different NFV orchestration algorithms to solve the VNE problem in 5G networks. In particular, it covers (*i*) the VNE problem accounting for 5G network slices' requirements; and (*ii*) the VNE problem for fog-oriented use cases, in particular, on a warehousing use case.

All the analyzed scenarios have been formulated as optimization problems to compare the optimality of the proposed embedding algorithms. The proposed solutions achieve near-optimal, if not optimal solutions with polynomial time guarantees. Additionally, they (*i*) meet KPIs of network slices; and (*ii*) deal with mobility-, coverage-, and volatility-related challenges of fog devices. Additionally, OKpi, has been validated not only via simulation, but on a real cloud robotic testbed.

[15] https://github.com/MartinPJorge/placement

Overall, the chapter proposes embedding algorithms for sliced, and edge/fog networks. Regarding OKpi, it is still left to study how the resolution parameter γ impacts the run-time of the algorithm. With that it would be possible to derive a run-time vs. optimality trade-off to reach adequate solutions under time-constrained computational times. About the proposed heuristic for the mobile robots warehousing solution, a first extension of the study would be to plug battery, radio coverage, and mobility models into the system model constraints, and check the complexity side-effects.

Additionally, both the two NFV orchestration algorithms presented in this chapter could be extended to account for the Radio Access Network (RAN) resource allocation, and pools of federated resources as discussed in the previous chapter 4.

6. Scaling of V2N services: a study case

The two previous chapters study and propose solutions to solve the Virtual Network Embedding (VNE) problem in 5G networks. In particular both solve the problem statically, i.e., only take once the decisions on which resources should be allocated to accommodate the traffic demand and requirements. For example, upon the orchestration of a Vehicle-to-Network (V2N) Network Service (NS), OKpi (see section 5.1) decides which network links are traversed by the V2N service packets, and the server resources to be allocated to process the traffic. Such allocation is performed throughout the whole NS lifetime, denoted with $\phi(\psi)$, but it assumes a static demand of traffic $l(\psi, v_1, v_2)$ that remains during the service lifetime. However, this is not usually the case, specially in the V2N NSs that are matter of study in the present chapter. In particular, the traffic demand of V2N services depend on the vehicles' flow across the roads. That is, a hazard warning or collision avoidance service will not only have to accommodate more link resources for the increase of cars' packets, but as well more computational resources to, for example, compute the distances among the increasing number of cars. Hence, it is necessary to scale the network resources assigned to offer an adequate V2N service as the traffic flow increases along the roads.

This chapter studies the scaling of V2N by means of computational resources, so as to satisfy the low latency requirements of these services. Section 6.1 provides some background and requirements of V2N NSs, before section 6.2 introduces the vehicular dataset used in this chapter to propose a tentative solution to scale V2N services. In particular, the solution is based on the forecasting of traffic flow in a specific road using different time-series techniques that are presented in section 6.4, and later used in section 6.5 to increase/decrease the number of CPUs accordingly.

6.1 V2N services' requirements

Connected Automated Vehicles (CAV)s represent one of the most significant transformations of the automobile industry, providing an opportunity to enhance monitoring of transportation network conditions, in order to improve safety and reduce pollution, energy consumption, and travel delays. 5G systems are gaining traction as a viable technological solution, due to the ubiquity of the cellular infrastructure. Moreover, by ensuring ultra-low latency and ultra-high reliability communications under high-density and high-mobility conditions, 5G systems will enable Cerllular Vehicle-to-Everything (C-V2X) communications.

The C-V2X technology has been designed by the Third Generation Partnership Project (3GPP) to allow vehicles to communicate with other vehicles (Vehicle-to-Vehicle (V2V)), road users (Vehicle-to-Pedestrian (V2P)), roadside infrastructure (Vehicle-to-Infrastructure (V2I)), and cloud/edge servers (V2N) [3GP19a]. The 3GPP specifies the Enhancement of 3GPP support for Vehicle-to-Everything (V2X) services (eV2X) in the following four areas: *(i)* Vehicles Platooning;

(ii) Advanced Driving; *(iii)* Extended Sensors; and *(iv)* Remote Driving [3GP19b]. The European Telecommunication Standard Institute (ETSI) Multi-Access Edge Computing (MEC) group has also collected the corresponding requirements for MEC, by analyzing the relevant V2X use cases [ETS18d] pertaining to safety, convenience, advanced driving assistance and vulnerable road users.

All aforementioned cases require process-intensive and low-latency, reliable computing and communication services. For example, remote driving applications require values of 99.999% reliability and 5ms End-to-End (E2E) latency with a rate of 25Mbps and 1Mbps for the Uplink (UL) and Downlink (DL) respectively.

Such characteristics prohibit cloud-based solutions, due to the long delays to reach the cloud. An effective approach to address such requirements is to leverage the edge computing paradigm [Shi+16]. Edge Computing provides a low-latency alternative to the cloud, moving computational resources to the edge, closer to where the data is being generated, processed, and most likely consumed. However, edge computing devices have limited processing resources. Therefore, careful resource scaling or (re)allocation is needed to meet both the computational and the latency requirements of vehicular applications. Indicatively, in [Pre+18], the efficient placement of edge computing devices is addressed for vehicular applications in an urban scenario, by mapping the average number of vehicles in each cell to Central Processing Unit (CPU) demand. However, to tackle such resource allocation problem optimally over time or to scale resources cost-effectively, the evolution of the network traffic demand is needed per cell.

This chapter compares different forecasting techniques to predict road traffic evolution over time. The per-cell granularity is granted thanks to per-road forecasts that use a Torino road traffic dataset (see section 6.2). Traffic forecasts are later used to dynamically scale-out the network slice computational resources (at the edge), supporting the requirements for three different Cerllular Vehicle-to-Network (C-V2N) use cases, drawn from the ETSI [ETS18d][ETS18c] and 3GPP[3GP19b] use case portfolio:

- *Remote Driving* to enable the remote operation of a vehicle by a remote driver or a cloud-based application. Different use case scenarios can benefit of such approach, ranging from the operation of vehicles that are located in dangerous environments, assistance of passengers that cannot drive by themselves, or support of autonomous vehicles that are in unexpected situations. Usually, the information exchanged between the V2X application in the vehicle and the V2X application server includes live video streams of on-board cameras and driving commands. These use cases can benefit from computational resources at the edge to locally process vehicle's information and compute the driving commands. Such applications require E2E latency below 5ms [3GP19b].
- *Cooperative awareness* to improve traffic efficiency. This also encompasses multiple use cases, with the goal of improving overall traffic efficiency by sharing specific pieces of information and/or coordinating actions of several vehicles. Examples of these applications are lane change assistance, co-operative glare reduction, co-operative merging assistance, traffic light optimal speed advisory, etc. [ETS09]. These use cases benefit from computational resources at the edge, to collect, process and decide what information/command to distribute and to which vehicles. These applications typically require latency between 100ms and 500ms.
- *Hazard warning* to share and distribute pieces of information for avoiding a risk (e.g., danger on a lane, intersection warning, collision avoidance) or minimizing the consequences of it (e.g., pre-configuration of seat-belts in preparation of an imminent accident). Computing resources at the edge can be used to extend the connected car cloud into the highly distributed mobile network environment. As described in [ETS18c], applications can be deployed on the edge to provide the roadside functionality: *(i)* receiving local messages directly from the applications in the vehicles and the roadside sensors; *(ii)* analyzing them; and *(iii)* propagating (with extremely low latency) hazard warnings and other latency-sensitive messages to other cars in the area. If provided within very low and strict latency constraints, vehicles in the

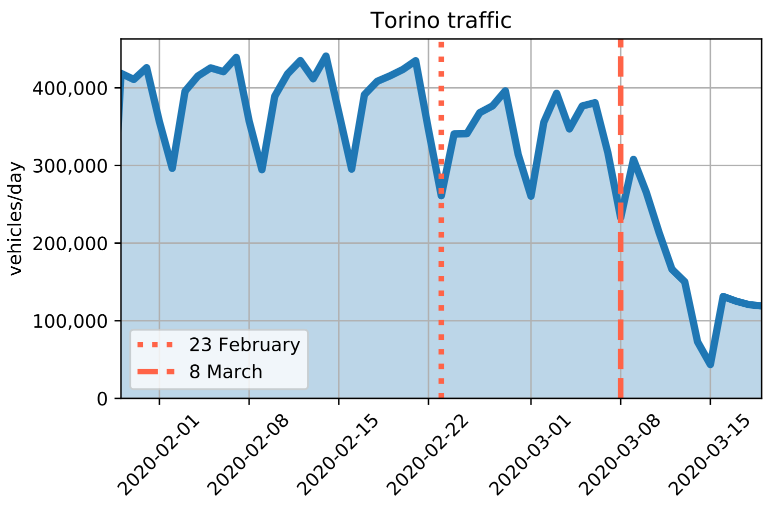


Figure 6.1: Impact of COVID-19 in Torino traffic flow. The figure plots the sum of all road probes' traffic flows.

local vicinity of a dangerous situation receive data in a matter of milliseconds, allowing the driver to immediately adopt preventive actions (e.g. switch lanes, slow down or change his route). Depending on the type of event, the speed of the road and the traffic density, these applications require latency between 10 and 50ms [ETS09].

6.2 Torino road traffic dataset

This section presents the real road traffic dataset used in this chapter. The dataset contains information of Torino city traffic flows, with measurements from more than 100 road probes reporting their location, traffic flow, and vehicles speed.

The data is gathered by using S.I.MO.NE protocol [CFA09], and exposed via an API[1]. Thus, probes' measurements are fetched every 5 minutes, and gathered in a common eXtensible Markup Language (XML) file. Each entry in the XML file represents the measurements that a probe aggregates every 5 minutes, containing fields [Arn+14, Table 8] such as: *(i)* probes' latitude and longitude; *(ii)* their identification codes, *(iii)* their offsets along the road; *(iv)* the road names; *(v)* average speed of vehicles; *(vi)* vehicles flow, i.e., vehicles per hour; *(vii)* measurements accuracy; and *(viii)* timestamp. In this work, all entries of the gathered XML files have been collapsed in a Comma-Separated Values (CSV) dataset with samples from 28/01/2020 to 25/03/2020.

Not every road probe reports data with the same frequency. Indeed, some of them suddenly stop reporting measurements during the night, or even worse, in the middle of rush hours. Such behavior might be caused by errors in the probes' hardware, or just by decisions of Torino city Mobility Central. Thus, the collapsed CSV dataset required of some filtering to remove spurious probes, i.e., probes reporting less than an 80% of the measurements they should have reported (one every 5 minutes). Among the 116 road probes present in the dataset, 24 of them did not satisfied such requirement and were removed from the CSV dataset.

Even after pruning spurious road probes, the dataset still had probes that spent up to a 20% of the time without reporting data. Thus, the dataset was sanitized to fill those missing values with the last reported probe measurement. However, this first sanitation sweep was not sufficient, because there were still missing timestamps. The first sweep just considered as timestamps those reported by any road probe of the dataset, hoping that every time instant would be present in the reported information. But, it happened that for some instants of time (e.g., at 2020-02-15 at 04:25) no road probe reported data (this might be because the company managing road probes might turn them off

[1] `http://opendata.5t.torino.it/get_fdt`

Table 6.1: Forecasting features

Feature	Name	Values	Description
ϕ_1	flow	integer	vehicles/hour
ϕ_2	accuracy	$\{0,\dots,100\}$	percentual accuracy of the reported measurement
ϕ_3	speed	float	average vehicles' speed (km/hour)
ϕ_4	distance to orbassano	[0,35]	distance to Corso Orbassano road probe (km)
ϕ_5	day_of_week	$\{1,\dots,7\}$	day of the week
ϕ_6	month	$\{1,\dots,12\}$	month of the measurement
ϕ_7	day	$\{1,\dots,31\}$	day of the measurement
ϕ_8	year	integer	year of the measurement
ϕ_9	hour_min	[0,24)	hour+minute/60

in the wee hours of the morning). Thus, the first sanitation sweep missed such values. To fix the issue, a second sanitation sweep looked for those missing timestamps all over the dataset, which are filled with the most recent information.

By the time the dataset was collected, the COVID-19 pandemic struck northern Italy, and, consequently, there was a significant drop of the flow of vehicles in the city of Torino. More precisely, on February 23rd the region of Piemonte (where Torino is) applied a Health Ministry order [SC20b] that temporarily suspended teaching activities in the region. And under the pandemic expansion, the 4th of March a new decree [Giu20b] extended the restrictions until the 15th of March. But it is not until the 8th of March when the vehicles per day show a significant drop in Torino city (see Figure 6.1). That day the government imposed a lockdown [Giu20c] in other northern regions of Italy that did not affect Torino, which did not experienced lockdown until the 11th of March [Giu20a]. But, even though Torino was not in lockdown by the 8th of March, the traffic decrease after then was more significant than the decrease experienced after the 23rd of February. Figure 6.1 shows a traffic decrease of a 9.82% from the 27th of February to the 8th of March, whilst after the 8th of March the decrease dropped down to 57.53%.

COVID-19 impact on the traffic of Torino is a relevant factor to consider for the forecasting techniques presented in section 6.3, as it allowed to assess how each technique is able to adapt to changing patterns. Thus, and based in the average traffic decrease, the remaining of this paper considers the 8th of March as the date splitting the dataset into non-COVID-19 (before), and COVID-19 (after) periods.

6.3 Forecasting techniques

This section presents the features of section 6.2 dataset, and how they are arranged in a matrix that is later fed as input to forecasting techniques. It describes the selected forecasting technique analyzed in the remainder of this chapter. Sections 6.3.1-6.3.2 describe DES, and TES time-series techniques; and section 6.3.3 explains the HTM solution. Finally, sections 6.3.4-6.3.7 detail the analyzed NN forecasting techniques, namely, LSTM, GRU, TCN, and TCNLSTM.

Each forecasting technique is used to forecast the vehicles/hour traffic flow f_t seen at Corso Orbassano road probe[2]. The set of features ϕ_i at their disposal are those reported by all road probes s_j $(s_1,\dots,s_{92})$ of the dataset. The numerical value of a feature reported by a probe at instant t is denoted as $x_t^{\phi_i,s_j}$. Table 6.1 enumerates the features $\phi_i,\ i=\{1,\dots,9\}$ used by the selected techniques. These consist in a subset of all the features inside the XML files used to build the CSV dataset (see section 6.2). The dataset granularity is of 5 min., and throughout this section $t+1$ represents the instant $t+5$ min.

[2] This is the road probe with highest number of reported measurements in Torino city.

Among all analyzed techniques, some of them can incorporate all features of past events to forecast the future flow of Corso Orbassano road. Thus, they take as input a matrix containing every feature reported by a road probe during the last h timestamps. The forecasting techniques described in the following sub-sections use matrix $X_{t,h}$ to denote the input.

$$X_{t,h} = \left(\begin{array}{ccc} x_{t-1}^{\phi_1,s_1} & \cdots & x_{t-1}^{\phi_9,s_1} \\ \vdots & \ddots & \vdots \\ x_{t-1}^{\phi_1,s_{92}} & \cdots & x_{t-1}^{\phi_9,s_{92}} \\ \hline \vdots & \vdots & \vdots \\ \hline x_{t-h}^{\phi_1,s_1} & \cdots & x_{t-h}^{\phi_9,s_1} \\ \vdots & \ddots & \vdots \\ x_{t-h}^{\phi_1,s_{92}} & \cdots & x_{t-h}^{\phi_9,s_{92}} \end{array}\right) \tag{6.1}$$

The dataset has been split in two portions: the training dataset and the test dataset. Every forecasting technique is trained with the training dataset. During the training, it calculates the parameters used to perform the forecasting in the test dataset. In case the forecasting technique updates its parameters as it performs forecasting, the technique uses an *online training* approach. In case the forecasting technique does not update its parameters during the forecasting, they follow an *offline training* approach. Each of the presented techniques explains how they implement both the *offline* and *online training*.

Note that only techniques described in sections 6.3.4-6.3.6 use the feature matrix $X_{t,h}$. For the techniques present in sections 6.3.1-6.3.3, only the traffic flow records f_t are used as input for their predictions.

6.3.1 Double Exponential Smoothing (DES)

[3]DES is a forecasting technique based on time series analysis. DES uses a smoothing S_t time scale with a smoothing factor $\alpha \in [0,1]$, and trend T_t with a trend factor $\beta \in [0,1]$ as described in Eqs. (6.2) and (6.3). The smoothing time scale is obtained based on the previous experienced time interval value of smoothing S_{t-1} and trend T_{t-1}. Note that in Eq. (6.2), the current value of time series (i.e., f_t) is used to derive the smoothing value S_t:

$$S_t = \alpha \cdot f_t + (1-\alpha) \cdot (S_{t-1} + T_{t-1}) \tag{6.2}$$

$$T_t = \beta \cdot (S_t - S_{t-1}) + (1-\beta) \cdot T_{t-1}. \tag{6.3}$$

Eq. (6.4) is used to obtain k timestamps ahead (namely *lead time*) forecast f_{t+k} in DES:

$$\widehat{f}_{t+k} = S_t + k \cdot T_t. \tag{6.4}$$

In DES, the *offline training* is determined by calculating the smooth (S_t) and the trend (T_t) as in the Eq. (6.2) and Eq. (6.3) using the training set. Then, Eq. (6.4) is applied to forecast the flow values based on the k timestamps ahead. Whereas in *online training*, the S_t and T_t values are updated at each time t even during the testing phase.

[3] The implementation of DES in the work presented in this chapter was done by Dr. Koteswararao Kondepu

6.3.2 Triple Exponential Smoothing (TES)

[4]TES is another time series analysis technique. As shown in Eqs. (6.5)-(6.7), TES exploits three different forecasting factors such as *smooth*, *trend*, and *seasonality*. Here, $\widehat{f}_{t+k}$ represents the forecast at time $t+k$ with k timestamps ahead, and the seasonality constant m (i.e., the number of observations per season). TES can be performed in two ways, namely *additive* and *multiplicative* techniques, depending on the seasonality effect. Note that the following equations are defined based on the *additive* technique:

$$S_t = \alpha \cdot (f_t - I_{t-m}) + (1-\alpha) \cdot (S_{t-1} + T_{t-1}) \tag{6.5}$$

$$T_t = \beta \cdot (S_t - I_{t-1}) + (1-\beta) \cdot T_{t-1} \tag{6.6}$$

$$I_t = \gamma \cdot (f_t - S_t) + (1-\gamma) \cdot I_{t-m} \tag{6.7}$$

$$\widehat{f}_{t+k} = S_t + k \cdot T_t + I_{t+k-m}, \tag{6.8}$$

where m is the length of the seasonal cycle, for $\alpha \in [0,1]$, $\beta \in [0,1]$, and $\gamma \in [0,1]$.

In TES, the *offline training* is performed by calculating S_t, T_t, and I_t with train set. After that, the $\widehat{f}_{t+k}$ is used to forecast the k timestamps ahead forecasted values. Whereas in *online training*, the S_t, T_t, and I_t are updated at each time t even during the testing phase.

6.3.3 Hierarchical Temporal Memory (HTM)

[5] The core component of the HTM forecaster [Ahm+17] is a temporal memory m_t consisting of a two-dimensional array of cells that can either be switched on or off and that evolves as the timestamp t increases ($t = 0, 1, 2, \ldots$). Cells can influence each other via synapses ω and update rules H and G. For a detailed description of the update rules H and G we refer to [Ahm+17], but in essence they embody how neurons influence each other. At time t during the training the temporal memory m_t is presented with a sparse bit string s_t as input that has the same dimensions as the number of columns in the temporal memory. For each sparse input bit string s_t in the sequence, the temporal memory produces an output sparse bit string $\hat{s}_{t,1} = H(s_t, m_t, \omega)$ (of the same length), where the second subscript denotes that the prediction pertains to one slot in the future and it updates the state of its memory as $m_{t+1} = G(s_t, m_t, \omega)$. The output sparse bit string $\hat{s}_{t,1}$ serves as a forecast for the next sparse bit string s_{t+1}. The temporal memory can forecast not only the next sparse bit string, but also the following ones by building upon consecutive forecasts, e.g., the forecast $k > 1$ time slots in future is obtained via $\hat{s}_{t,k} = H(\hat{s}_{t,k-1}, m_{t+k-1}, \omega)$ and $m_{t+k} = G(\hat{s}_{t,k-1}, m_{t+k-1}, \omega)$. Learning involves adjusting the synapses ω in such a way that the output bit strings $\hat{s}_{t,k}$ $(k \leq 1)$ resemble the actual input bit strings s_{t+k} $(k \leq 1)$ as much as possible. In that way the temporal memory learns to forecast the next sparse bit strings based on the patterns in the sequence of input bit string it saw.

When being presented with a sequence of floating point values f_t the HTM forecaster first translates these floating point values in sparse bit strings via an encoder $s_t = E(f_t)$. Once the values of the sequence are encoded in bit strings by this encoder, the temporal memory learns to forecast the sparse bit strings as described before.

Finally, the predicted bit strings $\hat{s}_{t,k}$ need to be translated to floating point variables again, which is the job of a classifier (a process that also referred to as decoding). The classifier chooses as forecast $\hat{f}_{t,k}$, k timestamps in future, a representative value of all values that encode into the predicted bit string $\hat{s}_{t,k}$.

Untrained the HTM forecaster sets $\hat{s}_{t,k} = s_t$, which is equivalent to a sample-and-hold strategy that we use as benchmark in Figure 6.3. After training, the forecasts are more accurate.

In HTM, we run through the training set once (and checked that running through it more did not have a beneficial effect) and then either stop the learning process before running through the test set (*offline training*), or continue training while running through the test set to obtain the (*online training*) results.

[4] The implementation of TES in the work presented in this chapter was done by Dr. Koteswararao Kondepu

[5] The implementation of HTM in the work presented in this chapter was done by Dr. Danny de Vleeschauwer

6.3.4 Long Short-Term Memory (LSTM)

LSTM is a special form of Recurrent Neural Network (RNN) that can learn long-term dependencies based on the information remembered in previous steps of the learning process. It consists of a set of recurrent blocks (i.e., memory blocks), each of the block contains one or more memory cells, and multiplicative units such as *input*, *output* and *forget gate*.

LSTM is one of most successful models for forecasting long-term time series, which can be characterized by different hyper-parameters, specifically the number of hidden layers, the number of neurons, and the batch size. Details of LSTM parameters and their impact on forecasting accuracy can be found in [HS97]. However, the process of finding optimal hyper-parameters which minimize the forecasting error could be time and resource consuming.

In the approach considered in this chapter, the LSTM inputs corresponds to h previous timestamps with the features described in Table 6.1 (i.e., $X_{t,h}$) and the output vector corresponds to k timestamps ahead. A *stacked LSTM model* is exploited with a single-step (i.e., $k = 1$) and a multi-step (i.e., $k > 1$) forecasting.

In *LSTM single-step forecasting*, a single timestamp is forecasted based on the history, as follows:

$$\widehat{f_t} = LSTM(X_{t,h}) \tag{6.9}$$

where $\widehat{f_t}$ is the forecast of the single time-stamp at t, and $X_{t,h}$ corresponds to the station features during the last h timestamps.

In *LSTM* multi-step forecasting*, LSTM* forecasts k future timestamps by considering h previous timestamps.

$$\widehat{f}_{t+k-1}, \widehat{f}_{t+k-2}, \dots, \widehat{f_t} = LSTM^*(X_{t,h}) \tag{6.10}$$

where k $>$ 1.

For the *offline training* approach, LSTM and LSTM* learn their neurons' weights running the back-propagation-through-time [CR13], over a training dataset $T = \{(X_{t,h}, f_t)\}_{t=0}^{N}$ Then, both techniques use the learned weights to forecast $\widehat{f}_{t>N}$. If LSTM and LSTM* use *online training*, the weights are updated following procedure illustrated in Figure 6.2. That is, upon each traffic flow forecast, LSTM and LSTM* update their parameters using an online training window, which is nothing but a sliding window that incorporates the most recent road probes' reports. Following the notation, the online training window T_t^O of size W is updated as follows:

$$T_{t+1}^{O} = \{(X_{t-i,h}, f_{t-i}) : \forall 0 \le i < W-1\} \tag{6.11}$$

i.e., the online window incorporates the latest observed features-flow, and discards the oldest features-flow pair.

6.3.5 Gated Recurrent Unit (GRU)

Gated Recurrent Units (GRUs) [Chu+14] are neurons used in RNNs, and as LSTMs cells, they store a hidden state that is recurrently fed into the neuron upon each invocation. The neuron uses two gates, namely, *(i)* the *update gate*, and *(ii)* the *reset gate*. The former gate is an interpolator between the previous hidden state, and the candidate new hidden state; whilst the latter gate decides what to forget for the new candidate hidden state.

GRUs were originally proposed at [Cho+14], and compared against the LSTMs. They intend to avoid ignoring features in the past during the training stage (i.e., the vanishing gradient problem [BSF94]). In other words, they keep track of as much information as possible of past events. Thus, their use in time-series forecasting is becoming popular in current state-of-the-art.

In a similar way to the LSTM approach, the implemented GRU solution uses the stations' features over the last h timestamps (i.e. $X_{t,h}$) to forecast the traffic flow in the next k timestamps, or to

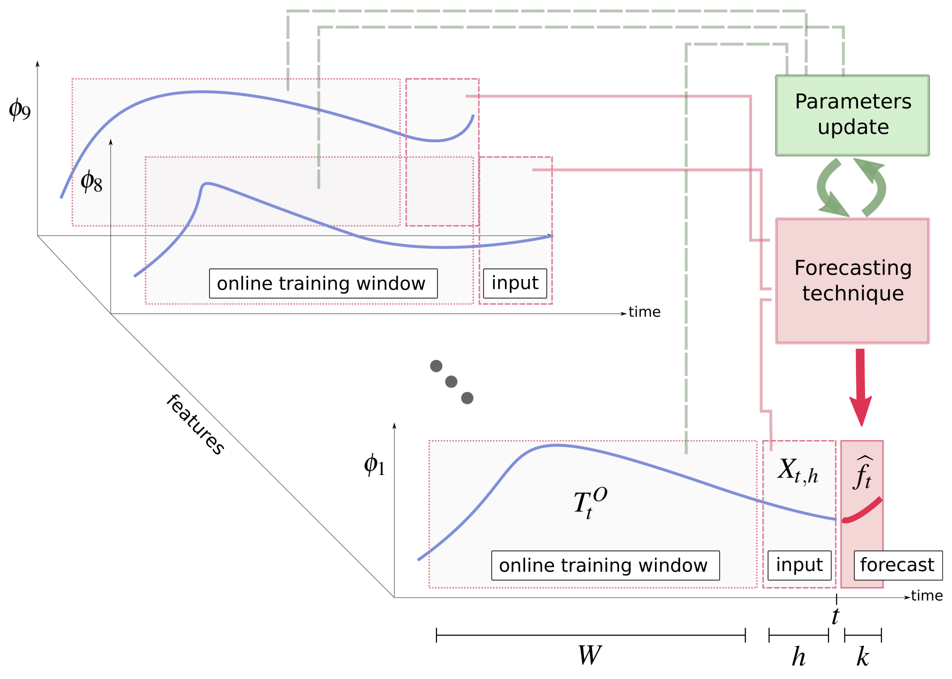


Figure 6.2: Online training & forecasting for NN techniques.

just forecast the traffic flow for the next time-stamp. Thus, both *multi-step forecasting* $GRU^*(X_{t,h})$, and *single-step forecasting* $GRU(X_{t,h})$ are implemented. But unlike the LSTM solution, the output of a GRU layer is not fed into a consequent GRU layer, but directly into the output layer that is densely connected with the neurons in the GRU layer. Regarding the *offline* and *online training*, GRU and GRU* implement the same solution as the LSTM.

6.3.6 Temporal Convolutional Networks (TCNs)

The usage of TCNs in this work relates to the results reported in [Aqi+17], where authors claimed that they implemented a traffic jam forecasting methodology using a deep learning infrastructure with 2 hidden layers, and convolutional neural networks. For the sake of comparison, their approach was included in our analysis.

The implemented version consists of a neural network with two hidden layers, namely, a TCN_h hidden layer, and another layer R to reduce TCN_h output dimension down to a k-vector. Thus the implemented single-step, and multi-step neural is written as

$$\begin{aligned} \widehat{f}_t &= TCN(X_{t,h}) = R \circ TCN_h(X_{t,h}) \\ \widehat{f}_{t+k-1}, \dots, \widehat{f}_t &= TCN^*(X_{t,h}) = R^* \circ TCN_h(X_{t,h}) \end{aligned} \tag{6.12}$$

Note that the second layer R^* is the one responsible of expanding the output of the TCN to a multi-step forecast. Moreover, the convolution operation [BKK18] performed by TCN_1 uses a $h/4$ convolution window operating in the time-domain of the input matrix $X_{t,h}$.

For the *online* and *offline training*, TCN and TCN^* follow the same approach as the described for $LSTM$ and $LSTM^*$ in section 6.3.4

6.3.7 Convolutional LSTM

In the convolutional LSTM, both TCN and LSTM models are combined into a single unified framework and trained together as described in [Sai+15]. The input features are feed to TCN layers, and the LSTM is taking as input the output of the TCN layers. Then, the output of the LSTM is fed into a dense layer. This model is considered to observe the advantage for mapping input features extraction with TCN and interpreting the features with LSTM model. In [Pas+14], it is shown that the LSTM performance can be improved by providing better features. Indeed, TCN helps by reducing the frequency variations in the input features. Here, the convolutional 1D structure with 64 filters is considered, and all other parameters are summarized in Table 6.2. The convolutional LSTM is referred to as TCNLSTM hereafter.

6.4 Road traffic flow forecasting

The techniques described in section 6.3 are applied to forecast the number of vehicles of Corso Orbassano road in Torino. We use the Root Mean Square Error (RMSE) metric to measure the accuracy of the forecasted road traffic flow, with respect to its real counterpart.

Table 6.2: Evaluation Parameters

Parameter	**Forecasting techniques**	**Value**
Level factor (α)	DES, TES	0.5, 0.5
Trend factor (β)	DES, TES	0.001, 0.001
Seasonality factor (γ)	TES	0.001 (3 days)
Hidden layers	TCN, LSTM, GRU, TCNL-STM	2,2, 1, 4
Neurons in hidden layer	TCN, LSTM, GRU, TCNL-STM	100
Epochs	TCN, LSTM, GRU, TCNL-STM	100
history Window size (h)	TCN, LSTM, TCNLSTM	60 min.
	GRU	120 min.
Batch size	TCN, LSTM, GRU	5
Temporal memory	HTM	32x2048
Encoder representation		1024 bit str

DES and TES were implemented in Python, HTM was tested using a proprietary solution, and neural network techniques were implemented by using Google's TensorFlow library, accessed through the Keras high-level front-end. Table 6.2 reports the hyperparameters that allowed to get the lowest RMSE for each forecasting technique. In HTM, we use a temporal memory of 2048 columns of 32 cells each and the encoder maps floating point values in bit strings of 1020 bits of which only 21 where non-zero.

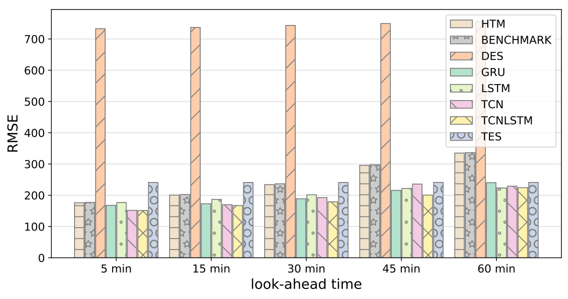
The forecasting techniques have been evaluated in two different scenarios with a 80% of training data, and a 20% of testing data, namely

- *non-COVID-19 scenario*
  - training: 28thJanuary - 28thFebruary
  - testing: 29thFebruary - 07thMarch
- *COVID-19 scenario*
  - training: 06thFebruary - 07thMarch
  - testing: 8thMarch - 15thMarch

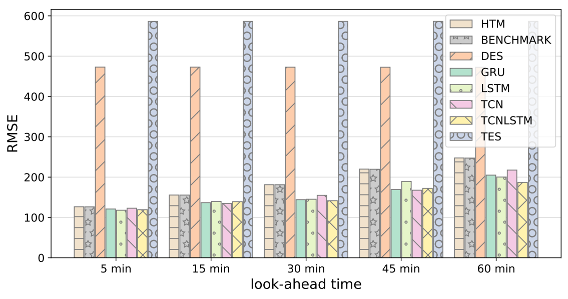
6.4.1 Look-ahead time impact

Throughout this section the k timestamps ahead parameter will be referred as the *look-ahead time*. That is, the future time for which the data need to be forecasted. Since the forecasted traffic flow will be later used to scale the different vehicular services, the *look-ahead time* will depend on multiple factors such as the time required to (de)allocate the necessary resources, the type of service, or the applied virtualization technology.

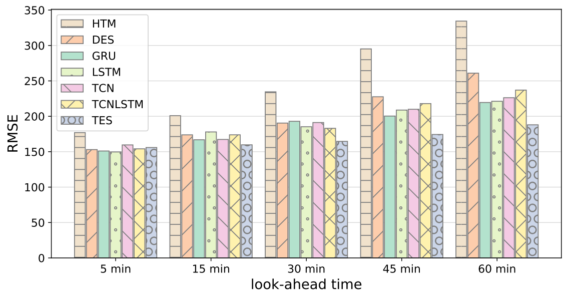
Results of Figure 6.3 illustrate how increasing the *look-ahead time* forecast leads to an increasing RMSE in every possible training and dataset combinations (*online*/*offline*, *COVID-19*/*non-COVID-19*), as it becomes more difficult to forecast the traffic further in the future.



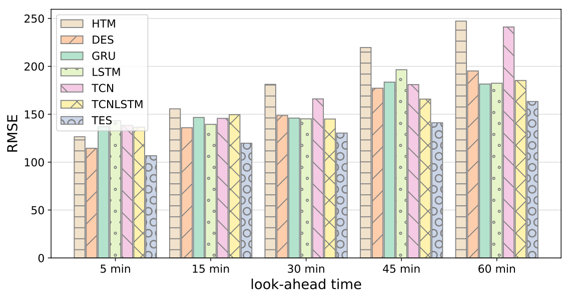
(a) Offline training and non-COVID-19 scenario



(b) Offline training and COVID-19 scenario



(c) Online training and non-COVID-19 scenario



(d) Online training and COVID-19 scenario

Figure 6.3: Accuracy of section 6.3 look-ahead forecasts.

Figures 6.3(a) and 6.3(b) show that the HTM technique did not manage to beat the sample-and-hold benchmark (i.e., $\hat{f}_{t+1} = f_t$). Moreover, in the *online training* scenarios, it yielded the worst performance among all analyzed techniques. For the rest of the techniques, the neural networks (NNs) solutions achieved the best performance for *offline training*. In the *offline training*, DES is not capable of capturing the trend, and the TES pitfalls in the *COVID-19* scenario. Unlike DES and TES, the NN solutions can capture the evolving traffic trend thanks to the update of their hidden states (except the TCN). This explains why the NNs achieve lower RMSE when using offline forecasting (see Figure 6.3(a) and Figure 6.3(b)). Figure 6.3(a) and Figure 6.3(b) show the RMSE values of *offline training* in *non-COVID-19* and *COVID-19* scenarios. The results presented in Figure 6.3(a) show that DES technique has highest RMSE values, because the smooth (S_t) and the trend (T_t) values initially calculated during the training, are not updated in the testing phase. The other time-series technique (i.e., TES) mitigates such problem since its seasonality factor can capture better the trend.

Figure 6.3(b) shows the RMSE values of the considered techniques in *offline training* with *COVID-19* traffic. The considered scenario does not show any seasonality during 8th Mar - 15th Mar due to the *COVID-19* lockdown mentioned in section 6.2. Thus, the obtained TES results exhibit the highest RMSE value compared to all other techniques. The detailed description about this behavior is discussed later in this section.

Figure 6.3(c) and Figure 6.3(d) show the RMSE values of *online training* in *non-COVID-19* and *COVID-19* scenarios. The TES outperforms all considered NN solutions even when the *look-ahead time* increases. In addition, the results show that TES does not increase the RMSE as much as the NN techniques. This is due to the fact that it captures faster the new trends of traffic over the time. Thus, the long *look-ahead time* forecasts are better as smoothing (S_t), trend (T_t), and seasonality (I_t) are updated for every data point in the test set. Even though the traditional time series techniques (DES/TES) are limited to uni-variate time series, the *online* update of their parameters achieve a better performance than the NN solutions that account for all features reported in Table 6.1.

Finally, Figure 6.4 shows the real and the forecasted traffic flow as a function of time. Here, the *look-ahead time* is set to 5 min., and *offline training* is used to forecast the traffic flow during

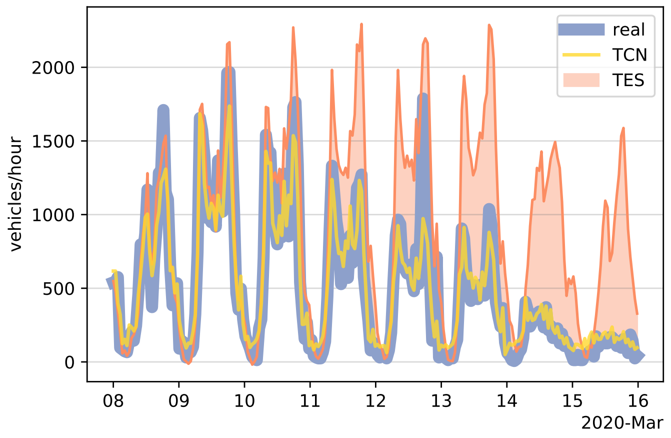


Figure 6.4: TES, TCN forecasts vs. real flow values. 5 min. *look-ahead* in *COVID-19* scenario using *offline training*.

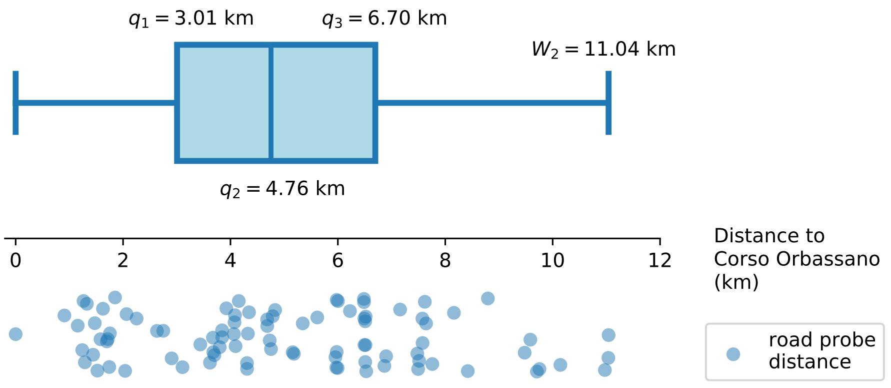


Figure 6.5: Distances of dataset probes towards Corso Orbassano road probe. Boxplot above illustrates the quantiles for the distances' distribution, along with W_2, i.e., the last road probe's distance that is below $1.5(q_3 - q_1) + q_3$.

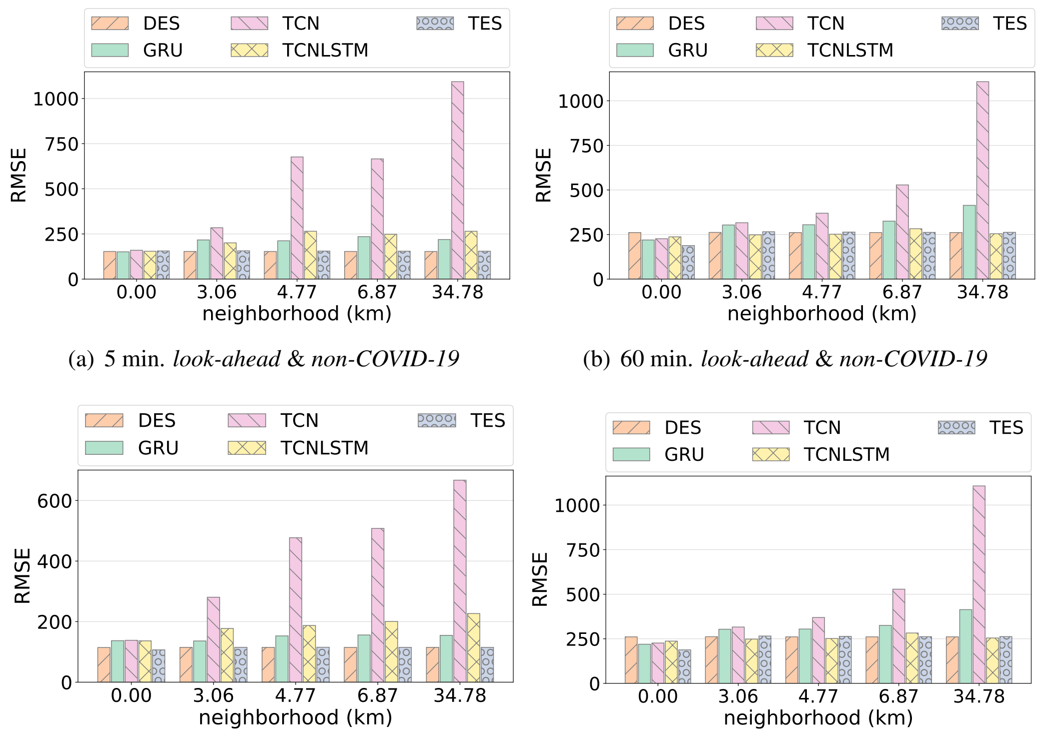
the *COVID-19* scenario (i.e., same conditions as in Figure 6.3(b)). It is possible to observe that the real traffic flow exhibits a seasonality pattern till 12thMar. However, later on traffic flow gradually decreases due to *COVID-19* lockdown. This explains why TES exhibits the highest RMSE values in Figure 6.3(b). Both HTM and NN forecasts adapt to the traffic decrease, and among all them, TCN was selected to show that it forecasts traffic flow better than TES. Because every technique uses *offline training*, TES keeps using the seasonality learned during the training phase, and it forecasts high traffic flows even after the decrease.

6.4.2 Using neighboring road probes

Results of Figure 6.6 show whether incorporating the information of neighboring road probes benefits Corso Orbassano traffic flow forecast.

Figures 6.6(a) and Figure 6.6(b) show the impact of the neighborhood size to the RMSE using *online training* in the *non-COVID-19* scenario. The experiment considered 5 min. and 60 min. *look-ahead time* values, and quantiles in Figure 6.5 as neighborhood distances.

The increase of the neighborhood leads to a growth of the training data, due to the additional information of the neighboring road probes. As shown in Figures 6.6(a)-6.6(d), no technique is capable of reducing the RMSE by having additional neighboring information. Among all of them, DES, TES and GRU do not decrease significantly their performance. But TCN does, since it convolves every feature present in the input matrix $X_{t,h}$, including also the distance to Corso Orbassano feature. By convolving such feature over the time domain, the NN cannot distinguish whether the input corresponds to a Corso Orbassano measurement or not. This phenomenon



(a) 5 min. *look-ahead* & *non-COVID-19*

(b) 60 min. *look-ahead* & *non-COVID-19*

(c) 5 min. *look-ahead* & *COVID-19*

(d) 60 min. *look-ahead* & *COVID-19*

Figure 6.6: Accuracy of 5 min. and 60 min. *look-ahead* forecasting from section 6.3, using *online training* and varying the neighboring stations.

prevents the TCN connections from giving less relevance to non-correlated measurements of irrelevant neighboring road probes. Note that HTM and LSTM results are not included, as HTM proprietary implementation could not receive multiple probes' flows as input, and both LSTM and GRU are achieving close performance results as shown in Figure 6.3. Most of the techniques discussed in this chapter should be able to achieve the same RSME value when they take the values of more stations into account than just the Orbassano station. Each technique can set the weights associated to the stations other than Orbassano to 0, washing out the influence of those additional stations completely. The fact that the training does not reach this situation (where all weights associated to stations other than Orbassano are set to 0) means that the training algorithm converges to local minimum rather than the global minimum, thus improvement of the training algorithm is possible.

6.5 V2N scaling with forecasting

To relate number of required resources with the quality of service, a queuing model is utilized. The cars represent the clients while the available automotive service instances represent the servers. A similarity between the handover process and service request is considered. It is assumed that when cars enter in the crossing area, they are requesting the service, as if mobile users handover into another cell.

For modeling the arrival process, it is necessary to select the arrival process of the cars into the service area. Previous studies have been conducted that provide careful models for automotive traffic [GFC14]. Similarly, the service time can be modeled as the residence time in a cell of a mobile user. Previous studies provided some models as reported, for example, in [KT02].

However, as this preliminary study focuses on assessing how forecasting can be beneficial for resource scaling, a simplified model for the arrival process and the service time is considered. The model is based on the $M/M/c$ queue, that is at the basis of circuit switching in communications

Algorithm 10: *n*-min. horizontal scaling

```
   Data: $n, \mu, T_0$
1  for $t \in \{i \cdot n/5min. : i > 0\}$ do
2      $\widehat{f}_{t+n-1}, \dots, \widehat{f}_{t+1}$ = forecast($X_{t,h}$);
3      $\widehat{F} = max\left\{\widehat{f}_{t+k}\right\}_{k=1}^{n-1}$;
4      $c = 1$;
5      $\rho = \widehat{F}/c\mu$;
6      while $\frac{1}{\mu} + \frac{P_Q}{c\mu - \widehat{F}} > T_0$ do
7          $c = c + 1$;
8      end
9      scale($c$);
10 end
```

networks [BGH92].

Thus, cars are assumed to enter in the coverage area of the service with a Poisson process with arrival rate λ and their residence time in the cell is exponentially distributed with average $\frac{1}{\mu}$. Consequently, the average time for which each vehicle spends in the system (i.e., waiting to receive the service and in the service) can be written as:

$$T = \frac{1}{\mu} + \frac{P_Q}{c\mu - \lambda}, \tag{6.13}$$

where c is the number of available servers and P_Q is the probability that an arrival finds all the services busy. The expression of P_Q is provided by the Erlang C formula:

$$P_Q = \frac{p_0 (c\rho)^c}{c!(1-\rho)}, \tag{6.14}$$

where $\rho = \frac{\lambda}{c\mu}$ and the probability p_0 of having zero clients in the system is:

$$p_0 = \left[\sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)}\right]^{-1}. \tag{6.15}$$

Given the flow rate of vehicles $\lambda = f_t$, and the latency specification T_0 of the V2N services explained in the beginning of this section (hazard warning, cooperative awareness, and remote driving), it is possible to derive the required number of virtualized service instances c to satisfy the average E2E latency ($T_0 = 5$ ms in the case of remote driving). More specifically, given the tuple (λ, μ), c is increased until the average delay formula reports a value of $T \leq T_0$. This is the approach used in Algorithm 10 to derive the required number of servers c in the horizontal scaling strategy presented in the next section.

Given the queuing theory framework, in this section it is presented how horizontal scaling is assessed for each of the three considered V2N services. The following paragraphs describe the procedure that uses the traffic forecasting results of section 6.4, to increase/decrease the number of servers c that will be needed to meet latency requirements. In particular, the proposed scaling solution leverages in the best forecasting techniques for each time-ahead and scenario – see Table 6.3.

5G-TRANSFORMER deliverable D5.4 [5GT19] reports the results of what is called an Enhanced Vehicular Service (EVS), that is a service that deploys sensing and video streaming and

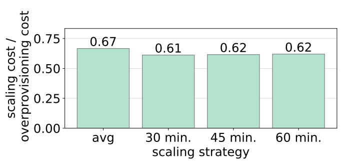
Table 6.3: Best traffic flow forecasting solutions

Forecasting task		Best solution	
Step-ahead	**Scenario**	**Technique**	**Online**
5 min	*non-COVID-19*	LSTM	✓
	COVID-19	TES	✓
15 min	*non-COVID-19*	TES	✓
	COVID-19	TES	✓
30 min	*non-COVID-19*	TES	✓
	COVID-19	TES	✓
45 min	*non-COVID-19*	TES	✓
	COVID-19	TES	✓
60 min	*non-COVID-19*	TES	✓
	COVID-19	TCNLSTM	✓

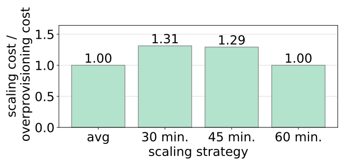
processing facilities in the edge. It reports not only the required physical resources to deploy an EVS service, but as well the flow of cars used to perform their evaluations. The document details that an EVS instance, i.e. $c = 1$ in our notation, offers a service rate of $\mu_{EVS} = 208.37$ vehicles/second. These values are taken as reference for the analysis of the three V2N services.

The whole purpose of the traffic flow forecasting of section 6.4 is to know whether a deployed V2N has enough resources to meet the E2E delay in an interval of up to 1 hour in the future. Thus, a V2N service can scale accordingly if the 5G network infrastructure receives as input the forecasting information. Three different scaling strategies are considered:

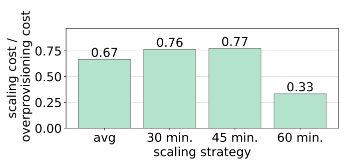
- *over-provisioning/max.scaling*: this strategy assumes that the V2N service is deployed with c instances capable of meeting the average E2E delay during peak hours of traffic;
- *avg. scaling*: the network dimensions the V2N service so that the c instances meet latency restrictions considering an average flow of vehicles; and
- *n-min. scaling*: based on the n-minutes ahead forecasting of section 6.4 techniques, the service is scaled to satisfy the peak of traffic forecasted for the next n minutes (see Algorithm 10).



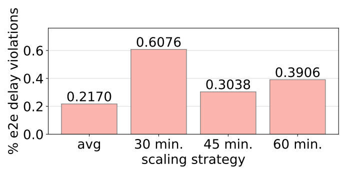
(a) Remote driving scaling savings



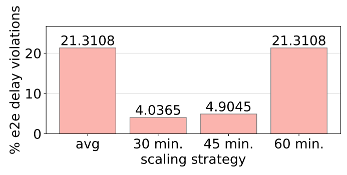
(b) Cooperative awareness scaling savings



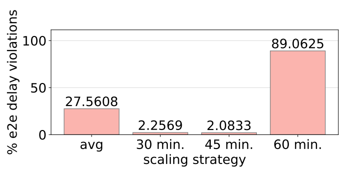
(c) Hazard warning scaling savings



(d) Remote driving scaling delay violations



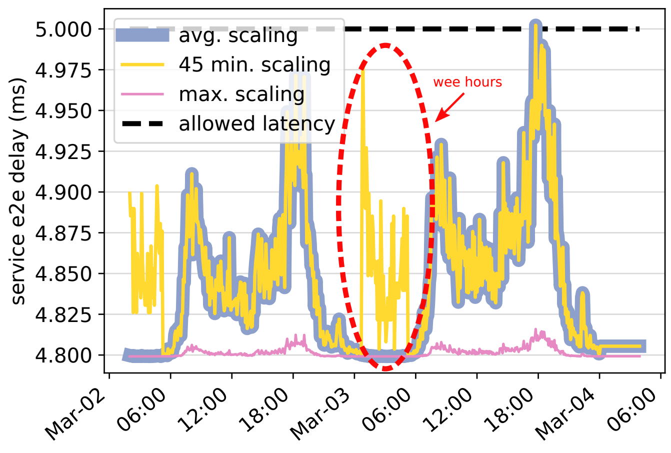
(e) Cooperative awareness delay violations



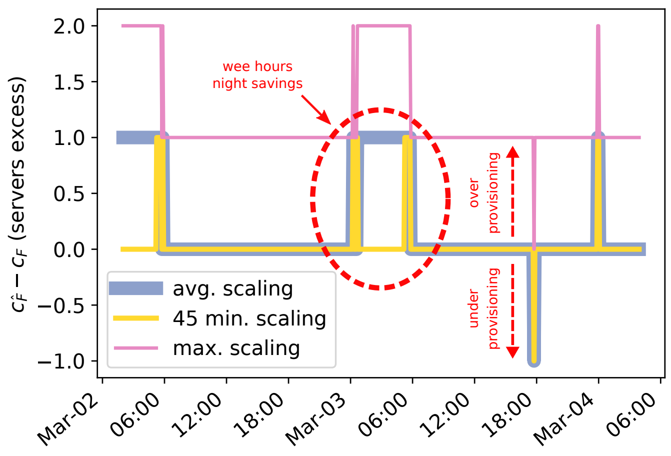
(f) Hazard warning delay violations

Figure 6.7: Cost savings and delay violations due to scaling. TES with online training was used for n-min. strategies (i.e., Algorithm 10).

Figure 6.7 compares the performance of *over-provisioning-scaling*, *avg. scaling*, and *n-min scaling* in the remote driving (see Figures 6.7 (a) and (d)), cooperative awareness (see Figures 6.7 (b) and (e)), and hazard warning (see Figures 6.7 (c) and (f)) V2N services. The three scaling strategies were tested under simulation in the *non-COVID-19* scenario, as the traffic flow was significantly higher than in the *COVID-19* scenario (see Figure 6.1). In every simulation n was set to 30, 45, and 60 minutes for the *n-min scaling* strategy, assuming scaling operations take less than 30 minutes. Figures 6.7 (a)-(c) compare the cost of *avg. scaling* and *n-min scaling* against *over-provisioning*



(a)



(b)

Figure 6.8: Impact of remote driving scaling on (a) delay violations, and (b) difference of forecasted $c_{\hat{F}}$ and required c_F servers. TES with online training was used for 45-min. scaling (i.e., Algorithm 10).

scaling; and Figures 6.7 (d)-(f) report the ratio of E2E violations.

For the remote driving simulations in Figures 6.7(a) and (d), the service rate, and target latency were set to $(\mu = \mu_{EVS}, T_0 = 5ms)$. Results show that *n-min.* reduces the costs with respect to both an *over-provisioning* strategy, and the *avg. scaling*. These savings are attained during the night, when the vehicular traffic on the streets drop and it is no longer necessary to have that many computing servers c to process the traffic. Figure 6.8(a) and Figure 6.8(b) depict the service e2e delay a excess of servers using the different scaling strategies. The night saving are appreciated in the wee hours of the morning of March-03 (see Figure 6.8(b)), when the *45 min. scaling* decreases by one the number of servers given the drop of traffic, whilst the *avg. scaling* keeps the same number of active servers, which results in a resource over provisioning leading to higher costs. Moreover, even though the *45 min. scaling* decreased the number of servers in the first hours of March-03, Figure 6.8(a) shows that still the service e2e delay remained below the 5 ms latency constraint. Although the reader might think that the *n-min. scaling* strategies might substantially increase the percentage of e2e delay violations, Figure 6.7(d) shows that at most, there is only an increase of $\leq 0.4\%$ of e2e delay violations over the simulated period. Additionally, such delay violations are assumable noticing the scaling savings, which go up to a 5% according to Figure 6.7(a).

In the cooperative awareness simulations' of Figure 6.7(b) and (e), the service rate and target latency were set to $(\mu = \mu_{EVS}/20, T_0 = 100ms)$. Given the target latency $T_0 = 100ms$, throughout the experiments it was enough to deploy only $c = 1$ instances of the service almost every time. Thus, both the *average scaling*, and *over-provisioning scaling* strategies cost the same (ratio equal to 1 in Figure 6.7(b)). In the case of the *n-min. scaling* strategy, setting up n to 30 and 45 minutes result into higher costs than the *over-provisioning scaling* (ratio above 1 on the left axis in Figure 6.7(b)), as both setups over-estimate the required resources. Consequently, *n-min. scaling* leads to less e2e delay violations than *avg. scaling* (above 4 times less, $\frac{21.3108\ \%}{4.9045\ \%} = 4.34$ to be specific).

For the last V2N service, the hazard warning, every simulation used a service rate and target latency of $(\mu = \mu_{EVS}/2, T_0 = 10ms)$. Figure 6.7(f) show that *n-min. scaling* with $n = 30$ and $n = 45$ achieve around 12 times ($\frac{27.5608\ \%}{2.2569\ \%} = 12.21$) less e2e delay violations than the *avg. scaling* strategy. The reduction of violations only incur in, at maximum, less than a 15 % ($\frac{0.77}{0.67} = 14.92\ \%$ of additional investment over the *avg. scaling* strategy. However, *n-min. scaling* underestimates the required resources when $n = 60$, and it incurs into more e2e delay violations than *avg. scaling*.

6.6 Conclusions

This chapter provides an analysis of state-of-the-art solutions to forecast the road traffic of Torino city, either leveraging on time-series analysis or neural networks. The performed analysis compares each forecasting technique's RMSE considering *(i)* forecasting intervals from 5 to 60 minutes, *(ii)* offline/online training; *(iii)* COVID-19 lockdown; and *(iv)* neighboring road probes. Results show that under offline training, neural network solutions outperform traditional time-series methods, especially during the COVID-19 lockdown, as they adapted to the Torino traffic drop. Whilst with online training, time-series techniques achieve results better or as good as the analyzed neural networks. However, none of the analyzed methods could benefit from information of neighboring stations. Experimental results confirm the benefits of using scaling based on traffic forecasting. Savings of up to a 5% only incur in an increase of $\leq 0,4\%$ of latency violations in the remote driving use case. For the cooperative awareness an extra 31% of investment achieved a 4-fold latency reduction, whilst for the hazard warning less than a 15% investment increase already resulted in a 12-fold latency reduction.

A first direction to extend this work is to find techniques that can incorporate neighboring road probes' information, such as spatial analysis techniques. Furthermore, the applicability of the presented techniques to different scenarios is also envisioned as a next step of this work. The use of different datasets, including operator records with respect to the base stations used by mobile phones to access the Internet, is going to be taken into consideration. In such scenario, forecasting the user density distribution along time would enable better decisions regarding the edge server placement and service migrations.

Similarly, to the adopted scaling strategy of this work, enhancing orchestration algorithms – see chapter 4 and chapter 5 – with forecasting information would contribute to a smarter orchestration and resource control. Resulting decisions would be impacted in terms of improved quality, accuracy, and optimality. Optimized deployment, enhanced management and control of elastic network slices that support dynamic demands and their respective Service Level Agreement (SLA)s, improved resource arbitration and allocation and maximized service request admission are some examples where forecasting information can impact the decisions.

The aforementioned mechanisms are going to be developed and leveraged in selected use cases in the scope of the 5Growth project, which comprises Industry 4.0, transportation and energy scenarios, targeting full support of automation and SLA control for vertical services life-cycle management. Hence, it would be worth-studying the probability of forecasting less demand than what is required by each use case, i.e., $\mathbb{P}(\hat{F} < F)$; so as to perform preemptive actions under high probabilities of forecasting error. Such a calculus deserves a detailed analysis on how to compute max-statistics for correlated random variables (e.g. speed and traffic flow) [MP14].

7. Conclusions

This thesis aims to contribute to the state of the art of Network Function Virtualization (NFV) orchestration algorithms proposing candidate solutions to assess the resource allocation for 5G Network Service (NS)s. In particular, it describes methods to generate 5G infrastructure graphs that are useful to evaluate the theoretical performance of NFV orchestration algorithms. The proposed graphs model 5G networks, and range from the generation of Base Station (BS)s, up to the aggregation and core rings of network operators. Additionally, this thesis also considers graphs of federated networks where each administrative domain has a partial view of the whole infrastructure, depending on the resources that peering domains' share in the pool of federated resources.

Leveraging on the proposed network graphs, this thesis proposes an enhanced version of Depth-First Search (DFS) and Breadth-First Search (BFS) NFV orchestration algorithms to solve the Virtual Network Embedding (VNE) problem in federated networks with multiple peering domains sharing resources. The polynomial run-time complexity of these algorithms meet high acceptance ratios, and are tested in stress situations with scarce of resources. Motivated by the federation problem, this thesis also studies the problem of delegating NSs to federated domains upon price changes of the peering domains, and presents a Deep Q-Network (DQN) agent that achieves near optimal decisions to decide whether to federate or locally deploy the NSs. Experiments show that the agent learned how to prevent the local domain from loses upon price peaks, thus, maximizing the local domain revenue.

Later, the thesis presents OKpi, a NFV orchestration algorithm with polynomial run-time complexity that achieves near optimal solutions for the VNE problem. OKpi does the allocation of both network, and computational resources to meet the service latency, and reliability requirements of 5G NSs belonging to different network slices. To do so, it accounts for edge and fog resources, and leverages on a expanded graph construction that seeks a path to steer traffic, and allocate computing resources. The accuracy of the solution search is tuned by a granularity parameter γ that allows to approximate to the optimal solution. OKpi was tested in both smart-factory robotic, and automotive scenarios; on small and large scale settings. Additionally, it was validated and integrated in a real testbed of a cloud-robotic NS. Following the VNE problem study, this thesis also studies more fog-oriented use cases in which the volatility and mobility of fog nodes is considered. In particular it proposes an optimization model that accounts for battery consumption of fog devices as robots, so as their mobility and radio coverage requirements. The thesis presents a heuristic algorithm based on a randomized-rounding solution of a variable-size variation of the bin packing problem, and compares its performance against optimal solutions, and a state of the art NFV orchestration algorithm designed for fog scenarios. Experiments show that with the proposed system model, the proposed heuristic achieves near optimal solutions that satisfy volatility, mobility, and radio coverage constrains in a simulated warehousing use case with cloud-robotics.

Finally, the thesis shifts its focus to the study of how to scale Vehicle-to-Network (V2N) services, so the allocated network resources meet the strict End-to-End (E2E) latency constraints upon peaks of traffic. To do so, this thesis studies and compare the accuracy of time-series techniques to forecast the vehicular traffic using a real dataset of Torino city, which includes data of the COVID-19 pandemic. The best traffic forecasting technique is later used to feed a $M/M/c$-based queuing model to assess the scaling of computing resources, and anticipate to increases/decreases of traffic flows. Results show that the proposed solution achieves either monetary savings or reduce the E2E latency violations in the analyzed V2N network services, with their respective latency requirements, namely, in the hazard warning, remote driving, and cooperative awareness V2N services.

8. Future work

The following paragraphs point out future research directions for the solutions proposed in this thesis.

Chapter 3 proposes a methodology to generate graphs that represent 5G reference infrastructures, and multi-domain networks. However, it misses creating graphs conveying both. That is, it is still left to generate 5G infrastructure graphs in which the data centers of various domains are considered. Such graphs would be beneficial for the evaluation of Network Function Virtualization (NFV) orchestration algorithms in federated 5G networks, as they could be tested on more realistic scenarios not only accounting for data centers, but also the network infrastructure that interconnects them with the users.

Regarding the multi-domain NFV orchestration algorithms presented in section 4.2.1, a future research work would be the analysis of their optimality gap, so as their evaluation in graphs accounting for the end to end 5G network infrastructure (as stated in the previous paragraph).

On the Network Service (NS) deployment delegation algorithms proposed in 4.3, the problem statement has to be modified to account for more than only an additional federated domain, so the federation agent (e.g., the Deep Q-Network (DQN) agent) faces a more realistic scenario. Additionally, the Q-table and DQN agents could be enhanced with time-series techniques to anticipate to pricing fluctuations. For example, the DQN agent could have an LSTM layer to record in memory cells temporal characteristics of the prices time-series. In such a way, the Neural Network (NN) will benefit to take more optimal decisions on whether to delegate or locally deploy the NSs.

About the OKpi heuristic proposed in section 5.1, future work can study the impact of the resolution parameter γ in the algorithm run-time. The optimality analysis showed that a higher value of γ led to solutions closer to optimality. However, OKpi run-time increases with γ, and large running times are unfeasible if the NS deployment decision must be taken in a short time-span. Thus, future work on OKpi should focus on studying the γ and run-time trade-off.

The NFV orchestration algorithm presented in section 5.2 was successfully evaluated using reference infrastructure graphs. But the coverage, and battery consumption functions were simplified. In particular, to determine whether the robot had coverage, the formulation only checked if the distance of the robot towards an antenna was lower than a threshold. But in a real scenario the coverage is impacted by interference, signal strength, and path loss. Therefore, a future direction would be to improve the coverage constraint considering analytical coverage models. Similarly, the battery consumption of a robot was simplified using a linear function decreasing with the number of hosted Virtual Network Function (VNF)s. Future versions of section 5.2 should consider analytical models on battery consumption on the battery level constraints.

Also, the two NFV orchestration algorithms of chapter 5 may be extended to account the allocation of Radio Access Network (RAN) resources. That way, the NSs embedding solutions will be more reliable, as they will also consider the impact of latency and reliability in the user to RAN communication, something both solutions miss in their analysis (or simplify by having a fixed parameter of latency and reliability in the user to RAN communication).

Last of all, the Vehicle-to-Network (V2N) forecasting and scaling procedure of chapter 6 can be extended by checking other prediction algorithms as Spatio-Temporal Graph Convolutional Networks (STGCN)s [YYZ18], or Facebook prophet [TL18]. Moreover, the $M/M/c$-based scaling mechanism may be further improved, as it is designed to meet on average the latency constraints. But this may not be enough to meet 99.9999% reliability imposed by some V2N services. Thus, future work should base the scaling decision using the probability distribution function of the service time in $M/M/c$ systems, so the 99.9999% of V2N users experience the required latency.

Bibliography

References

Articles

[AD18] M. Afshang and H. S. Dhillon. 'Poisson Cluster Process Based Analysis of HetNets With Correlated User and Base Station Locations'. In: *IEEE Transactions on Wireless Communications* 17.4 (April 2018), pages 2417–2431. ISSN: 1536-1276. DOI: 10.1109/TWC.2018.2794983 (cited on page 82).

[Aga+19] Satyam Agarwal et al. 'VNF Placement and Resource Allocation for the Support of Vertical Services in 5G Networks'. In: *IEEE/ACM Trans. Netw.* 27.1 (February 2019), pages 433–446. ISSN: 1063-6692. DOI: 10.1109/TNET.2018.2890631. URL: https://doi.org/10.1109/TNET.2018.2890631 (cited on page 76).

[Ahm+17] Subutai Ahmad et al. 'Unsupervised real-time anomaly detection for streaming data'. In: *Neurocomputing* 262 (2017). Online Real-Time Learning Strategies for Data Streams, pages 134–147. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.04.070 (cited on page 172).

[Ant+20] K. Antevski et al. 'A Q-learning strategy for federation of 5G services'. In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 2020, pages 1–6. DOI: 10.1109/ICC40277.2020.9149082 (cited on pages 14, 116, 117, 121).

[Ant+18] K. Antevski et al. 'Resource Orchestration of 5G Transport Networks for Vertical Industries'. In: *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2018, pages 158–163. DOI: 10.1109/PIMRC.2018.8581029 (cited on page 14).

[Aqi+17] Muhammad Aqib et al. 'Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs'. In: *International Conference on Smart Cities, Infrastructure, Technologies and Applications*. Springer. 2017, pages 139–154 (cited on page 174).

[Avi+19] G. Avino et al. 'A MEC-based Extended Virtual Sensing for Automotive Services'. In: *IEEE Transactions on Network and Service Management* 16.4 (2019), pages 1450–1463 (cited on page 143).

[Aya+20] J. A. Ayala-Romero et al. 'vrAIn: Deep Learning based Orchestration for Computing and Radio Resources in vRANs'. In: *IEEE Transactions on Mobile Computing* (2020), pages 1–1. DOI: 10.1109/TMC.2020.3043100 (cited on page 59).

[Bau+19] Matt Baughman et al. 'Deconstructing the 2017 Changes to AWS Spot Market Pricing'. In: *Proceedings of the 10th Workshop on Scientific Cloud Computing*. ScienceCloud '19. Phoenix, AZ, USA: Association for Computing Machinery, 2019, pages 19–26. ISBN: 9781450367585. DOI: 10.1145/3322795.3331465. URL: https://doi.org/10.1145/3322795.3331465 (cited on page 111).

[BEL57] RICHARD BELLMAN. 'A Markovian Decision Process'. In: *Journal of Mathematics and Mechanics* 6.5 (1957), pages 679–684. ISSN: 00959057, 19435274. URL: http://www.jstor.org/stable/24900506 (cited on pages 115, 117, 118).

[BSF94] Yoshua Bengio, Patrice Simard and Paolo Frasconi. 'Learning long-term dependencies with gradient descent is difficult'. In: *IEEE transactions on neural networks* 5.2 (1994), pages 157–166 (cited on page 173).

[Cam+13] Hadrien Cambazard et al. 'Bin Packing with Linear Usage Costs - An Application to Energy Management in Data Centres'. In: *Principles and Practice of Constraint Programming - 19th International Conference* (2013). Best Application Paper Award, pages 47–62. URL: https://hal.archives-ouvertes.fr/hal-00858159 (cited on page 155).

[CDJ17] F. Carpio, S. Dhahri and A. Jukan. 'VNF placement with replication for Loac balancing in NFV networks'. In: *2017 IEEE International Conference on Communications (ICC)*. 2017, pages 1–6. DOI: 10.1109/ICC.2017.7996515 (cited on page 74).

[Cas+18] Claudio Casetti et al. 'Arbitration Among Vertical Services'. In: *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2018, pages 153–157. DOI: 10.1109/PIMRC.2018.8580852 (cited on page 13).

[Cav+17] F. Cavaliere et al. 'Towards a unified fronthaul-backhaul data plane for 5G The 5G-Crosshaul project approach'. In: *Computer Standards & Interfaces* 51 (2017), pages 56–62. ISSN: 0920-5489. DOI: 10.1016/j.csi.2016.11.005 (cited on page 90).

[CL19] Yan-Ting Chen and Wanjiun Liao. 'Mobility-aware service function chaining in 5g wireless networks with mobile edge computing'. In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pages 1–6 (cited on page 159).

[Cho+14] Kyunghyun Cho et al. 'On the Properties of Neural Machine Translation: Encoder–Decoder Approaches'. In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (2014). DOI: 10.3115/v1/w14-4012 (cited on page 173).

[CFA09] Massimo Cocozza, Giovanni Foti and Fabrizio Arneodo. 'S.I.MO.NE. Innovative System for Metropolitan Area Mobility Management'. In: *16th ITS World Congress and Exhibition on Intelligent Transport Systems and ServicesITS AmericaERTICOITS Japan*. 2009 (cited on page 169).

[Coh+15a] R. Cohen et al. 'Near optimal placement of virtual network functions'. In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. 2015, pages 1346–1354. DOI: 10.1109/INFOCOM.2015.7218511 (cited on page 75).

[Coh+15b] R. Cohen et al. 'Near optimal placement of virtual network functions'. In: *IEEE INFOCOM*. 2015 (cited on pages 133, 136).

[Com+18a] L. Cominardi et al. 'Understanding QoS Applicability in 5G Transport Networks'. In: *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. June 2018, pages 1–5. DOI: 10.1109/BMSB.2018.8436847. URL: https://e-archivo.uc3m.es/bitstream/handle/10016/27393/understanding_BMSB_2018_ps.pdf (Retrieved: 10th January 2019) (cited on pages 60, 83, 88, 90, 92).

[Com+18b] L. Cominardi et al. 'Understanding QoS Applicability in 5G Transport Networks'. In: *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. June 2018, pages 1–5. DOI: 10.1109/BMSB.2018.8436847 (cited on pages 150, 160).

[Con20] Contreras, Luis M. and Cominardi, Luca and **Martín Pérez, Jorge** and Bernardos, Carlos J. 'Applicability of SDN and NFV Techniques for a Virtualization-Based Roaming Solution'. In: *Journal of Network and Systems Management* 28 (July 2020), pages 576–604. DOI: 10.1007/s10922-020-09534-z. URL: https://doi.org/10.1007/s10922-020-09534-z (cited on pages 10, 103).

[CB18] Angelo Corsaro and Gabriele Baldoni. 'fogØ5: Unifying the computing, networking and storage fabrics end-to-end'. In: *2018 3rd Cloudification of the Internet of Things (CIoT)*. IEEE. 2018, pages 1–8 (cited on page 56).

[CN06] Gabor Csardi and Tamas Nepusz. 'The igraph software package for complex network research'. In: *InterJournal* Complex Systems (2006), page 1695. URL: http://igraph.org (cited on page 98).

[Fai08] Florian Fainelli. 'The OpenWrt embedded development framework'. In: *Proceedings of the Free and Open Source Software Developers European Meeting*. sn. 2008, page 106 (cited on page 146).

[Fen+17] Hao Feng et al. 'Approximation algorithms for the NFV service distribution problem'. In: *IEEE INFOCOM*. 2017 (cited on page 136).

[FGK93] Robert Fourer, David M Gay and Brian W Kernighan. 'AMPL. A modeling language for mathematical programming'. In: (1993) (cited on page 160).

[Fra+17a] V. Frascolla et al. '5G-MiEdge: Design, standardization and deployment of 5G phase II technologies: MEC and mmWaves joint development for Tokyo 2020 Olympic games'. In: *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*. 2017, pages 54–59. DOI: 10.1109/CSCN.2017.8088598 (cited on pages 44, 49).

[Fra+17b] V. Frascolla et al. '5G-MiEdge: Design, standardization and deployment of 5G phase II technologies: MEC and mmWaves joint development for Tokyo 2020 Olympic games'. In: *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*. September 2017, pages 54–59. DOI: 10.1109/CSCN.2017.8088598 (cited on page 82).

[Geo72] Arthur M Geoffrion. 'Generalized benders decomposition'. In: *Journal of optimization theory and applications* 10.4 (1972), pages 237–260 (cited on page 76).

[Geo+19a] Gareth George et al. 'Analyzing AWS spot instance pricing'. In: *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE. 2019, pages 222–228 (cited on page 111).

[Geo+19b] Gareth George et al. 'Analyzing AWS spot instance pricing'. In: *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE. 2019, pages 222–228 (cited on page 111).

[GMZ16] Chaima Ghribi, Marouen Mechtri and Djamal Zeghlache. 'A Dynamic Programming Algorithm for Joint VNF Placement and Chaining'. In: *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*. CAN '16. Irvine, California, USA: Association for Computing Machinery, 2016, pages 19–24. ISBN: 9781450346733. DOI: 10.1145/3010079.3010083. URL: https://doi.org/10.1145/3010079.3010083 (cited on page 73).

[GCR17] Fabio Giust, Xavier Costa-Perez and Alex Reznik. 'Multi-Access Edge Computing: An Overview of ETSI MEC ISG'. In: *IEEE 5G Tech Focus* 1.4 (2017) (cited on page 81).

[GFC14] M. Gramaglia, M. Fiore and M. Calderon. 'Measurement-Based Modeling of Interarrivals for the Simulation of Highway Vehicular Networks'. In: *IEEE Communications Letters* 18.12 (2014), pages 2181–2184 (cited on page 178).

[Gro+21a] Milan Groshev et al. 'COTORRA: COntext-aware Testbed fOR Robotic Applications'. In: *MobiSys2021: the 19th ACM International Conference on Mobile Systems, Applications, and Services*. Accepted in the 1st Workshop on Serverless mobile networking for 6G Communications. Association for Computing Machinery, 2021. arXiv: 2101.07676 [cs.RO] (cited on page 14).

[Gro+21b] Milan Groshev et al. 'Towards Intelligent Cyber-Physical Systems: Digital Twin meets Artificial Intelligence'. In: *IEEE Communications Magazine* (2021). Accepted with major review (cited on page 15).

[Gur15] Incorporate Gurobi Optimization. 'Gurobi optimizer reference manual'. In: *URL http://www. gurobi. com* (2015) (cited on page 160).

[HSS08] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart. 'Exploring Network Structure, Dynamics, and Function using NetworkX'. In: *Proceedings of the 7th Python in Science Conference*. Edited by Gaël Varoquaux, Travis Vaught and Jarrod Millman. Pasadena, CA USA, 2008, pages 11–15 (cited on page 107).

[Hal+18] B. Halvarsson et al. '5G NR Testbed 3.5 GHz Coverage Results'. In: *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. June 2018, pages 1–5. DOI: 10.1109/VTCSpring.2018.8417704 (cited on page 161).

[Hem+05] Stephen Hemminger et al. 'Network emulation with NetEm'. In: *Linux conf au*. 2005, pages 18–23 (cited on page 146).

[HK16] A. Hirwe and K. Kataoka. 'LightChain: A lightweight optimisation of VNF placement for service chaining in NFV'. In: *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. 2016, pages 33–37. DOI: 10.1109/NETSOFT.2016.7502438 (cited on page 73).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. 'Long Short-Term Memory'. In: *Neural Comput.* 9.8 (November 1997), pages 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735 (cited on page 173).

[Hu+15] Yun Chao Hu et al. 'Mobile edge computing – A key technology towards 5G'. In: *ETSI White Paper* 11 (2015) (cited on page 81).

[IEE13] A. M. Ibrahim, T. ElBatt and A. El-Keyi. 'Coverage probability analysis for wireless networks using repulsive point processes'. In: *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. September 2013, pages 1002–1007. DOI: 10.1109/PIMRC.2013.6666284 (cited on page 82).

[JPP16] Fatma Ben Jemaa, Guy Pujolle and Michel Pariente. 'QoS-aware VNF placement optimization in edge-central carrier cloud architecture'. In: *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2016, pages 1–7 (cited on page 133).

[Jin+20] P. Jin et al. 'Latency-aware VNF Chain Deployment with Efficient Resource Reuse at Network Edge'. In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 2020, pages 267–276. DOI: `10.1109/INFOCOM41043.2020.9155345` (cited on page 77).

[Jor+21a] **Jorge Martín-Pérez** et al. 'Dimensioning of Vehicle-to-Network (V2N) Services in 5G Networks through Forecast-based Scaling'. In: (2021). Submitted. arXiv: `2105.12527 [cs.NI]` (cited on pages 10, 16).

[Jor+20] **Jorge Martín-Pérez** et al. 'DQN Dynamic Pricing and Revenue driven Service Federation Strategy'. In: (March 2020). Submitted (cited on page 10).

[Jor+21b] **Jorge Martín-Pérez** et al. 'KPI Guarantees in Network Slicing'. In: *IEEE/ACM Transactions on Networking* (2021). Accepted with minor changes (cited on page 11).

[KYM19] Khashayar Kamran, Edmund Yeh and Qian Ma. 'DECO: Joint Computation, Caching and Forwarding in Data-Centric Computing Networks'. In: *ACM MobiHoc*. 2019 (cited on page 131).

[Keh+15] Ben Kehoe et al. 'A survey of research on cloud robotics and automation'. In: *IEEE Transactions on automation science and engineering* 12.2 (2015), pages 398–409 (cited on page 147).

[Kle67] Leonard Kleinrock. 'Time-shared systems: A theoretical treatment'. In: *Journal of the ACM (JACM)* 14.2 (1967), pages 242–261 (cited on page 76).

[KT02] S. Kourtis and R. Tafazolli. 'Modelling cell residence time of mobile terminals in cellular radio systems'. In: *Electronics Letters* 38.1 (2002), pages 52–54 (cited on page 178).

[Kuh55] Harold W Kuhn. 'The Hungarian method for the assignment problem'. In: *Naval research logistics quarterly* 2.1-2 (1955), pages 83–97 (cited on page 76).

[Kuo+16] T. Kuo et al. 'Deploying chains of virtual network functions: On the relation between link and server usage'. In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 2016, pages 1–9. DOI: `10.1109/INFOCOM.2016.7524565` (cited on page 74).

[Lan+19] Jeffrey Lancon et al. 'AWS EC2 Instance Spot Price Forecasting Using LSTM Networks'. In: *SMU Data Science Review* 2.2 (2019), page 8 (cited on page 111).

[Lei85] Charles E Leiserson. 'Fat-trees: universal networks for hardware-efficient supercomputing'. In: *IEEE transactions on Computers* 100.10 (1985), pages 892–901 (cited on pages 75, 99, 100, 106).

[LQ15] X. Li and C. Qian. 'The virtual network function placement problem'. In: *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2015, pages 69–70. DOI: `10.1109/INFCOMW.2015.7179347` (cited on page 73).

[Ma+17] W. Ma et al. 'Traffic aware placement of interdependent NFV middleboxes'. In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 2017, pages 1–9. DOI: `10.1109/INFOCOM.2017.8056993` (cited on page 76).

[Mac+67] James MacQueen et al. 'Some methods for classification and analysis of multivariate observations'. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Volume 1. 14. Oakland, CA, USA. 1967, pages 281–297 (cited on page 77).

[MP14] Satya N Majumdar and Arnab Pal. ‘Extreme value statistics of correlated random variables’. In: *arXiv preprint arXiv:1406.6768* (2014) (cited on page 182).

[MC19] F. Malandrino and C. Chiasserini. ‘Getting the Most Out of Your VNFs: Flexible Assignment of Service Priorities in 5G’. In: *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*. 2019, pages 1–9. DOI: `10.1109/WoWMoM.2019.8792983` (cited on page 76).

[Mal+20] M. Malinverno et al. ‘Edge-Based Collision Avoidance for Vehicles and Vulnerable Users: An Architecture Based on MEC’. In: *IEEE Vehicular Technology Magazine* 15.1 (2020), pages 27–35 (cited on page 143).

[Man+19] J. Mangues-Bafalluy et al. ‘5G-TRANSFORMER Service Orchestrator: design, implementation, and evaluation’. In: *2019 European Conference on Networks and Communications (EuCNC)*. 2019, pages 31–36. DOI: `10.1109/EuCNC.2019.8802038` (cited on pages 14, 44, 45).

[MB18] **Martín-Pérez, J**. and Carlos J. Bernardos. ‘Multi-Domain VNF Mapping Algorithms’. In: *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. 2018, pages 1–6. DOI: `10.1109/BMSB.2018.8436765` (cited on page 9).

[Mar+20] **Martín-Peréz, J.** et al. ‘OKpi: All-KPI Network Slicing Through Efficient Resource Allocation’. In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 2020, pages 804–813. DOI: `10.1109/INFOCOM41043.2020.9155263` (cited on page 11).

[Mar+19a] **Martín-Pérez, Jorge** et al. ‘5GEN: A tool to generate 5G infrastructure graphs’. In: *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*. 2019, pages 1–4. DOI: `10.1109/CSCN.2019.8931334` (cited on pages 9, 16).

[Mar+19b] **Martín-Pérez, Jorge** et al. ‘Modeling Mobile Edge Computing Deployments for Low Latency Multimedia Services’. In: *IEEE Transactions on Broadcasting* 65.2 (2019), pages 464–474. DOI: `10.1109/TBC.2019.2901406` (cited on page 9).

[Mar+19c] Jorge Martıén-Pérez et al. ‘Modeling mobile edge computing deployments for low latency multimedia services’. In: *IEEE Transactions on Broadcasting* (2019) (cited on page 144).

[MGZ16] M. Mechtri, C. Ghribi and D. Zeghlache. ‘A Scalable Algorithm for the Placement of Service Function Chains’. In: *IEEE Transactions on Network and Service Management* 13.3 (2016), pages 533–546. DOI: `10.1109/TNSM.2016.2598068` (cited on page 73).

[Mni+13] Volodymyr Mnih et al. ‘Playing atari with deep reinforcement learning’. In: *arXiv preprint arXiv:1312.5602* (2013) (cited on page 118).

[Muh+18] A. Muhammad et al. ‘On the Scalability of Connectivity Services in a Multi-Operator Orchestrator Sandbox’. In: *2018 Optical Fiber Communications Conference and Exposition (OFC)*. 2018, pages 1–3 (cited on page 13).

[Nem+21] Balazs Nemeth et al. ‘Delay and reliability-constrained VNF placement on mobile and volatile 5G infrastructure’. In: *IEEE Transactions on Mobile Computing* (2021), pages 1–1. DOI: `10.1109/TMC.2021.3055426` (cited on page 10).

[NS58] Jerzy Neyman and Elizabeth L. Scott. ‘Statistical Approach to Problems of Cosmology’. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 20.1 (1958), pages 1–43. ISSN: 00359246. URL: `http://www.jstor.org/stable/2983905` (cited on page 82).

[NJ14a] Vladimir Nikolikj and Toni Janevski. ‘A cost modeling of high-capacity LTE-advanced and IEEE 802.11 ac based heterogeneous networks, deployed in the 700 MHz, 2.6 GHz and 5 GHz Bands’. In: *Procedia Computer Science* 40 (2014), pages 49–56 (cited on page 161).

[NJ14b] Vladimir Nikolikja and Toni Janevski. ‘A Cost Modeling of High-Capacity LTE-Advanced and IEEE 802.11ac based Heterogeneous Networks, Deployed in the 700 MHz, 2.6 GHz and 5 GHz Bands’. In: *Procedia Computer Science* (2014) (cited on page 142).

[Niu+15] Yong Niu et al. ‘A survey of millimeter wave communications (mmWave) for 5G: opportunities and challenges’. In: *Wireless networks* 21.8 (2015), pages 2657–2676 (cited on page 34).

[Nog+18] Borja Nogales et al. ‘Adaptable and automated small uav deployments via virtualization’. In: *Sensors* 18.12 (2018), page 4116 (cited on pages 150, 153).

[Oli+18] Antonio de la Oliva et al. ‘5G-TRANSFORMER: Slicing and orchestrating transport networks for industry verticals’. In: *IEEE Communications Magazine* 56.8 (2018), pages 78–84 (cited on pages 62, 98).

[Oli+19] Pierre Olivier et al. ‘A binary-compatible unikernel’. In: *Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. 2019, pages 59–73 (cited on page 56).

[Olj+17] Dejene Boru Oljira et al. ‘A model for QoS-aware VNF placement and provisioning’. In: *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE. 2017, pages 1–7 (cited on page 133).

[Pas+14] Razvan Pascanu et al. ‘How to construct deep recurrent neural networks’. English (US). In: *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*. 2014 (cited on page 174).

[Pat+18] N. Patriciello et al. ‘5G New Radio Numerologies and their Impact on the End-To-End Latency’. In: *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. September 2018, pages 1–6. DOI: `10.1109/CAMAD.2018.8514979` (cited on page 161).

[Pre+18] Gopika Premsankar et al. ‘Efficient placement of edge computing devices for vehicular applications in smart cities’. In: *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE. 2018, pages 1–9 (cited on page 168).

[Qaz+] Zafar Ayyub Qazi et al. ‘SIMPLE-fying middlebox policy enforcement using SDN’. In: *ACM SIGCOMM computer communication review* () (cited on page 133).

[RB14] Tiago Rosado and Jorge Bernardino. ‘An overview of openstack architecture’. In: *Proceedings of the 18th International Database Engineering & Applications Symposium*. 2014, pages 366–367 (cited on page 41).

[Sac+18] J. Sachs et al. ‘5G Radio Network Design for Ultra-Reliable Low-Latency Communication’. In: *IEEE Network* 32.2 (March 2018), pages 24–31. ISSN: 0890-8044. DOI: `10.1109/MNET.2018.1700232` (cited on page 91).

[Sai+15] T. N. Sainath et al. ‘Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks’. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pages 4580–4584 (cited on page 174).

[San+20] Victor Sanchez-Aguero et al. ‘Energy-Aware Management in Multi-UAV Deployments: Modelling and Strategies’. In: *Sensors* 20.10 (2020), page 2791 (cited on page 153).

[San+17a] Y. Sang et al. 'Provably efficient algorithms for joint placement and allocation of virtual network functions'. In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 2017, pages 1–9. DOI: 10.1109/INFOCOM.2017.8057036 (cited on page 74).

[San+17b] Yu Sang et al. 'Provably efficient algorithms for joint placement and allocation of virtual network functions'. In: *IEEE INFOCOM*. 2017 (cited on page 136).

[SLK18] Alkharif Sarah, Kyungyong Lee and Hyeokman Kim. 'LSTM model to forecast time series for EC2 cloud price'. In: *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE. 2018, pages 1085–1088 (cited on page 111).

[SYC18] V. Sciancalepore, F. Z. Yousaf and X. Costa-Perez. 'z-TORCH: An Automated NFV Orchestration and Monitoring Solution'. In: *IEEE Transactions on Network and Service Management* 15.4 (2018), pages 1292–1306. DOI: 10.1109/TNSM.2018.2867827 (cited on page 77).

[Sei95] Raimund Seidel. 'On the all-pairs-shortest-path problem in unweighted undirected graphs'. In: *Journal of computer and system sciences* (1995) (cited on page 141).

[Sga+19] A. Sgambelluri et al. 'Orchestrating QoS-based Connectivity Services in a Multi-Operator Sandbox'. In: *J. Opt. Commun. Netw.* 11.2 (February 2019), A196–A208. DOI: 10.1364/JOCN.11.00A196. URL: http://jocn.osa.org/abstract.cfm?URI=jocn-11-2-A196 (cited on page 14).

[Sga+17] A. Sgambelluri et al. 'Orchestration of Network Services across multiple operators: The 5G Exchange prototype'. In: *2017 European Conference on Networks and Communications (EuCNC)*. 2017, pages 1–5. DOI: 10.1109/EuCNC.2017.7980666 (cited on pages 13, 44, 45).

[Shi+16] W. Shi et al. 'Edge Computing: Vision and Challenges'. In: *IEEE Internet of Things Journal* 3.5 (2016), pages 637–646 (cited on page 168).

[Shi+10] Z. Shi et al. 'Effects of Packet Loss and Latency on the Temporal Discrimination of Visual-Haptic Events'. In: *IEEE Transactions on Haptics* 3.1 (January 2010), pages 28–36. ISSN: 1939-1412. DOI: 10.1109/TOH.2009.45 (cited on page 81).

[SC20a] A. Solano and L. M. Contreras. 'Information Exchange to Support Multi-Domain Slice Service Provision for 5G/NFV'. In: *2020 IFIP Networking Conference (Networking)*. 2020, pages 773–778 (cited on pages 112, 113, 120, 121, 124–126).

[Son+15] Balázs Sonkoly et al. 'Multi-Domain Service Orchestration Over Networks and Clouds: A Unified Approach'. In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. SIGCOMM '15. London, United Kingdom: Association for Computing Machinery, 2015, pages 377–378. ISBN: 9781450335423. DOI: 10.1145/2785956.2790041. URL: https://doi.org/10.1145/2785956.2790041 (cited on page 98).

[SMF15a] V. Suryaprakash, J. Møller and G. Fettweis. 'On the Modeling and Analysis of Heterogeneous Radio Access Networks Using a Poisson Cluster Process'. In: *IEEE Transactions on Wireless Communications* 14.2 (February 2015), pages 1035–1047. ISSN: 1536-1276. DOI: 10.1109/TWC.2014.2363454 (cited on page 82).

[SMF15b] V. Suryaprakash, J. Møller and G. Fettweis. 'On the Modeling and Analysis of Heterogeneous Radio Access Networks Using a Poisson Cluster Process'. In: *IEEE Transactions on Wireless Communications* 14.2 (2015), pages 1035–1047. DOI: 10.1109/TWC.2014.2363454 (cited on page 152).

[SRF15] V. Suryaprakash, P. Rost and G. Fettweis. 'Are Heterogeneous Cloud-Based Radio Access Networks Cost Effective?' In: *IEEE Journal on Selected Areas in Communications* 33.10 (October 2015), pages 2239–2251. ISSN: 0733-8716. DOI: 10.1109/JSAC.2015.2435275 (cited on page 82).

[Sut+19] Marco Suter et al. 'Fog Application Allocation for Automation Systems'. In: *2019 IEEE International Conference on Fog Computing (ICFC)*. IEEE. 2019, pages 97–106 (cited on page 160).

[SBD18] Meenakshi Syamkumar, Paul Barford and Ramakrishnan Durairajan. 'Deployment Characteristics of "The Edge" in Mobile Edge Computing'. In: *Proceedings of the 2018 Workshop on Mobile Edge Communications*. MECOMM'18. Budapest, Hungary: ACM, 2018, pages 43–49. ISBN: 978-1-4503-5906-1. DOI: 10.1145/3229556.3229557. URL: http://doi.acm.org/10.1145/3229556.3229557 (cited on page 82).

[TVi75] T.Vicenty. 'DIRECT AND INVERSE SOLUTIONS OF GEODESICS 0N THE ELLIPSOID WLTH APPLICATION OF ESTED EQUATIO S'. In: *DMAAC Geodetic Survey Squadron* (1975) (cited on page 96).

[Tan11] O. Tange. 'GNU Parallel - The Command-Line Power Tool'. In: *;login: The USENIX Magazine* 36.1 (February 2011), pages 42–47. URL: http://www.gnu.org/s/parallel (cited on page 107).

[TL18] Sean J Taylor and Benjamin Letham. 'Forecasting at scale'. In: *The American Statistician* 72.1 (2018), pages 37–45 (cited on page 186).

[Tok+20] Laszlo Toka et al. 'Pricing games of NFV infrastructure providers'. In: *Telecommunication Systems* (2020), pages 1–14 (cited on page 111).

[Ume88] S. Umeyama. 'An eigendecomposition approach to weighted graph matching problems'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10.5 (1988), pages 695–703. DOI: 10.1109/34.6778 (cited on page 74).

[Vin75] T. Vincenty. 'Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations'. In: *Survey Review* 23.176 (1975), pages 88–93 (cited on page 91).

[Vle+21] Danny de Vleeschauwer et al. '5Growth Data-driven AI-based Scaling'. In: *2021 European Conference on Networks and Communications (EuCNC)*. Accepted for publication. 2021 (cited on pages 15, 16).

[XL13] Hong Xu and Baochun Li. 'Dynamic cloud pricing for revenue maximization'. In: *IEEE Transactions on Cloud Computing* 1.2 (2013), pages 158–171 (cited on pages 112, 121).

[XF10] J. Xu and J. A. B. Fortes. 'Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments'. In: *2010 IEEE/ACM Int'l Conference on Green Computing and Communications Int'l Conference on Cyber, Physical and Social Computing*. 2010, pages 179–188. DOI: 10.1109/GreenCom-CPSCom.2010.137 (cited on page 74).

[Xue+07] Guoliang Xue et al. 'Finding a path subject to many additive QoS constraints'. In: *IEEE/ACM Transactions on Networking* (2007) (cited on pages 136, 138, 139).

[YYZ18] Bing Yu, Haoteng Yin and Zhanxing Zhu. 'Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting'. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (July 2018). DOI: 10.24963/ijcai.2018/505. URL: http://dx.doi.org/10.24963/ijcai.2018/505 (cited on page 186).

[ZLZ19] Q. Zhang, F. Liu and C. Zeng. 'Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices'. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 2019, pages 2449–2457. DOI: `10.1109/INFOCOM.2019.8737660` (cited on page 78).

[ZD18] P. Zhao and G. Dán. 'A Benders Decomposition Approach for Resilient Placement of Virtual Process Control Functions in Mobile Edge Clouds'. In: *IEEE Transactions on Network and Service Management* 15.4 (2018), pages 1460–1472. DOI: `10.1109/TNSM.2018.2873178` (cited on page 76).

Books

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009 (cited on pages 70, 72).

[BeW07] A. Baddeley, Centro internazionale matematico estivo and W. Weil. *Stochastic Geometry: Lectures Given at the C.I.M.E. Summer School Held in Martina Franca, Italy, September 13-18, 2004*. Lecture Notes in Mathematics / C.I.M.E. Foundation Subseries. Springer, 2007. ISBN: 9783540381747 (cited on pages 82, 84, 85).

[BRT15] Adrian Baddeley, Ege Rubak and Rolf Turner. *Spatial Point Patterns: Methodology and Applications with R*. London: Chapman and Hall/CRC Press, 2015. URL: `http://www.crcpress.com/Spatial-Point-Patterns-Methodology-and-Applications-with-R/Baddeley-Rubak-Turner/9781482210200/` (cited on page 94).

[BGH92] Dimitri P Bertsekas, Robert G Gallager and Pierre Humblet. *Data networks*. Volume 2. Prentice-Hall International New Jersey, 1992 (cited on page 179).

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004 (cited on pages 140, 141).

[BF11] R.L. Burden and J.D. Faires. *Numerical Analysis*. Brooks/Cole, Cengage Learning, 2011. ISBN: 9780538735643. URL: `https://books.google.it/books?id=KlfrjCDayHwC` (cited on page 91).

[CR13] Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology press, 2013 (cited on page 173).

[GLC13] Forbes Guthrie, Scott Lowe and Kendrick Coleman. *VMware vSphere design*. John Wiley & Sons, 2013 (cited on page 41).

[HT11] Harri Holma and Antti Toskala. *LTE for UMTS: Evolution to LTE-advanced*. John Wiley & Sons, 2011 (cited on page 35).

[Kle75] Leonard Kleinrock. *Theory, Volume 1, Queueing Systems*. USA: Wiley-Interscience, 1975. ISBN: 0471491101 (cited on page 76).

[Mat86] B. Matern. *Spatial Variation*. (Meddelanden från Statens Skogsforskningsinstitut). Springer New York, 1986 (cited on page 84).

[PS98] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998 (cited on pages 73, 74, 77, 78, 136).

[SW08] Rolf Schneider and Wolfgang Weil. *Stochastic and Integral Geometry*. Probability and its Applications. Berlin, Germany: Springer, 2008 (cited on page 85).

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cited on page 117).

[Tru13] Richard J Trudeau. *Introduction to graph theory*. Courier Corporation, 2013 (cited on page 99).

Standards

[3GP20a] 3GPP. *Application layer support for Vehicle-to-Everything (V2X) services.* Technical Specification (TS) 23.286.v16.5.0. 3rd Generation Partnership Project (3GPP), December 2020 (cited on page 39).

[3GP20b] 3GPP. *Architecture enhancements for 5G System (5GS) to support Vehicle-to-Everything (V2X) services.* Technical Specification (TS) 23.287 v16.5.0. 3rd Generation Partnership Project (3GPP), December 2020 (cited on page 36).

[3GP17a] 3GPP. *Enhancement of 3GPP support for V2X scenarios; Stage 1.* Technical Specification (TS) 22.186 v15.0.0. 3rd Generation Partnership Project (3GPP), March 2017 (cited on page 34).

[3GP19b] 3GPP. *Enhancement of 3GPP support for V2X scenarios; Stage 1.* Technical Specification (TS) 22.186 v16.2.0. 3rd Generation Partnership Project (3GPP), June 2019 (cited on pages 39, 168).

[3GP20c] 3GPP. *Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3.* Technical Specification (TS) 28.541 v16.7.0. 3rd Generation Partnership Project (3GPP), December 2020 (cited on page 62).

[3GP20d] 3GPP. *Management and orchestration; 5G performance measurements.* Technical Specification (TS) 21.552 v16.8.0. 3rd Generation Partnership Project (3GPP), December 2020 (cited on page 63).

[3GP20e] 3GPP. *Management and orchestration; Concepts, use cases and requirements.* Technical Specification (TS) 28.530 v16.4.0. 3rd Generation Partnership Project (3GPP), December 2020 (cited on pages 60, 61).

[3GP20f] 3GPP. *Management and orchestration; Generic management services.* Technical Specification (TS) 28.532 v16.6.0. 3rd Generation Partnership Project (3GPP), December 2020 (cited on page 63).

[3GP20g] 3GPP. *Management and orchestration; Performance assurance.* Technical Specification (TS) 28.550 v16.7.0. 3rd Generation Partnership Project (3GPP), December 2020 (cited on page 63).

[3GP19c] 3GPP. *Release description; Release 15.* Technical Report (TR) 21.915 v15.0.0. 3rd Generation Partnership Project (3GPP), October 2019 (cited on pages 35, 36, 38, 45, 75).

[3GP20h] 3GPP. *Release description; Release 16.* Technical Report (TR) 21.916 v1.0.0. 3rd Generation Partnership Project (3GPP), July 2020 (cited on pages 35, 36, 38, 39, 62, 75).

[3GP17b] 3GPP. *Revised WID: Enhancements for high capacity stationary wireless link and introduction of 1024 QAM for LTE DL.* Work Item 171738. 3rd Generation Partnership Project (3GPP), September 2017 (cited on page 35).

[3GP20j] 3GPP. *Service requirements for cyber-physical control applications in vertical domains.* Technical Specification (TS) 22.104 v16.5.0. 3rd Generation Partnership Project (3GPP), July 2020 (cited on page 61).

[3GP16b] 3GPP. *Service requirements for next generation new services and markets.* Technical Specification (TS) 22.261 v1.0.0. 3rd Generation Partnership Project (3GPP), December 2016 (cited on page 61).

[3GP20k] 3GPP. *Service requirements for the 5G system.* Technical Specification (TS) 22.261 v16.13.0. 3rd Generation Partnership Project (3GPP), October 2020 (cited on pages 37, 38).

[3GP20l] 3GPP. *Study on Communication for Automation in Vertical Domains*. Technical Report (TR) 22.804 v16.3.0. 3rd Generation Partnership Project (3GPP), July 2020 (cited on page 54).

[3GP18c] 3GPP. *Study on Integrated Access and Backhaul; Release 16*. Technical Report (TR) 38.874.v16.0.0. 3rd Generation Partnership Project (3GPP), December 2018 (cited on pages 37, 49).

[3GP18d] 3GPP. *System Architecture for the 5G System*. Technical Specification (TS) 23.501 v15.4.0. 3rd Generation Partnership Project (3GPP), December 2018 (cited on page 81).

[3GP20m] 3GPP. *Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS)*. Technical Specification (TS) 28.622 v16.6.0. 3rd Generation Partnership Project (3GPP), December 2020 (cited on pages 62, 63).

[Con17] OpenFog Consortium. *OpenFog Reference Architecture for Fog Computing*. OPFRA001.020817. Architecture Working Group, February 2017 (cited on pages 53–55).

[ETS20a] ETSI. *5G; NR; Physical layer procedures for data*. Technical Specification (TS) 38.214.v16.2.0. European Telecommunications Standards Institute (ETSI), July 2020 (cited on page 38).

[ETS18a] ETSI. *5G; Service requirements for enhanced V2X scenarios*. Technical Specification (TS) 122.186.v15.3.0. European Telecommunications Standards Institute (ETSI), July 2018 (cited on page 46).

[ETS09] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions*. Technical Report (TR) 102 638 v1.1.1. European Telecommunications Standards Institute (ETSI), June 2009 (cited on pages 168, 169).

[ETS19a] ETSI. *Management and Orchestration; Network Service Templates Specification*. Group Specification (GS) 014 v3.3.1. European Telecommunications Standards Institute (ETSI), September 2019 (cited on page 43).

[ETS19b] ETSI. *Management and Orchestration; VNF Descriptor and Packaging Specification*. Group Specification (GS) 011 v3.3.1. European Telecommunications Standards Institute (ETSI), September 2019 (cited on page 44).

[ETS18b] ETSI. *MEC Deployments in 4G and Evolution Towards 5G*. White Paper 24. European Telecommunications Standards Institute (ETSI), February 2018 (cited on pages 50, 51).

[ETS19c] ETSI. *Mobile Edge Computing (MEC); Framework and Reference Architecture*. Group Specification (GS) 003 v2.1.1. European Telecommunications Standards Institute (ETSI), January 2019 (cited on pages 46, 47, 51, 52, 57).

[ETS20b] ETSI. *Multi-access Edge Computing (MEC); MEC 5G Integration*. Group Report 031 v2.1.1. European Telecommunications Standards Institute (ETSI), October 2020 (cited on page 49).

[ETS18c] ETSI. *Multi-access Edge Computing (MEC); Phase 2: Use Cases and Requirements*. Group Specification (GS) MEC 002 v2.1.1. European Telecommunications Standards Institute (ETSI), October 2018 (cited on pages 48, 49, 168).

[ETS18d] ETSI. *Multi-access Edge Computing (MEC); Study on MEC Support for V2X Use Cases*. Group Report. September 2018 (cited on page 168).

[ETS19d] ETSI. *Multi-access Edge Computing (MEC); Support for network slicing*. Group Report 024 v2.1.1. European Telecommunications Standards Institute (ETSI), November 2019 (cited on page 63).

[ETS14] ETSI. *Network Functions Virtualisation (NFV); Management and Orchestration*. Group Specification (GS) 001 v1.1.1. European Telecommunications Standards Institute (ETSI), December 2014 (cited on pages 40–42, 45, 52, 56, 67, 98).

[ITU17] ITU-R. *Minimum requirements related to technical performancefor IMT-2020 radiointerface(s)*. M Series Mobile, Radiodetermination, Amateur and Related Satellite Services M.2410-0. International Telecommunication Union - Radiocommunication Sector (ITU-R), November 2017 (cited on page 59).

[ITU18] ITU-T. *Consideration on 5G transport network reference architecture and bandwidth requirements*. Study Group 15 Contribution 0462. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), February 2018 (cited on pages 90, 144, 150, 160).

White papers

[EAN16] EANTC. *Juniper Networks MPC9E: EANTC Performance, Scale and Power Test Report*. White Paper v1.0. European Advanced Networking Test Center, April 2016 (cited on page 90).

[Mou+18] A. Mourad et al. *5G-CORAL initial system design, use cases, and requirements*. Deliverable 1.1. 5G-Coral consortium, February 2018. URL: `http://5g-coral.eu/wp-content/uploads/2018/04/D1.1_final7760.pdf` (Retrieved: 10th January 2019) (cited on pages 55, 56, 142, 149).

[NGM15] NGMN. *5G White Paper*. White Paper v1.0. Next Generation Mobile Networks Alliance, February 2015 (cited on page 88).

[NGM16] NGMN. *Description of Network Slicing Concept*. White Paper v1.0. Next Generation Mobile Networks Alliance, February 2016 (cited on page 88).

[Sam19] Samer Talat and Ibrahiem Osamah and and Milan Groshev and Luca Cominardi and Giovanni Rigazzi and Charles Turyagyenda and Riccardo Ferrari and Giacomo Parmeggiani and Saptarshi Hazra and Bengt Ahlgren and Chenguang Lu and Gyanesh Patra and Daniel Cederholm and Gian Michele Dell'Aera and Chi-Yu Li and Hsu Tung Chien and Chung-Hau Lee and Ivan Paez and Alan Chen and Aitor Zabala and Pedro Bermúdez. *D4.2–5G-CORAL Proof of concept and future direction*. Deliverable 4.2. 5G-Coral consortium, August 2019. URL: `%7Bhttp://5g-coral.eu/wp-content/uploads/2019/09/D4.2_FINAL.pdf%7D` (cited on page 56).

[Sol+20] Alberto Solano et al. *5G-DIVE architecture and detailed analysis of vertical use cases*. Deliverable 1.1. 5G-DIVE consortium, March 2020. URL: `https://5g-dive.eu/wp-content/uploads/2021/01/D1.1_Final.pdf` (cited on page 57).

Acronyms

Symbols	
3GPP	3rd Generation Partnership Project
5GA	5G Access Network
5GC	5G Core
A	
AF	Application Function
AI	Artificial Intelligence
AMF	Access and Mobility Management Function
AP	Access Point
API	Application Programming Interface
AR	Augmented Reality
B	
BBU	Baseband processing Unit
BFS	Breadth-First Search
BGP	Border Gateway Protocol
BiS-BiS	Big Switch with Big Software
BS	Base Station
BSS	Business Support System
C	
C-V2N	Cerllular Vehicle-to-Network
C-V2X	Cerllular Vehicle-to-Everything
CAPEX	Capital Expense
CAPIF	Common API Framework
CAV	Connected Automated Vehicles
CDN	Content Delivery Network
CFS	Customer Facing Service
CPD	Connection Point Descriptor
CSC	Communication Service Consumer
CSP	Communication Service Provider
CU	Central Unit

D	
D2D	Device-to-Device
DEEP	5G-DIVE Elastic Edge Platform
DFS	Depth-First Search
DL	Downlink
DPI	Deep Packet Inspector
DQN	Deep Q-Network
DU	Distributed Unit
E	
E2E	End-to-End
EFS	Edge and Fog computing System
eMBB	enhanced Mobile Broadband
eNB	Evolved Node B
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
G	
gNB	Next Generation NodeB
GSMA	Global System for Mobile Communications
GST	Generic network Slice Template
H	
HPM	Hardware Platform Management
I	
IAB	Integrated Access and Backhaul
IIC	Industrial Internet Consorcium
ILP	Integer Linear Programming
IoT	Internet of Things
IP	Internet Protocol
K	
KPI	Key Performance Indicator
L	
LTE	Long Term Evolution
M	
MANO	Management and Orchestration
MCP	Multi Constrained Path
MCT	Mobile Communication Transport
MdO	Multi Domain Orchestrator
MDP	Markov Decision Process
MEC	Multi-access Edge Computing
MEO	MEC Orchestrator
MILP	Mixed Integer Linear Programming
MIoT	Massive Internet of Things
ML	Machine Learning
MME	Mobility Management Entity

mMTC	Massive Machine-Type Communications
mmWv	Millimeter Wave
MOI	Managed Object Instance
MPLS	Multiprotocol Label Switching
MTC	Machine Type Communication
N	
NEF	Network Exposition Function
NEST	Network Slice Template
NF	Network Function
NFV	Network Function Virtualization
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
NN	Neural Network
NP	Non-deterministic polynomial time
NPN	Non Public Network
NR	New Radio
NS	Network Service
NSA	Non-Stand Alone
NSaaS	Network Slice as a Service
NSD	Network Service Descriptor
NSSAA	Network Slice-Specific Authentication and Authorization
NSSI	Network Slice Subnet Instance
O	
OCS	Orchestration and Control System
OPEX	Operational Expense
OSS	Operation Support System
P	
PDU	Protocol Data Unit
PGW	Packet Data Network Gateway
PNF	Physical Network Function
PNFD	Physical Network Function Descriptor
PoC	Proof of Concept
PoP	Point of Presence
PP	Point Process
PPP	Poisson Point Process
PS	Processor Sharing
Q	
QoE	Quality of Experience
QoS	Quality of Service
R	
RAN	Radio Access Network
RAT	Radio Access Technology
RTT	Round Trip Time
RU	Radio Unit
S	

SA	Stand Alone
SAP	Service Access Point
SBA	Service-Based Architecture
SCS	Subcarrier Spacing
SDN	Software Defined Networking
SEAL	Service Enabler Architecture Layer for Verticals
SFC	Service Function Chain
SGW	Serving Gateway
SLA	Service Level Agreement
SMF	Session Management Function
SP	Service Provider
STGCN	Spatio-Temporal Graph Convolutional Networks

T

TS	Technical Specification
TTI	Transmission Time Interval

U

UAV	Unmaned Aerial Vehicle
UDM	Unified Data Management
UE	User Equipment
UL	Uplink
UPF	User Plane Function
URLLC	Ultra-Reliable and Low Latency Communications

V

V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VDU	Virtualization Deployment Unit
VIM	Virtualised Infrastructure Manager
VL	Virtual Link
VLAN	Virtual LAN
VM	Virtual Machine
VNE	Virtual Network Embedding
VNF	Virtual Network Function
VNFC	Virtual Network Function Component
VNFD	Virtual Network Function Descriptor
VNFFG	VNF Forwarding Graph
VNFM	VNF Manager
VR	Virtual Reality
VSB	Vertical Service Blueprint
VXLAN	Virtual Extensible Local Area Network

W

WG	Working Group