H2020-ICT-2016-2
5G-MoNArch
Project No. 761445

5G Mobile Network Architecture

for diverse services, use cases, and applications in 5G and beyond

Deliverable D4.1

Architecture and mechanisms for resource elasticity provisioning

Contractual Date of Delivery	2018-05-30
Actual Date of Delivery	2018-06-08
Work Package	WP4 – Resource elasticity
Editor(s)	Ghina Dandachi (CEA), Nicola di Pietro (CEA), David Gutierrez Estevez (SRUK)
Reviewers	Gerd Zimmermann (DT), Peter Rost (NOK-DE)
Dissemination Level	Public
Type	Report
Version	1.0
Total number of pages	82

Abstract: Vertical markets and industries are addressing a large diversity of heterogeneous services, use cases, and applications in 5G. It is currently common understanding that for networks to be able to satisfy those needs, a flexible, adaptable, and programmable architecture based on network slicing is required. Moreover, a softwarisation and cloudification of the communications networks is already happening, where network functions (NFs) are transformed from programs running on dedicated hardware platforms to programs running over a shared pool of computational and communication resources. However, this novel architecture paradigm requires new solutions to exploit its inherent flexibility. In this deliverable, we introduce the concept of resource elasticity as a key means to make an efficient use of the computational resources in 5G systems. Besides establishing a definition as well as a set of requirements and key performance indicators (KPIs), we propose an elastic functional architecture addressing the three different dimensions, namely computational elasticity in the design and scaling of NFs, orchestration-driven elasticity by flexible placement of NFs, and slice-aware elasticity via cross-slice resource provisioning mechanisms. Moreover, we analyse the solution space by proposing specific mechanisms for the exploitation of elasticity in the above mentioned three different dimensions.

Keywords: 5G Network Architecture, Resource Elasticity, Computational Elasticity, Orchestration-driven Elasticity, Slice-aware Elasticity.

Executive Summary

As part of the EU H2020 5G-PPP initiative Phase 2, this public deliverable constitutes a further step in the 5G-MoNArch research project output. The key goal in the 5G-MoNArch project is to propose a flexible, adaptable, and programmable fully-fledged architecture for 5G mobile networks. This proposal should be grounded upon and demonstrate three key enabling innovations: (i) inter-slice control and cross domain management, (ii) experiment-driven optimisation, (iii) cloud-enabled protocol stack, (iv) resilience, and (v) resource elasticity, as they are linked to two different testbeds that will demonstrate those innovations. This document, as the first public deliverable of work package 4 (WP4), focuses on one of the five key innovations of 5G-MoNArch, namely the above-mentioned resource elasticity.

The novelty and relevance of the concept of resource elasticity needs to be framed in the context of 5G network architecture, where virtualisation and cloudification have become central features. Indeed, the virtualised and cloudified infrastructure of 5G systems requires a tremendous level of flexibility if the 5G challenge of hosting a great variety of verticals with very heterogeneous requirements is to be met. Resource elasticity is thus a novel concept that leverages on those two new features seeking to dramatically increase resource efficiency by reducing over-provisioning while allowing for network self-dimensioning and smart resource distribution and orchestration. While this required flexibility has been tackled for radio resources in the research community during past years, much less work has been devoted to the elasticity of computational resources, and this issue has become very relevant in the context of cloud and virtualised architectures where computational resources play a key role.

In addition to the introduction, conclusions, and references sections, this document describes the technical innovations developed within WP4 as follows.

- **Chapter 2 covers new technical and business requirements for resource elasticity.** The first section, focused on technical requirements, provides a definition for elasticity in the context of networking, an analysis on requirements at three different levels (namely virtual network function (VNF), intra-slice, and infrastructure levels), and a list of relevant key performance indicators (KPIs). The second section focuses on business requirements by analysing the economic implications of the concept of elasticity via a techno-economic system dimensioning as well as economic views on the different technologies relevant to elasticity.
- **Chapter 3 provides a novel functional architecture proposal for resource elasticity.** The chapter is structured in two sections: the first one provides a logical description of the architectural interactions needed for the different elasticity dimensions; the second section focuses on the architecture itself as follows: A description of the different existing layers in the 5G-MoNArch architecture is first provided (including service, management and orchestration (M&O) layer, controller, and network layers), followed by an in-depth analysis of the architectural elements that are needed to exploit resource elasticity in its three different dimensions. Furthermore, the addition of an artificial intelligence (AI) and big data analytics engine, as part of the elasticity provisioning enhancements, is also examined.
- **Chapter 4 presents specific novel mechanisms for resource elasticity provisioning**. The proposed mechanisms address the issue of elasticity along its three identified dimensions, namely computational elasticity, orchestration-driven elasticity, and slice-aware elasticity. For each dimension, a review of the state of the art is first provided, followed by a set of novel techniques addressing the problem of elasticity via a wide variety of mechanisms. The presented mechanisms represent a step forward in the attainment of elasticity via the three above mentioned dimensions. However, the mechanisms vary in the employed analytical and mathematical tools; a few examples are multi-objective optimisation, AI and machine learning (ML), big data analytics or game theory. The common objective of all of them, however, is optimising the elastic operation of an architecture based on network slicing.

Overall, this public deliverable represents the first comprehensive guide on the proposed architecture and mechanisms for resource elasticity provisioning in 5G networks, thus highlighting the related innovations under development in WP4 of the 5G-MoNArch project.

List of Authors

Partner	Name	E-mail
NOK-DE	Irina Balan	irina.balan@nokia-bell-labs.com
UC3M	Marco Gramaglia Pablo Serrano	mgramagl@it.uc3m.es pablo@it.uc3m.es
DT	Paul Arnold Michael Einhaus Igor Kim Mohamad Buchr Charaf	Paul.arnold@telekom.de einhaus@htl-leipzig.de igor.skh@gmail.com besh.sh91@gmail.com
SRUK	David Gutierrez Estevez	d.estevez@samsung.com
ATOS	Joanna Bednarz	joanna.bednarz@atos.net
CEA	Ghina Dandachi Antonio De Domenico Nicola di Pietro	ghina.dandachi@cea.fr antonio.de-domenico@cea.fr nicola.dipietro@cea.fr
CERTH	Anastasios Drosou Stavros Papadopoulos Asterios Mpatziakas Eleni Ketzaki	drosou@iti.gr spap@iti.gr ampatziakas@iti.gr eketzaki@iti.gr
MBCS	Dimitris Tsolkas Odysseas Sekkas	dtsolkas@mobics.gr sekkas@mobics.gr
RW	Simon Fletcher Julie Bradford	simon.fletcher@realwireless.com julie.bradford@real-wireless.com
NOMOR	Sina Khatibi	khatibi@nomor.de

Revision History

Revision	Date	Issued by	Description
1.0	08.06.2018	5G-MoNArch WP4	Final submitted version 1.0

List of Acronyms and Abbreviations

3G	3rd Generation mobile wireless communication system (UMTS, HSPA)
3GPP	3rd Generation Partnership Project
4G	4th Generation mobile wireless communication system (LTE, LTE-A)
5G	5th Generation mobile wireless communication system
5G-PPP	5G Infrastructure Public Private Partnership
ACO	Ant Colony Optimisation
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AR	Augmented Reality
BWP	Bandwidth Parts
CAPEX	CAPital EXpenditure
CFS	Completely Fair Scheduler
COTS	Commercial off-the-shelf
CP	Control Plane
CPU	Central Processing Unit
CQI	Channel Quality Indicator
C-RAN	Cloud Radio Access Network
CSCM	Cross-Slice Congestion Manager
CSMF	Communication Service Management Function
CU	Centralised Unit
DL	DownLink
DU	Distributed Unit
E2E	End-to-End
eMBB	Enhanced Mobile BroadBand
eNB	Evolved Node B
ENI	Experiential Network Intelligence
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
FDD	Frequency Division Duplexing
FFT	Fast Fourier Transformation
FIFO	First In - First Out
FR-A	FlexRAN Agent
FR-C	FlexRAN Controller
gNB	Next Generation Node B
HARQ	Hybrid Automatic Repeat Request
HSS	Home Subscriber Server
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IFFT	Inverse Fast Fourier Transformation
InP	Infrastructure Provider
ISC	Intra-slice Controller
KPI	Key Performance Indicator
LTE	Long Term Evolution
LTE-A	LTE Advanced
M&O	Management and Orchestration (*5G-MoNArch layer*)

MAC	Medium Access Control
MANO	Management and Orchestration (*ETSI NFV stack*)
MCC	Mobile Cloud Computing
MCS	Modulation and Coding Scheme
MEC	Mobile Edge Computing, Mobile Edge Cloud
ML	Machine Learning
MME	Mobility Management Entity
mMTC	massive Machine-Type Communication
MNO	Mobile Network Operator
MRT	Maximum Ratio Transmission
MSP	Mobile Service Provider
MU	Multi-User
MU-MIMO	Multi-User Multiple Input Multiple Output
MUPF	Multi-User Proportional Fair
NF	Network Function
NFV	Network Function Virtualisation
NFV-O	Network Function Virtualisation Orchestrator
NGMN	Next Generation Mobile Networks
NI	National Instruments
NR	New Radio
NSI	Network Slice Instance
NSMF	Network Slice Management Function
NSSI	Network Slice Subnet Instance
NSSMF	Network Slice Subnet Management Function
OAI	OpenAirInterface
QAM	Quadrature Amplitude Modulation
OFDM	Orthogonal Frequency Division Multiplexing
OPEX	OPerational EXpenditure
PBCH	Physical Broadcast Channel
PDCP	Packet Data Convergence Protocol
PDSCH	Physical Downlink Shared Channel
PID	Process Identifier
PF	Proportional Fair
PHY	Physical Layer
PNF	Physical Network Function
PRB	Physical Resource Block
PSS	Primary Synchronisation Channel
QAM	Quadrature Amplitude Modulation
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RAN	Radio Access Network
RBG	Resource Block Group
RCPSP	Resource-Constrained Project Scheduling Problem
RL	Reinforcement Learning
RR	Round Robin
RRC	Radio Resource Control
RTT	Round Trip Time
RX	Reception

SDK	Software Development Kit
SDN	Software Defined Networking
SDO	Standards Development Organisation
SDR	Software Defined Radio
SINR	Signal to Interference plus Noise Ratio
SLA	Service Level Agreement
SLR	Signal to Leakage Ratio
SNR	Signal to Noise Ratio
SRS	Sounding Reference Signal
S/P-GW	Serving/Packet Gateway
SS	Sub-Slice
SSS	Secondary Synchronisation Channel
SU-MIMO	Single User Multiple Input Multiple Output
SVM	Support Vector Machine
TBS	Transport Block Size
TCP	Transmission Control Protocol
TDD	Time Division Duplexing
TTI	Transmission Time Interval
TX	Transmission
UDP	User Datagram Protocol
UE	User Equipment
UL	UpLink
USRP	Universal Software Radio Peripheral
UP	User Plane
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VNFM	Virtual Network Function Manager
VR	Virtual Reality
XSC	Cross-slice Controller
ZF	Zero Forcing
ZF-MU	Zero Forcing-Multi User

Table of Contents

List of Figures

List of Tables

1 Introduction

The 5th generation (5G) of cellular systems will change the access to communication services for users, vertical markets and industries. Thanks to the 5G-enabled technical capabilities, they will experience a drastic transformation that will trigger the development of cost-effective new products and services. A large number of use cases and corresponding requirements for representative vertical markets such as automotive, health, factories of the future, energy, and media and entertainment will need agile access to network support functionalities [5G-PPP16]. This will require a fundamental rethinking of the mobile network architecture and interfaces. The expected diversity of services, use cases, and applications in 5G requires a flexible, adaptable, and programmable architecture. To this end, network architecture must shift from the current network of entities to a network of capabilities.

In the context of 5G network architecture, a few key concepts have been introduced in the last years by standards development organisations (SDOs) and research activities. The first one is the concept of network slicing [NGMN15], which allows the mobile network operator (MNO) to run multiple logical network instances in parallel. It was introduced as an effective way to meet all of the heterogeneous requirements from supported use cases and services by means of a cost-effective multi-tenant shared network infrastructure. Another fundamental enabler that emerged as an initiative from the industry to increase the deployment flexibility and the agility with which a new service is deployed within the network is network function virtualisation (NFV) and its management and orchestration (MANO) architecture [HB16]. NFV is a framework where NFs that traditionally used dedicated hardware are now implemented in software that runs on top of a pool of hardware that can be dynamically allocated, effectively enabling a hardware-software separation that reduces both capital and operational expenditures (i.e., CAPEX and OPEX).

This public deliverable is part of the 5G-PPP Phase 2 5G-MoNArch project output. 5G-MoNArch is working to design an architecture that will combine today's accepted and designed concepts (such as virtualisation, slicing and orchestration of access and core functions) with three enabling innovations that fill gaps not addressed yet by academia and industry: (i) inter-slice control and cross-domain management, to enable the coordination across slices and domains, (ii) experiment-driven optimisation, to leverage experimental results to design highly performing algorithms, and (iii) cloud-enabled protocol stack, to gain flexibility in the orchestration of virtualised functions. To this end, 5G-MoNArch will deploy and test the devised architecture in two testbeds: (i) the sea port, representative of a vertical industry use case, and (ii) the touristic city, representative of a MNO deployment. For each testbed, 5G-MoNArch will instantiate the architecture and complement it with a use case specific functionality: (i) resilience and security, needed to meet the sea port requirements, and (ii) resource elasticity, to make an efficient use of the resources in the touristic city. This public deliverable deals with the latter key functional innovation of the project.

Elasticity is a well-studied concept in cloud computing systems, and in that context it has traditionally been defined as the degree up to which a system is able to adapt to workload changes by allocating and de-allocating resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible [HKR13][CSR+15]. In networks, temporal and spatial traffic fluctuations require that the network efficiently scales resources such that, in case of peak demands, the network adapts its operation and re-distributes available resources as needed, gracefully scaling the network operation. This feature is particularly useful when a network slice is instantiated in an environment where overprovisioning of resources is not an option and the nature of the service allows for a flexible management of the resources without incurring in critical service level agreement (SLA) violations. We refer to this flexibility, which could be applied both to computational and communications resources, as resource elasticity. Although elasticity in networks has already been exploited traditionally in the context of communications resources (e.g., where the network gracefully downgrades the quality for all users if resources such as spectrum are insufficient), in this report we focus on the computational aspects of resource elasticity, as we identify the management of computational resources in networks as a key challenge of future virtualised and cloudified 5G systems.

In addition to providing a definition and a set of technical and economic requirements and key performance indicators (KPIs) for resource elasticity, in this public deliverable we present two main innovations around the above topic of resource elasticity: i) an elastic functional architecture, i.e., the

implications required for 5G architecture to support three dimensions of elasticity, namely computational elasticity, orchestration-driven elasticity, and slice-aware elasticity, and ii) a set of novel mechanisms to provision resource elasticity along the three mentioned dimensions. In the following, we present a brief summary of all presented mechanisms highlighting its key idea and novelty.

Contributions on computational elasticity:

- *Modulation and coding scheme (MCS) selection for C-RAN*, and *scheduling under resource constraints*: These two contributions address the design of the internals of the mentioned VNFs to take into account the constraints related to the cloud resources availability.
- *Rank control for multi-user multiple input multiple output (MU-MIMO)*: A new scheduling design of a MU-MIMO scheduler with adaptive rank control is proposed considering the usage of computational resources besides satisfying traditional KPIs. Therefore, the scheduler is designed as an elastic VNF.
- *Model for slice-aware elastic resource management in radio access network (RAN)*: The model for elastic resource management approach proposed in this report addresses the assigning the available computational resources in a resource pool to the different slices based on their requirements and SLAs.
- *Computational resource control for a virtualised mobile network with containers*: The overall target of this contribution is on the one hand to derive a design of an algorithm which controls radio and computational resources simultaneously and on the other hand find requirements for a definition of open interfaces among virtualised RAN functionalities designed as VNFs.

Contributions on orchestration-driven elasticity:

- *Proactive resource allocation and relocation*: A functionality of 5G-MoNArch's Management and Orchestration (M&O) Layer is introduced, called Cross-Slice Congestion Manager (CSCM). The CSCM is part of the global slice orchestrator that controls the lifecycle of network slices, monitors the buffer of slice requirements, and manages the resource allocation and availability. We describe here the role of the CSCM in enabling orchestration-driven elasticity and we give a preliminary mathematical formulation of the VNF relocation problem.
- *Machine learning for network function (NF) scaling and orchestration*: Artificial intelligence (AI) and machine learning (ML) techniques are explored for scaling and orchestration of virtual network functions (VNFs). In particular, a novel aspect of this mechanism is its ability to differentiate between horizontal and vertical scaling approaches, using a reinforcement learning (RL) approach.
- *Orchestration through live migration of network functions*: Live migration consists in moving application instances or virtual machines (VMs) between hosts without service interruption. Full live migration of VMs, unikernels and containers for VNFs, possibly aided by AI techniques, are explored in this contribution.

Contributions on slice-aware elasticity:

- *Multi-objective optimisation for slice-aware resource orchestration*: A multi-objective optimisation approach is used in order to produce a set of solutions for the orchestration of the network resources, in a slice-aware context, based on the minimisation of network KPIs. The algorithm will output a set of optimal solutions that represent different ways to map the resources to the slices while retaining all information about each KPI behaviour. This allows a better understanding and perception of the traffic conditions, which in turn leads to a clearer comprehension of the trade-offs between various resource allocation decisions.
- *Computational resource utilisation optimisation in slice-enabled systems*: A mathematical formulation of the following problem is provided, namely the allocation of VNFs of different slices/services to a set of available computational resources / computation machines to achieve the minimum time to complete the execution of all the VNFs. Ant Colony Optimisation has been identified as a candidate semi-optimal solver of the problem.
- *Slice-aware automatic RAN configuration*: In a scenario where the several slices are sharing the limited pool of RAN resources, the goal of the slice-aware elastic function is to adapt the

network configuration in a proactive and automated fashion to match the dynamic changes in the service specific needed capacity. We explore the solution of adapting the beam patterns to achieve an optimal capacity/coverage trade-off.

- *Slice blueprint analytics and classification*: A method to enable slice-aware elasticity based on the exploitation of similarity of service requests is introduced. The main idea is to cluster slice requests that have a high similarity to reduce the resources used by each network slice, whenever resource isolation across slices is not a constraining factor. In this scenario, resource utilisation efficiency will increase if two slice requests that share the same VNFs are allocated on common virtual resources at the same time, inducing a cost reduction in the process of slice deployment.
- *Big data analysis for network slice characterisation*: Big data analytics are used in this contribution to learn from previous network utilisation across different services to devise optimal orchestration policies.
- *A market-based approach to network slicing*: In this contribution, we model the 5G resources (both cloud and spectrum) as a market of resources where players (tenants) either bid or buy for creating a slice and the infrastructure provider look for maximizing the monetisation.

The remainder of this document is organised as follows. Chapter 2 analyses resource elasticity requirements both from a technical and an economic perspective. It provides a definition, related metrics and KPIs, as well as an in-depth analysis of techno-economic aspects of resource elasticity. Chapter 3 provides a novel functional architecture proposal for resource elasticity, covering both the logical description of the three elasticity dimensions and its interfaces, as well as an in-depth analysis of the architectural elements needed to exploit resource elasticity in the three above-mentioned dimensions. Chapter 4 is composed of three sections, one-to-one mapped to the three major dimensions of resource elasticity provisioning, namely computational elasticity, orchestration-driven elasticity, and slice-aware elasticity. For each one of these sections the current state of the art and a set of innovative approaches adopted by the 5G-MoNArch project are included. Chapter 5 summarises the contributions of the document and provides concluding remarks.

2 Technical and Economic Requirements for Elasticity

2.1 Resource Elasticity Definition and Related KPIs

With the rise of network programmability, enabled by technologies such as software defined networking (SDN) and Network Function Virtualisation (NFV), Network Functions (NFs) may be easily moved around in a cloudified environment. On the other hand, the load fluctuations that different network slices will introduce in the network will impact the way on how such NFs are located in the cloudified environment. To efficiently deal with changing load in such cloudified scenario, which in turn translates into a varying resource consumption, we introduce the concept of resource elasticity. Resource elasticity has different dimensions that can be evaluated with novel KPIs.

2.1.1 Resource Elasticity: A Definition

The resource elasticity of a communications system can be defined as the ability to gracefully adapt to load changes in an automatic manner such that at each point in time the available resources match the demand as closely and efficiently as possible. Hence, elasticity is intimately related to the system response when changes occur in the amount of available resources. We employ the term gracefully in the definition of elasticity to imply that, for a relatively small variation in the amount of resources available, the operation of the service should not be disrupted. If the service produces a quantifiable output, and the resource(s) consumed are also quantifiable, then the gracefulness of a service can be defined as the continuity of the function mapping the resources to the output; sufficiently small changes in the input should result in arbitrarily small changes in the output (in a given domain) until a resource shortage threshold is met where the performance cannot keep up. We refer to this resource shortage threshold as minimum footprint. Figure 2-1 shows a conceptual example of the operation of an elastic system compared to a non-elastic one, where the elastic performance is capable of achieving graceful degradation with resource shortages until the minimum footprint is met. An elastic VNF should thus be able to cope with variations in the availability of resources without causing an abrupt degradation in the outputs provided by the function.

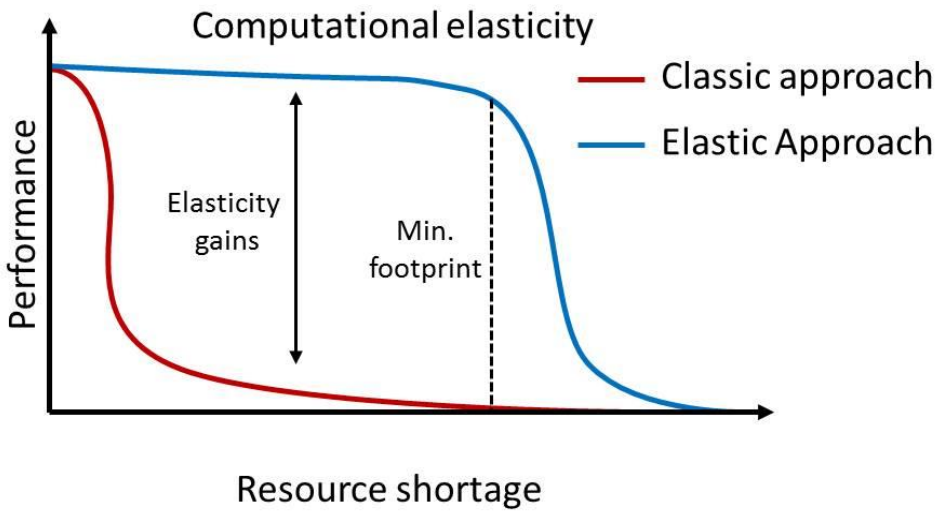


Figure 2-1: Illustration of gains achieved by elastic computation.

2.1.2 Elastic Operation Requirements

Resource elasticity can be exploited from different perspectives, each of them being a fundamental piece required to bring overall elasticity to the network operation. In this section, we describe in detail these different perspectives (referred to as *elasticity dimensions*), which, in turn, generate several innovation opportunities, as we show in Chapter 4:

- Elasticity at the VNF level,
- Elasticity at intra-slice level
- Elasticity at infrastructure level.

The first requirement for an elastic network operation is the need for *elasticity at the VNF level*. In general, the concept of elasticity for a NF is not directly applicable to legacy physical NFs (PNFs), that refers to a specific purpose-built hardware box that provides a well-defined networking functions.NF. Especially for the case of distributed NFs, the functionality is provided by a physical box that is the result of a thorough joint hardware/software design. Therefore, they have traditionally been designed without any major constraint on the available execution resources as they were expected to be always available by design, given some boundaries on the i.e., maximum number of user devices allowed. In addition, in networks with virtualised NFs, the joint hardware/software design is not possible anymore: VNFs are pieces of software that run on virtual containers on heterogeneous cloud platforms with standard interfaces. Therefore, in this new but already widely adopted scenario, expecting all the needed resources to be always available by design is not reasonable anymore. Furthermore, current VNFs (especially those in the RAN) have been designed under the assumption that required computational resources are always available and they may not be prepared for a shortage of computational resources. Indeed, when such resource outages occur (e.g., lack of CPU availability), current virtualised RAN implementations such as OpenAirInterface (OAI [OAI]) just drop the frame being processed, and as a result they see their performance severely degraded [NMM14]. This requirement is addressed by the **computational elasticity** innovation area described in Chapter 3 and Section 4.1.

A second requirement for elastic network operation can be characterised as *elasticity at intra-slice level*. The elastic design of a VNF has an impact on the elasticity of a network slice, defined as the chain of VNFs that provides a telecommunication service. Indeed, chaining and orchestrating a sequence of VNFs with different elastic KPIs (as described in Section 2.1.3) will result in an overall elasticity associated to a tenant running a service using a single network slice. This ultimately affects the quality of experience/service (QoE/QoS) perceived by users, who may experience different performance degradations according to the elasticity level provided by the tenant. This fact has an impact on the orchestration of a hierarchical cloud architecture such as the one defined in [RBB+16], in which the mobile network stack is decomposed into atomic VNFs to better exploit the location diversity and provide service-tailored orchestration. That is, orchestration algorithms may locate VNFs with strong elasticity characteristics where the operational cost is higher or avoid the co-location of inelastic VNFs in the same infrastructure. This requirement is addressed by the **orchestration-driven** elasticity innovation area described in Chapter 3 and Section 4.2.

The last requirement for elastic operation is *elasticity at the infrastructure level*, i.e., a requirement that involves the infrastructure on which elastic VNFs run. The choice of how many network slices are hosted in the same infrastructure depends on the infrastructure provider (InP) who runs e.g., admission control algorithms to guarantee that the SLA or security requirements with the various tenants are always fulfilled. Elasticity at the infrastructure level is a metric that involves both business and technical KPIs. By leveraging multiplexing gains, more network slices can be hosted on the same infrastructure (thus providing higher revenues), but it comes at the cost of having to resort to more elastic VNFs. This requirement is addressed by the **slice-aware elasticity** innovation area described in Chapter 3 and Section 4.3.

2.1.3 Measuring Elasticity: Technical KPIs

Another consequence of the elastic approach to the network design is the need for a new framework to (i) quantify its behaviour, (ii) assess its design, and (iii) serve as input for the orchestration algorithm, which would decide where to place the VNFs depending on their elasticity. In this section, we discuss how to evaluate the benefits of the two strategies proposed above, especially focusing on the elastic VNF design, as it has more cross-fertilisation opportunities. These novel elasticity KPIs in some cases may be just mutated from the traditional definitions provided by major SDOs such as 3GPP or ETSI, but some of them are native to this new framework.

A first category of KPIs includes metrics already established such as the service creation time or the availability, the latter defined as the relative amount of time that the function under study produces the

output that it would have produced under ideal conditions. A second category, however, includes brand new KPIs that shall be defined to measure the advantages introduced by the elastic operation of the network and the elasticity level of each VNF as defined in [5GM17-D6.1]. Native elasticity KPIs measure mostly the resource savings achieved by the elastic operation, defined as the average cost of deploying and operating the network infrastructure to support the foreseen services. An elastic system should also be able to be optimally dimensioned such that less resources are required to support the same services; furthermore, in lightly loaded scenarios the elastic system should avoid the usage of unnecessary resources and reduce the energy consumption by e.g., consolidating the load, hence also limiting the OPEX.

The cloud computing community has indeed long worked on the definition of elasticity, this generally being defined as the ability to provision and de-provision resources to match the demand at each time instant as closely and efficiently as possible [HKR13][CSR+15]. Elasticity is also related to resiliency, scalability, and efficiency, but with key differences: resiliency is the ability to recover from failures or to adjust easily to them, but it does not deal with efficiency; scalability is the ability to meet a larger load demand by adding a proportional amount of resources, but it does not consider temporal aspects of how fast and how often scaling actions can be performed. Finally, although a better elasticity should result in a higher efficiency, the opposite is not necessarily true. Building on this definition, it is possible to assess the degree of elasticity of e.g. a certain system, quantifying thus its ability to match the demand.

However, the relatively coarse timescales of the traditional cloud operation prevent a direct mapping to our vision, where VNFs operate on much shorter ones. In the following, we provide some considerations on the required space of metrics to quantitatively characterise these new VNFs.

In general, the resources supporting the execution of a VNF are a heterogeneous set (e.g., CPU, RAM, spectrum, transport bandwidth), thus care should be taken when measuring them or varying their availability performing experiments, to perform fair comparisons. A VNF should be characterised by its *minimum footprint*, defined as the minimum combination of resources needed to provide any output. During its regular operation, the *footprint* is temporal sequence of the resource occupied because of the execution of a function.

Under ideal circumstances, i.e., no shortage or variation of the resources available, a cloud-aware VNF has to operate as reliably as a traditional NF. With this being the benchmark, we can define *reliability* as the percentage of time that a VNF is providing the expected output. However, while in the traditional approach this reliability referred to the availability of a communication resource (e.g. “five nines reliability”), in our vision there are more categories of resources that impact the operation apart from congestion or link degradation.

Arguably, the most distinct feature of an elastic VNF is how it relates the above two points, i.e., the shape of the function that maps the available resources to the obtained outputs. This degradation function characterises the way in which performance degrades as resources lack, and depending on its actual shape we could characterise different quantitative behaviours: e.g., a *graceful degradation* might be defined by a percentage decrement of resources causing the same or a smaller percentage of reduction in performance.

Additional aid in achieving resiliency can be obtained via orchestration mechanisms, that horizontally (up/down) or vertically (in/out) scale the containers executing the VNF, but usually require a relatively long-time scale to operate. For that reason, not all VNFs could easily “be rescued”' in cases of low resource availability. The *rescuability* of a VNF is hence its capability of overcoming an outage by providing a limited degradation, until new resources are available.

Finally, we also envision elasticity-related business-driven KPIs such as the price of resource overbooking, the average performance loss of a single slice in comparison with the monetary gains that additional network slices may provide, or the specific number of network slices that can be hosted given a total amount of infrastructure. We summarise the proposed KPIs next.

Table 2-1: Elasticity technical KPIs

KPI name	Meaning
Minimum footprint	Minimum resource set to provide any output
Reliability	Percentage of time in which a VNF provides optimal operation
Graceful degradation	Perceived degradation utility vs resource shortage
Rescuability	Ability of overcoming an outage until new resources are available
Cost efficiency	Increased number of slices to be hosted on the same infrastructure

2.2 Economic Implications of Resource Elasticity

2.2.1 Techno-Economic Drivers for Network Elasticity

The ultimate goal of the techno-economic analysis is to establish the factors that influence the performance of the cost vs revenue characteristics of mobile networks over time and to understand how the 5G-MoNArch innovations can improve these and hence the overall business case for mobile networks and their evolution of 5G-MoNArch like architectures.

Figure 2-2 presents the existing cost and revenue trends over time as can be seen in the mobile industry today. It also illustrates the general objectives of flexible, virtualised networks like 5G-MoNArch in improving these.

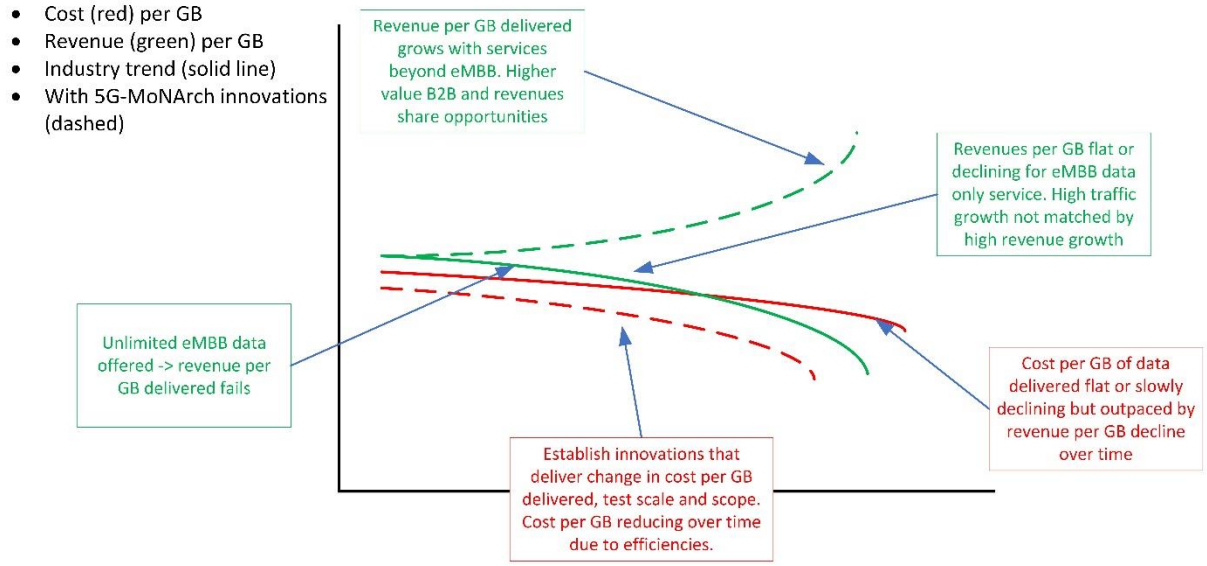


Figure 2-2: Trends and costs vs revenue over time.

Looking at revenue, the opportunity presented by multi-service 5G networks to MNOs, based on their use case interest, is to establish greater value for their services alongside these greater cost efficiencies. This combination of delivering new higher revenues per GB services alongside greater cost efficiencies should overall de-risk and strengthen the long-term business case for mobile networks.

The effects of the 5G-MoNArch innovations on revenue and costs will be investigated and quantified in WP6 where a series of evaluation cases are being defined. These are currently set exemplary for Hamburg city and, relevant to WP4's network elasticity innovations, will include evaluation cases which consider:

- Temporary demand hotspots where network elasticity may help reduce over-dimensioning the network for these peak cases.
- Delivery of a range of services to different tenants (including for example sea port and smart city services) and investigation of how the benefits of resource orchestration and elasticity vary when considering use cases and networks of different geographic size (scale), mix of services (scope) and depths of tenant control over their individual network slices.

The cost structure of the network cases to be evaluated in WP6 to determine the business models include CAPEX and OPEX. CAPEX is predominantly in the infrastructure layer, like site costs, specialist RF hardware, spectrum, compute, networking, and storage; referred to as resource in this section. The economic merit of elasticity will emerge predominantly through effects that are measured by impact on OPEX. The following KPIs will be evaluated within WP6 to investigate the anticipated impact on business case as outlined above:

- **Revenue per GB -** The revenues expected to be received for services on the 5G-MoNArch platform per unit of data delivered. Some services will be highly commoditised and have a low revenue per GB whereas other newer business to business services may perform better against this metric.
- **Cost per GB -** The costs expected to be incurred in delivering services on the 5G-MoNArch platform per unit of data delivered. This will increase with the service requirements. For example, high reliability of a service will come at increased cost. Ideally the revenues from new services will outweigh the costs of adding them to multi-service networks benefiting from economies of scope. Also, ideally costs will decrease compared to non-5G-MoNArch architectures for some services due to the 5G-MoNArch innovations such as elasticity.
- **Social Benefits** - A financial assessment of the social benefits achieved through new 5G services from the 5G-MoNArch platform. These may include for example savings from more efficient energy usage via smart meters or reduced costs from road traffic accidents due to the introduction of vehicular services. These social benefits need to be considered in association with incremental revenues and network costs to understand the full value proposition for 5G-MoNArch. In the cases of some services the incremental revenues may be low but social benefits high implying some form of governmental intervention might be required.

2.2.2 The Economics of Slicing and Elasticity and the Interaction Between These

The effects of resource elasticity should be visible through the business models envisaged under 5G-MoNArch. These business models are underpinned by new stakeholders (tenants, mobile service providers (MSPs), data centre operators, VNF providers, InPs etc.) introduced via the new multi-tiered stakeholder model enabled by flexible virtualised multi-tenant and multi-service 5G networks supporting network slicing. The initial framing of the multi-tiered stakeholder model and stakeholder roles within this comes from those that are envisaged in [5GM17-D6.1].

The 5G-MoNArch architecture has the potential to unlock value generation opportunities in the above multi-tiered stakeholder framework as follows:

- Enabling new higher value revenue streams beyond enhanced mobile broadband (eMBB) only consumer services: This is done by providing end to end networking slicing that provides guarantees of service (with an appropriate choice of quality, availability, performance) that can be tailored to a tenant's individual requirements. Network slicing also enables tenants to have different depths of control on the slices being delivered to them. Some may be happy with SLAs where the MSP handles the instantiation and orchestration of network functions and resources. Other tenants, such as consumer of industrial high security services, may wish to implement some of the higher layers of the protocol stack handling security functions on their own infrastructure. At one extreme the tenants may be existing MNOs entering into network sharing

agreements and hence requiring much deeper control on the implementation of their network slices.

- Resource savings: This is the amount and type of resources consumed by an elastic function to perform a successful operation as compared to its inelastic counterpart (e.g. percentage of saved resources while providing 99% of the performance of the inelastic counterpart). Resource utilisation efficiency provides an effective metric to measure these effects.
- Cost efficiency gain: This metric measures the average cost of deploying and operating the network infrastructure to support the foreseen services. An elastic system should be able to be optimally dimensioned such that fewer resources are required to support the same services. In addition, in lightly loaded scenarios the elastic system should avoid the usage of unnecessary resources and reduce the energy consumption (thus limiting the operation expenditure). The new multi-tiered ecosystem enables this by introducing InPs separate to the MSPs so that those implementing the end to end network functionality can acquire resources from a set of infrastructure providers more dynamically. These savings can be delivered via:
  - Resource savings due to elastic NFs that track and adapt to observed and changing demand levels.
  - Intra-slice orchestration ensuring that NF placement on resources is managed and updated to always use the most cost-effective resources for the observed demand level and requirements of the individual slice.
  - Cross-slice orchestration ensuring that the network resources are shared and re-used as cost-effectively as possible across slices to deliver economies of scope.

From the above points there is an interaction and interdependency between the benefits of network slicing, network virtualisation and network elasticity. Network slices enable a wider mix of services to be offered from a single network platform and packaged and tailored towards the need of individual tenants. Virtualisation and network elasticity allow these network slices to be efficiently managed on a shared set of resources helping to maximise economies of scope and with the potential to acquire network resources more dynamically depending on the relationship with InPs. However, one important aspect of network slicing is the ability for different tenants to have different depths of control of the network slice depending on their needs. Giving tenants more control over their acquired slices may raise the value of the slice to the tenant commanding higher revenue. However, greater control by the tenant may also reduce the scope for the MSP to achieve cost efficiencies as their ability to flexibly make use of and orchestrate resources within NFs, slices and across slices will be limited if tenants require more control over slices. Therefore, the cost efficiencies that can be delivered by network elasticity from a MSP's perspective will in practice depend on the use case and corresponding set of tenants being served and their required level of slice control and needs to be balanced with potentially higher revenues from delivering this greater depth of slice control to tenants. Additionally, there should also be the opportunity for tenants to use network elasticity techniques across the depth of slice that they control.

3 Elastic Functional Architecture

This and the follow-up sections establish the main technical basis for resource elasticity. In this section, we provide a description of the elastic functional architecture developed within 5G-MoNArch while the next section is dedicated to present the specific mechanisms proposed to achieve elasticity in the three above mentioned dimensions, namely computational elasticity, orchestration-driven elasticity, and slice-aware elasticity.

Table 3-1: Innovation areas, challenges and potential solutions towards an elastic 5G architecture

Innovation areas	**Challenges**	**Potential solutions**
Computational elasticity	Graceful scaling of computational resources based on load	Elastic NF design and scaling mechanisms
Orchestration-driven elasticity	NF interdependencies	Elastic cloud-aware protocol stack
Slice-aware elasticity	End-to-end (E2E) cross-slice optimisation	Elastic resource provisioning mechanisms exploiting multiplexing across slices

Before getting into technical details, a look at Table 3-1 is useful to better understand the problems addressed by each elasticity dimension as well as the type of solutions proposed under each of the categories.

In that respect, a first challenge in virtualised networks is the need to perform graceful scaling of the computational resources required to execute the VNFs according to the load. The computational elasticity innovation acts at the VNF level by introducing the ability to scale them and their complexity based on the available resources: in case of resource outage, NFs would adjust their operation to reduce their consumption of computational resources while minimizing the impact on network performance.

The second challenge can be illustrated with the current 4G/LTE design of the protocol stack, where the NFs co-located in the same node are interdependent, i.e., interact and depend on each other. One example of logical dependencies within the stack is the recursive interaction between MCS, segmentation, scheduling, and radio resource control (RRC). In addition to logical dependencies, traditional protocol stacks also impose stringent temporal dependencies, e.g., the hybrid automatic repeat request (HARQ) requires a receiver to send feedback informing of the decoding result of a packet within 4 ms after the packet reception. Indeed, traditional protocol stacks have been designed under the assumption that certain functions reside in the same (fixed) location and, while they work close to optimality as long as such NFs are co-located in the same node, they do not account for the possibility of placing these NFs in different nodes. To deal with this challenge, a new protocol stack, adapted to the cloud environment, needs to be designed. This new protocol stack relaxes and potentially removes the logical and temporal dependencies between NFs, with the goal of providing a higher flexibility in their placement. This elimination of interdependencies among VNFs allows the orchestrator to increase its flexibility when deciding where to place each VNF, hence the name orchestration-driven elasticity.

A final challenge of the envisioned 5G architecture appears at the intersection of virtualisation and network slicing, i.e., the need for E2E cross-slice optimisation such that multiple network slices deployed on a common infrastructure can be jointly orchestrated and controlled in an efficient way while guaranteeing slice isolation. To address this challenge, it is important to devise functions that optimise the network sizing and resource consumption by exploiting statistical multiplexing gains. Indeed, due to load fluctuations that characterise each slice, the same set of physical resources can be used to simultaneously serve multiple slices, which yields large resource utilisation efficiency and high gains in network deployment investments, as long as resource orchestration is optimally realised.

It should seem apparent from the above description that the three elasticity dimensions are not independent from each other. As an example, computational elasticity may solve resource scarcity problems at short timescales and be sufficient as long as the resource shortage is small; however, with

longer timescales or extreme resource shortage, it may be better to re-orchestrate the network and move NFs out of the region with resource shortages. Hence, these interdependencies will be necessarily investigated throughout the course of the project.

The next sections deal with the logical relation between elasticity dimensions and their mapping to the 5G-MoNArch reference architecture.

3.1 Logical Interactions among Elasticity Dimensions

Implementing elasticity in a network is a challenging task that involves several elements in the architecture. Still, their high-level interaction can be summarised as depicted in the figure below.

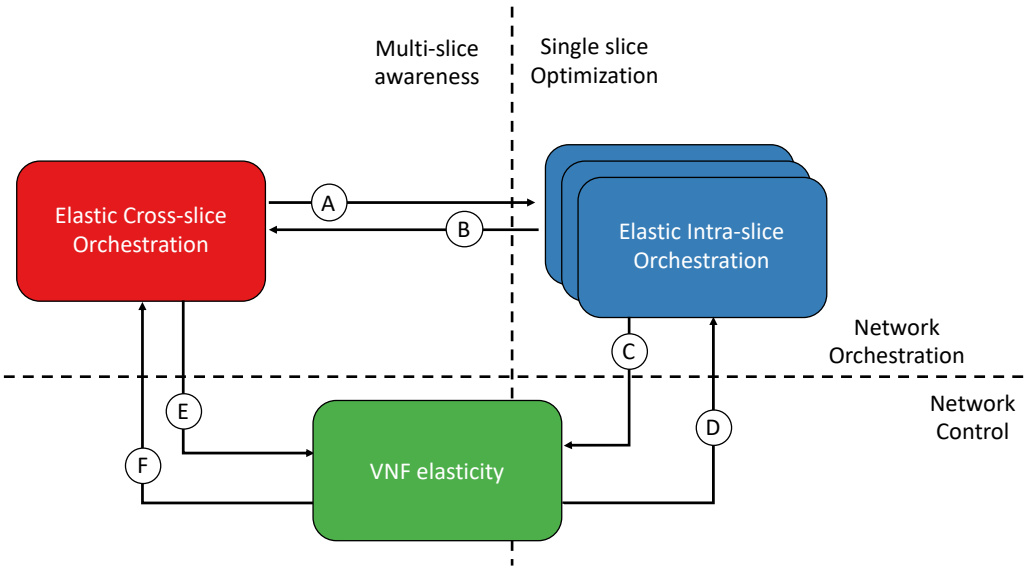


Figure 3-1: High level interactions across elastic modules

As described in Chapter 2, elasticity happens at different levels: at the VNF level, at intra-slice level and at infrastructure level across slices. Their interactions form hence a triangle, in which each vertex is represented by one of the dimensions of the elasticity. The edges are the needed functional interfaces that shall be included in the overall architecture, to finally enable an elastic operation.

Elasticity mainly addresses two domains: network orchestration and network control. The former shall incorporate the elements needed to (i) flexibly assign resources to different slices and (ii) find the best location of a VNF belonging to a certain network slice within the infrastructure. The latter, instead, shall provide an inner loop control of network functions, to enforce elasticity at faster time scales, such as the ones needed in the RAN.

Also, the dimensions of elasticity span across different domains with respect to the network sharing policies. Some elasticity aspects have necessarily to take into account different services at the same time while performing resource assignment or when to decide “elasticity level” of a network slice, as it is composed by a set of elastic VNFs. On the other hand, modules that are devoted to intra-slice elasticity targets one slice only, at least from the logical perspective. Then, depending on the implemented service provisioning model (e.g., Network Slice as a Service, or Infrastructure as a Service), these logical entities may be operated by different stakeholders.

The figure above also shows the required interfaces among modules: we describe them in the following.

- Interfaces A and B: these interfaces regulate the information exchange between the cross-slice elastic orchestration modules and the (logically) different intra-slice orchestration modules that take care of the intra-slice orchestration. Interface A transfers information related to the instantiation of an elastic NS on a shared infrastructure, i.e., the selected VNFs needed to provide a given service, and the associated resources. This information is needed by the intra-slice orchestration module to perform then an optimisation within the resources granted to the

slice. Also, this interface is used to exchange the information related to the lifecycle management of the network slice. The cross-slice orchestrator has an E2E view of the available resources, so it can command to a given slice to acquire or release new resources, as needed. In turn, interface B has mostly the role of reporting the used resources by a given network slice, that is eventually used by the cross-slice orchestrator to keep the resource map up-to-date. Also, this interface is used to deliver re-orchestration triggers coming from inside a slice (e.g., when the computational resources originally granted to a slice are not enough anymore to provide the required service).

- Interfaces C and D: these interfaces cross the Orchestration / Controller domain shall be used to exchange information regarding the behaviour of specific elastic VNFs that build a network slice. For instance, interface C is used to transmit to the VNF the information needed for the elastic behaviour such as the total amount of available computational resources or memory. Also, interface C logically covers the one used to instantiate a VNF in a specific location of the system (e.g., in the edge). On the other hand, interface D is used mainly for reporting by the VNFs: information about the current resource utilisation, together with other network related metrics (e.g., in case of a RAN function, the number of used physical resource blocks (PRBs) or the average signal to noise ratio (SNR) of each user). In general, this information is the used by the AI and big data analytics modules. Finally, information needed for the experiment-driven optimisation, as defined in [5GM17-D2.1], shall also be provided through this interface.
- Interfaces E and F: these logical interfaces are not mapped to a specific interface in the overall 5G-MoNArch architecture, but rather specify a functional behaviour. Interface E represents the abstraction of a network slice blueprint that is composed by a certain number of elastic VNFs and, finally, achieve an elasticity level defined as a function of them. Similarly, the information about the elastic behaviour of each VNF shall be available at the cross-slice orchestration. That is, the information about the used resources and the graceful degradation trends of a VNFs need to be taken into account by the cross-slice orchestrator when performing the slice admission control and onboarding operations.

These interfaces will be further detailed in the following sections and fully specified by the end of the project lifetime.

3.2 Architectural Implications for Resource Elasticity

3.2.1 Layers in 5G-MoNArch Overall Architecture

The interactions among elasticity dimensions described above describe high level functionalities that are valid independently of the underlying architecture and the chosen business model. In this section, we further specify our architecture with respect to the 5G-MoNArch overall architecture as defined in [5GM17-D2.1]. The overall architecture has the explicit goal of accommodating any kind of network slice, including resilient network slices as defined in [5GM18-D3.1]. In this section, we analyse the mapping of the elements and the interfaces specified above with the ongoing work of the overall architecture. Being the 5G-MoNArch overall architecture built with the goal of enhancing the state of the art while maintaining the backward compatibility, it is expectable that the elements defined as a result of this work may find their way into standardisation.

The 5G-MoNArch architecture includes four different layers that decompose the main functionalities of a softwarised 5G: Service Layer, M&O Layer, Controller Layer and Network Layer. In the following, we review them from a network elasticity perspective.

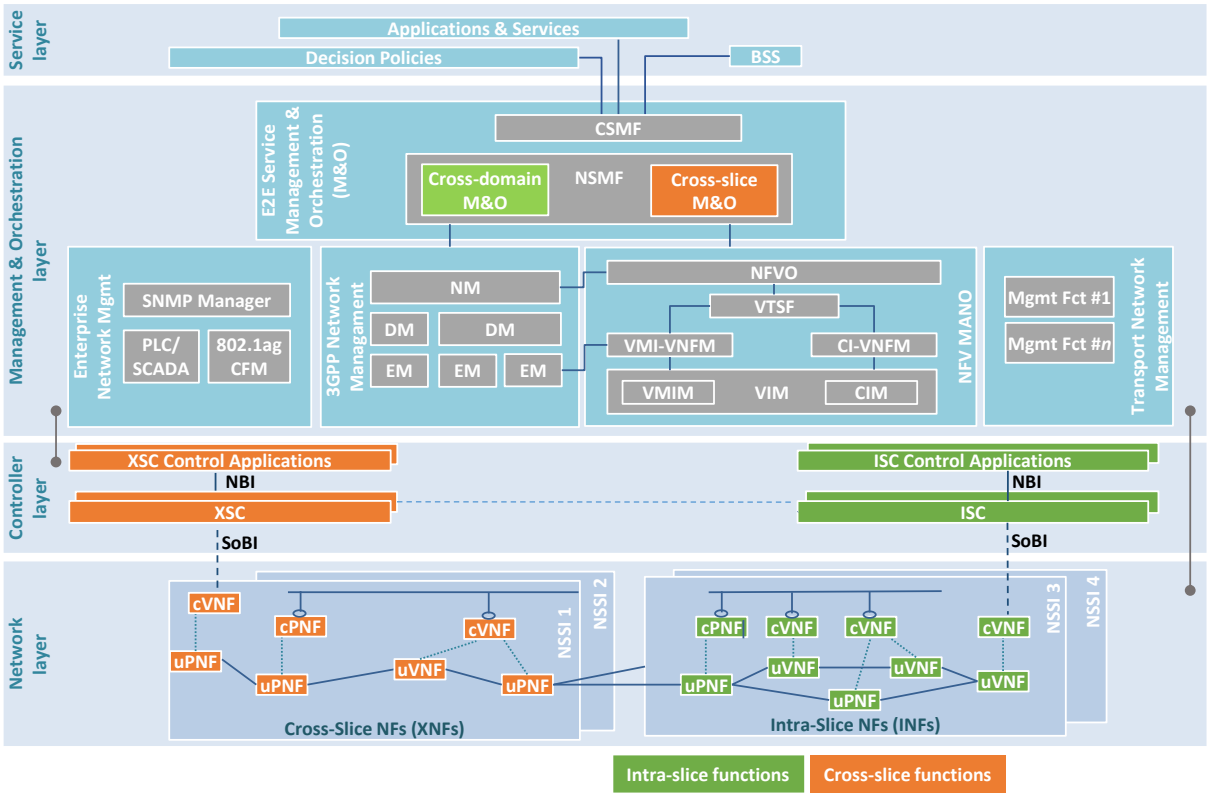


Figure 3-2: Intermediate 5G-MoNArch overall functional architecture as defined in [5GM18-D2.2]

Service Layer

The Service Layer, at the top of this structure, comprises business-level decision functions, applications, and services, operated by a tenant or other external entities. Such functions and services are applied to the network through operations in the M&O Layer. This layer provides a multi-tenant, multi-service environment that enables E2E service and resource orchestration.

The Service Layer is the "door" to the tenant and implements all the functionalities needed for example to perform network slice admission control and the mapping of a specific slice type to a well-defined chain of VNFs. The functionalities implemented in this layer mainly fall into the elasticity at cross slice level and the afferent interfaces. For example, an ML admission control system as proposed in Section 4.3.2.6 can be implemented in this layer, for that purpose, it will need information about (i) the used and available resources (spectrum, but most importantly computational resources) and (ii) the elasticity level of the VNF that will compose the slice to be admitted.

M&O Layer

Similarly to the ETSI NFV MANO [ETSI14-NFV] architecture, the M&O Layer in Figure 3-2 incorporates components that deal with the life cycle management of the virtual resources via the virtual infrastructure manager (VIM), the life cycle management of the VNFs via the virtual network function manager (VNFM), and the overall orchestration of the resources and the services on top of those managers via the network function virtualisation orchestrator (NFV-O). More specifically, the M&O architecture developed by this project divides this layer into two specific sub-parts: (i) the slice specific sub-part, a full MANO stack composed by NFV-O, VNFM and VIM that orchestrates resources within a network slice and (ii) a cross-slice orchestrator that deals with resource assignment across different slices running on the same infrastructure. This dichotomy fits the two levels of elasticity we described in Chapter 2. This layer accomplishes the functionalities performed in the cross-slice and intra-slice elastic elements. For instance, the elastic resource assignment across network slices shall be decided at the cross slice orchestrator and enforced by the MANO stack. Similarly, elastic orchestration algorithm will involve the different elements of a MANO stack. Finally, the interaction among the cross-slice and intra-slice modules will closely follow the interfaces A and B of Figure 3-1 above.

Controller and Network Layers

Following the recent trends in network softwarisation, 5G-MoNArch employs a controller-based architecture that applies the network programmability concept to all the VNFs of a network slice. These controllers centralise the network control logic provided by the application running on top of a northbound interface and enforce these decisions in the agents placed in the Network Layer. We further split controllers into two categories, depending on their role: Intra-slice controllers (ISCs) control dedicated NFs (i.e., the ones that are assigned to only one slice) and cross-slice controllers (XSCs) that actuate on shared NFs across slices.

Among the functionalities fulfilled by the controllers there is the required translation of the northbound management and orchestration services into commands that are applied to the actual VNFs and PNFs. The VNFs and PNFs compose the lower layer in the reference architecture, referred to as Network Layer. In order to abide by the fundamental 5G direction for multi-tenancy support on top of a softwarised and slice-enabled network, the Network Layer incorporates legacy control plane (CP) and user plane (UP). Implementing elasticity at the bottom level of the architecture is mostly achieved by the elastic VNF design and its interaction with the M&O Layer. Elastic VNFs needs a constant monitoring of the computational resources, possibly through the controllers of the Controller Layer. In the next section, we provide more details on how this behaviour can map into specific network procedures involving the architectural elements.

3.2.2 Architectural Components Enabling Elasticity

Within each of the constituent layers of the 5G system architecture shown in Figure 3-2, several components are essential to provide and/or exploit elasticity in the system. In particular, the following components should be highlighted:

- Elastic VNFs: as explained later in Section 4.1, VNFs can be (re-)designed with elastic principles in mind such that (i) the computational resources available for its execution are taken into account, or (ii) its temporal and/or spatial interdependencies with other VNFs are mitigated, hence allowing its orchestration to be much more flexible when deciding for a location for its execution. From the above it follows that the existence of this type of VNFs is needed to exploit computational elasticity.
- Elastic intra-slice orchestrator: elasticity-aware algorithms are needed to orchestrate the different VNFs that are part of the same slice. The tasks of such an elasticity-aware orchestrator may include re-locating VNFs (from central to edge cloud and vice versa, or from one server to another) depending on available resources, horizontally or vertically re-scaling the amount of resources allocated to one particular VNF or a set thereof, clustering and joining resources from different locations, etc. Hence, this module would be responsible for implementing the dimensions of elasticity described in Section 4.2.
- Elastic cross-slice orchestrator: the cross-slice orchestrator is in charge of performing the management and control of the multiple slices that share the architecture, i.e., enabling slice-aware elasticity as described in Section 4.3. Some or all of these slices may be elastic, i.e., slices that do not have totally stringent requirements but rather admit graceful degradation. For those cases, specific orchestration algorithms need to be designed.
- Elastic controller: the SDN-like centralised controller is responsible for carrying out the control of the elastic VNFs within a slice, as well the shared elastic VNFs across slices, ensuring a correct multi-tenant operation. This is done through applications that run on top of the controller and implement the logic of the elastic VNFs.

The rest of this section describes in detail which are the architectural implications of each of the elasticity dimensions referred to throughout this document.

Computational Elasticity

To introduce computational elasticity, several requirements to the 5G-MoNArch architecture have been identified. In general, a VNF itself needs to have additional functionalities to react on dynamic shortages of computational resources, to reach graceful degradation in radio performance. Furthermore, it is necessary to have a centralised entity, which controls multiple VNFs to react on short-term dynamics.

The central entity needs to react on lack of resources being aware of the impact on the radio performance. VNF individual decision might cause a lack of resources for other VNFs or unwanted decreased radio performance and thus possible SLA violations. A VNF will have a certain degree of resource isolation (e.g., CPU pinning). However, in a non-pinned setup, this may happen. Therefore, it is mandatory for the controller to have knowledge about the behaviour of the VNFs. In other words, the containers and VMs need to be associated with the functionalities per VNF running inside to react from a short-term perspective. To interact between the M&O and the Controller Layer for re-/orchestration and life cycle management procedures, there is a need to define interfaces between those layers. Such interfaces need the ability to trigger re-orchestration, influencing and monitor VNF behaviour as well as resource consumption. The necessary interaction among the considered layers to enable computational elasticity are described in the following.

- The M&O Layer especially the VIM needs to generally allocate the necessary computational resources among the VNFs during re- /orchestration phase (long term control loop).
- The Controller Layer needs to support flow control to react on the radio performance, dynamically. This is done within the given maximum amount of resources allocated by the VIM in the M&O Layer for each VNF (short term control loop).
- The Controller Layer can set parameters of the VNF how to handle the available computational capabilities (short term control loop).
- It is for further study if the Controller Layer might dynamically reassign computational resources in tight arrangement with the VIM or if the VIM can react dynamically on a fast time scale.
- The VNF needs to be able to report on the virtual resource consumption to the Controller Layer.
- The Controller Layer needs to be able to request more computational resources from the MANO stack or at least needs to trigger a re-orchestration of specific VNFs.
- The Controller Layer needs to be able to influence the behaviour of the VNF to react on short term.
- The Controller Layer needs to know the influence on the radio performance of the computational resource shortages of specific VNFs.
- The Controller Layer needs the ability to influence VNF behaviour to react on computational resource shortages with respect to the radio performance.

Based on the aforementioned necessary requirements to enable computational elasticity, Figure 3-3 shows exclusively the affected layers of the functional overall 5G-MoNArch architecture given in Figure 3-2.

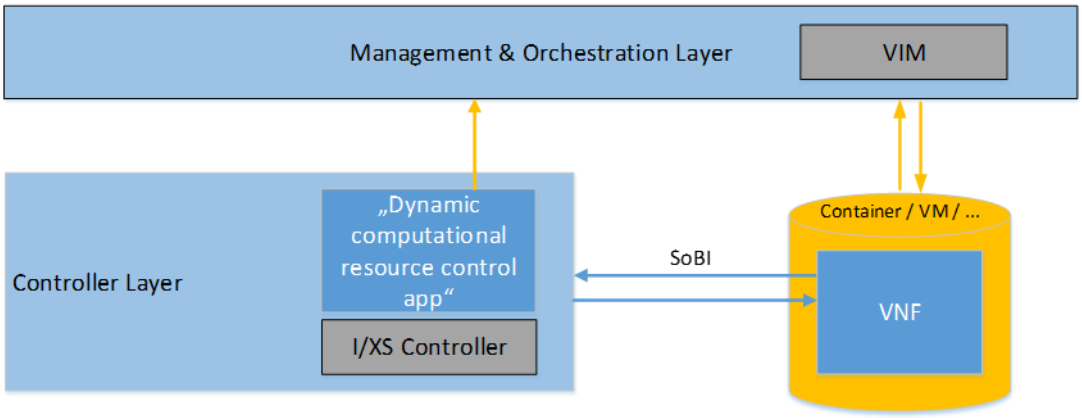


Figure 3-3: Exclusively affected layers to introduce computational elasticity

It is for further study if additional information exchange or functionality is needed to be introduced. For instance, if the VIM cannot react dynamically enough on changing container/VM settings, the Controller Layer might take additional action to improve the resource allocation. Especially, if the intention is to introduce functional separation in the lower layer radio protocol stack the round trip time (RTT) between MANO and Network Layer might become critical.

Orchestration-driven Elasticity

Orchestration-driven elasticity is enabled by the cooperation of several blocks of the 5G architecture, belonging to different Network Layers. This level of elasticity is exploited as a measure to free physical resources at some location of the network, by moving running VNFs to other locations (for example, from the edge of the network towards the central cloud). For this purpose, the network needs to monitor the state of the resources and take re-orchestration decisions whenever necessary, based on the incoming new service requests or on the modifications of the performance requirements of running services. The blocks of the 5G-MoNArch architecture involved in the implementation of orchestration-driven elasticity mechanisms are represented in Figure 3-4.

Orchestration-driven elasticity is performed either at the Cross-Slice M&O by the global resource orchestrator or at an intra-slice level by the Cross-Domain M&O. Both entities are part of the network slice management function (NSMF), which splits service requirements as received from the communication service management function (CSMF) and coordinates with multiple management domains for E2E network slice deployment and operation (see Figure 3-2 and [5GM17-D2.1]). The Cross-Slice M&O has a global view on the network and on the running slices and is also responsible for the application of slice-aware elasticity mechanisms (see Section 4.3). It can command the relocation of VNFs either in conjunction with or independently from the application of cross-slice resource allocation algorithms. In the case of a non-shared set of NFs and associated resources (a Network Slice Instance, NSI) or whenever the re-location of VNFs can be handled internally to a network slice without passing by the global orchestrator, orchestration-driven elasticity can be managed by Cross-Domain M&O.

Orchestration-driven elasticity mechanisms can be triggered by three kinds of events:

- The arrival of new slice requests from the CSMF [5GM17-D2.1], which demand the allocation and activation of new services.
- The request for an update of the requirements of already-running NSIs, transmitted to the NSMF by the CSMF [5GM17-D2.1].
- The request for an update of the requirements of already-running NSIs, forwarded to the NSMF by the Network Slice Subnet Management Function (NSSMF, belonging to the 3GPP Network Management block [3GPP17-28801]) and related to the cloud domain. This event may be due to a request either of the NFV-O or from the Network Layer, which asks for additional computational resources. Notice that for the sake of simplification, the NSSMFs are not explicitly shown in Figure 3-2, although they belong to the network architecture. They appear instead in Figure 3-4 to make clear their monitoring role at a sub-slice level: Sub-Slice (SS) Resource Monitoring and Network Slice Subnet Instance (NSSI) Performance Monitoring are the two NSSMF's functionalities devoted to control and evaluate the state of available/busy resources and of the already-running NSIs at a sub-slice level. Multiple NSSMF take care of different sets of subnets and can be organised as needed according to different reasons: technologies, processes, practical needs (vendors).

The decision on how and where to relocate VNFs is taken at the NSMF level by considering the (possibly updated) slice requirements, the new slice instantiation requests, and the resource availability. All along the network operations, the NSSMF monitors the status of resources, collecting data from the NFV MANO and from the Network Layer; the NSSMF filters all the data from the network management system and provides a view at sub-slice level. This information is periodically forwarded to the NSMF, which takes it into account for the implementation of the orchestration-driven elasticity algorithm. The output of this algorithm is eventually passed back to the NSSMF and then implemented through the NFV MANO.

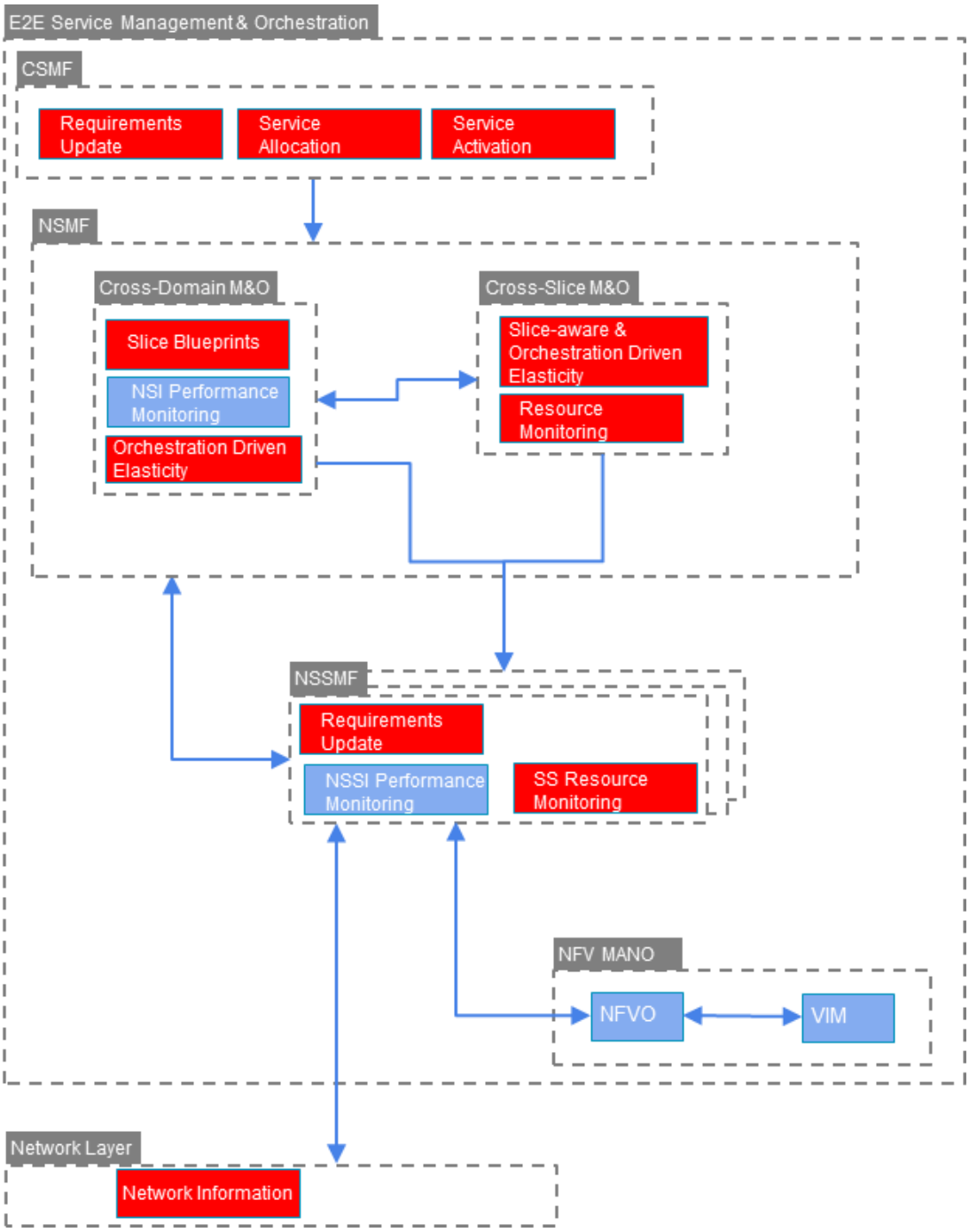


Figure 3-4: Orchestration-driven and slice-aware elasticity enabling architecture

Slice-aware Elasticity

Slice-aware elasticity refers to the ability to serve multiple slices over the same physical resources while optimising the allocation of computational resources to each slice based on its requirements and demands. The algorithms/policies for slice aware elasticity reside at the E2E service M&O Layer and more precisely at the Cross-Slice M&O entity, which, triggers the above-mentioned resource optimisation via the 3GPP Network Management or the NFV MANO entities, as depicted in Figure 3-2.

In 5G-MoNArch, slice-aware elasticity is supported through three different approaches that can be mapped to three different phases of the lifecycle of a slice instance (Figure 3-5):

- *Preparation-time approach:* Slice behaviour analysis can be a critical asset for elasticity provisioning, since statistics can be exploited in the slice preparation phase to efficiently decide the basic configurations and set the network environment.
- *Instantiation-time approach:* Flexible slice admission control and network slice blueprint[1] or template[2] analysis (as the scheme proposed in Section 4.2.2.1) are applied during the slice instantiation and configuration phase, so as the activated slices have reduced probability to experience a computational resource outage.
- *Run-time approach*: Advanced sharing of computation resources among VNFs of multiple slices provide resource elasticity as the involved slices are in operation, by exploiting multiplexing capabilities.

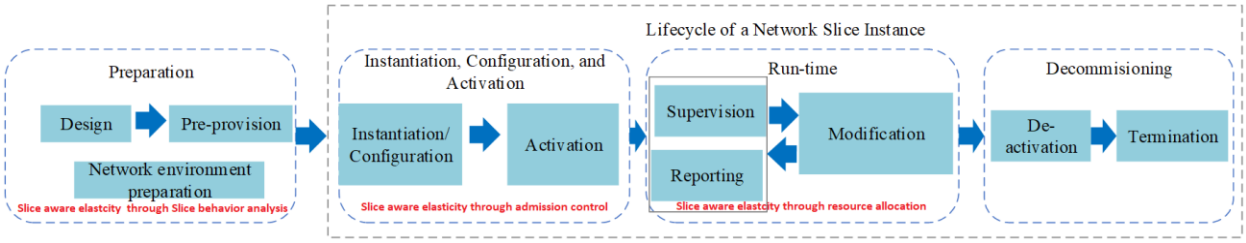


Figure 3-5: Slice aware elasticity support in different phases of the lifecycle of network slice instances (based on [3GPP17-28801])

Referring to the architectural components that undertake the major tasks for each one of the approaches, the functionality needed for the run-time approach is part of the NSMF and resides at the Cross-layer M&O. The input information needed is available at the NSMF via the NSSMF, while the output commands target the NFVO of the NFV MANO. Regarding the instantiation-time approach, input from the NSSMF or NFV MANO can be used, while info from CSMF, regarding the service requirements, might be exploited. Again, the output commands target the NFVO of the NFV MANO. Finally, the functionality needed for the preparation-time approach can be provided by the AI and Big Data Analytics Engine, an architectural entity described in Section 3.3.

3.3 AI & Big Data Analytics Engine

In 5G-MonArch, we foresee that all the above elasticity-related functionalities could be greatly enhanced with an AI and Big Data Analytics Engine. This activity is in line with the goal of the ETSI ENI (Experiential Network Intelligence) [ETSI17-ENI].

Focused on optimising the operator experience, this engine would be equipped with big data analytics and AI capabilities that could enable a much more informed elastic management and orchestration of the network, often allowing proactive resource allocation decisions based on the history rather than utilising reactive approaches due to changes in load.

More specifically, the ETSI ENI focuses on improving the operator experience, adding closed-loop algorithms controlled through AI solutions, which exploit context-aware and metadata-driven policies to more quickly recognise and incorporate new and changed knowledge, and hence, make actionable decisions. In addition, ENI will specify a set of use cases, and the architecture, for a network supervisory assistant system based on the 'observe-orient-decide-act' control loop model (see Figure 3-6). This model can assist decision-making systems, such as network control and management systems, to adjust services and resources offered based on changes in user needs, environmental conditions and business goals.

[1] **Network slice blueprint:** A complete description of the structure, configuration and the plans/work flows for how to instantiate and control the Network Slice Instance during its life cycle. A Network Slice Blueprint enables the instantiation of a Network Slice, which provides certain network characteristics (e.g. ultra-low latency, ultra-reliability, value-added services for enterprises, etc.) [NGMN16].

[2] **Network slice template:** description of the structure (and contained components) and configuration of a network slice [3GPP17-28801].

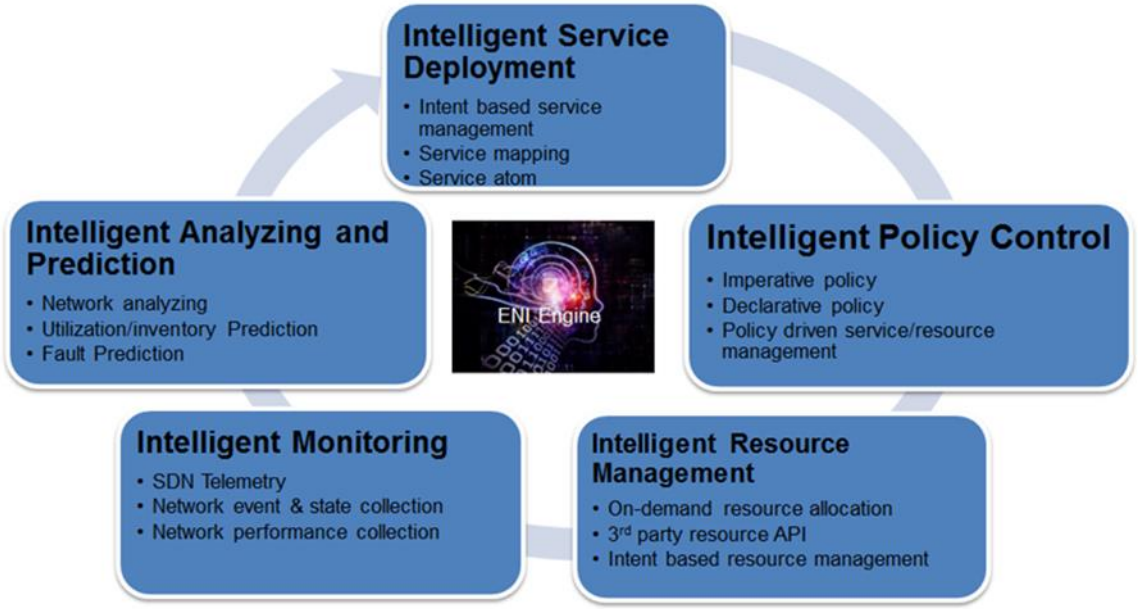


Figure 3-6: Control loop model in the ETSI ENI Engine [ETSI17-ENI].

Concerning the ENI use cases, it is worth highlighting that the Use Case #3-2 focuses on "intelligent network slicing management", which completely fits with our activity: placing or adjusting the NSI (e.g., reconfiguration, VNF scale-in, scale-out) to achieve an optimised resource utilisation with changing context. Also, ETSI identified three main blocks within the ENI engine: the inference system, the data analysis, and the optimisation block (see Figure 3-7:).

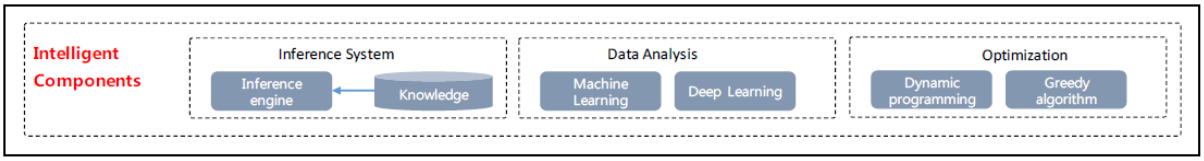


Figure 3-7: Intelligence components in the ETSI ENI Engine [ETSI17-ENI].

The first block enables the system to acquire useful knowledge, identify features and detect anomalies. The unsupervised ML mechanisms are algorithms part of the AI family that do not require labelled training data and can be classified as being either parametric or non-parametric. Some exemplary schemes that can be successfully implemented for these goals are presented in [BLJ13]. The second block acts on the acquired information to predict future status of the system, and autonomously detect anomalies and faults. Supervised ML algorithms require training with labelled data and include, among others, the artificial neural network (ANN) and support vector machine (SVM) algorithms [BLJ13], and can be successfully implemented in the data analysis block. ML solutions such as ANN and clustering schemes are identified as potential solutions for the slice blueprint analytic problem discussed in Section 4.3.2.4, which targets slice-aware elasticity.

The last block is related to the decision-making tools, such as optimised network control and management. Dynamic programming and RL are tools that can be used as optimal decision tools. Dynamic programming schemes can be used to solve problems that can be modelled as Markov Decision Process, and require complete knowledge of the system. RL is a technique that permits an agent to modify its behaviour by interacting with its environment [SB98]. This type of learning can be used by agents to learn autonomously without supervision. In this case, the only source of knowledge is the feedback an agent receives from its environment after executing an action. RL techniques are further

discussed in the context of computational elasticity in Section 4.1, and are identified as solutions to deal with the proactive (re)allocation of virtual resources (see Section 4.2.2.1).

A first mapping of the AI & Big Data Analytics Engine in the 5G-MoNArch M&O Layer is presented in Figure 3-8:. This block will include the ML and RL functionalities that will be developed in WP4 to achieve elasticity by orchestrating and managing virtual and physical resources across multiple slices.

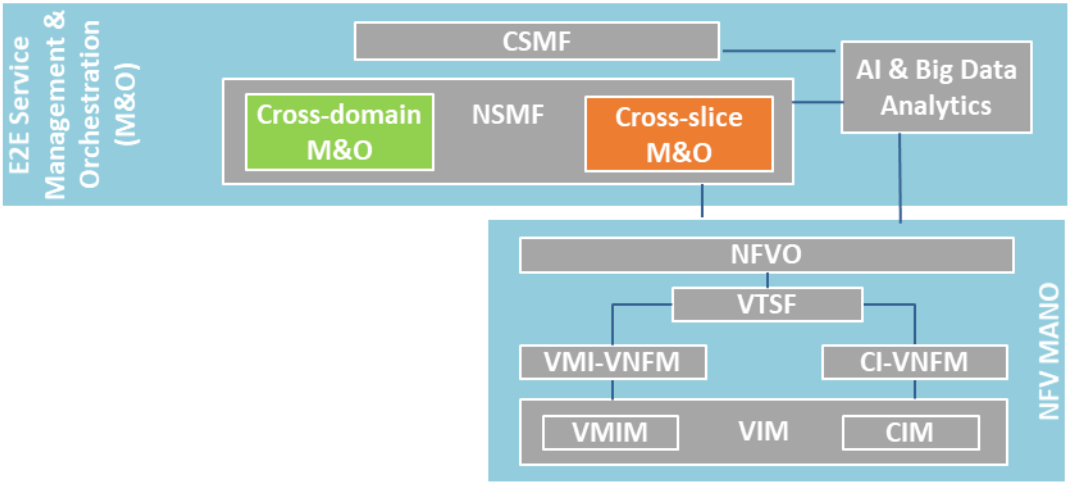


Figure 3-8: Mapping of the AI & Big Data Analytics in the 5G-MoNArch architecture.

4 Mechanisms for Resource Elasticity Provisioning

In this chapter, we provide a set of mechanisms to provision resource elasticity along the three dimensions of elasticity recurrently named throughout this deliverable. Table 3-1 already anticipated a summary of the challenges and the type of solutions that elasticity addressed. This chapter follows up in detail that summary and is organised as follows. Each of the following sections is dedicated to one of the aforementioned elasticity dimensions and includes a state of the art analysis as well as the latest status of ongoing investigations within the project on possible technical solutions.

4.1 Computational Elasticity

In the context of wireless communications, the concept of elasticity usually refers to a graceful degradation in performance, for example when the spectrum becomes insufficient to serve all users. However, in the context of a cloudified operation of mobile networks, when addressing elasticity to resource shortages, we also consider other kinds of resources that are native to the cloud environment, such as computational and storage resources. One of the most appealing advantages of a cloudified network is the possibility of reducing costs by adapting and re-allocating resources following (and even anticipating) temporal and spatial traffic variations in a centralised manner. However, it is expectable that the cloud resource assignment is occasionally exceeded by the induced burden. This is particularly true for Cloud RAN (C-RAN) deployments, which are known to be highly variable [CCY+14]. In this scenario, allocating resources based on peak requirements would be highly inefficient. VNFs, instead, shall efficiently use the resources they are assigned, and become computationally elastic, i.e., adapt their operation when temporal changes occur in the load and hence in the available resources. Elasticity has also been considered by non-VNFs cloud operators, but as mentioned earlier our concept deviates very much from theirs: the time scales involved in RAN functions are significantly more stringent than the ones required by e.g., a Big Data platform or a web server back-end. Another key difference is that resources are sparser (e.g. they are distributed across the "edge clouds"), which reduces the possibility of damping peaks by aggregating resources.

In order to overcome such computational outages, one goal of 5G-MoNArch will be to design VNFs that can gracefully adjust the amount of computational resources consumed while keeping the highest possible level of performance. RAN functions have been typically designed to be robust against shortages on communication resources; hence, 5G-MoNArch aims at making RAN functions also robust to computational shortages, by adapting their operation to the available computational resources.

4.1.1 State of the art

C-RAN based architectures, possible functional RAN protocol splits as well as CP and UP separation, enable the possibility to apply network slicing in future mobile networks. A customised set of VNFs as well as PNFs need to be chained to fulfil contradictory service requirements for different logical networks belonging to different tenants. Support of network elasticity and real-time processing are seen as major challenges in this context [FNK+16]. C-RAN is seen as one potential technique to gain from computational resource pooling with graceful degradation in performance [PHT16][BCK+12]. Possible radio protocol split options are required to find trade-offs between centralised and distributed functional placement to satisfy contradictory service requirements [RBD+14][ABB+17].

When it comes to practical considerations, a first LTE based VNF system is implemented based on OAI [KNS+17][OAI]. Each network element, such as eNB, mobility management entity (MME), serving/packet gateway (S/P-GW), and home subscriber system (HSS), runs as a chain of multiple VNFs. Dynamic RAN function chaining is seen as crucial to enable network slicing, as well, but also seen as critical when it comes to managing resources at runtime. Real-time constraints in the base station at lower layers impose extreme challenges.

A promising approach is described in [FNK+16]. Besides a network-slicing enabled 5G architecture a summary of possible technologies and frameworks can be found to enable cloud and NFV control when slicing the RAN. Currently available southbound application programming interfaces (APIs) and protocols are seen as not useful in that case. Therefore, a novel southbound control protocol, called FlexRAN, is proposed which enables management for time critical physical (PHY) and medium access control (MAC) layer VNFs during runtime. FlexRAN provides an open source software defined RAN

implementation to separate CP and UP in the RAN. A CP design with support for virtualised real-time RAN CP is currently developed, and a new southbound agent API is designed especially for real-time critical processing functions, such as MAC scheduling.

However, there are obvious gaps in the NF design that are orthogonal to the architectural construction as well as to novel NFV control enabling APIs. Some elements need additional components to cope with elasticity at NF level. In addition, the amount of resources available at any time should be constantly monitored by the monitoring module, which should provide useful information to both the controllers and the orchestration. Hence, novel elastic functions have to be designed as well as mechanisms for NF scaling. Furthermore, with NR Rel.15 novel RAN functionalities are defined, which provide higher degrees of freedom, thinking about NF design considering computational elasticity in the RAN, in addition to LTE-A pro, such as higher flexibility in numerology, slot based scheduling and bandwidth adaptation [3GPP17-38300].

4.1.2 Mechanisms for Computational Elasticity

We next discuss some of the potential benefits of the introduction of elasticity in the VNF design. To this aim, we revisit the non-negligible CPU resources needed for their operation which, in the traditional approach, is 100% guaranteed by the tight integration with the hardware.

While elasticity can be introduced to any function of a softwarised network, we currently investigate two important RAN functionalities with huge influence on the number of computational resources needed to process data in a sufficient manner. On the one hand, the processing power needed to modulate and demodulate as well as encode and decode the radio signal is one of the most consuming functionalities in RAN signal processing. This depends on the number of spatial layers (ranks) which are used for higher spectral efficiency and thus higher cell capacity comes with a higher computational effort. We remark that both MCS selection and rank usage are sensitive functionalities for spectral efficiency. For both, novel strategies need to be explored how to handle scarcity of computational resources under consideration of the loss in spectral efficiency.

Furthermore, we present an analytical model for computational resource allocations to the network slices based on their SLAs. Finally, we provide a preliminary evaluation of computational resources control for container-based virtualisation.

4.1.2.1 MCS Selection for C-RAN

We consider a C-RAN scenario where the scheduling of a number of base stations is done by a central entity. In a traditional approach, the VNF performing frame decoding has to be dimensioned for peak capacity as follows from Section 2.3, i.e., all PRB using the highest MCS, which corresponds to ideal radio conditions for all users (having a set of users with good channel conditions is a common assumption for schedulers that rely on Opportunistic Scheduling techniques [AM13]). However, planning for peak capacity not only requires prior knowledge of the users' demand, which is a difficult problem *per se*, but also results in resource wastage when mobile traffic falls below this peak.

Thus, let us assume a C-RAN scenario where resources are not over-provisioned, and the scheduling function for the UL is serving a large enough set of base stations. Our formulation focuses on uplink scheduling, as the load at the cloud-computing platform is dominated by uplink frames [BCK+12]; however, the formulation could be easily extended to incorporate downlink scheduling by simply adding downlink transmissions with their (low) computational cost. Under these conditions, it will be likely that during short periods of time a set of users (experiencing good channel quality and hence using high MCS) require more capacity than available, as higher MCS require more iterations to be decoded [BCK+12]. A non-elastic function would fail to e.g. decode the PRB, this resulting in an abrupt degradation of performance.

A cloud-aware MCS selection function helps to address this challenge more efficiently. Given that the "disruption" is caused by a relatively large set of users using high MCS, one elastic strategy is to purposely "downgrade" some of the MCS to be used in the PRBs, to find a combination that can be supported by the available computational resources. This version of the function, originally proposed in [RSV15], might have better short-term fairness properties than the previous one, as users are still scheduled but at a lower rate. Also, this might support a more graceful degradation in the absence of

resources. However, one key challenge is to find the most appropriate set of MCS to be used, as by definition the computational capacity is limited and therefore, the algorithm has to find the solution in short time.

4.1.2.2 Scheduling Under Resource Constraints

Running a non-elastic C-RAN scheduling function under drastic traffic variations (a typical condition nowadays) is challenging, as it results in abrupt disruptions when there are not enough resources to perform a computation. We discuss here another way to introduce elasticity in the scheduling function, which assumes that the amount of computational resources available, i.e., the capacity, it is known. This capacity is denoted as C.

Like in the previous section, the scheduler starts by selecting those users as in a regular mode of operation, and checks if the amount of required resources to perform this decoding (i.e., iterations), denoted as $\tilde{C}$ is less than the capacity C.

This estimation of $\tilde{C}$ can be done following e.g. the model introduced in [RSV15], that provides the computational complexity as a function of the SNR and the selected MCS. If $\tilde{C} < C$, it proceeds as a regular scheduler would do; otherwise, a different scheduling decision has to be taken (in the previous section, elasticity was achieved by downgrading all the MCS). So, another way to introduce elasticity in the scheduling function would be to use the following algorithm. Building on the widely-used Proportional Fair (PF) scheduling [Sto05], the function selects the candidate set of users U that maximises performance.

Then, it follows these steps:

- Compute the cost of scheduling of U, namely, $\tilde{C}$. If $\tilde{C} < C$, then U is scheduled and the algorithm stops.
- Otherwise, discard the elements in U in increasing order of the PF metric (i.e., the lowest first), until $\tilde{C} < C$. Note that larger PF metrics are usually provided by higher MCS values or starving users.
- Add to U those users that would maximise performance, but not accounting for those discarded in the previous step (that are kept in memory).
- Repeat 1--3 until all base stations are scheduled.

Note that in Step 2, the algorithm discards some users with relatively high PF metric (i.e., the instantaneous rate over long term rate ratio), this resulting in a degradation of performance as compared with the default PF behaviour in traditional networks. We next quantify how this degradation occurs as the capacity available C is reduced. To this aim, we simulate a C-RAN scenario with 5 base stations serving 5 users each, where user i (with $i = 1 \ldots 5$) is located at a distance $i \cdot d_0$, with d_0 being the distance at which the SNR is 17.6 dB (i.e., the maximum SNR defined for the highest MCS by 3GPP). The wireless channel follows the Rayleigh model [Rap11] with $\gamma = 3$ and $\sigma = 6$ dB, while the MCS complexity model follows the ones presented in [RSV15].

We first evaluate the degradation function of this scheduling algorithm, as discussed in Section 2.1.3. To this aim, we first assume an unconstrained scenario to compute the reference performance and required capacity. Then, we reduce the capacity by a given *shortage*, and compute the resulting performance for the same PF scheduler (that fails to serve users if $\tilde{C} < C$) and our elastic scheduler. The resulting degradation in terms of performance are illustrated in Figure 4-1, for the "Standard PF" scheduler and the "Elastic PF" scheduler.

The traditional scheduler exhibits a proportional degradation of performance with the resource shortage, e.g., for a 20% reduction in resources, performance is reduced by approx. 20%. In contrast, the elastic scheduler provides a sub-linear degradation, thanks to the selection of a feasible set of users. We argue that this sub-linearity corresponds to graceful degradation of performance provided by the elasticity of this approach, which discards users that are likely to have good channel conditions (and hence higher MCS) to favour others with a better decoding efficiency (i.e., PF metric over complexity ratio).

We next evaluate the timings associated with the operation of the scheduling algorithm. To this aim, we initially assume the same scenario as above, with a 0% shortage, and then we periodically add a new

base station with 5 more users. We illustrate in Figure 4-2 the resulting performance degradation for the two scheduling functions, where dashed lines mark when a new base station is activated.

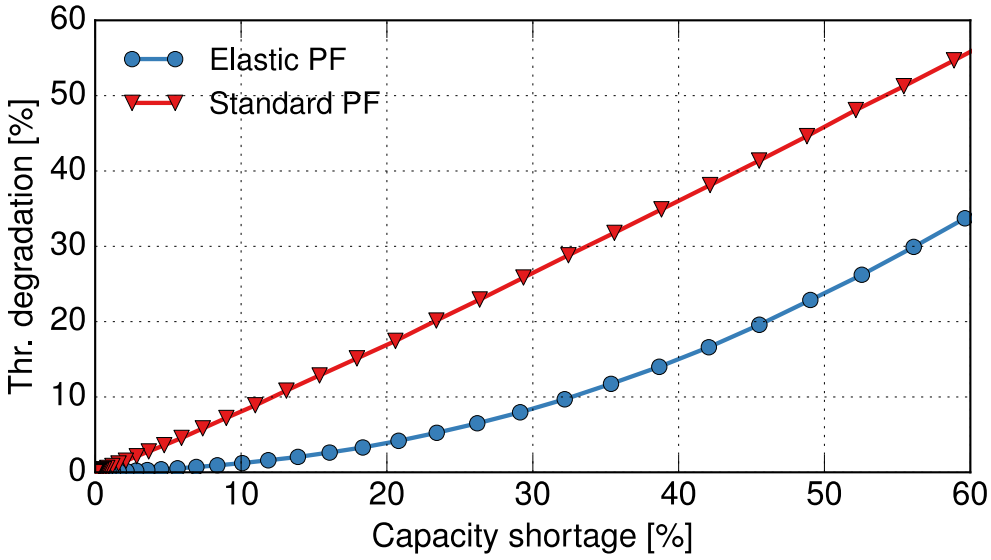


Figure 4-1: The elastic scheduling behaviour

As the figure illustrates, an elastic approach can prevent abrupt reductions in performance, this way facilitating *rescuability* as the smooth degradation supports providing more resources with a lower chance to incur into user SLA violations.

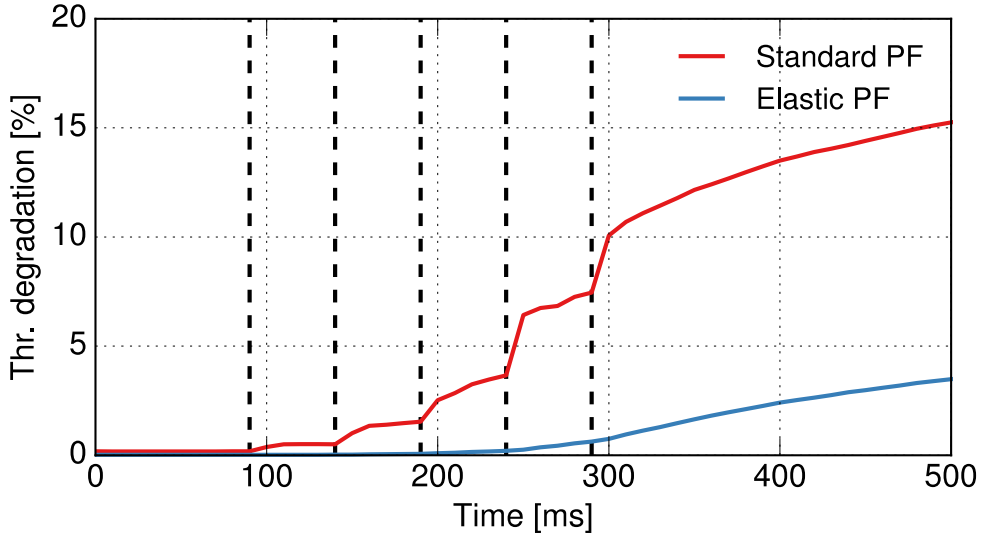


Figure 4-2: Graceful performance degradation with increasing load

4.1.2.3 Rank Control for MU-MIMO

MU-MIMO is a promising technique to increase the spectral efficiency in mobile communication systems. The principle idea is to schedule multiple UEs on the same radio resources. The spatial dimension is used to separate signals to or from co-scheduled UEs by beamforming, as depicted in Figure 4-3. Each beam corresponds to an antenna weight vector, also called precoder. For MU-MIMO no coordination among UEs is necessary, as the whole complexity is typically located in the e/gNB. Typical techniques to create UE specific beams are zero forcing (ZF) and/or maximum ratio transmission (MRT) [SSH04][BSH+14]. ZF is used to mitigate inter-user interference, while MRT is used to increase the signal strength of the serving link. Furthermore, MU-MIMO can be applied

complementary to single user (SU)-MIMO transmission modes and is already supported in LTE. Dependent on the number of users (further defined as a MU-group) and their individual number of antenna elements, with nearly orthogonal (spatially separated) precoding vectors a number of parallel data streams can be used for transmission. The spatial precoding vectors can be derived from the reported spatial covariance matrix in frequency division duplexing (FDD) or the measured sounding reference signal (SRS) transmission in time division duplexing (TDD). To minimise inter-user interference in a MU-group it is necessary to recalculate the precoding vector for each user of a MU-group (e.g., for signal to leakage ratio (SLR) or ZF-MU scheme). After that, a scheduling metric needs to be calculated (e.g. a MUPF metric) for each MU-group. Finally, each sub-band is allocated to the MU-group with the maximum MUPF metric under consideration of frequency selectivity. The basic calculation steps of a MU-MIMO scheduler are depicted in Figure 4-4. Dependent on the chosen MU-groups for each sub-band, a number of parallel data streams needs to be processed. For instance, in LTE Rel. 14 the maximum number of spatial streams supported was increased to 32, which requires additional computational resources to, in the first step find a sophisticated scheduling decision and last but not least process the total number of streams in parallel.

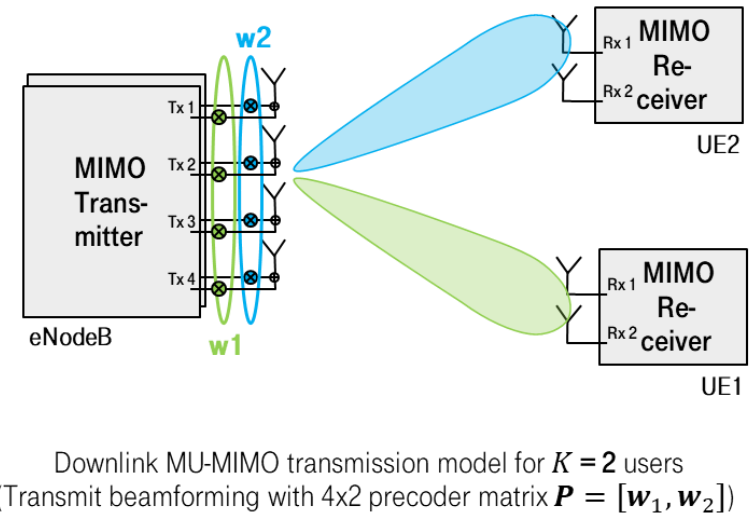


Figure 4-3: Principle of MU-MIMO transmission based on orthogonal precoding vectors

In a C-RAN scenario with shared computational resources, possible scheduling decisions, which maximise the traditional KPIs, such as throughput and delay, might overload the available computational power. Therefore, a new scheduling design of a MU-MIMO scheduler is required, which optimises the usage of computational resources besides satisfying traditional KPIs. Figure 4-5 shows in principle the extension of a MU-MIMO scheduler being aware of the available computational resources. The idea is to run a reference system level simulation and profiling the algorithm of the scheduler, which calculates how many ranks are used on the RBs for a specific user group (the simulation includes 3D channel model and supports LTE Rel.14). After the reference is profiled, the needed computational power can be derived which is needed to process this complex calculation. Then, performance results of different scheduling strategies can be compared, if not enough computational resources are available, e.g. only 80% of the computational power of the reference is available during the simulation.

Dependent on the strategy of the scheduler (e.g. limit the number of available ranks) to react on the limitation of the computational resources, the system performance should be least affected negatively but coincidently the system should be kept alive. Therefore, a novel extension of a MU-MIMO scheduling algorithm will be suggested, which also takes into account the available amount of computational resources in its scheduling decisions.

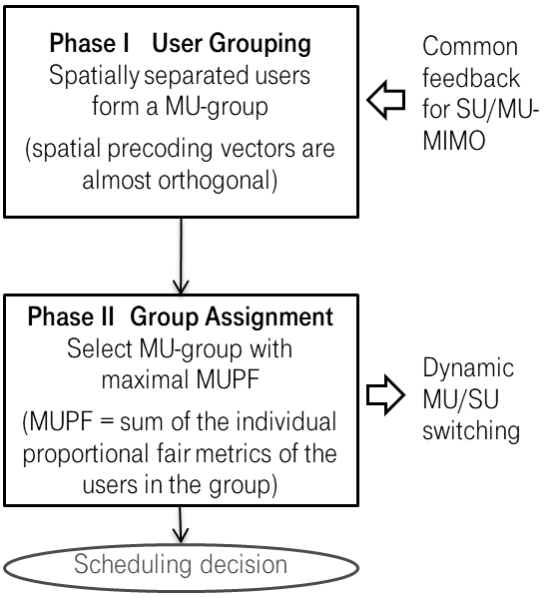


Figure 4-4: Principle calculation steps of a MU-MIMO scheduler

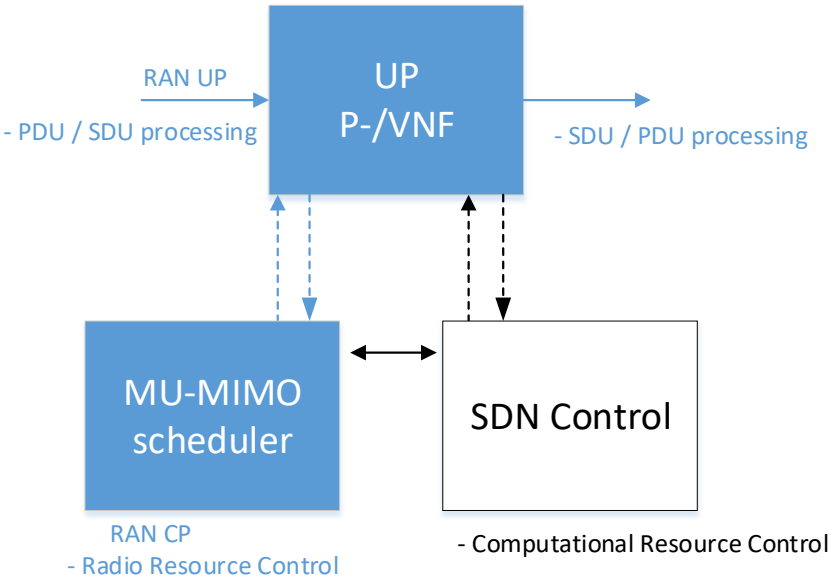


Figure 4-5: Principle logical extension of a computational resource aware MU-MIMO scheduler

4.1.2.4 A Model for Slice-aware Elastic Resource Management in RAN

This section describes the optimisation procedure to achieve the slice-aware elastic resource management with the focus on RAN. The goal is to develop an algorithm, which considers each network slice's requirements, SLA, and demands to allocate/deallocate the available (computational) resources to each of them. The required computational resources in this problem are dependent on many parameters, e.g., the traffic demands, the channel quality of terminals, and the network slice services. These parameters change over time, therefore the problem to optimise the allocation of resources is a probabilistic decision-making problem.

The shared VNFs (i.e., the VNFs that serve multiple slices) are the critical elements in the problem above. The allocation of resources to these VNFs has to be done considering all the sharing slices requirements. Some of VNFs, such as FFT/IFFT, require a constant amount of computational resources regardless of the input variables. The algorithm has to allocate the required computational capacity once. Since their computational resource requirements do not change at run-time, they can be excluded from the total elastic resource management.

Based on [FLG+17], the resource management in the cloud environment is the allocation of the m available physical machines to the n VNFs (also known as jobs). It is worth noting that a machine can host multiple VNFs.

$$\{M_1, M_2, \dots, M_m\} \rightarrow \{VNF_1, VNF_2, \dots, VNF_n\},$$

where M_i is the *i-th* available physical machine and VNF_i is a generic VNF.

The algorithm for slice-aware elastic resource management includes the following main steps:

Forming the available resource pool: In the first step, the algorithm has to identify the available physical resource and form the shared resource pool for the serving slices. Based on the slices' requirements and their SLAs, the algorithm allocates the available computational resources to each slice. This step is actually forming the state of nature in Bayesian Decision Theory.

Estimating the total computational capacity: in this step, the algorithm maps the total computational capacity to the slice requirements (e.g., NF processing time as the function of the input variables such as allocated PRBs in RAN). The algorithm uses the output of this step to decide whether to admit any new slice in addition to determining the service level each slice can receive.

Allocation of available computational resource to different slices: The algorithm allocates the required computational resource to the NFs of each slice ensuring the acceptable total processing time. The allocation procedure should consider SLAs type and slices' priorities. Based on [KC15], there are three main SLA types, as follows:

- Guaranteed slice, where the offered service quality is guaranteed to be kept in between the minimum and maximum level.
- Best-effort with a minimum guaranteed slice: the SLAs of these type of slices guarantees them receiving the minimum QoS. However, the higher QoS will be provided in a best-effort manner.
- Best-effort: the slice will be served in best effort manner and the guaranteed on the QoS is offered beside as soon as resources become available the slice will be served

While the SLAs define the constraints for resource allocation optimisation problems, defining serving weight for each NF of each of slices is necessary to enable prioritisation among them. The NFs with higher serving weight have higher priority in the resource allocation process. Consequently, the violation weights define the order of importance to violate the NFs requirements in the undesirable case where the shortage of resources happens.

Observe the network performance and re-allocate the computational resources: based on the changes of demands. The resource management algorithm observes the changes in the network performance as the results of the changes in the resource demands or resource availability and updates the resource allocation accordingly.

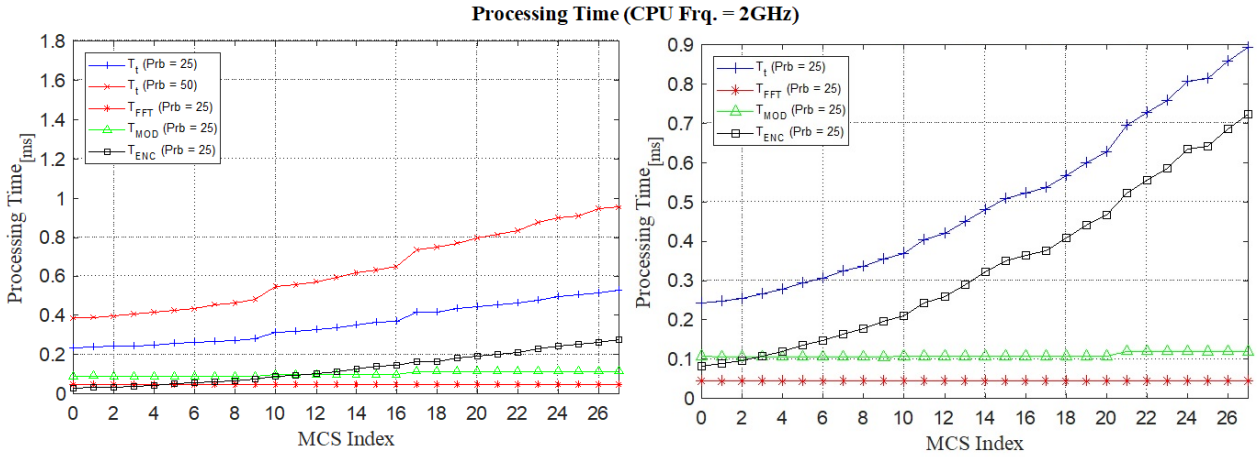
Estimation of Computational Capacity

The first step in the estimation of the computational capacity is the studying of the computational complexity of each of the VNFs in the RAN. The profiling of the processing time for different functions over different infrastructure (i.e., CPU frequency and the number of CPU cores) can provide an acceptable complexity estimation. This section addresses the details of profiling and the achieved results.

The performed profiling is based on OAI [OAI] emulator to measure the processing time of the LTE PHY layer given different load configurations, the number of PRBs and MCS. The used physical machine for profiling experimentation has Intel® Core™ i7-4790 CPU @ 3.60GHz, and 16 GB RAM. The machine was operated using Kubuntu 16.04 LTS with Kernel version 4.4.0-96 generic. Virtualisation was conducted using docker 17.06.1-ce. The CPU above has 4 physical cores and 8 logical threads. The processor also has 8 MB of Cache Memory and supports instruction set extensions SSE4.1/4.2, AVX 2.0.

Also, to measure the gain introduced by utilising AVX 2.0 instruction set a brief experimentation on a bare metal OS on an Intel® Core™ i7-2600 CPU @ 3.4 GHz (no AVX 2.0) compared to the primary testbed. Using AVX 2.0 instructions shows significant improvements in the individual functions and the overall processing time of the baseband unit. The results show utilisation of CPUs with AVX 2.0 can reduce the processing time into half. In the following, the profiling results are presented.

The first set of experiments was done on the bare metal OS to investigate the main affecting parameters on the RAN functions and the effect of using AVX 2.0 optimised instructions. To study the impact of channel quality on the required computational resources, MCS are swept from 0 to 27 and the required processing time per subframe is measured. Results for PDSCH in Figure 4-6 show that the processing time required for one subframe increases linearly with the choice of the MCS index. Figure 4-6 presents the processing time in DL and UL as the function of the MCS. It is apparent from the figure that increasing the MCS, the processing time increases.



a) Downlink ***b) Uplink***

Figure 4-6: Processing time as the function of MCS index

Furthermore, the figure shows the measured time for encoding, IFFT and modulation. It is apparent that the encoding time increases almost linearly with increment in MCS. However, the time taken by IFFT process remains constant throughout the experiment, since the IFFT block size is independent of the choice of the MCS index.

The processing time required by modulation can be divided into three sets. The time required by MCS indices 0 to 9 is nearly the same as QPSK modulation is performed for all of the cases. A similar explanation can be provided for MCS indices 10 to 16 which are modulated using 16QAM and MCS indices 17 to 27 are modulated using 64QAM.

It is worth noting that the processing time required for decoding is almost double than encoding. Hence, decoding has been the most critical process from profiling point of view. The required time by FFT is the same as IFFT in case of encoding.

Figure 4-7 shows the effect of running the PDSCH and PUSCH profiling on a container in comparison with Bare metal OS. The results show that at high CPU operation frequencies the overhead is negligible and the processing time of doing the operations in the containers is almost equivalent to that of Bare Metal. However, in lower CPU operation frequencies, in PDSCH in particular, the overhead effect starts to be evident.

Having a realistic estimation of required computational resources for RAN NFs is essential for provisioning the computational resource pool as well as dynamically allocating them to different network slices. The given input variables in the said estimation are network bandwidth (number PRBs), expected traffic demands, and CPU working frequencies. Based on the numeric results, there are three classes of RAN NFs:

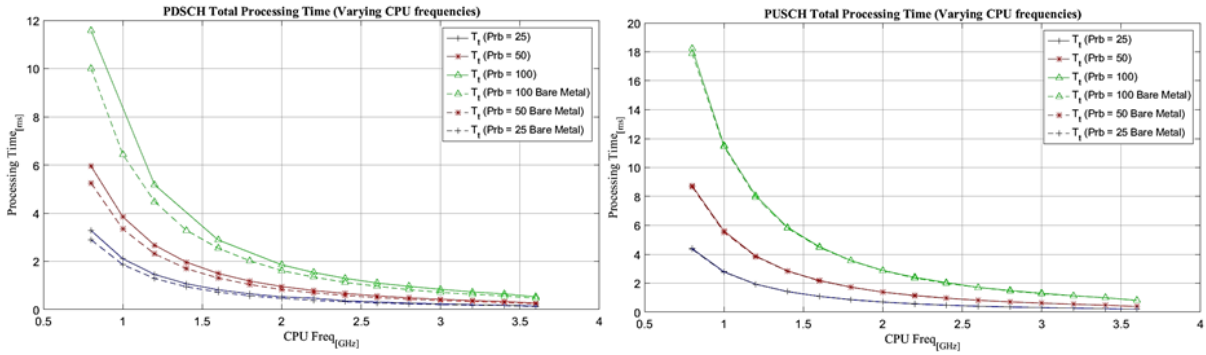


Figure 4-7: The total processing time as the function CPU frequency

- NFs with constant complexity: these NFs' (i.e., FFT/IFFT) complexity only depends on the network's configuration and bandwidth. The processing time for these NFs is independent of the network's load or the selected MCS. The required computational resources for these NFs do not change during run-time.
- NFs with complexity depending on the demand and channel quality. The processing time of NFs including (de)modulation and encoding/decoding functional block depends on the number of allocated PRBs and the MCS index. The traffic demands plus the effect of the link adaptation algorithms jointly estimate the required computational resources.
- NFs with complexity depending only on the throughputs such as the NFs in Layer 2. The required computational resources for these NFs are estimated just based on the demands. These NFs are going to be the next study topic.

In next step, the experimental results are used to fit a polynomial equation and estimate the processing time as the function of CPU frequency and MCS index. Examining the experimental readings suggests that they can be normalised to the number of PRBs. The only exceptions are the FFT/IFFT NFs. While analytical analyses show these blocks are logarithmic time complex, the linear approximation introduces negligible differences. The input variables described above are used to form the design matrix for regression and the lasso technique is used to fit a curve to the experimental results. The equation below presents the polynomial function of MCS index that can estimate the processing time in UL and DL.

$$\tau_{p_{[\mu s]}} = \frac{N_{PRB}}{f^2_{[GHz]}} \sum_{i=0}^{2} \alpha_i (i_{MCS})^i,$$

where:

- τ_p: pressing time,
- N_{PRB}: Number of PRBs
- f: the working frequency of CPU,
- i_{MCS}: the MCS index,
- α_i: polynomial coefficients,

Table 4-1 presents the coefficients' value in UL/DL for total, FFT/IFFT, modulation/demodulation, and encoding/decoding.

The above equation provides a closed-form approximation for processing time in C-RAN. The InPs can use this equation to determine the required number of CPUs and their working frequency for implementing each VNF in provisioning phase. The algorithms for elastic computational resource management can use it to estimate the effect computational resource changes and make decisions to cope with networks changes (either the changes in demand or MCS) while meeting the processing time requirements. During the network's run-time, based on the time measurements and observations and using machine learning techniques the approximation can be improved and optimised for the specific running situation (e.g., different implementation of NFs or CPU architecture).

Table 4-1: The coefficients for processing time

	α_0	α_1	α_2
T_t^{DL}	32.583	1.072	0.03
T_{FFT}^{DL}	7.609	0	0
T_{Enc}^{DL}	3.907	0.773	0.027
T_{mod}^{DL}	13.851	0.133	0.002
T_t^{UL}	35.545	1.623	0.086
T_{FFT}^{UL}	6.957	0	0
T_{Enc}^{UL}	10.512	1.631	0.083
T_{mod}^{UL}	17.494	-0.08	0.006

To this end, the following section describes the initial setup of a test bed, which will be used to run a computational resource control algorithm in a real environment, such as intended above. Therefore, an experimental testbed has been developed, in which the results of the experiments explained above are proved and additional measurements regarding Docker Containers and bandwidth part adaptation as a first example of an NR feature are analysed.

4.1.2.5 Computational Resource Control for a Virtualised Mobile Network with Docker Containers

The first specifications of 3GPP for the NR air interface for 5G and the corresponding RAN architecture have just been published in December 2017, incorporating concepts motivated by the virtualisation and cloudification paradigm. Special emphasis is thereby put on the functional split between distributed and centralised units (CU and DU, respectively) within logical base station entities [3GPP17-38300]. As investigated for example in [ABB+17], the CU/DU split has to be conducted under accurate consideration of processing times of individual functions within a base station.

An example implementation and extension of an interface between CU and DU based on OAI [OAI] and the corresponding FlexRAN implementation [3GPP17-38211] is studied in this section. It provides an extensible framework for SDN-based RAN concepts that is well in line with the proposed 5G-MoNArch architecture introducing a controller, such as XSC/ISC, described and illustrated in Section 3.2.1. The second part of the performance study addresses an initial evaluation of the impact of containerisation of OAI eNBs with Docker [Doc]. Finally, an outlook on the future work is presented.

Evaluation and Development Environment

The test setup that has been used for the performance study without Docker based containerisation is shown in Figure 4-8. It consists of three computers with 4 x 4 GHz cores for evolved packet core (EPC), FlexRAN Agent (FR-A), and FlexRAN Controller (FR-C), respectively. All computers run with Ubuntu 16.04 and Linux kernel version 4.10.0-28-lowlatency. The computers are interconnected via Gigabit Ethernet. The software defined radio (SDR) device is an NI USRP (Universal Software Radio Peripheral) 2944R [NI] which is connected via a ten Gigabit Ethernet connection to the computer that is running the FR-A with the embedded eNB implementation. The ten Gigabit Ethernet connection is required in order to guarantee sufficient stability for the transfer of I/Q samples between eNB and USRP in both UL and downlink (DL) direction. This directly reflects the challenges faced by fronthaul implementations for lower layer splits in C-RAN architectures as discussed in detail in [ATM18]. The testbed architectures used for the performance study with Docker is shown in Figure 4-9. In this case only one computer (Docker host) is used, and all elements of the network (eNB, MME, etc.) are running in individual Docker containers (marked as grey boxes in the figure).

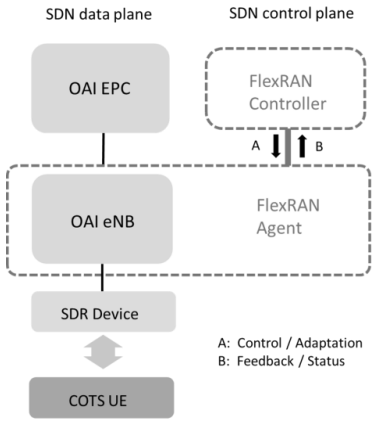


Figure 4-8: Experimental testbed based on FlexRAN

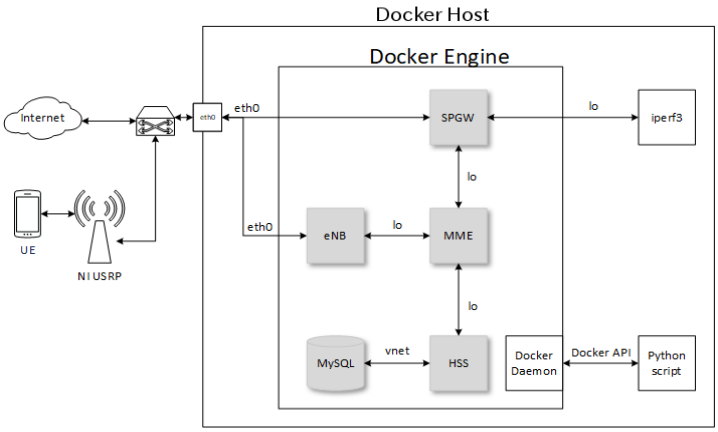


Figure 4-9: Experimental testbed for running OAI in Docker containers

OpenAirInterface

The OAI Software Alliance [OAI] provides a comprehensive development environment for software defined radio incorporating concepts such as SDN and NFV. The intention of the OAI project is the development of a fully real-time 5G protocol stack for running on commercial off-the-shelf (COTS) hardware by means of open source software. Throughout the study, the LTE implementation of OAI has been used.

The general setup of an OAI based system that has been used for the work presented in this report consists of an OAI EPC implementation, including MME, S-GW, and P-GW, HSS, and an eNB implementation.

FlexRAN

FlexRAN [FNK+16] is an open source project that provides a flexible and extensible SDN-based RAN platform as a reference platform for researchers and developers. For that purpose, the framework basically extends the OAI implementation with SDN functionality.

Following the SDN paradigm of separating CP and UP [JSS+14], the FlexRAN implementation introduces FR-As and a FR-C, where the latter represents the SDN controller

The communication between FR-C and FR-A forms the CP while the LTE data traffic flow within the OAI components (EPC, eNB) would represent the UP within the SDN context. The asynchronous communication between FR-C and FR-A is facilitated by a proprietary FlexRAN protocol which uses Google Protocol Buffer [PB] for message implementation and parsing.

In the context of the CU/DU split that is currently discussed at 3GPP as one of the essential enablers for meeting the stringent RAN requirements in terms of spectral efficiency, latency and energy efficiency, multiple optional splitting options are under discussion. These range from low layer splits between physical layer baseband processing (lower PHY) and RF functionality to higher layer splits between RRC and packet data convergence protocol (PDCP) [3GPP17-38300].

Docker

Docker is a platform to run and deploy applications using Linux containers. In comparison to VMs they share the kernel of a host operating system and can easily be distributed and replicated. This lightweight, portable and scalable approach, make this technology a reasonable choice for virtualizing RAN functionalities. Another advantage of container-based approach described in [GMG+17] is that containers introduce less additional delay compared to other virtualisation methods, such as VMs.

The testbed in Figure 4-10 shows the experimental environment of running the OAI EPC and the OAI eNB in a containerised environment. The grey boxes indicate here the functional entities running in separate Docker containers. In order to monitor CPU utilisation, run and change settings of Docker

containers, the Docker API and the Python software development kit (SDK) [SDK] have been used. Python scripts are used for monitoring the CPU utilisation of the Docker containers every second and saving the data for further evaluation.

Docker Compose has been used to prepare Docker containers for deployment. It is a tool for defining and running a multi-container application in terms of services by describing service features, such as virtual networks and connections between related services [DocO]. For each Docker container in Figure 4-10, a corresponding Docker file [DocB] that contains necessary installation instructions has been prepared. The typical Docker Compose build process is shown in Figure 4-10.

Currently all Docker containers are running on the same physical machine (Docker host), however Docker Compose can be used as well for Docker Swarm [Lov10] orchestration in order to distribute containers over different hosts and perform load-balancing

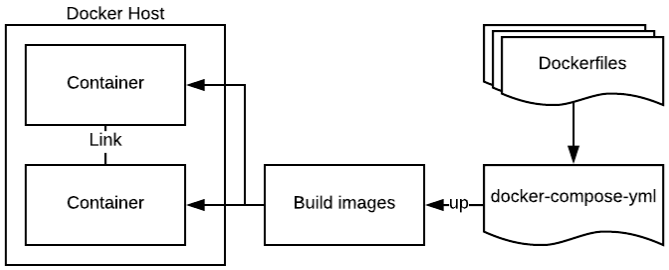
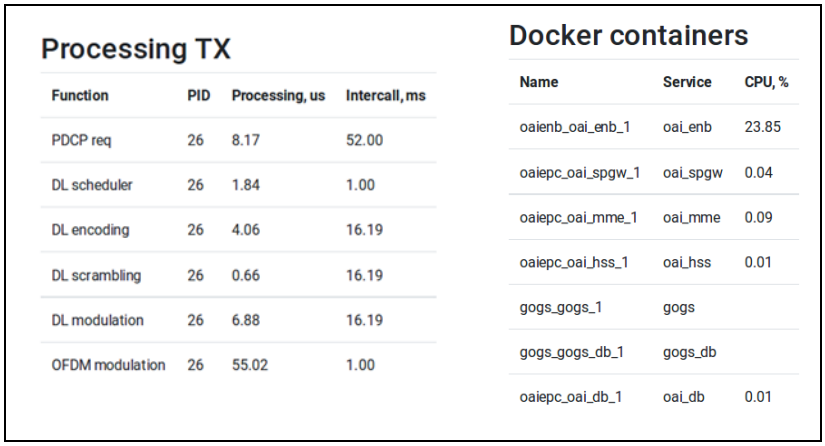


Figure 4-10: Docker Compose build process

Within the scope of this work, a process monitoring and control tool has been developed. The screenshots in Figure 4-11 and Figure 4-12 show an exemplary snapshot of real-time monitoring of processing duration and inter-call duration of different processes and functions within an OAI eNB. UE states and the CPU utilisation of individual Docker containers are visualised as well.

The inter-call duration evaluation indicates the duration between two successive calls of a specific function within the corresponding eNB process. The shown process ID (PID) furthermore indicates whether multi-threading is used for these functions. Different PIDs indicate different threads in the Linux kernel. As shown in Figure 4-11 and Figure 4-12, all investigated functions below the PDCP layer are running within the same thread, which means that they will run sequentially in a chain and the corresponding processing durations will add up.



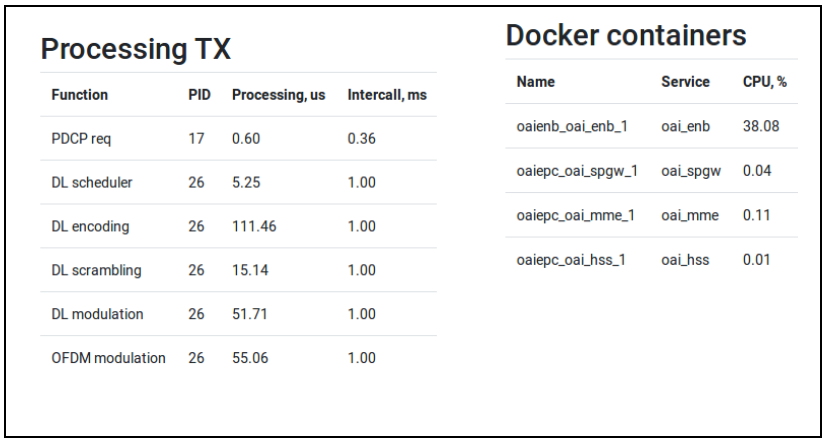
Processing TX

Function	PID	Processing, us	Intercall, ms
PDCP req	26	8.17	52.00
DL scheduler	26	1.84	1.00
DL encoding	26	4.06	16.19
DL scrambling	26	0.66	16.19
DL modulation	26	6.88	16.19
OFDM modulation	26	55.02	1.00

Docker containers

Name	Service	CPU, %
oaienb_oai_enb_1	oai_enb	23.85
oaiepc_oai_spgw_1	oai_spgw	0.04
oaiepc_oai_mme_1	oai_mme	0.09
oaiepc_oai_hss_1	oai_hss	0.01
gogs_gogs_1	gogs	
gogs_gogs_db_1	gogs_db	
oaiepc_oai_db_1	oai_db	0.01

Figure 4-11: Evaluation without traffic load



Processing TX

Function	PID	Processing, us	Intercall, ms
PDCP req	17	0.60	0.36
DL scheduler	26	5.25	1.00
DL encoding	26	111.46	1.00
DL scrambling	26	15.14	1.00
DL modulation	26	51.71	1.00
OFDM modulation	26	55.06	1.00

Docker containers

Name	Service	CPU, %
oaienb_oai_enb_1	oai_enb	38.08
oaiepc_oai_spgw_1	oai_spgw	0.04
oaiepc_oai_mme_1	oai_mme	0.11
oaiepc_oai_hss_1	oai_hss	0.01

Figure 4-12: Evaluation with one UE connected and iPerf3 downlink user datagram protocol (UDP) test

Communication Interface for Computational Elasticity

The performance evaluation with focus on computational elasticity has been conducted with extensions of the FlexRAN interface that have been developed within the scope of the 5G-MoNArch project. The communication between FR-C and FR-A has been extended in both directions in order to support adaptation of the resource scheduling by means of bandwidth restrictions and evaluation of computational resource utilisation in terms of processing time. Corresponding messages and handlers have been implemented in both FR-C and FR-A.

Bandwidth Part Adaptation

The possibility to configure and adapt bandwidth parts (BWPs) within the system bandwidth is an essential feature of the new NR air interface. According to [3GPP17-38211], a BWP is a set of contiguous PRBs selected from a contiguous subset of the common PRBs with a size that is lower than the total carrier bandwidth.

In order to implement flexible BWPs within the OAI framework for LTE, the interface between FR-A and FR-C has been extended so that the controller can configure the BWPs for the agent during runtime.

In order to comply with the resource block group (RBG) concept of LTE according to resource allocation type 0 in the specification [3GPP18-36213], the BWP configuration and adaptation has been done on RBG level. In case of 10 MHz system bandwidth which has been used for the following performance study, an RBG comprises three consecutive PRBs. It is important to keep in mind during the following performance study that the RBG determines the resource allocation granularity during DL scheduling while the BWP determines the set of consecutive RBGs that are used for DL scheduling. RBGs have so far not been specified for NR.

The BWP implementation used here is not the exact realisation of NR BWPs since the latter would for example contain additional reference and synchronisation signals that cannot be used in combination with LTE COTS UEs.

The FR-C is basically responsible for assigning BWPs consisting of consecutive RBGs to the FR-A which encompasses the OAI eNB implementation. This BWP allocation can be adapted anytime during runtime of the OAI eNB within the FR-A. The BWP concept implementation and the corresponding adaptation are both transparent for the DL resource scheduler. The latter just operates on a set of PRBs that are available for DL resource allocations within a transmission time interval (TTI). The BWP concept as applied in the performance study is in this sense from scheduler point of view therefore basically a PRB set restriction. The BWP concept in NR will furthermore allow the configuration of different OFDM subcarrier spacing for different BWPs.

Modulation and Coding Scheme Adaptation

In addition to the implemented BWP adaptation, it is furthermore possible to set the MCS per scheduled UE. The MCS levels are used according to the LTE specification given in [3GPP18-36213]. Table 4-2

shows the mapping of MCS levels to specific modulation schemes. Within a subset of MCS levels for a specific modulation scheme, the levels differ in the code rate of the channel encoder. 256QAM is not taken into account in the performance evaluations presented in this report.

Table 4-2: Modulation and coding schemes

MCS level	Modulation scheme
0 - 9	QPSK
10 - 16	16QAM
17 – 28	64QAM

Processing Time Reporting

The time required for performing different PHY functions within the eNB are reported periodically every TTI, corresponding to an LTE subframe of 1 ms duration, from the FR-A to the FR-C.

The processing time reporting comprises DL resource scheduling, transport block encoding, transport block scrambling, transport block modulation, and the IFFT processing of the OFDM transmitter in the DL transmission chain. The three functions are called sequentially per scheduled UE during the TTI construction.

The spatial precoding is not considered in the current performance study since a system configuration with a single TX antenna port is used, meaning that no spatial multiplexing or transmit diversity schemes will be applied. This corresponds to Transmission Mode 1 [3GPP18-36213] which does not introduce additional complexity in term of spatial processing in the transmission chain.

Performance Study without Docker

The performance measurements were done with different commercial UEs (Google Nexus, Samsung Galaxy S7, Samsung Galaxy Tab active) operating in LTE FDD Band 7 (2.66 GHz DL, 2.54 GHz UL). The system bandwidth has been set to 50 PRBs, corresponding to a nominal channel bandwidth of 10 MHz, and the BWP configuration has been varied between 50 PRBs and 6 PRBs. All UEs are scheduled within the same BWP.

The UEs perform periodic wideband channel quality indicator (CQI) reporting, and the link adaptation algorithm running in the eNB sets the DL MCS directly according the one suggested by the UE within the CQI report.

Since the maximum DL throughput performance is addressed in this study, the UEs have been positioned in a way that provides sufficient signal to interference plus noise ratio (SINR) levels (> 20 dB) for the highest MCS level (28) supported by all used UEs and the used OAI implementation.

Downlink Throughput Evaluation

The DL throughput performance has been evaluated depending on number of active UEs and number of PRBs available for resource allocation in a configured BWP. The throughput on the physical downlink shared channel (PDSCH) has been measured on TTI level using the Accuver tool XCAL-M [Acc] which provides direct access to the UE chipsets for comprehensive monitoring.

The traffic load has been generated by using the iPerf3 tool [iPer] between the S-GW and the UEs with an appropriate configuration for full buffer UDP traffic. The latter is required in order to achieve maximum throughput. Especially the use of transmission control protocol (TCP) during different tests has shown significantly lower throughput results due to the inherent traffic load reduction by means of TCP flow control.

The content of Table 4-3 provides the maximum achievable DL throughput (in Mbit/s) on the LTE PHY. The throughput values have been determined under the assumption of fair resource sharing between the UEs, meaning that every UE will get in average the same DL throughput, and the assumption that the transport block size (TBS) is always chosen according to the highest MCS (28) which corresponds to 64QAM assuming no support for 256QAM [DT10]. The actual channel code rate depends on multiple factors such as control region size and reference symbol mapping.

Table 4-3: Maximum theoretical downlink throughput (in Mbps)

		Number of PBRs				
		6	18	27	39	50
Number of UEs	1	4.4	13.5	19.8	29.3	36.7
	2	2.2	6.8	9.9	14.6	18.3
	3	1.5	4.5	6.6	9.8	12.2

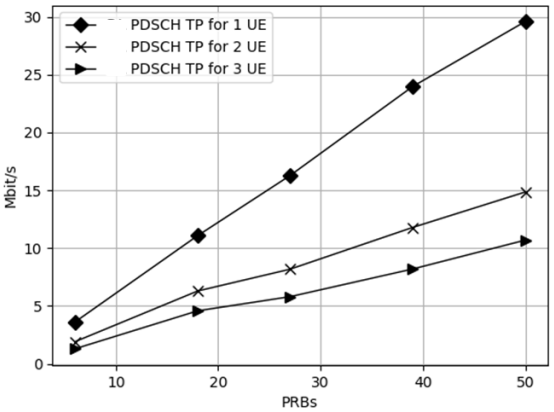


Figure 4-13: Downlink throughput per UE depending on bandwidth part configuration

The results of the DL throughput evaluation with OAI and COTS UEs are shown in Figure 4-13. The results confirm that the throughput grows linearly with the number of PRBs and that the resources are shared in fair fashion between all active UEs with DL traffic. The difference between theoretical throughput limit and the actual observed DL throughput corresponds to approximately 20% loss. The reason for this is given by the fact that the OAI DL resource scheduler that has been used does not schedule subframes for DL transmissions that carry primary synchronisation signal (PSS), secondary synchronisation signal (SSS) and physical broadcast channel (PBCH). Since these are transmitted in every fifth subframes, it yields a user throughput performance degradation of 20%.

Processing Time Evaluation

The processing time is evaluated on TTI level, meaning that for every TTI the time required for performing the DL resource scheduling is measured. Since the construction of a DL subframe has to be done every ms in LTE, the overall processing time should never exceed this limit. The processing time requirement will become even more demanding under consideration of fronthaul latencies between CU and DU as discussed in [ABB+17]. The NR air interface will furthermore support flexible TTI durations down to the scheduling of mini slot with 0.1 ms duration.

Figure 4-14 and Figure 4-15 show the processing time duration with two and three active UEs depending on the BWP configuration in terms of number of assigned PRBs, respectively. In addition to the average processing time per TTI, the figure shows the standard deviation of the processing time for encoding, scrambling and modulation. It can clearly be seen that the encoding entails to largest variance of required processing time while the scrambling shows negligible variance. It is for further study to which reason the variance of the considered functionalities behave different. All DL throughput performance evaluation tests were running for 20 s, corresponding to 20.000 TTIs of 1 ms duration in order to collect a sufficiently large number of processing time reporting samples.

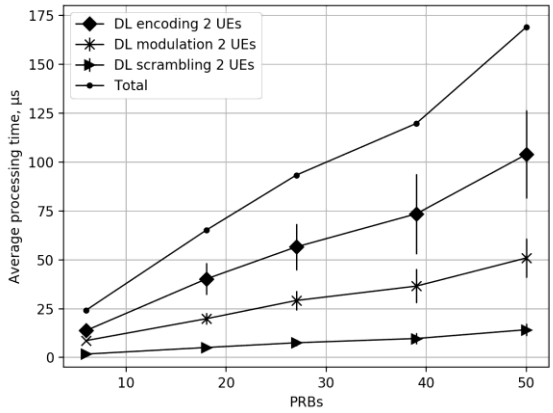


Figure 4-14: Processing time evaluation with two active UEs depending on BWP

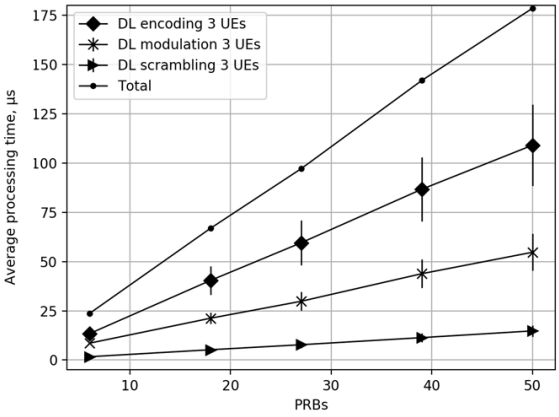


Figure 4-15: Processing time evaluation with three active UEs depending on BWP

Figure 4-16 and Figure 4-17 show the average processing time depending on the used MCS for one and two active UEs with DL traffic. The corresponding observed standard deviation of the processing time is shown as well.

Both figures clearly show that the DL encoding is the most time consuming of the considered functions in the PHY layer of the eNB for higher MCS levels (>13). The average processing time for the encoding ranges from 8 µs for MCS 0 to 115 µs for MCS 28 in case of one connected UE. In case of two UEs, a small increase of about 2-3 µs could be observed.

The processing time required for modulation and scrambling in case of one UE range from 38 µs to 54 µs and from 4.5 µs to 15.5 µs, respectively, depending on the used MCS. In case of two UEs, some slight increase of processing time could be observed as well, corresponding to the observations regarding the encoding.

The overall processing time comprising encoding, scrambling and modulation are in all cases below 200 µs and grows approximately linearly with the MCS level.

Very similar processing time values can be observed for MCS ranges 0-9, 10-16, and 17-28, respectively. This is based in the fact that the same modulation schemes (QPSK, 16QAM, or 64QAM) are used within these ranges, as shown in Table 4-2.

Performance Study with Docker

As a first step, it has been evaluated how the limitation of computational resources affects the performance of the OAI eNB running within a Docker container. That was done by generating DL TCP and UDP traffic with iPerf3 [iPer] for one UE. The time that expires until the eNB application starts transmitting and receiving I/Q samples was measured as well. Corresponding to the previous performance evaluation without Docker, the system bandwidth was set to 10 MHz.

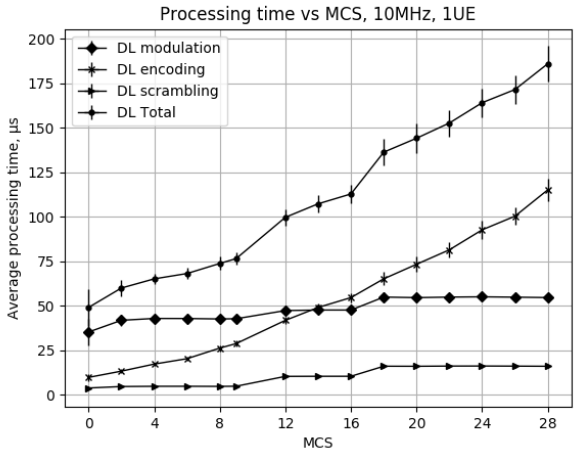


Figure 4-16: Processing time evaluation with two active UEs depending on MCS

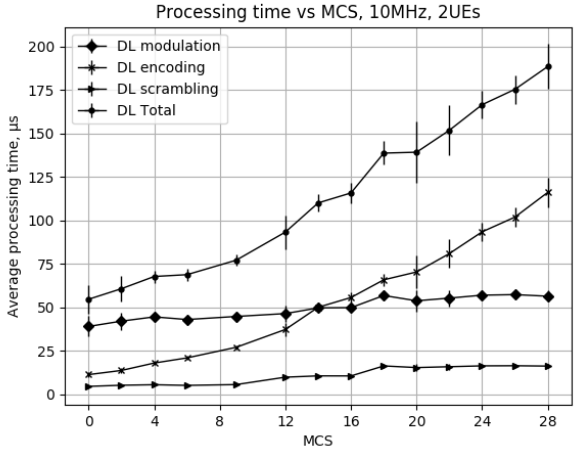


Figure 4-17: Processing time evaluation with three active UEs depending on MCS

In order to limit the computational resources for processes, Docker uses underlying cgroups technology [Man] by defining quotas that define the duration in µs that a process is allowed to run on a CPU within a defined time period.

In the set of conducted experiments we limited processes controlled by completely fair Linux scheduler (CFS) [Lov10], while the OAI eNB is a real-time process and scheduled by real-time round robin (RR) and first in - first out (FIFO) [Lov10] and therefore is not directly affected by applied limitations.

The initial reference evaluation of TCP and UDP throughput between a Docker container and a host operating system is shown in Figure 4-18. Here it has to be taken into account that the traffic does not go over a specific physical network interface. Within the Docker host, network interfaces are emulated by inter-process communication concepts, such as for example shared memory access. This can yield significantly higher throughput values than in case of physical network interfaces.

The results reveal that the container network performance significantly decreases with the reduction of available CPU resources. This plays a critical role because traffic (in terms of I/Q samples) to and from the SDR equipment also goes through the network interface using UDP socket connections. As the UDP throughput drops down, the eNB might become inoperable due to packet (and hence I/Q sample) loss or significantly delayed delivery of the latter. Further comprehensive investigations are required regarding this aspect.

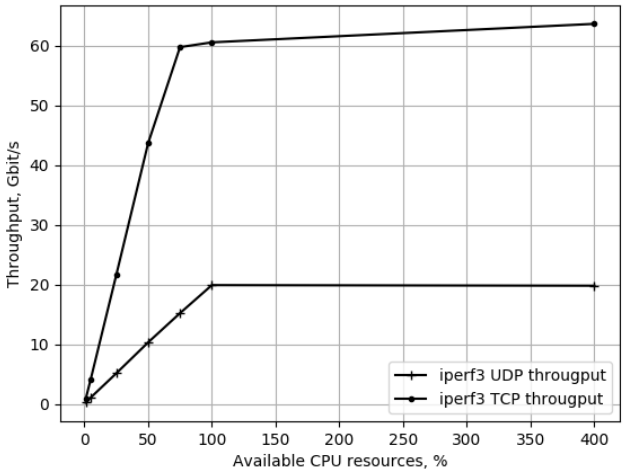


Figure 4-18: Docker container throughput evaluation vs. available CPU resources

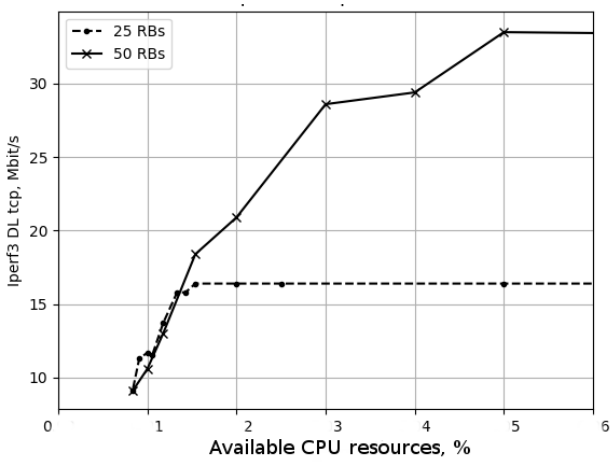


Figure 4-19: Downlink throughput depending on CPU resource limitation

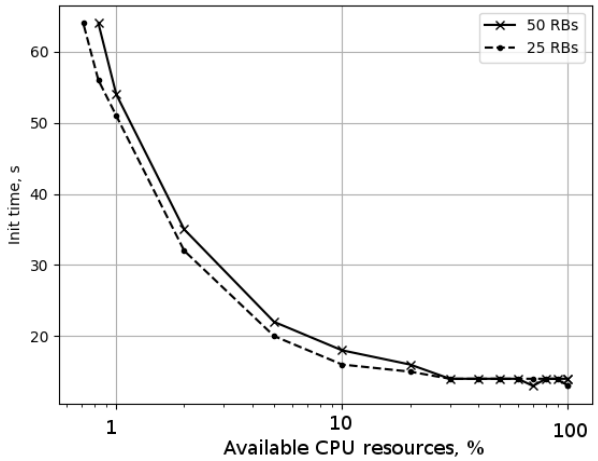


Figure 4-20: Initialisation time depending on CPU resource limitation

Figure 4-19 shows the DL TCP throughput results with one UE, OAI eNB and OAI EPC components running in Docker containers corresponding to the setup in Figure 4-9. It can be seen that the throughput starts to degrade when the maximum allowed CPU utilisation is restricted to less than 5%. In addition

to 10 MHz system bandwidth, 5 MHz system bandwidth has been evaluated as well. The system bandwidth settings correspond to 50 PRBs and 25 PRBs, respectively.

Figure 4-20 shows how the initialisation time of an OAI eNB within a Docker container is gradually increasing when the available CPU resources are reduced. These results have significant impact on further considerations of Docker usage since virtualised/containerised functional entities in the RAN should be available within a short amount of time after start-up.

Figure 4-21 and Figure 4-22 show an exemplary trace of the CPU utilisation of an OAI eNB running in a Docker container with maximum allowed CPU utilisation. In comparison with this, Figure 4-22 shows the CPU utilisation trace of the same OAI eNB running in a Docker container with a CPU utilisation limited to a maximum of 1 %. The red line in Figure 4-22 is the amount of time for which process has been throttled, which means that it has exhausted assigned quota and has not been allowed to use CPU resources until the next period. The traces consist of four phases:

1. Initialisation,
2. Start RX/TX,
3. iPerf3 DL,
4. iPerf3 UL.

The first phase comprises the initial start-up of the eNB. In the second phase, the eNB is active and sends synchronisation signals, reference signals and broadcast information. It is furthermore listening for UL transmissions. The latter two phases correspond to the eNB state during DL and UL speed tests. It can be seen in the traces that the eNB switches back to the second phase between DL and UL speed tests, and after the last speed test, respectively.

Overall, the results clearly show that the UL processing requires significantly more computational resources than the DL processing.

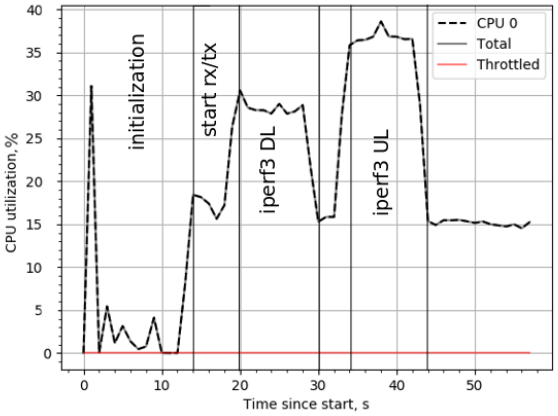


Figure 4-21: CPU utilisation trace without limitation

Future Work

The overall target is on the one hand to derive a design of an algorithm which controls radio and computational resources simultaneously and on the other hand find requirements for a definition of open interfaces among virtualised RAN functionalities. The initial performance results presented in this document address the processing time evaluation of DL eNB functions under consideration of eNB parameter adaptations such as bandwidth and modulation and coding scheme. The study has been conducted with and without containerisation based on Docker. Next steps will focus on additional evaluation of the UL processing in the eNB and additional configuration and adaptation possibilities, such as for example spatial precoding. Further extensions of the related interface between CU and DU are currently under development, as well as running the separated RAN protocol stack functionalities within an eNB in a multiple container environment. The first tests in this direction will focus on the separation of the link adaptation (MCS and spatial precoder selection) from the resource allocation.

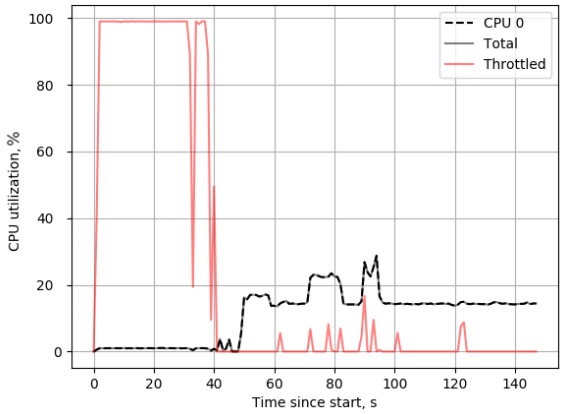


Figure 4-22: CPU utilisation with limitation of 1% quota per period

4.2 Orchestration-driven Elasticity

This set of innovations focuses on the ability to re-allocate VNFs within the heterogeneous cloud resources located both at the central and edge clouds, taking into account service requirements, the current network state, and implementing preventive measures to avoid bottlenecks. The algorithms that implement orchestration-driven elasticity need to cope with the local shortage of computational resources by moving some of the VNFs to other cloud servers which are momentarily lightly loaded. This is particularly relevant for the edge cloud, where computational resources are typically more limited than in the central cloud. In case of scarce resource availability, the execution of NFs with tight latency requirements should be guaranteed at the edge by offloading other elastic VNFs without such tight timescale constraints to central cloud servers. To efficiently implement such functionalities, special attention needs to be paid to:

- The trade-off between central and edge clouds and the impact of choosing one location for a given function.
- The coexistence of mobile edge computing (MEC) and RAN functions in the edge cloud.

This may imply scaling the edge cloud based on the available resources, clustering and joining resources from different locations, shifting the operating point of the network depending on the requirements, and/or adding or removing edge nodes [OCB16].

In addition, VNF scaling is a key functionality of the elastic orchestrator. Indeed, scaling mechanisms need to be investigated in horizontal and vertical dimensions, i.e., the ability to scale either the number of VMs or containers executing the functions (horizontal) or the resource capabilities of the allocated VMs or containers (vertical); furthermore, it is also important to study how these approaches impact NF performance.

4.2.1 State of the art

Orchestrating VNFs, i.e., moving the location of the orchestrated VNFs, requires a migration process of the virtual environment that hosts them. This introductory section gives a brief description of the different techniques that can be employed to perform VNF migration, purposely presenting the topic from a rather technical and computer-scientific point of view. Migration of virtual environments implies several technical challenges and requires different solutions depending on the application scenario.

In general, it has been assumed that the type of virtual environment performing the VNF hosting is a VM, and the problem of VM migration has been an active field of research for already a few years. A number of different techniques have been proposed in the literature to perform migration, each of which presents certain characteristics with implications on the orchestration result [CSR+15][AGH+15]. Most importantly, the literature makes the following classification:

- *Live vs non-live migration*, depending on whether the VM can keep uninterruptedly running throughout the complete migration process.
- *Pre-copy vs post-copy migration*, depending on how the VM keeps running (i.e., during a live migration) while the memory pages are copied to the destination VM; in pre-copy migration the pages are copied from source to destination while the VM keeps running on the source node, while post-copy migration suspends the VM at the source node and a minimum subset of the execution state of the VM is copied to the destination node.
- *Reactive vs proactive migration*, depending on whether the migration responds to changes in the system parameters (load, resource utilisations, SLA violations, etc.) or rather it anticipates them based on the history.

An additional note worth adding relates to the migration of containers. A container is a lightweight (compared to VM) virtualisation technique which is capable to run an application irrespective of the host environment. Key motivations behind consideration of container for RAN virtualisation are short instantiation time and much less overhead compared to the VMs [FFR+15]. Even though container-based virtualisation is a relatively newer virtualisation technology, several migration techniques exist, similar to those adapted to VMs. A comparison between pre-copy and post-copy methods is proposed below in tabular form [Pok]:

Table 4-4: Performance comparison between pre-copy and post-copy

Scenario	**Pre-copy**	**Post-copy**
Destination Node Failure	Can recover	Cannot recover
Downtime	Long	Short
Up state after migration	Fast	Slow
Write Intensive application	Bad	No difference
Read Intensive application	Good	No difference

Checkpoint and restoring mechanism is a technique used for live containers migration. Running containers are frozen and all the operating system related states and information is saved on the disk, known as checkpointing. After successful completion of checkpointing, a checkpointed file is transferred to a new machine and restarted. “CRIU” [Criu] and “P.Haul” [Haul] are the two open source tools capable enough to save low level system variables and save them to disk. Live migration can be successfully performed with CRIU and P.Haul with Docker and OpenVZ containers.

One of the limitations of CRIU migration is, it expects same library at destination machine with the same version as source machine. In case of any mismatch, running application may crash. This issue can be handled by migration of libraries together with application. For the applications dealing with peripheral hardware another limitation of CRIU is, plugins have to be developed in order to save Hardware State dealing with connected peripheral hardware [RedH]. Providers of containers have already developed frameworks for containers management that includes support for migration, but the technology is still less advanced and optimised than their VM counterparts.

Besides the topics of migration and containerisation, further research work exists on orchestration and optimisation of the exploitation of MEC resources. Orchestration of virtualised resources can be modelled as a problem of joint optimisation of edge and centralised cloud resources. Solutions to this problem can be inspired by or generalise techniques and analyses typical of the MEC scenario. For example, in [OCS+15], MEC communication and computation resources are jointly allocated for optimal results. Complementarily, in [OCB15], federation of computation resources physically situated in different places is proposed as a method for balancing the total computation load and improving the system performance. In analogous scenarios, federation and clustering of computation caching resources is studied in [OC17]. Strategies closely related to the works mentioned so far can be applied to the problem of orchestration and migration of network slices and VNFs to enable system elasticity. Moreover, following the approach of [OCD+14], the opportunity of relocating VNFs from the edge of the network to the centralised cloud can be evaluated in terms of the trade-offs between the advantages of relocation and the energy or delay backhauling costs: indeed, according to the specific case, relocation

may be feasible in terms of availability of resources at the central cloud, but also be inconvenient in terms of migration costs, which can be measured, e.g., in terms of total latency or energy consumption.

4.2.2 Mechanisms for Orchestration-driven Elasticity

In this section, we briefly introduce and discuss a few different approaches to model orchestration-driven elasticity and enable it in 5G networks. In particular, Section 4.2.2.1 introduces the idea of "slice blueprints", which will appear again in Section 4.3.2.4; the resource requests specified by slice blueprints are handled by an *orchestration algorithm* at the NSMF (see Figure 3-2 and Figure 3-4), whose main task is to manage the resource allocation and to enable orchestration-driven elasticity by relocating VNFs whenever necessary. As we describe in Chapter 2, elasticity can also be obtained by a smart management of VMs and containers scaling when more computational capacity is required. Therefore, a fundamental aspect to introduce computational elasticity to VNFs in general is, how to scale VNFs in an appropriate manner. Within Section 4.2.2.2 we present one possible solution for VM scaling based on a ML approach. Finally, VNF relocation also involves the migration of the virtual environments that host the running VNFs; a brief description of the environments on which we focus our attention is provided in Section 4.2.2.3.

4.2.2.1 Proactive Resource Allocation and Relocation

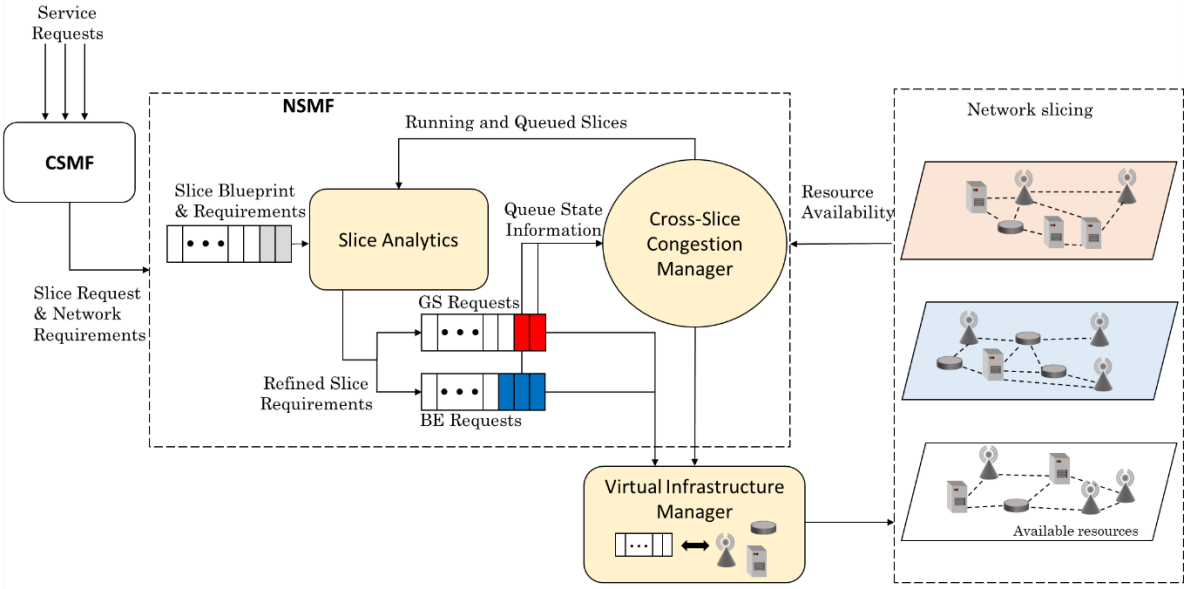


Figure 4-23: From service requests to resource allocation and network slicing

In the 5G-MoNArch framework, the Communication Service Management Function (CSMF) represents an intermediary function between the Service Layer and the M&O Layer [5GM17-D2.1]. It transforms consumer-facing service descriptions into resource-facing service descriptions, which are in turn mapped to network slice "blueprints" or "templates", in the NGMN or 3GPP terminologies, respectively. Another example of these descriptions are the ETSI NFV Network Service Descriptors [ETSI16-NFV]. From a very general point of view, in our model, we are interested in two components of the network slice blueprints:

- A list of the NFs that each slice request requires to be run.
- A list of inputs for each NF being part of the service, characterising the parameters of their specific instances.

First, the system takes the requests and translates them into network slice blueprints, which contain all requirements needed by the slices to be run. These requirements are "readable" by the CSCM, one of the functionalities of 5G-MoNArch's NSMF devoted to the slice lifecycle management. This functionality, based on the blueprint requirements, distributes each request to available pools of

resources, trying to maximise the set of slices that can be handled at the same time, given some performance constraints (e.g., on latency). This procedure needs to be carefully managed, in order to optimise the resource exploitation and to avoid bottlenecks or service refusals.

Figure 4-23 represents a logical diagram showing how service requests are taken into account by the network and transformed into running slices. At the M&O Layer, the algorithm that handles the service requests is made of three main logical blocks, represented in yellow in Figure 4-23.

They are:

- The *Slice Analytics*: it is part of the AI & big data analytics defined in Section 3.3. It receives the slice blueprints and requirements defined by the NSMF from the slice requests and the network requirements. It is also aware of running and queued slices. Its main role is to enable slice-aware elasticity (see Section 4.3) and evaluate the amount of resources to be allotted to each NF. Specifically, it classifies the NFs between elastic and non-elastic and processes the service requests, also focusing on evaluating resource request correlation.
- The *CSCM:* it is a global resource orchestrator, aware of both running and queued slices. Its role is to control the lifecycle of network slices, to monitor the buffer of slice requirements, and to manage the resource allocation and availability. The orchestrator de facto enables all levels of elasticity, by sending resource allocation instructions to the NFV MANO. These instructions concern the slice admission, the NF deployment, and their scaling.
- The *VIM*: it receives the allocation instructions from the NSMF and executes them. Its job is to concretely handle the dynamic resource orchestration and the allocation (or re-allocation) of virtual resources to satisfy the service requests.

To face the challenge of orchestration-driven elasticity, we will focus on the role of the CSCM. From a high-level perspective, the latter can be viewed as the application of four main mechanisms, related either to the re-organisation of a given pool of resources or to the relocation of NFs from and towards different areas of the network:

1. Clustering of virtual resources.
2. Sparsification of virtual resources, intended as the reduction of assigned resources to the execution of already-running NFs, possibly gracefully reducing their performance, exploiting trade-offs among performance constraints; freed resources can be used for the execution of other NFs.
3. Re-allocation of NFs within nodes of the same edge cloud.
4. Re-direction/migration of NFs towards different locations of the network (from central cloud to edge cloud and vice versa).

Notice that, at the present moment, we are considering NFs as “atomic”: we are making the assumption that a whole NF is running at the same place, even if this is done by a few different virtual components; if needed, all these components are relocated together.

The previous points can be seen as complementary parts of a more general paradigm. Based on different use cases and different states of the network, the four previous action points are alternately chosen and applied by the CSCM. Our work is motivated by the search for a mathematical formalisation of this scenario and aims at designing algorithms that implement one or more of the previous action points.

Our goal is to design an orchestration algorithm that considers as inputs:

- The resources requirements or slice blueprints, which contain the list of NFs to be run for the related services and the corresponding inputs.
- The constraints associated with the blueprints (e.g., on latency, on computational elasticity).
- The state of the network resources (e.g., amount of free resources, estimated running time of busy resources, state of buffers/queues).

The aim of the orchestration algorithm is to dynamically optimise the allocation of resources to satisfy the service requests and, at the same time, to avoid the creation of bottlenecks that may force the system to refuse future incoming requests. The main idea is to design a system that, whenever some local resources begin to run low, is capable of migrating requests with high elasticity to further available

nodes of the network, freeing occupied resources for the treatment of requests with tighter constraints. Nonetheless, before migrating NFs to far locations, the possibility of redistributing the work load within the same "local" resources need to be considered.

The main challenge that we will face is the design of a system which applies orchestration-driven elasticity in a proactive (and not only reactive) manner. Control conditions need to be identified that entail activation of the re-allocation or migration of NFs to other resources before criticalities happen (resource congestions, "traffic jams", drastic performance reductions, service denials). Ideally, the system will exploit statistics on the past workload and resource occupation level to predict future states of the network. Reinforcement Learning strategies for orchestration management need to be investigated, based on the strategy that the CSCM is "rewarded" if it manages to avoid bottlenecks, to serve all the incoming requests, and to maintain an acceptable QoS, while respecting – or, whenever possible, gracefully relaxing – the constraints imposed by the slice blueprints.

In the following, we will give a preliminary formulation of the problem from a mathematical point of view. Further details and results will be made available in future deliverables. In our architecture, in contrast to the more classical centralised radio access network, a distributed and heterogeneous cloud infrastructure is considered. This choice enables to move NFs between central and distributed cloud units to allow load management, performance optimisation, and adapt the system configuration to the momentary load of the transport network. Let us denote as $\mathcal{N} = \{1,2, \dots, N\}$ the set of NFs that compose the 5G network. Each request by a NF $n \in \mathcal{N}$ is characterised by specific computational requirements x_n in terms of number of operations to be processed.

Then, we indicate as $\mathcal{I} = \{e, c\}$ the set of available (edge and central) cloud units. Each cloud unit $i \in \mathcal{I}$ is identified by its computational capacity C_i, which represents the maximum number of computation operations per second that it can support.

In the current technology, NFs are characterised by strict inter-dependencies, i.e., signalling and data need to go through neighbouring NFs within specific timing constraints, in order to guarantee reliable network operations. The most stringent constraint exists at the lowest level (the physical layer) of the access network protocol stack, i.e., between the first NF that can be centralised and the functions located at the access point. To consider this, we define an additional NF, denoted NF 0, which includes those functions that cannot be virtualised.

In the considered architecture, NFs are executed in a serial flow; the latency experienced by this flow includes both the computational delay l^C and the communication delay l^T of each NF. The former depends on the computational requirement of a NF and on the amount of computational resources that it receives. The latter is experienced when the two communicating NFs are not located in the same cloud unit and the flow needs to be transferred through a communication link. To model the communication delay, we consider a transport network technology with rate R^T [bps/Km] connecting the edge and the central cloud nodes, and the cloud units with the access network. In addition, we consider that the output data w_n [bits] generated by the NF n, which has to be transferred to its neighbouring NFs, linearly depends on the computational requirements x_n, i.e., $w_n = \alpha x_n$, where α [bits per operation] is a fixed system parameter.

Here, we characterise and analyse the problem of optimizing the deployment of a set of NFs over the cloud units jointly with the associated resource allocation. To strike a balance between the resources used at the central and edge cloud, we consider a convex price function $P_i(x)$ for all $i \in \mathcal{I}$ characterised by a two-part tariff [MV95]:

$$P_i(x) = c_i^d x^2 - g_i x + c_i^c,$$

where $g_i x$ describes the gain due to the usage of the computational resources x, $c_i^d x^2$ denotes the cost for deploying x computational resources, and c_i^c represents the cloud service connection cost.

Since the central cloud can be located in a suburban or rural area, its connection costs are larger than those for the edge cloud, $c_c^c > c_e^c$; in contrast, we assume that the resource deployment costs are larger for the edge cloud, $c_e^d > c_c^d$; however, the gain provided by the usage of computational resources at the edge cloud is larger than the one provided by the central cloud due to the reduced communication latency, $g_e > g_c$.

According to this model, the joint NF deployment and resource allocation problem can be formulated as follows:

$$\min_{\mathbf{C},\mathbf{a}} \quad \sum_{i\in\mathcal{I}} P_i\left(\sum_{n\in\mathcal{N}} C_{i,n}\right)$$

$$\begin{aligned} \text{subject to:} \quad & l_n^C + l_{n,n+1}^T \le l_{n,n+1}^* \quad \forall\, n \in \mathcal{N}\setminus\{N\} \\ & l_n^C + l_{n,n-1}^T \le l_{n,n-1}^* \quad \forall\, n \in \mathcal{N}\setminus\{1\} \\ & l_1^C + l_{1,0}^T \le l_{1,0}^* \\ & a_n \in \{0,1\} \qquad\qquad \forall\, n \in \mathcal{N}, \end{aligned}$$

where $C_{i,n}$ denotes the amount of computational resources allocated by the node i to the NF n and a_n is a binary deployment variable with $a_n = 1$ if the NF n is executed at the edge cloud and $a_n = 0$ otherwise. Moreover, the first three constraints describe the latency constraints between a NF n and its neighbouring NFs $(n-1, n+1)$ with computational delay

$$l_n^C = \frac{a_n x_n}{C_{e,n}} + \frac{(1-a_n)x_n}{C_{c,n}} \quad \forall\, n \in \mathcal{N},$$

and delays related to the communication links

$$\begin{aligned} l_{n,n-1}^T &= \frac{w_n d_{e,c}}{R^T}\left(a_n(1-a_{n-1}) + a_{n-1}(1-a_n)\right) \quad \forall\, n \in \mathcal{N}\setminus\{1\} \\ l_{n,n+1}^T &= \frac{w_n d_{e,c}}{R^T}\left(a_n(1-a_{n+1}) + a_{n+1}(1-a_n)\right) \quad \forall\, n \in \mathcal{N}\setminus\{N\} \\ l_{1,0}^T &= \frac{w_1}{R^T}(a_1 d_e + (1-a_1)d_c); \end{aligned}$$

$d_{e,c}$, d_e, and d_c denote respectively the distances [Km] between the edge and cloud nodes and the between the edge and central clouds and the access network.

Solving the mixed-integer problem described above is very challenging for three main reasons: first of all, the binary variable a_n makes it combinatorial; second, in the definition of l_n^C there is a coupling between the two optimisation variables, which makes the problem also non-convex; finally, there are additional couplings between a_n and a_{n-1} in the definition of $l_{n,n-1}^T$ and between a_n and a_{n+1} in the definition of $l_{n,n+1}^T$.

At this stage, we have identified two possible approaches to solve the joint NF deployment and computational resource allocation problem: the first approach targets to find a near-optimal solution with a low complexity by relaxing the integer variable and reformulating the minimisation problem into a convex optimisation problem; the second approach is based on matching theory [GSB+15], a mathematical framework that provides tractable solutions for combinatorial problems related to the matching of players belonging two in two distinct sets. Solving this problem and analysing the proposed solution is left as a future work.

4.2.2.2 Machine Learning for NF Scaling and Orchestration

Key enablers to provide computational elasticity are the scaling mechanisms that modify the amount of computational resources that are allocated to a NF. They should be able to help exploiting the elasticity of the system if they are properly designed. Coarsely described, there are two significant ways to scale a NF: (i) *horizontal scaling*, where the system is *scaled up/down* by adding or removing new virtual or physical nodes to execute a NF, and (ii) *vertical scaling*, where the system is *scaled out/in* by increasing or decreasing the allocated resources to the existing virtual or physical nodes [Wal12]. In both cases, when dealing with VNFs, the virtual instance hosting the NF might be a VM, as in the most traditional architectures, or a container, the faster and lightweight version of a VM [HB16]. Furthermore, virtualisation also implies that horizontal scaling can be easily achieved by replicating the virtual environment, whereas vertical scaling, which usually means modifying the allocated amount of CPU, RAM, or storage, could be achieved either by *resizing* the environment (changing assigned resources at runtime, possibly without even rebooting) or by *replacing* the environment (i.e., changing less powerful servers with more powerful ones) [CSR+15].

Based on the above definitions, we describe the objective of this contribution in terms of the differentiation of the two scaling approaches' performances when dealing with NFs, i.e., the need to provide insight to the key question of which NF scaling approach is preferred for each NF. As it turns out, the answer to this question does not only depend on the NF itself but also on the location of the NF (e.g., in the central or edge clouds, or at the load conditions of the specific server where it is being executed), a parameter that can be modified at least in the case of many VNFs via appropriate orchestration mechanisms. This could be achieved by migrating the virtual instance executing the NF from one server/cloud to another, and the benefits may include availability of resources and lower NF delays, load balancing, and power savings via server consolidation. Hence the relationship of this contribution with the orchestration-driven dimension of elasticity and with Section 4.2.2.3, where live migration of VMs are discussed. In the ETSI NFV MANO reference architecture [ETSI14-NFV], the tasks of NF scaling are handled by the VNFM, so any novel AI-enhanced scaling algorithm should be placed in such a module (or one with equivalent tasks). The beneficiaries of this innovation would be the end users who would get proper QoE even under very high load scenarios, and operators who would make a more efficient use of the system.

However, the investigated scaling mechanisms in this contribution are not constrained to VNFs since it is well known that NFs at the RAN may not be fully virtualisable or not virtualisable at all. Nevertheless, the most appropriate scaling mechanism may still depend on the type of NF and the network operation conditions. As an example of scaling decisions in the RAN, a need to attain higher system throughput can be achieved via different scaling mechanisms, e.g., by adding additional base stations, which is considered horizontal scaling and may be preferable if the current base station is overloaded, or by introducing an increase in the operating bandwidth of the system, which is considered a vertical scaling mechanism and may be adopted when there is bandwidth available in the system.

As described in Section 4.1.1, cloud computing scaling solutions have been developed in the past to provide elasticity when managing data centre resources [PHL+14]. ML techniques, and in particular, RL, provides (i) a promising model-free framework for learning the optimal scaling policies depending on the context, (ii) the ability to reason under uncertainty based only on environmental observations, and (iii) the adaptability to suit the environment based on its own experience. Other model-based methods (e.g. threshold-based) may need to change the model when load request model or VNF change, which makes its operation much more complex. The RL approach, instead, will adapt to suit the environment based on its own experience.

An RL formulation contains an intelligent agent that automatically learns from an environment. The agent interacts with the environment by applying an action and learning from the reward (positive or negative) awarded by the environment. At each time interval, the environment provides the state to the agent. The agent applies an action and receives a reward, and the environment moves to a new state. Examples of actions could be increasing/decreasing the number of (virtual or physical) nodes, increasing/decreasing the resources assigned to a node, or migrating the virtual environment to another physical server. Furthermore, additional factors to capture in the formulation should be (i) the NF time scales, since the time involved in providing new resources (VM or container, CPUs/RAM/disk) might not be fast enough for a NF to provide the required output, and (ii) the location of the NF, as resources might be limited at the edge cloud, but execution delays would be shorter due to the edge proximity to the RAN. Finally, as the agent chooses the action based on a policy, the objective of the agent is to learn the optimal policy to achieve maximum reward in the long run [UPS+05].

In the absence of a complete environmental model, the model-free reinforcement learning algorithm Q-learning [SB98] can be used to generate optimal policies, as it has the capability of being able to make predictions incrementally and in an online fashion. In general, a Q-learning framework requires a function $Q(s, a)$ denoting the expected return from taking scaling action a in state s. For each interval, the Q values are updated using the Q-learning technique, where a simple value iteration updated is performed: It assumes the old value and makes a correction based on the new information. A learning factor α is usually considered to balance the trade-off exploration vs exploitation, and β is a reward discount factor of rewards seen in previous intervals, as shown in the following algorithm:

1. Initialisation:

2. *Q(s,a)* ← *0*
3. Initialise *s* (current state)
4. *r(s,a):* reward function
5. For each interval, do
6. *a* ← choose action using policy derived from *Q*
7. Take action a and observe *r(s,a)* and *s'*
8. $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r(s,a) + \beta \max_{a'} Q(s',a') - Q(s,a)\right]$

In previous works using RL for scaling of cloud applications, a Q-learning solution approach has been developed for allocating resources to applications in the cloud. For example, [DKM+11] developed an automated controller capable of adding and removing VMs based on a variable workload model. Due to a large state space in the problem and long VM provisioning times, a number of heuristics are applied to bootstrap Q-learning and speed up learning [BHD13]. As learning is performed from the actions taken due to heuristics, the number of explored actions increases and learning becomes more predominant.

RL Problem Formulation

The problem of scaling and moving (i.e., orchestrating) the VNFs hosted by the corresponding VMs is defined as follows. The reward function is based on individual VM's performance and aggregated over the set allocated to each VNF, which is in turn calculated assuming a performance profile of the elastic VNF is available. The state spaces are the configurations of such set of VMs allocated to each VNF. The actions are the possible configuration modifications that can be carried out, namely horizontal and/or vertical scaling, or moving the VNFs to a different location. With this in mind, an RL formulation is proposed as follows.

*The reward function***:** The long-term cumulative *reward* is the optimisation target of the RL formulation. The immediate rewards are the summarised VNFs performance feedbacks on the resulted new configuration of their allocated VMs. The throughput of an individual VNF is measured via a *score* assuming an elastic behaviour, i.e., a certain shortage of resources is tolerated provided the amount of allocated resources R_a (i.e., CPU) is greater than the minimum footprint R_{mf} defined in Section 2.1.3. R_N is the amount of resources needed for achieving the reference VNF performance (throughput). The reference throughput (*ref_thrpt*) values are the maximum achievable VNF performance under SLA constraints. The reference for one application can be obtained by dedicating the physical host and giving more than enough resources to the corresponding VM(s). α_e is a parameter between 0 and 1 that represents the level of elasticity of a function (i.e., how well it can keep a graceful degradation), and it is estimated by profiling the VNF. A low score indicates lack of resources, which should be avoided in making allocation decisions. As suggested by virtualisation benchmarks for summarised performance, the reward calculation uses weights ω_i for each VNF. We refuse the configurations of negative scores (i.e., violation of SLA) by assigning a reward of −1. Furthermore, we capture possible penalties when response time (*resp*) based SLAs are violated.

$$reward = \begin{cases} \sqrt[n]{\prod_{i=1}^{n} \omega_i \cdot score_i} & if \ \forall\, score_i > 0 \\ -1 & otherwise \end{cases}$$

$$score = \begin{cases} ref_{thrpt} - penalty, & R_a \geq R_N \\ ref\_thrpt \dfrac{(1-\alpha_e)R_a}{R_n - R_{mf}} - penalty, & R_N > R_a \geq R_{mf} \\ 0, & R_a < R_{mf} \end{cases}$$

$$penalty = \begin{cases} 0, & resp \leq SLA \\ \dfrac{resp}{SLA}, & resp > SLA \end{cases}$$

The state space: The state space is defined to be the set of possible VNF configurations, which should be fully observable and hence simplify the RL problem. Each VNF is allocated a set of VMs, i.e., $VNF_i = \{VM_1^i, \dots, VM_n^i\}$, where each VM is characterised by the amount of CPU that it has been allocated, i.e., $VM_x^i = \{cpu_x^i, cloud_n^i\}$. Extensions of this formulation would include other resources such as the amount of memory or storage, or the scheduler credit.

The actions: For each VNF, the configuration of its allocated VMs can be modified with different actions. Basically two main options exist, namely scaling the VMs or moving the same. If the first option is chosen, the scaling can be performed horizontally or vertically. We quantify the possible increments of both CPU and allocated VMs but allowing an increase or decrease of one unit at a time. The second option allows the VNF to be moved from the central to the edge cloud and vice versa. Below there is a summary of all possible actions for this RL problem:

- Scaling
- Horizontal scaling
- Increase allocation of VM/container to VNF by one unit
- Decrease allocation of VM/container to VNF by one unit
- Vertical scaling
- Increase amount of CPU allocated to VNF
- Decrease amount of CPU allocated to VNF
- Moving
- Move a VNF from central to edge clouds
- Move a VNF from edge to central clouds
- None

The above problem formulation will be refined, implemented and evaluated during the remainder of the project, with results expected to be obtained in the next deliverable.

4.2.2.3 Orchestration through Live Migration of Network Functions

In order to satisfy the service performance requirements, certain VNFs will have to be re-orchestrated and migrated to edge network depending on the nature and availability of the resources and network conditions. This will improve the performance by decreasing the demand of resources in the core network and decrease the overall latency of the service.

The placement of VNF is a complex issue. The orchestrator should be aware of the network state and the infrastructure as well as taking into account the machine states. The algorithm should take several aspects into account, like resource consumption, number of hops, and bandwidth consumption. NFs for which the latency is the most critical requirement, for example EPC routing components should be the ones that are moved to the edge cloud and closer to the user.

The challenge is to decide which procedure is most suitable for a particular case where several options for re-orchestration can be considered: horizontal scaling – adding and removing of VNF instances, effectively increasing the resources of the slice, vertical scaling allocation or removal of the resources such as computing or storage resources to/from a VNF instance and live migration of VNF instances.

The functionality of moving the VNFs from central cloud to edge cloud is not new and has been investigated in several studies. However, each of the proposed solutions has its limitations and cannot satisfy all the requirements of the diverse VNF that compose the 5G system. For example, 5G NORMA [5GN17-D6.2] implemented live migration of VNFs to specific compute node based on OpenStack live migration functionality. The dynamic reallocation of the VNFs was achieved and it was demonstrated that the relocation without service interruption between the edge and the central cloud is possible.

However, in order to get acceptable booting times and performance it was necessary to introduce some limitations in the architecture, including "shared storage live-migration" strategy or the use of the same hardware architecture in source and destination. With these limitations the performance that was achieved was acceptable and the booting time was in range of a few ms. One of the drawbacks of this approach is that this solution imposes strict limitations in the architecture and in network topology, and reduces network security and reliability [5GN17-D6.2].

In this work, we present an orchestration framework that accounts both for the horizontal and vertical scaling options for a NF as well as for the live migration option for VMs. In particular, the RL-based algorithms of Section 4.1.2.4 will be expanded to include VM (i.e., the hosted NF) migration as an additional orchestration option to exploit the elasticity of the system.

In addition, another important fact that should be taken into consideration is that the delays were slightly higher during the migration of the VM. Depending on the service requirements and the role of the migrated function in the network, this increased latency may impact the overall functionality of the network.

Taking into account all the limitations and drawbacks of the solution described in [5GN17-D6.2], we propose to investigate other options in order to substitute the VMs whose size is big – indicatively, tens of GBs. One of the obvious solutions is to reduce the size of the VMs that we want to migrate, since this seems to be the biggest factor that limits the scalability. The idea is to minimise the VMs image size and memory footprint. To achieve this, we can take into account three options: containers, unikernels and lightweight VMs.

Containers are getting more and more popular in cloud implementations because they are lighter than traditional VMs. By eliminating the guest OS, they optimise the resources and can be implemented fast. Among the drawbacks of containers, we could include issues with the security and isolation, since container share the same kernel [MLS+17].

Unikernels, on the other hand, are tiny VMs based on library operating systems designed for single purpose applications [MMR+13]. This is a new approach that could be of big interest in 5G networks.

Some of the advantages of unikernels are:

- High security: compared to containers, the unikernels have small amount of code so there is no attack surface.
- Small image size and memory footprint: comparing to VM images that occupy tens of GBs, unikernels can occupy around 5MB depending on the functionality.
- High level of optimisation.
- Boot time in the range of ms.
- Cost reduction by minimalizing the usage of the resources allowing high density.

Among the drawbacks, we can include the fact that unikernels are limited to single applications and linking them to construct more complex functionalities might require time [MLS+17]. That is why another option can be considered, that is, to create lightweight VMs (tens of MB) based on an optimised Linux kernel. [MMR+13] compares booting times of three different web server implementations: unikernel based on MirageOS, minimal Linux kernel and Debian Linux running Apache2. All of them were running on XEN hypervisor and the best performance was achieved for unikernels.

In this section, we presented three approaches considering live migration. Taking into account the characteristics of unikernels described above, it would be interesting to investigate the possibilities of implementing some of the NFs, especially those that are latency sensitive, using this technology. Due to the short booting time, they are suitable for re-orchestration in the edge cloud. However, this is an emerging technology and the orchestration mechanisms should be adapted however to support unikernel solutions. On the other hand, containers are an interesting approach and this technology is more mature than the one involving unikernels. Containers could be suitable for the re-orchestration mechanisms we want to demonstrate in orchestration-driven elasticity functionality. For this purpose, the 5G-MoNArch architecture is being updated with modules that will allow the deployment and management of containers, as will be shown in future deliverables.

4.3 Slice-aware Elasticity

A key elastic requirement is the ability to serve multiple slices over the same physical resources while optimising the allocation of computational resources to each slice based on its requirements and demands. Offering slice-aware elastic resource management facilitates the reduction of CAPEX and OPEX by exploiting statistical multiplexing gains. Indeed, due to load fluctuations that characterise each slice, the same set of physical resources can be used to simultaneously serve multiple slices, as Figure 4-24 illustrates.

Adaptive mechanisms that exploit multiplexing across different slices must be designed, aiming at satisfying the slice resource demands while reducing the amount of resources required. Hence, the solutions must necessarily dynamically share computational and communications resources among slices whenever needed. These algorithms in general have to obtain an estimation of the available resource pool in first step and define series of policies or strategies for allocating these resources to different slices on-demand based on the slices requirements and their priorities. In the following sections, different resource management techniques based on analytical modelling, game theory, or machine learning are briefly described. Furthermore, an elastic admission control system would be also required, as elastic slices need not have the same amount of available resources as e.g., a highly resilient slice where all resource demands must be fully satisfied at each point in time.

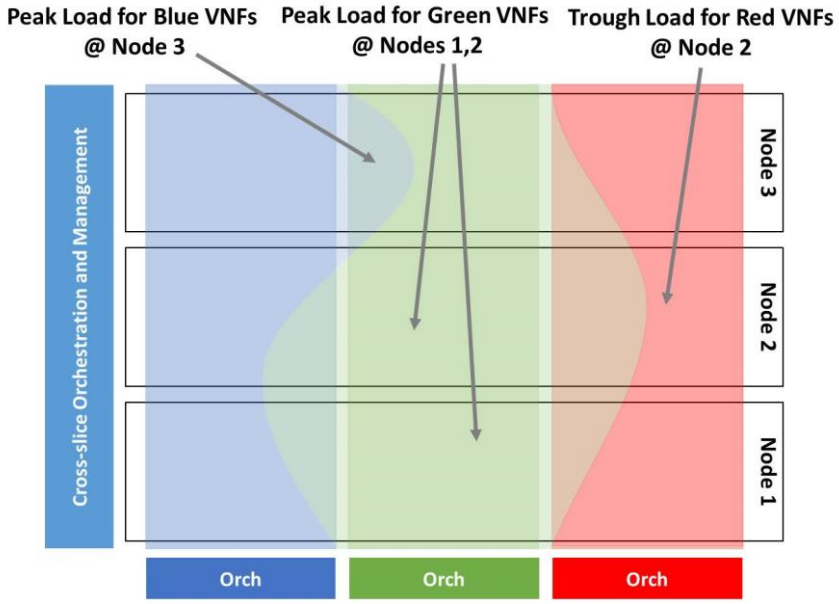


Figure 4-24: Illustration of slice-aware elasticity

4.3.1 State of the Art

This section provides an overview on different resource orchestration scheme that can be used as tools to achieve slice-aware elasticity. More specifically, we focus on both single objective and more complex multi-objective optimisation solutions. Finally, we will focus on ML schemes that can be powerful tool to classify and estimate resource requirements and improve the results of the resource allocation methods.

The paragraphs that follow present resource orchestration methods that allocate resources to the different network components and network slices based on three approaches, namely, 1) *Application-based methods:* Application-aware resource orchestration methods of elastic and inelastic adaptive real-time traffic. 2) *Game theory-based methods*: Economic and pricing models addressing resource management in the 5G wireless networks, and 3) *Optimisation-based methods:* Methods in which the resource allocation problem is formulated as an optimisation problem.

Application-based methods: Erpek et al. [EAC14] propose the use of logarithmic and sigmoidal utility functions in order to represent the different applications that are running on the mobile devices on the

different slices. The goal is to allocate the resources to each device in a way that the network objective (i.e., the product of the individual mobile device utilities), is maximised. Abdelhadi et al. [AC14] propose a combination of content-aware, time-aware and location-aware resource allocation methods for improved results. Shajaiah et al. [SAC14] propose a method that assigns a utility function that represents the application type running on each mobile device. Each mobile device assigns a weight to each of its applications based on their usage percentage. The final goal is the optimal allocation of resources to the devices and their applications based on a utility proportional fairness policy.

Game theory-based methods: Wang et al. [WMC+17] propose a method for resource allocation in Mobile Cloud Computing (MCC) environments over 5G architectures. Specifically, the authors propose a method that maximises the utility function of MCC networks based on a multi-leader multi-follower two-stage Stackelberg game model. Singh et al. [SKT14] propose a distributed spectrum sharing method based on multiple one-shot repeated games between two operators. In this approach, the operators are free to decide whether they share spectrum or not, and exchange favours asked and received.

Optimisation-based methods: Niu et al. [NZS+16] utilise a mobility prediction method for traffic demand prediction. Based on this method, the authors propose a multi-timescale resource sharing mechanism, consisting of multiple resource allocation processes, namely, a global resource allocation process and multiple local resource allocation processes that are deployed at different time scales. The resulting problem is solved by an increment-based greedy allocation algorithm. Ha et al. [HL16] proposed the use of non-linear integer programming for solving an optimisation problem in order to determine the transmission rates and the numbers of quantisation data bits for all users. The goal is the maximisation of the total transmission rate. Wang et al. [WNW+16] propose a beamforming scheme in order to coordinate multiple remote radio heads in C-RAN, with the goal of improving the QoE of users. The QoE of each user if modelled as sigmoidal function and the problem is formulated as a non-convex optimisation problem that is solved using an iterative algorithm.

The majority of the aforementioned methods that deal with slice-aware resource allocation in mobile networks optimise only with respect to a single objective. This does not always correspond to the desired result, since there usually exist multiple conflicting objectives that must be simultaneously optimised (minimised or maximised), i.e., energy utilisation versus latency. These types of problems are solved using multi-objective optimisation techniques.

One simple approach to deal with multiple conflicting objectives is the use of scalarisation, i.e., a combination of the objectives into a single objective. This single objective is afterwards minimised using traditional single-objective optimisation methods. The most commonly used scalarisation methods are the weighted sum and the constraint method [Ehr05]. The main problem of the scalarisation methods is that they involve the introduction of preferences for each objective, e.g. the value of the weights in the weighted sum. However, the automatic selection of specific preferences for each objective is not a trivial task and manual intervention is often introduced to determine them.

To overcome this limitation, multiple approaches have been considered in the literature, which do not include any preferences at all, but instead they calculate a set of multiple solutions, each one of them corresponding to different trade-offs between the multiple objectives [Ehr05]. This set of solutions is called the Pareto set. The Pareto set is optimal in the sense that no other solution can be found that surpasses any solution of the Pareto set in all objectives simultaneously. In order to calculate this set, a commonly used method employed is the genetic algorithm [CLV07][ZLB04]. The determination of fitness function of the genetic process is performed using weighted sum of the objectives with varying weights [HL92], alternating between the objective minimisation [ZLB04], and using dominance relations [ZLT01].

In order to optimise the resource utilisation efficiency, the network has to properly estimate the resources to be allocated to a given slice such that the SLA is satisfied. To achieve such goal, we foresee that the slice blueprint analytics is able to identify correlation between the different requests and to scale the resource requirement accordingly, by means of inter-slice resource elasticity. Note that making this analysis across all the slice requests may be computationally infeasible; therefore, identify and cluster the slice requests is a potential enabler to simplify this task. However, this approach will potentially lead to sub-optimal solution when compared to a scheme that processes all the slices in the network. For this

reason, the choice of a good cluster remains a major challenge, and requires optimising the trade-off between complexity and performance.

Note that this step goes beyond the simple mapping of a service to the operator slice template, e.g., eMBB or massive machine-type communication (mMTC), and requires to analyse the NFs, which compose each request.

ML algorithms can be a proper enabler for achieving this goal by offering higher scalability, faster response time due to offline training, and convergence reliability. Among the popular ML algorithms, we mainly note neural networks, k-means, and fuzzy logic controllers. In the context of the 5G, Bendriss et al. have applied ANNs for SLA enforcement of networking services involving VNFs [BYC+17]. In addition, Persico et al. have proposed a resource scaling solution for public clouds that adapts allocated resources to workload dynamics by leveraging fuzzy logic [PGP+17].

After this first step, a second challenge concerning this proposed inter-slice elasticity scheme will be to precisely evaluate the similarity index between the slices in a cluster, by identifying the common NFs and potentially whether two slices have the same inputs for the same NF. In the latter case, the same virtual resource can be allocated for a common NF to serve multiple slices. More specifically, the proposed similarity index allows scaling down resource assignment and thus increasing resource utilisation. The correlation based similarity presented by Hassanzadeh et al. in the context of the coded caching is a potential solution to evaluate the similarity index across multiple slices [HTL+16]. Another solution may combine both distance based Gaussian similarity and cosine similarity between the NFs inputs [EBS17].

4.3.2 Mechanisms for Slice-aware Elasticity

This section presents the methods proposed in the context of 5G-MoNArch for slice-aware resource orchestration, RAN configuration, and slice admission control. There are many methods proposed for the solution of each of these problems, each one approaching the problem with a different perspective resulting in different advantages/disadvantages and performance/accuracy trade-offs (experimental results are expected on subsequent deliverables). Under this scope, the proposed approaches are complementary, and the selection of specific methods depends on the specific scenario.

Concerning slice-aware resource orchestration, Section 4.3.2.1 presents a multi-objective optimisation approach, which outputs several solutions that abide by specific constraints, and allows the network operator to select one of these solutions based on its needs interactively. On the other hand, Section 4.3.2.2 proposes a method for resource orchestration without network operator intervention. It proposes an algorithmic approximation method. Section 4.3.2.3 presents a method for automatic RAN parameter configuration based on service constraints concerning throughput, reliability, and latency. Section 4.3.2.4 proposes a method to address the high dimensionality of resource orchestration directly, by clustering the slices into groups of common NFs, which can be handled as a single slice by the resource allocation algorithms. Section 4.3.2.5 focuses on the use of predictive analytics to enable the resource orchestration algorithms to respond faster to changes based on these predictions. Section 4.3.2.6 proposes the use of game-theoretic concepts for resource orchestration which allows tenants to customise their allocations. Furthermore, it presents a method for slice admission control using deep learning.

4.3.2.1 Multi-objective Optimisation for Slice-aware Resource Orchestration

Problem statement

The majority of the existing methods for resource allocation in mobile networks provide a single solution by using traffic and demand related information to compute KPIs of interest. For the determination of the solution, a trade-off between the various KPIs needs to be selected, corresponding to the importance of each KPI with respect to the specific problem, and in most cases only a single KPI is optimised for. Even if the trade-off between KPIs is selected accurately, the fact that the result is a single solution means that some information from the original KPIs is lost. For the task of resource allocation, this information could be useful for the MNO, in order to have a better understanding and perception of the traffic condition.

Figure 4-25 shows a schematic description of the different behaviour of the traffic of a four-slice network, with and without slice-aware elasticity enabled. Available resources limits for each slice, e.g. available CPU, are depicted with a red dashed line. In the case of the non-elastic network shown in Figure 4-25(a), once traffic reaches the limit of available resources allocated to the service slice, further requests cannot be served (red areas). However, in an elastic network, shown in Figure 4-25(b), if resources are available from other slices, they are allocated between different slices according to an optimisation process resulting to all requests being served.

Since in the resource allocation problem different KPIs are generally conflicting, multi-objective optimisation methods can be used to deal with them. The effectiveness of multi-objective optimisation in fields such as economics, engineering and multi-modal analysis [KDT14], suggests that its use for the problem of multi-objective resource allocation in mobile networks is promising. An approach using multi-objective optimisation is presented in the rest of the section.

Approach

The first step of the proposed approach is the identification of the available physical resources and the formulation of the shared resource pool that will be used for serving the slices. The available pool P includes both network (e.g. bandwidth) and computational (e.g. CPU, memory) resources, located on either the edge or the central cloud:

$$P=\{r_1,r_2,\ldots,r_{|P|}\}$$

As noted earlier, this formulation takes into account also the location of the resources, and thus, also considers the placement of the slices to either the edge or the central cloud resources. Each slice corresponds to a service offered by the network, e.g., virtual reality (VR) or mobile browsing. Users request these services and the mechanism for handling these requests is described in Section 4.2.2.1.

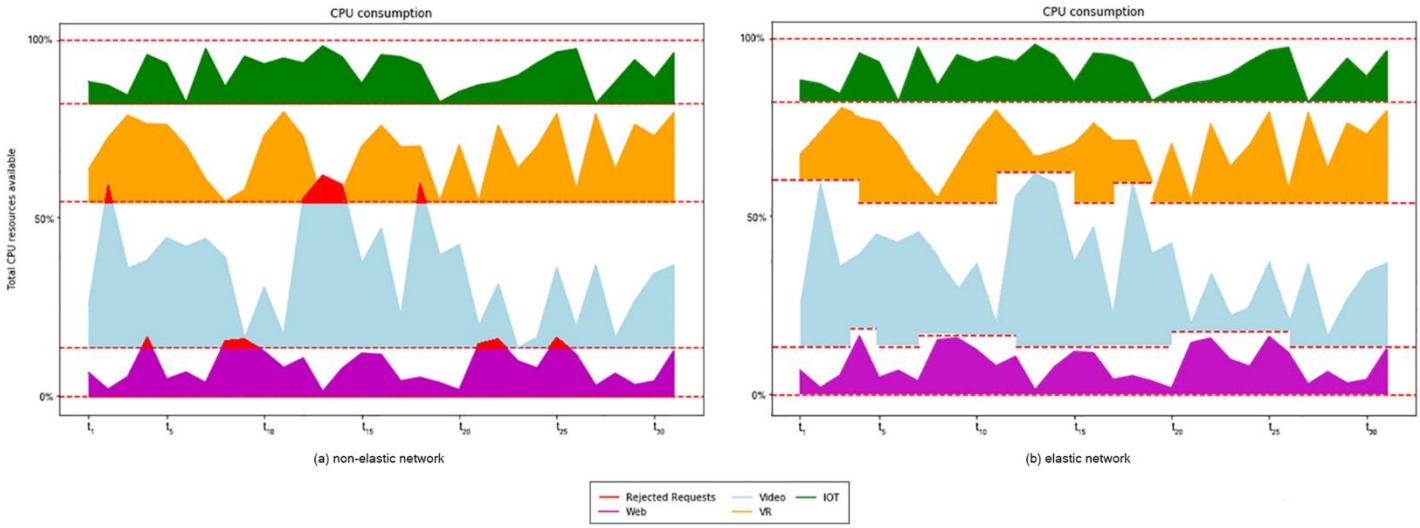


Figure 4-25: Schematic description of the behaviour of (a) a non-elastic and (b) an elastic network in a scenario where the legacy network cannot serve all requests, since slices need more than allocated resources. Dashed red lines show available resource limits and red areas show service degradation or rejection cases.

Afterwards, the total demand per slice is defined as follows. Given the set of slices $S = s_1,..,s_i$, the set of time periods $T = t_1,..,t_{|T|}$, and the set of user service requests $U_{qs_i}=\{u_{1s_i},u_{2s_i},...,u_{|U|s_i}\}$, define as $D_{s_i}^{t_k}$ the vector that represents the list of requests that wait in the execution queue of slice s_i at time t_k. The order of the requests in this vector represents their priority in the queue:

$$D_{s_i}^{t_k}=[u_m,\ldots,u_n]$$

For each slice, the combined total of requests, requires an amount of resources in order to be served $E_{s_i}=\{e_{1s_i},e_{2s_i},...,e_{|E|s_i}\}$ with E being a subset of the available resources set P, $E \subseteq P$. Not all requests can be accommodated, so for each request $u_j \in U_{qs_i}$ per slice $s \in S_i$, function x_{u_j,S_I} is defined, where:

$$x_{u_j,S_i}=\begin{cases}1,\text{if request } u_j \text{ for slice } S_i \text{ can be accommodated,}\\0,\text{if not}\end{cases}$$

The next step includes defining of the objective metric for the optimisation procedure. The resource allocation problem is formulated as a minimisation problem, with respect to minimising a list of KPIs of interest, e.g. latency, energy consumption etc. Since the resource allocation procedure can optimise for either a single or multiple KPIs each time, the set of minimisation objectives is defined as:

$$J=\{J_1,..J_{|J|}\}\equiv\{KPI_1,...,KPI_{|J|}\}$$

where each J_i represents a specific KPI, i.e. KPI_i.

The desired result of the algorithm is the definition of a mapping function $f: P\times D_{s_i}^{t_k}\rightarrow P(P)$, for all $t_k\in T, s_i\in S$, and $\mathcal{P}(P)$ is the powerset of P. In other words, given the demand $D_{s_i}^{t_k}$ and the available resources P, function f returns a list of resources allocated to slice s_j at time t_i, while also minimising the list of objectives J. In this respect, the actual values of the KPIs utilised in set J depends on the selection of the mapping function f, i.e. J is a function of f: J(f). Finally, the algorithm allocates the required computational resources to each slice ensuring the acceptable performance metric as follows:

$$\min_f J(f) \text{ subject to } KPI_J^{min}\leq KPI_J\leq KPI_J^{max}$$

KPI_J^{min} *and* KPI_J^{max} in this equation, represent the minimum / maximum values of a KPI and they depend on the scenario examined and/or SLAs, e.g., augmented reality (AR) speed > 2 Mbps, error rate < 10^{-6} etc. The proposed minimisation procedure observes the network performance J(f) and re-allocates the computational resources based on the changes of demands $D_{s_j}^{t_k}$. It should be noted that the optimisation problem defined in the previous equation does not constrain the method to formulate the function f. More specifically, the resource allocation function f can be either algorithmically defined (e.g. Resource Constrained Project Scheduling problem), defined using a model (e.g. probabilistic model, neural network etc.), or simply be rule-based, i.e. a sequence of if-then statements. As mentioned earlier, the KPI-based constraints in the previous equation do not only depend on the scenario, but also the SLAs. As mentioned later in Section 4.3.2.3 there are three types of SLAs: 1) guaranteed, 2) best-effort with a minimum guaranteed, and 3) best-effort. The first two SLAs provide additional constraints to the last optimisation equation, namely, 1) $KPI_I^{min}(s_i)\leq KPI_I(s_i)\leq KPI_I^{max}(s_i)$ and 2) $KPI_I(s_i)\leq KPI_I^{max}(s_i)$ respectively for a slice s_i.

The problem formulation described, can be further clarified by the example that follows. Specifically, let the required resources and cost per slice of a network be:

$$\text{CPU requirements per slice } s_i : e_{1S_i}(u_{qs_i})=\sum_1^j \text{CPU required} * x_{u_j,s_i}$$

$$\text{Memory requirements per slice } s_i : e_{2S_i}(u_{qs_i})=\sum_1^j \text{Memory required} * x_{u_j,s_i}$$

$$\text{Bandwidth requirements per slice } s_i : e_{3S_i}(u_{qs_i})=\sum_1^j \text{Bandwidth required} * x_{u_j,s_i}$$

$$\text{Power requirements per slice } s_i : e_{4S_i}(u_{qs_i})=\sum_1^j \text{Power required} * x_{u_j,s_i}$$

$$\text{Total cost of resources required per slice } s_i : C_{S_i}(u_{qs_i})=\sum_1^j \text{Cost of required resources} * x_{u_j,s_i}$$

while consumption for all slices:

$$\text{CPU consumption is } e_{1_{tot}}=e_{1_{S_1}}+e_{1_{S_2}}+...+e_{1_{S_n}},$$
$$\text{Memory consumption is } e_{2_{tot}}=e_{2_{S_1}}+e_{2_{S_2}}+...+e_{2_{S_n}},$$
$$\text{Bandwidth consumption } e_{3_{tot}}=e_{3_{S_1}}+e_{3_{S_2}}+...+e_{3_{S_n}},$$
$$\text{Power consumption } e_{4_{tot}}=e_{4_{S_1}}+e_{4_{S_2}}+...+e_{4_{S_n}}$$

The network has limited resources to offer: let r_1,r_2,r_3,r_4 be the max CPU, Memory, Bandwidth and power resources available respectively and A be the area covered by the network base stations. Then, for every time point, $t_k \in T$ we can define our KPIs as objective functions and specify existing constraints:

Area Capacity - KPI_1, *number of users* × *bandwidth served per* m^2 *in area*:

$$\text{maximise} \sum_{i=1,j=1}^{i=n,j=m} \frac{x_{u_j,s_i} * e_{3S_i}}{A}$$

UE data rates - KPI_2, *mean bps served*:

$$\text{maximise} \sum_{i=1,j=1}^{i=n,j=m} \frac{e_{3S_i}}{n}$$

Cost efficiency - KPI_3, *mean cost*:

$$\text{minimise} \sum_{i=1,j=1}^{i=n,j=m} \frac{C_{S_i}}{n}$$

Resource utilisation efficiency - KPI_4, *total resources allocated to slice minus resources used:*

$$\text{minimise} \sum_{i=1,j=1}^{i=n,j=m} (r_1-e_{1_{tot}})+(r_2-e_{2_{tot}})+(r_3-e_{3_{tot}})+(r_4-e_{4_{tot}})$$

Resource consumption cannot exceed available resources:

$$P_{tot} \leq P_{max}$$
$$R_{tot} \leq R_{max}$$
$$B_{tot} \leq B_{max}$$
$$E_{tot} \leq E_{max}$$

As mentioned, additional constrains can be imposed either by SLAs, e.g. AR requires bandwidth > 4 Mbps or by required QoS, e.g., mean user data rates should be between 1.5 and 4 Mbps ($1.5 \leq KPI_2 \leq 4$). KPIs can be adversarial, i.e. one cannot improve the value of one objective, without worsening the value of another objective, in this case increasing UE data rates, decreases cost efficiency. Using the definitions given above, we can form the final problem:

$$\text{minimise}_f \, J=\text{minimise}[-KPI_1,-KPI_2,KPI_3,KPI_4]$$
$$\text{minimise}_f \, J = \text{minimise} \, [-KPI_1, -KPI_2, KPI_3, KPI_4]$$

which can then be solved.

The presented problem formulation takes into account multiple objectives J that must be minimised at the same time. There might be cases however, in which these objectives are adversarial, i.e. one cannot improve the value of one objective, without worsening the value of another objective. For example, there might be the case that one might need to find a trade-off between latency and energy consumption, since increasing energy consumption (i.e. higher computation rate) reduces latency, while reducing energy consumption (i.e. lower computation rate) can increase latency. In this example, a single mapping function f that can simultaneously optimise both objectives, does not exist. However, there exists a set of optimal solutions, called Pareto set, in which each solution represents different trade-offs between the different objectives. In other words, the optimisation equation is a multi-objective optimisation problem, which has as a solution a set of Pareto optimal solutions.

Figure 4-26 shows an illustration of the problem formulation and the set of Pareto optimal solutions. Specifically, Figure 4-26(a) shows an illustration of the problem formulation procedure presented on the approach section. Given multiple monitoring metrics, the set of minimisation objectives (KPIs) is constructed. Afterwards, taking into account these KPIs, the available resources and the traffic demand at each time period, the proposed multi-objective processing approach provides a set of Pareto optimal solutions that represent different ways to map the resources to the slices. Figure 4-26(b) shows the list of Pareto optimal solutions, given a solution space (grey area) with respect to only two objectives J_1 and J_2 for illustration purposes. Specifically, solutions f_2 and f_3 belong to the Pareto front, since they are not comparable with respect to both objectives. Specifically, f_2 is better with respect to objective J_1, while f_3 is better with respect to objective J_2. On the other hand, all the solutions in the light-grey area of Figure 4-26(b) are dominated by solution f_2, e.g. solution f_1 is clearly worse than solution f_2 taking into account both objectives (f_2 has lower J_1, J_2 values compared to f_1).

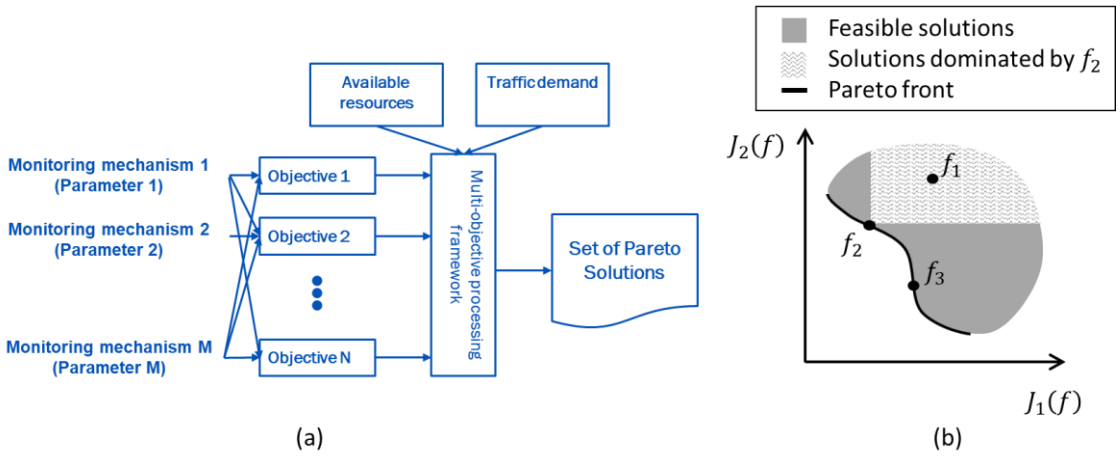


Figure 4-26: Illustration of the problem formulation and the set of Pareto optimal solutions. (a) The multi-objective processing framework takes as input multiple objectives, the available resources and the traffic demand, and returns the set of Pareto solutions. (b)

4.3.2.2 Optimising Computational Resource Utilisation in Slice-enabled Systems

In this section, we focus on the problem of allocating VNFs of different slices to a set of available computational resources, structured to a set of computation machines. As the flexible utilisation of the computational resources is an integral part of 5G networks (virtualisation techniques are exploited), the solution of such a problem is needed during the service establishment phase. Although for this allocation, we can theoretically assume infinite resources at a centralised cloud, it is not the case when edge cloud/fog nodes undertake the processing load. It is also important to utilise the resources in an efficient way, and thus, make room for additional processes to be allocated and increase elasticity when additional processing power is requested. We formulate the aforementioned problem as a resource-constrained project scheduling problem (RCPSP) and examine a heuristic solution to solve it. Note that the formulation proposed here specifies a single optimisation functions that potentially can feed the multi-objective optimisation framework provided in Section 4.3.2.1. However, to avoid confusion, the reader should be noted that the notation in Section 4.3.2.1 is not adopted here.

The RCPSP in this scenario refers to the problem of allocating VNFs that compose slices to computation resources (or computation machines) and it is stated as follows:

- Let a set of computation machines $M^T = \{m_1, m_2, m_3, \dots m_k, \dots, m_M\}$, that refer to processing machines on which the VNFs can run, and a set of services $J = \{J_1, J_2, J_3, \dots J_i, \dots, J_N\}$.
- Each service consists of a sequence of n_i VNFs.
- Each VNF_{ij}, $i = 1,2,3, \dots, N$, $j = 1,2,3, \dots, n_i$ can be allocated on an arbitrary computation machine m_k of a set $M^s (M^s \subseteq M^T)$, at a corresponding setup time S_{ijk}, and be processed for a

given uninterrupted processing time P_{ijk}. Two dummy VNFs are assumed to indicate the start and the end of the entire procedure, respectively.

- Let also C_i be the completion time of job J_i, and $C_{max} = \max_{1 \le i \le N}(C_i)$ the makespan, i.e., the total time needed to complete the execution of all the VNFs (or the time needed for completing the "longer" service).

The objective is to devise a compromise schedule that minimises or maximises an objective function subject to a set of constraints. Mathematically:

$$\text{Max or Min } f$$

subject to:

$$ST_{ij} + \sum_{k=1}^{m}\left[x_{ijk}\left(S_{ijk} + P_{ijk}\right)\right] \le ST_{i(j+1)} + \sum_{k=1}^{m}\left[x_{i(j+1)k}\left(S_{i(j+1)k} + P_{i(j+1)k}\right)\right]$$

$$\forall i \;\; and \;\; j = 1,2,..n_i - 1$$

$$\sum_{k=1}^{m} x_{ijk} = 1 \text{ with } \forall \text{i}, j$$

$$S_{ijk} = S_{ij}\, x_{ijk} \text{ with } \forall i, j, k$$

where x_{ijk} equals 1 if VNF_{ij} is finished on computation machine k, and 0 otherwise.

The inequality above indicates the operation sequence constraint, while the two equations indicate that each computation machine m_k, $k = 1,2,3, \dots, M$ can process at most one VNF at a time, and at most one VNF of each service J_i can be processed at a time, respectively. The optimisation function may refer to the resource utilisation efficiency, or another elasticity-related KPI.

According to the computational complexity theory, the RCPSP belongs to the class of problems that are NP-hard in the strong sense. An optimisation problem is NP-hard (in the strong sense) if its decision version is NP-complete (in the strong sense). The proof that the decision problem is NP-complete has been provided by Garey and Johnson [GJ79]. Another, easy way to prove that is through graph theory, since the RCPSP is mapped to the graph colouring problem. This analogous trigger the design of graph-based solutions that can approximate the optimal solution of the problem. A key paradigm in this category of solutions is the ant colony optimisation (ACO), inspired by the behaviour of real ants searching for food. According to [MGd04], "*The main underlying idea, loosely inspired by the behaviour of real ants, is that of a parallel search over several constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained result.*" If the VNFs are looked at as ants, it can be seen easily that there exist lots of similarities between an ant colony's foraging process and RCPSP. VNFs must search for proper machine to process them. Like ants, they want to search for the shortest path. Ants' nest and food source are similar with start and end dummy operation, respectively. If we look at an operation as an ant's path for foraging food, then the relation of any two operations can be looked at as an alternative path, and different processing time of all the operations on the machine just like different length of paths. The key steps of the basic ACO are the calculations of transition probability, visibility, and pheromone amount. An example of the approach that can be followed is given in [LTP+14]. The ACO provides the advantage of reaching a close to optimal solution in a distributed way. However, the cost is the time needed to converge to this solution. A cost that, in the framework in which we apply the solution, may affect the service creation time.

Overall, the allocation of VNFs of different slices/services to a set of available computational resources/computation machines can be formulated mathematically as described above. This formulation reveals the complexity of the problem as well as the implications (e.g., the correlation with the input and a set of constraints) that should be considered for its solution.

4.3.2.3 Slice-aware Automatic RAN Configuration

In 5G systems, due to the at least initially higher operating frequencies compared to 4G, multiple beams are needed to cover a cell area and serve the users. Within a cell, beam management (Layer 2 mechanism) is used for ensuring service continuity for moving users. The handover of users between two cells is handled by Layer 3 (as in legacy LTE) but is based on compounded beam measurements (referred to cell level quality) of the source and target cell.

On the user end, different services may have very different requirements and optimal operating regions (in terms of latency, throughput, tolerable interruption times, etc.). The assumption is that each service may be mapped to an individual slice and all slices could share the same RAN. Users could be grouped based on their service/traffic type and dedicated beams set up for serving them. Beam management needs to be matched to the different operational region and consider multiple sites/cells. Inter-service (/slice) beam management is needed to fulfil different service requirements simultaneously.

This means that a solution is needed to properly set up the RAN parameters to satisfy the requirements of the current services. In other words, the proposed solution will find the optimal cell operating region corresponding to the ongoing services and dynamic network conditions and reconfigure the RAN accordingly, in a proactive and automated fashion. The idea can be explained using Figure 4-27 below.

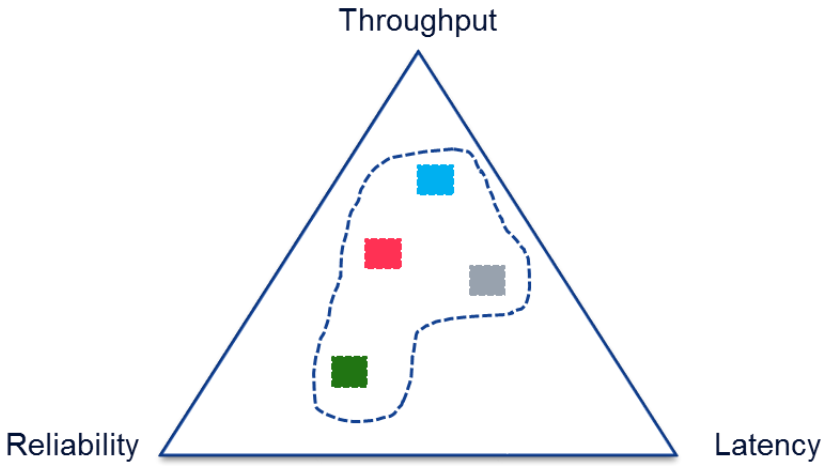


Figure 4-27: Service-aware cell operational region

Each coloured square in the figure corresponds to the **operational region** (i.e. that satisfies all requirements) of a given service ongoing in the current cell. The dotted area is the **cell operational region** that fulfils the requirements of **all** the ongoing services. This region may vary from cell to cell and depending on dynamic network conditions such as time of day, geo location, load, a service appearing/disappearing from the cell, etc. Ideally such an area can be constructed but in some cases (e.g. orthogonal requirements) this may be possible. In such cases, an alternative solution must be found (e.g. prioritisation). The desired end result is to provide users with a smooth QoE while using the RAN resources are used in an efficient manner.

In more concrete terms, the planned contribution will look at an outdoor touristic city scenario comprising of several cells and two active slices, a browsing slice and a tourist slice.

Users mapped onto the browsing slices are assumed to have a normal distribution in the cell and will generate background traffic. This means that their traffic demand will be relatively stable during different times of the day, weekdays and weekends and will have a minimum throughput requirement.

The tourist slice is serving tourist buses moving through the city and offering AR/VR applications on phone/tablet. The applications could for example show past and future of buildings in the tour and surroundings and offer interactive features. As such, the requirements of this slice will be quite different, i.e., high throughout and low latency. The traffic density for this slice will be significantly higher than the browsing slice but at the same time will be a transient presence in the cell. For instance, browsing slice traffic will be limited to day time only and have a higher foot print on weekend and holidays, less in off-season.

As users from both slices move within the network, the goal is to adapt the network configuration (i.e. the 'virtual' cell coverage) in a proactive and automated fashion to match the dynamic changes in the slice specific needed capacity.

This can be done by adapting the beam patterns of individual cells as the hotspot of tourist slice users is moving through the network. The coverage of the cell where the hotspot is currently located will be minimised whilst the coverage of the surrounding cells will be maximised. This will insure that the central cell serves (almost) exclusively the tourist slice users. Browsing users will be 'pushed' to be served by surrounding cells. In order for this solution to work, the cell group cell pattern change needs to be either controlled by a central entity or direct communication between in involved cells/nodes needs to be implemented.

4.3.2.4 Slice Blueprint Analytics and Classification

In our contribution, a slice is composed of a set of NFs described in a slice blueprint. In this section, we consider that network slice functions can be grouped into two main types: access network, and core network related NFs (see Figure 4-28). Eventually, each of these groups is composed of a finite number of NFs and a slice demand will require a set of these functions. The main idea in this work is to cluster slice requests that have a high similarity to reduce the resources used by each network slice.

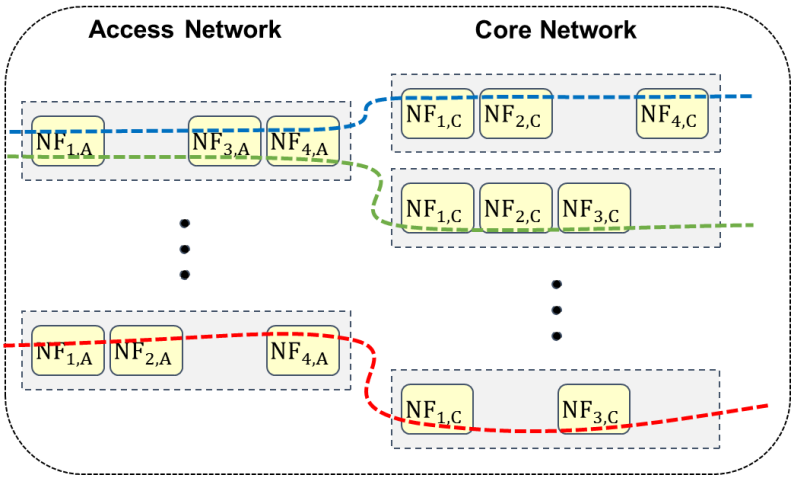


Figure 4-28: Mapping of slices and network function requests

Among different slice requests, we can have shared functions such as $NF_{1,T}$ as well as dedicated functions such as $NF_{2,T}$. When resource isolation across slices does not have to be guaranteed, resources utilisation efficiency will increase if two slice requests that shared NFs are allocated on common virtual resources at the same time. From an operator's point of view, this means this approach enables to increase the number of accepted service requests (i.e., higher user satisfaction), decrease a slice deployment cost, and the service creation time.

We use the following the mathematical model to characterise a slice request.

Let η be the set of all the $K = K_R + K_T$ NFs in the system:

$$\eta = \eta^R \cup \eta^T = \{NF_k\}_{k=1}^{K}$$

$$= \{NF_k\}_{k=1}^{K} = \{NF_{1,R}, \dots, NF_{K_R,R}\} \cup \{NF_{1,C}, \dots, NF_{K_C,C}\}$$

where η^R and η^C are the set of NFs in the access and core networks respectively.

For the n^{th} slice request, we define the related slice blueprint by introducing the NF request D_n and the associated parameters X_n as follows:

$$D_n = \{NF_{i\in I}\}, I \subseteq \{1,..,K\}$$

$$X_n = \{X_{i\in I}\}, I \subseteq \{1,..,K\}$$

Then, we represent the NFs composing a request D_n as a vector $\sigma_n \in \{0,1\}^K$ such as its j^{th} entry is defined as:

$$\sigma_n^j = \begin{cases} 1 \; if \; NF_j \in D, \forall j \in \{1, \cdots, K\} \\ 0 \; otherwise \end{cases}$$

If $NF_i \in D_n$, i.e., the slice n requests a certain NFs NF_i, each of its parameters $X_{n,i}$ is defined by b_i binary labels:

$$X_{n,i} = \left(l_{n,i}(1), l_{n,i}(2), \cdots, l_{n,i}(b_i)\right), \mathrm{i} \in \mathrm{I} \subseteq \{1, .., K\}.$$

For NF_i we map its parameters with some radio, computing, and storage resource requirements, as follows:

$$d_{r,n,i} = \rho_i + f_{r,i}\left(X_{n,i}\right)\alpha_r$$
$$d_{c,n,i} = \chi_i + f_{c,i}\left(X_{n,i}\right)\alpha_c$$
$$d_{m,n,i} = \mu_i + f_{m,i}\left(X_{n,i}\right)\alpha_m$$

where ρ_i χ_i, and μ_i are the minimum required resources for activating a given NF i, and $f_{r,i}\left(X_{n,i}\right)\alpha_r$, $f_{c,i}\left(X_{n,i}\right)\alpha_c$ and $f_{m,i}\left(X_{n,i}\right)\alpha_m$ are the required resources as a function of NF's parameters $X_{n,i}$. In summary, the total resources demand of a slice request is presented as follows:

$$T_n = \left(\sum_{i \in I} d_{r,n,i} \; , \sum_{i \in I} d_{c,n,i} \; , \sum_{i \in I} d_{m,n,i}\right)$$

Our goal is to separate the slice requests in Figure 4-29(a) into several clusters that share a given set of NFs, i.e. require less resources for deploying these functions, in order to reduce the cost of the slices deployment.

Accordingly, we define in the following a two-step procedure that enables the system to analyse the blueprints of deployed and queued slices and identify whether the available resources can be shared across different slice instances.

Step 1: Cluster the slice requests in big clusters, Figure 4-29(b), based on the common NFs regardless the resources required. This will offer a sub-optimal solution.

Step 2: For each cluster study the similarity between a new slice request and the running slice for which it was assigned in order to determine the new reduced amount of resource required.

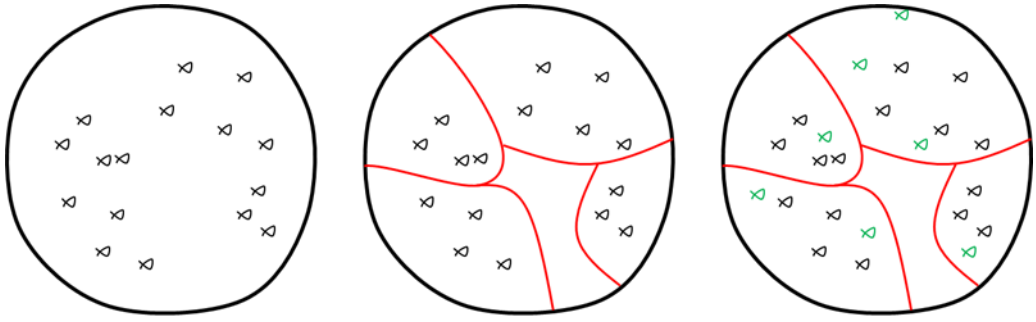


Figure 4-29: Slice clustering

In step 1, when a slice demand arrives, the most preferred cluster is obtained by calculating the similarity with all running slice instances and is assigned to the cluster that has highest similarity. The similarity between arriving slice demand σ_n and a deployed slice $\sigma_{n'}$ is evaluated using Jaccard similarity $p_{n,n'}$:

$$p_{n,n'} = \frac{\sigma_n \cap \sigma_{n'}}{\sigma_n \cup \sigma_{n'}}$$

Regarding Step 2, we evaluate the input similarity between different slices of the same cluster at each NF as follows:

$$S(X_{n,i}, X_{n',i}) = g_i\left(h_{i,1}(l_{n,i}(1), l_{n',i}(1)), \cdots, h_{i,J}(l_{n,i}b_i), l_{n',i}(b_i)\right)$$

Where $h_{i,j}$ and g_i are to be specified according to the NF.

By determining the similarity between the slice request n and the set $\mathcal{K}_{\ell,i}$ of slices n' in the cluster ℓ using NF_i, we will be able to scale down the request requirement as follows:

$$d_{r,n,i} = (1 - \max_{\forall n' \in \mathcal{K}_{\ell,i}} S(X_{n,i}, X_{n',i}))f_{r,i}\left(X_{n,i}\right)\alpha_r$$
$$d_{c,n,i} = (1 - \max_{\forall n' \in \mathcal{K}_{\ell,i}} S(X_{n,i}, X_{n',i}))f_{c,i}\left(X_{n,i}\right)\alpha_c$$
$$d_{m,n,i} = (1 - \max_{\forall n' \in \mathcal{K}_{\ell,i}} S(X_{n,i}, X_{n',i}))f_{m,i}\left(X_{n,i}\right)\alpha_m$$

Now by applying this scaling down policy at all the NFs' requirement of a new slice request, we recalculate the new demand T'_n, and apply the admission control accordingly with prioritising the requests based on their SLA.

In conclusion, in this slice-aware elasticity solution, we propose to decrease the resource requirement of a slice request by clustering the slices based on their common NFs so that we decrease the required resources for a slice activation by running requests with common NFs on the slice assigned to a given cluster. We propose also to add elasticity to the resources requirement related to a NF input by studying the similarity between two requests on the same cluster. We believe that this solution will offer a close-to-optimal utilisation of the network resources, and aim to prove this by simulations.

4.3.2.5 Big Data Analysis for Network Slice Characterisation

In this section we discuss the potential benefits of the characterisation of network slices according to their spatiotemporal behaviour. More specifically, we detail the performed analysis, and which impact it may have on the network orchestration.

Methodology

As the mobile Internet traffic grows along with the quantity and diversity of offered services, it becomes increasingly important to understand the demands generated by such services.

Indeed, characterising the traffic dynamics associated to different mobile services is of paramount importance in order to properly dimension and orchestrate the mobile network, and also offers an opportunity to unravel broader societal behaviours in general.

In future-generation mobile networks, the understanding of when, where and how different mobile services are consumed is essential to dynamically tailor resources to the actual fluctuations of the subscribers' activity. Indeed, many novel architectural paradigms are aimed at enabling the dynamic management of system resources, at the network edge or core as well as encompassing multiple NFs. For instance, an effective orchestration of network slices builds on the spatial complementarity of the demands for the different services.

This activity will be based on the work performed in [MGF+17], in which the initial characterisation of network slices has been performed.

We analyse data coming from real 3G/4G cellular deployments. More specifically we use data coming from common traffic probes located at the GGSN or P-GW to retrieve information on the amount of exchanged data (in UL and DL), time information (i.e., when data is exchanged) and position information (i.e., where data is exchanged). Traffic is aggregated at per-application level (i.e., YouTube, Netflix) and several antenna sector (i.e., small administrative domains). In this way, end-user privacy is ensured by means of aggregation (we do not have information about individual user sessions).

We use the exchanged bytes in UL and DL as a proxy for the real load inserted into the system by a given application that is mapped to one of the well-known services definition (e.g., video streaming maps to an eMBB slice). That is, a traffic peak for a given service means both a higher usage of PRBs (that concerns VNFs like scheduling) and the CPU supporting those VNFs. We will exploit this information for two outcomes i) understand the characteristics of a network slice (how it moves, how it develops over time) and ii) enhance the orchestration algorithms by leveraging this information.

Services behaviour initial evaluation

The outcomes of the work we performed in [MGF+17] already unveiled interesting characteristics of network slices (classified as mobile application) in terms of time and space usage. Therefore, we investigated the temporal dynamics of different mobile services at the national scale, aggregating the weekly demand for each service in space, over all administrative areas. Examples of the resulting time series are shown in Figure 4-30 for four sample mobile services in DL.

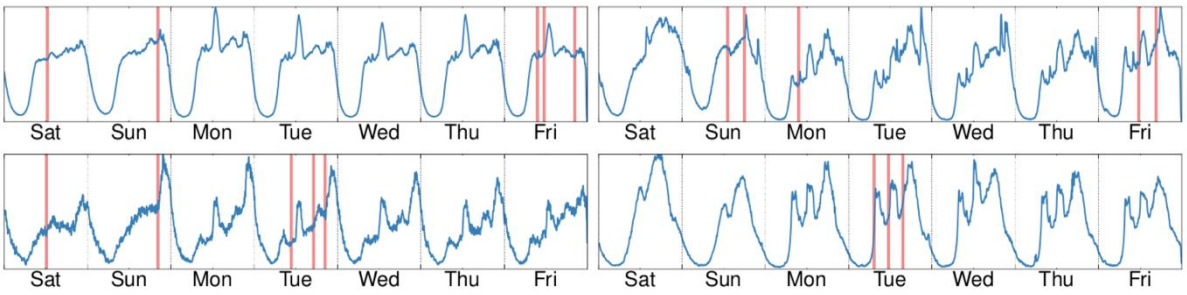


Figure 4-30: Sample time series of mobile services left to right and top to bottom, Facebook, SnapChat, Netflix and Pokemon Go.

We first observed the typical weekend/weekday patterns. For instance, the first plot in Figure 4-30, which refers to Facebook, displays a major traffic peak at midday of working days, plus several other minor peaks. However, other services in Figure 4-30 show other traffic peak arrangements.

Motivated by this last observation, we tried to understand the reasons behind this behaviour. As a matter of fact, in [MGF+17] we observed that this behaviour applies to all the applications in our dataset. Our results suggest that the cause for the above behaviour may lie in the different patterns of activity peaks that characterise each service. In order to verify this possibility, we rigorously detect the activity peaks in the per-service time series using the smoothed z-score algorithm [ZScA].

Examples of peaks inferred by the smoothed z-score are in Figure 4-30, where vertical red lines denote the rising front of peaks (for the sake of clarity, Figure 4-30 only shows one detected peak per type in each sample time series). Interestingly, by applying this methodology to all mobile services, we find that peaks only appear at seven specific moments during the week: at midday (around 1pm) and evenings (9pm) during weekends, and during the morning commuting time (8am), morning break (10am), midday (1pm), afternoon commuting time (6pm) and evenings (9pm) during working days.

We summarise the peak patterns observed for all services as done in Figure 4-31. In this figure, each sector refers to one mobile service, and each ring to a different topical time, as per the legend. We remark that: (i) individual services tend to have very diverse patterns even when looking only at when they show peaks of activity; (ii) this heterogeneity also separates services that belong to a same macro-category, e.g., video streaming behaves quite differently in YouTube, Facebook, Instagram, Netflix and iTunes platforms.

We then tried to cluster the time series, using the well-known *k-shape* algorithm, which is the current state-of-the-art unsupervised technique for time series clustering, as proven by extensive tests.

We then carry out an exhaustive search for mobile service clusters, by testing *k*-shape on all possible values of k, in combination with multiple indices of clustering quality. The latter are the (modified) Davies-Bouldin, Dunn, and Silhouette indices, which constitute a representative selection of popular indices used in the literature to rank different cluster sets generated from the same original elements [MC85].

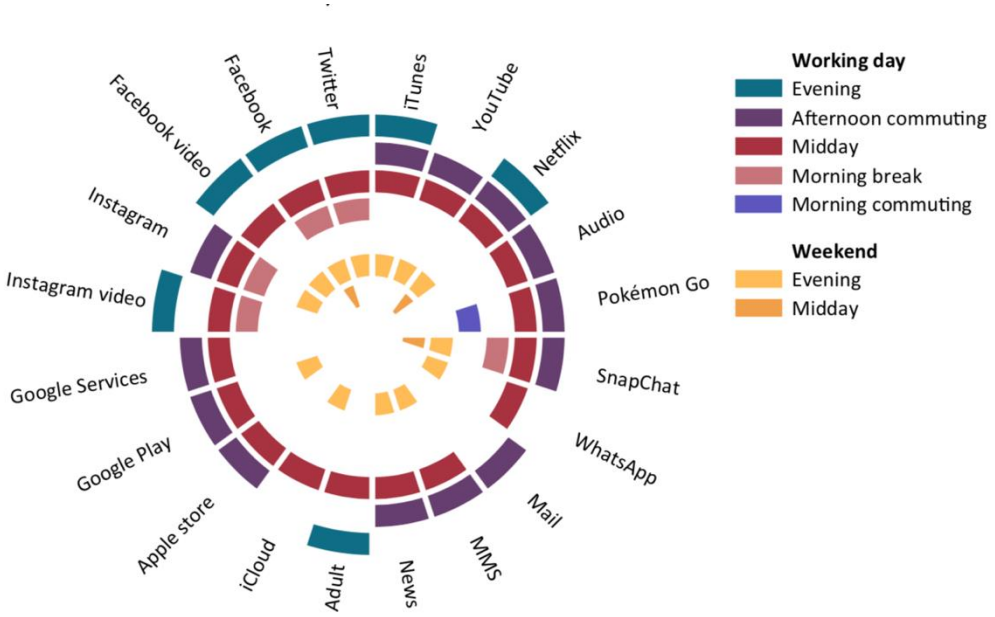


Figure 4-31: Activity peak times of mobile services

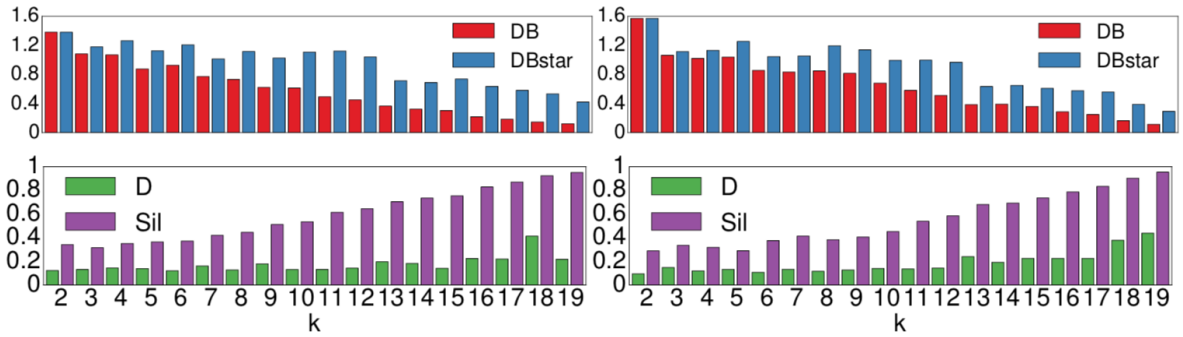


Figure 4-32: Clustering quality indices versus the cluster number, in downlink (left) and uplink (right). Plots for Davies-Bouldin, modified Davies-Bouldin (top, minimum is best) and Dunn, Silhouette (bottom, maximum is best) indices

As for all the indices the best decision is to create k=20 clusters, we can conclude that no two services exhibit similar time patterns in their nationwide aggregate traffic: although expected for different service categories, this is less obvious for akin services. This information can be exploited by orchestration algorithms to e.g. maximising multiplexing gains.

A second step we performed in [MGF+17] was to analyse the spatial behaviour of services, in terms of bytes/subscriber. We also observe here a high variability in the service usage. Figure 4-33 shows the map of the weekly per-subscriber Twitter traffic in DL. The distribution is highly skewed: subscribers in half of the communes consume a negligible weekly Twitter load below 1 byte, whereas users in other areas download tens of Mbytes of Twitter contents per week.

This is due to the inherent diversity in population densities (we normalise by using the registered people) and cellular coverage. However, if we compare services among specific service pairs: we experience low correlation only with Netflix (almost completely absent in rural areas) and iCloud (pushing UL data from all iPhones, and thus more uniformly distributed over the country). These outlier cases apart, our results let us conclude that mobile services tend to be consumed similarly. Such mobile data consumption features geographical distributions of the per-user traffic that are highly skewed, with subscribers within cities and on inter-city routes that tend to be much more active than those in other areas. This information can be used to carefully design orchestration algorithms that assign resources to slices on a geographical basis.

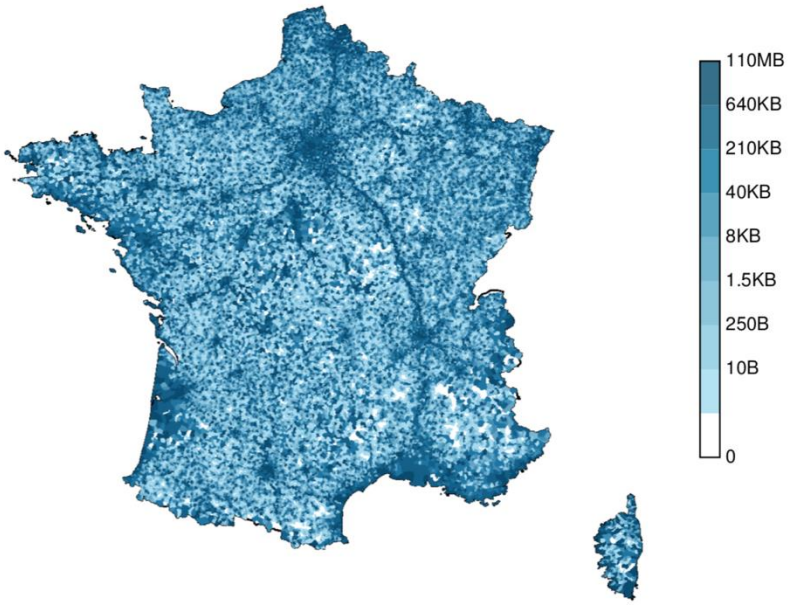


Figure 4-33: Map of the average per-subscriber activity, for downlink Twitter

Achievable enhancements

The accurate spatiotemporal maps of the resource demand of different services allows to a precise orchestration and thus a greater advantage for those services, such as eMBB, that are likely to be orchestrated in a way such that multiplexing gains can be exploited more efficiently. Accurate characterisation of traffic demands over a given area supports a more efficient planning of network resources. This can be achieved in several ways. The legacy one is the provisioning of more physical infrastructure (i.e., antennas, band) that can be shared by those services (i.e., slices) that need it in particular times or location of the network. However, a softwarised network such as the one targeted by 5G-MoNArch allows for different kinds of strategies to increase the area capacity. An operator may decide to provide more cloud resources according to the real demand. Or reassign cloud resources to a specific slice on demand: this network sharing strategy allows for an economically feasible network deployment. Ultimately, an operator can provide higher area capacity (and hence monetise it) with less infrastructure, resulting thus in a better €/bytes ratio. Efficient orchestration algorithms that exploit the knowledge on how network slices (in terms of user demand) move in space and time allow for the maximisation of multiplexing gains that naturally happen in the network. This can be applied at any level, ranging from spectrum at the antenna level to the cloud resources (i.e., memory or CPU) in centralised deployments.

Besides the large-scale deployment of central clouds that process data (e.g., I/Q samples for a C-RAN scenario) the accurate characterisation supports a very efficient deployment of resources over time, for capacity-limited setups. That is, without any prior knowledge and efficient orchestration algorithms that exploit this information, an operator may consider adding more cloud capacity to extreme areas not economically relevant. The availability of an orchestration algorithm that exploits past knowledge to infer the expected load and assign resources accordingly, can make this deployment efficient and thus exploitable. Similar considerations apply for the inclusion of small cell deployments, with edge cloud or MEC functionality.

Also, in a multi-slice scenario, the admission into the network of new slices necessarily entails new resources to be added. By knowing how typical slices develop in space and time, the amount of additional needed resources will be minimised. This also allow for a better monetisation of the network by the InP.

4.3.2.6 A market-based Approach to Network Slicing

A network slice is a collection of resources and functions that are orchestrated to support a specific service. This includes software modules running at different locations as well as the nodes' computational resources, and communication resources in the backhaul and RAN. The intention is to only provide what is necessary for the service, avoiding unnecessary overheads and complexity. Thus, network slices enable tenants to compete with each other using the same physical infrastructure but customising their slices and network operation according to their market segment's characteristics and requirements.

A key problem underlying network slicing is enabling efficient sharing of mobile network resources. One of the approaches considered in 3GPP suggests that resources could be statically partitioned based on fixed 'network shares' [3GPP14-22852]. However, given that slices' loads may be spatially non-homogenous and time varying, it is desirable to allow resource allocations to be 'elastic', e.g., dependent on the slices' loads at different base stations. At the same time, tenants should be protected from one another, and retain the ability to autonomously manage their slice's resources, in order to better customise al- locations to their customers. To that end, it is desirable to adopt resource allocation models in which tenants can communicate their preferences to the infrastructure (say by dynamically subdividing their network share amongst their customers) and then have base stations' resources allocated according to their preferences (i.e., proportionally to the customers' shares).

Under such a dynamic resource allocation model, a tenant might exhibit strategic behaviour, by adjusting its preferences depending on perceived congestion at resources, so as to maximise its own utility. Such behaviour could in turn have adverse effects on the network; for instance, the overall efficiency may be harmed, or one may see instability in slice requests. Therefore, we aim to evaluate this simple resource allocation model, and validate its feasibility as a means to enable tenants to customise resource allocation within their slice while protecting them from one another. One of the key features of our approach is the ability of tenants to customise their allocations; there is wide consensus in the standardisation community that this is needed to efficiently satisfy their very diverse requirements (see, e.g., [5G-PPP16] for examples of possible vertical tenants).

We will analyse the network slicing market from a game-theoretic standpoint, trying to find theoretical guarantees for the Nash Equilibrium, if e.g. tenants apply specific admission control. Based on this framework, which may include different admission control policies we can derive a resource allocation mechanism, and a strategy to drop users when rate guarantees are infeasible. These game-theoretic approaches are expected to achieve significant gains in terms of utility, throughput and block probability (i.e., the probability of having spare resources in the network, but not when the tenant needs) when compared with simpler resource assignment schemes that statically slice the network.

The outcome of this research work has a clear impact on the orchestration algorithm design that can assign resources according to this strategy.

As stated previously, a market based architecture is a valid strategy for the monetisation of network resources. However, the same problem can be tackled from a different perspective. We can think at the 5G infrastructure (i.e., spectrum, transport network, cloud resources) as a market in which the InP tries to maximise his revenues by selling resources to tenants that, in turn, run network slices. Therefore, a legitimate question is how to decide whether to accept or not requests for resources coming from the MSP on behalf of the tenants. That is, an InP could reject a request coming from a tenant (waiving the income opportunity), keeping resources free if in the future an offer with a bigger associated revenue (either because it is associated with a more remunerative slice or because the slice size is larger). Choosing the best admission control strategy is thus a fundamental problem that has to be solved that entails the solution of two subproblems:

- The definition of an *admissibility region* that limits the number of network slices that can be hosted in an infrastructure given the kind of slices (i.e., their requirements) and the available infrastructure.
- The design of *an admission control algorithm* that, given the admissibility region, decides whether to accept or not a slice request. Figure 4-34 depicts the proposed architecture.

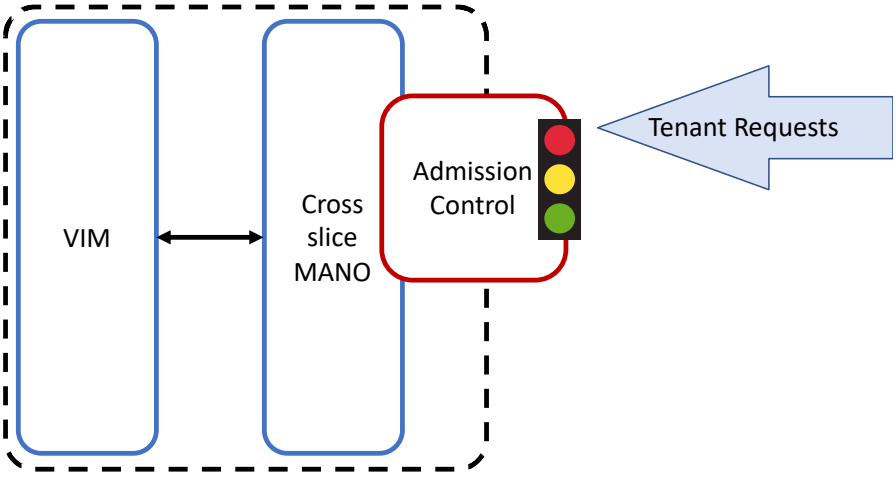


Figure 4-34: The proposed architecture

In [BGB+17] we provided a holistic solution for this problem, proposing an admissibility region model for elastic and inelastic traffic and a ML (Q-Learning) algorithm for the admission control algorithm. Results showed how the proposed algorithm could actually outperform any naïve or smart policy in terms of revenues, even with variable arrival rates.

However, the solution in [BGB+17] did not deal with elasticity requirements as we defined in this document. Moreover, the Q-learning approach developed in [BGB+17] may fail to scale on larger scenarios, with a more heterogeneous network slice classification. Therefore, an elastic admission control algorithm should:

- Consider a broader space of network slice types, with different latency and bandwidth requirements.
- Provide a scalable algorithm for this richer network slices ecosystem.

For the former point, we plan to include in the formulation the network elasticity concepts described, while for the latter we will work on a Deep Learning extension of the model, to increase scalability and efficiency.

5 Conclusions

This report provides the main principles of the resource elasticity concept in 5G networks, focuses on the architectural elements and functionalities that enable it, and proposes a set of innovative mechanisms for resource elasticity provisioning. More precisely, it includes: i) a conceptual study of resource elasticity provisioning, a description of its KPIs, and an analysis of its economic implications; ii) the 5G-MoNArch approach on the operational requirements needed for an elastic 5G network, with particular attention devoted to the architectural innovations that enable it; and iii) a set of mechanisms for resource elasticity provisioning.

As explained in this report, the first requirement for an elastic network or network slice is the use of elastic VNFs, i.e., the VNFs to be able to cope with variations in the availability of resources without causing an abrupt degradation in their outputs. This is because a network or network slice is composed of a set of VNFs that, chained, provide a telecommunication service. The second requirement for an elastic network or network slice is the use of an elastic infrastructure on which the VNFs run. In this context, orchestration-driven mechanisms can exploit cloud capabilities and allocate the elastic and non-elastic VNF to the available infrastructure in a cost-effective way. To take advantage of the elasticity gains in a 5G network interventions are also needed in the 5G architecture. As explained, enhancements in the architecture and interfaces of the Service and M&O Layers, in Controller and Network Layers, and at data analysis level are required, leading to the elasticity-enabled 5G architecture of 5G-MoNArch. This sets a third requirement for an elastic network or network slice, referring to the exploitation of AI and big data analytics, an approach that abides by the efforts of the ENI ISG of ETSI for 5G networks. Network elasticity needs to be supported by efficient monitoring mechanisms, which constantly update the network orchestrator on the state of VNFs, slices, and resources. The ad-hoc designed architectural enhancements proposed by 5G-MoNArch aim at facilitating the exchange and collection of contextual information on the state of the network; at the same time, they are essential to enable an effective application of the three complementary levels of elasticity in a coordinated and integrated manner, adapting to the contingent requirements imposed by slices.

In this report both business and technical KPIs for resource elasticity are described, including also techno-economic factors that are affected. More precisely, two main categories of resource elasticity KPIs have been identified. The first category includes more conventional metrics, such as the service creation time or the availability, while the second category, is more focused on the impact of elastic operations to a network.

Having identified the major requirements for a resource elastic network and the KPIs, this report highlights state-of-the-art solutions to fulfil these requirements and proposes a set of innovative solutions, organised into three categories: computational, orchestration-driven, and slice-aware.

- For computational elasticity, a lot of room for improvement beyond the state of the art is made, focusing mainly on the RAN functions. Abiding by the 5GNR Rel.15, novel RAN functionalities are defined, providing higher degrees of freedom, enabling the potential to redesign them in a more elastic way. Novel ideas for advanced VNF at RAN are presented, including functions related to the MU-MIMO, the MCS selection, and scheduling.
- For the orchestration-driven elasticity the main challenges are related to techniques for cloud-enabled VNF migration. The proposed approaches examine new proactive and live VNF relocation approaches, while describe mechanisms for resource sharing and admission control.
- The third category of resource elasticity incorporates algorithms and mechanism for elasticity-based resource utilisation in a slice-aware environment. As explained in this report, the computational resource allocation in such environments is subject to a set of constraints and can be faced through different approaches. To deal with this, the report examines the adoption of multiple tools, including multi-objective optimisation, heuristic algorithms, clustering mechanisms, and big data analysis.

Overall, this report is a comprehensive guide on resource elasticity provisioning in 5G networks, while it serves as a beacon for next deliverables, highlighting the related innovations under development in WP4 of the 5G-MoNArch project.

6 References

[3GPP14-22852] 3GPP "Study on Radio Access Network (RAN) sharing enhancements," TR 22.852, v13.1.0, Sep. 2014.

[3GPP17-28801] 3GPP Technical Report, TR 28.801 v15.0.0, "Telecommunication management; Study on management and orchestration of network slicing for next generation," Sep. 2017.

[3GPP17-38211] 3GPP Technical Specification Group Radio Access Network, TS 38.211 v15.0.0, "NR; Physical channels and modulation," Dec. 2017.

[3GPP17-38300] 3GPP TS 38.300 V15.0.0, "NR and NG-RAN Overall Description; Stage 2 (Release 15)," Dec. 2017.

[3GPP18-36213] 3GPP Technical Specification Group Radio Access Network, TS 36.213 v15.1.0, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures," March. 2018.

[5G-PPP16] 5G-PPP, "5G Empowering Vertical Industries," White Paper, Feb. 2016.

[5GM17-D6.1] 5G-MoNArch Deliverable 6.1, "Documentation of requirements and KPIs and definition of suitable evaluation criteria", Sept. 2017.

[5GM17-D2.1] 5G-MoNArch Deliverable 2.1, "Baseline architecture based on 5G-PPP Phase 1 results and gap analysis," Oct. 2017.

[5GM18-D3.1] 5G-MoNArch Deliverable 3.1, "Initial resilience and security analysis," Jan. 2018.

[5GM18-D2.2] 5G-MoNArch Deliverable 2.2, "Initial overall architecture and concept for enabling innovations," Jun. 2018.

[5GN17-D6.2] 5G Norma Deliverable D6.2, "Demonstrator design, implementation and final results," Nov. 2017.

[ABB+17] P. Arnold, N. Bayer, J. Belschner, and G. Zimmermann, "5G Radio Access Network Architecture Based on Flexible Functional Control / User Plane Splits," In Proc. European Conference on Networks and Communications (EuCNC), June 2017.

[AC14] A. Abdelhadi and C. Clancy, "Context-aware resource allocation in cellular networks," arXiv:1406.1910v1, Jun. 2014

[Acc] Accuver, www.accuver.com

[AGH+15] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, F. Xia, and S. A. Madani, "Virtual machine migration in cloud data centres: a review, taxonomy, and open research issues," The Journal of Supercomputing 71, no. 7, Jul. 2015, pp. 2473-2515.

[AM13] A. Asadi and V. Mancuso, "A survey on opportunistic scheduling in wireless communications," IEEE Communications surveys & tutorials, 15(4), Jan. 2013.

[ATM18] I. A. Alimi, A. L. Teixeira, P. P. Monteiro, "Toward an Efficient C-RAN Optical Fronthaul for the Future Networks: A Tutorial on Technologies, Requirements, Challenges, and Solutions," IEEE Comm. Survey & Tutorials, vol. 20, no. 1, First Quarter 2018.

[BCK+12] Bhaumik et. al, "CloudIQ: A Framework for Processing Base Stations in a Data Centre," in Proc. 18th annual international conference on Mobile computing and networking, Aug. 2012.

[BGB+17] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis and X. Costa-Perez, "Optimising 5G Infrastructure Markets: The Business of Network Slicing," In Proc. IEEE INFOCOM 2017, May 2017.

[BHD13] E. Barrett, E. Howley, and J. Duggan, "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud," Concurrency and Computation: Practice and Experience, 25(12), Aug. 2013, pp.1656-1674.

[BLJ13] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A Survey on Machine-Learning Techniques in Cognitive Radios," in IEEE Communications Surveys & Tutorials, vol. 15, no. 3, pp. 1136-1159, Third Quarter 2013.

[BSH+14] Björnson, Sanguinetti, Hoydis, Debbah, "Designing multi-user MIMO for energy efficiency: When is massive MIMO the answer?," In Proc. IEEE Wireless Communications and Networking Conference (WCNC), Apr. 2014.

[BYC+17] J. Bendriss, I. G. B. Yahia, P. Chemouil, and D. Zeghlache, "AI for SLA management in programmable networks," in Proc. 13th International Conference on the Design of Reliable Communication Networks (DRCN), Mar. 2017.

[CCY+14] Checko, A. et al., "Cloud RAN for Mobile Networks: A Technology Overview," IEEE Communications surveys & tutorials 17, no. 1, Second Quarter 2015.

[CLV07] C. Coello, G. Lamont, and D. Van Veldhuizen, "Evolutionary Algorithms for Solving Multi-Objective Problems," New York, NY, USA: Springer, 2007, vol. 5.

[Criu] https://criu.org/Live_migration

[CSR+15] E. F. Coutinho et al., "Elasticity in cloud computing: A survey," Annals of Telecommunications, vol. 70, no. 7-8, Aug. 2015.

[DKM+11] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck. "Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow," In Proc. ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems, May 2011.

[Doc] https://www.docker.com/

[DocB] https://docs.docker.com/engine/reference/builder/

[DocO] https://docs.docker.com/compose/overview/

[EAC14] T. Erpek, A. Abdelhadi, and T. C. Clancy, "An optimal application aware resource block scheduling in LTE," In Proc. 2015 IEEE International Conference on Computing, Networking and Communications (ICNC), Feb. 2015.

[EBS17] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," In Proc 2017 European Conference on Networks and Communications (EuCNC), Jun 2017.

[Ehr05] M. Ehrgott, "Multicriteria Optimisation". Berlin, Germany: Springer, 2005, vol. 2.

[ETSI14-NFV] ETSI, Technical Report, "Network Functions Virtualisation (NFV): Management and Orchestration," Dec, 2014, see http://www.etsi.org/

[ETSI16-NFV] ETSI GS NFV-IFA 014, "Network Functions Virtualisation (NFV); Management and Orchestration; Network Service Templates Specification," Oct. 2016.

[ETSI17-ENI] ETSI White Paper, "Improved operator experience through Experiential Networked Intelligence (ENI)," 1st Edition – October 2017.

[FFR+15] W. Felter, A. Ferreira, R. Rajamony, J. Rubio, "An updated performance comparison of virtual machines and Linux containers," In Proc. 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Mar. 2015.

[FLG+17] X. Fang, J. Luo, H. Gao, W. Wu, and Y. Li, "Scheduling multi-task jobs with extra utility in data centres," EURASIP Journal on Wireless Communications and Networking, vol. 2017, p. 200, Nov. 2017.

[FNK+16] X. Foukas, N. Nikalein, M. Kassem, M. Marina, and K. Kontovasilis, "FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks," In Proc. 12th International Conference on emerging Networking EXperiments and Technologies, Dec. 2016.

[GJ79] M. R. Garey, D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman & Co. New York, NY, USA, 1979.

[GMG+17] H. Gupta, D. Manicone, F. Giannone, K. Kodepu, A. Franklin, P. Castoldi, and L. Valcarenghi, "How much is fronthaul latency budget impacted by RAN virtualisation?," In Proc. 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Nov. 2017

[GSB+15] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: Fundamentals and applications," in IEEE Commun. Mag., vol. 53, no. 5, pp. 52-59, May 2015.

[Haul] https://criu.org/P.Haul

[HB16] J. G. Herrera, and J. F. Botero. "Resource allocation in NFV: A comprehensive survey," IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 518-532, Sept. 2016.

[HKR13] N. R. Herbst, S. Kounev, and R. H. Reussner, "Elasticity in Cloud Computing: What It Is, and What It Is Not," In Proc. International Conference on Autonomic Computing (ICAC), vol. 13, pp. 23-27, Jun. 2013.

[HL16] V. N. Ha and L. B. Le, "Resource allocation for uplink ofdma c-rans with limited computation and fronthaul capacity," in IEEE ICC, May 2016.

[HL92] P. Hajela and C.-Y. Lin, "Genetic search strategies in multicriterion optimal design," Struct. Optimiz., vol. 4, no. 2, pp. 99–107, Jun. 1992.

[HTL+16] P. Hassanzadeh, A. Tulino, J. Llorca, and E. Erkip, "Cache-aided coded multicast for correlated sources," In Proc. 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), Sept 2016.

[iPer] iPerf3, https://iperf.fr/

[JSS+14] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," Computer Networks, vol. 72, Oct 2014.

[KC15] S. Khatibi and L. M. Correia, "A model for virtual radio resource management in virtual RANs," EURASIP Journal on Wireless Communications and Networking, vol. 2015, p. 68, 2015.

[KDT14] I. Kalamaras, A. Drosou, and D. Tzovaras, "Multi-objective optimization for multimodal visualization," IEEE Trans. Multimed., vol. 16, no. 5, pp. 1460–1472, Aug. 2014.

[KNS+17] K. Katsalis et. al, "Network Slices toward 5G Communications: Slicing the LTE Network," IEEE Communications Magazine, 55(8), pp.146-154, Aug. 2017.

[Lov10] R. Love, "Linux Kernel Development," Addison-Wesley, 2010

[LTP+14] E. Liotou, D. Tsolkas, N. Passas and L. Merakos, "Ant colony optimisation for resource sharing among D2D communications," In Proc. IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Dec. 2014.

[Man] http://man7.org/linux/man-pages/man7/cgroups.7.html

[MC85] G.W. Milligan, M.C. Cooper, "An Examination of Procedures for Determining the Number of Clusters in a Data Set," Psychometrika, 50(2):159--179, Jun. 1985.

[MGd04] V. Maniezzo, L. Gambardella, and F. de Luigi, "Ant colony optimisation," in New Optimisation Techniques in Engineering, Springer Berlin Heidelberg, vol. 141, pp. 101-121, 2004.

[MGF+17] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, C. Ziemlicki and Z. Smoreda, "Not All Apps Are Created Equal: Analysis of Spatiotemporal Heterogeneity in Nationwide Mobile Service Usage," In Proc. ACM CONEXT, Nov. 2017.

[MLS+17] F. Manco, C. Lupu, F. Schmidt, J. Mendes, S. Kuenzer, S. Sati, K. Yasukata, C. Raiciu, F. Huici "My VM is Lighter (and Safer) than your Container," in Proc. 26th Symposium on Operating Systems Principles, Oct 2017.

[MMR+13] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft. 2013. "Unikernels: library operating systems for the cloud," SIGARCH Comput. Archit. News 41, 1, pp. 461-472, Mar. 2013.

[MV95] J. K. MacKie-Mason and H. R. Varian, "Pricing congestible network resources," in IEEE J. Sel. Areas Commun., vol. 13, no. 7, pp. 1141-1149, Sep. 1995.

[NGMN15] Next Generation Mobile Networks (NGMN) Alliance, "5G WhitePaper", Feb. 2015, Available Online:
https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2015/NGMN_5G_White_Paper_V1_0.pdf

[NGMN16] Next Generation Mobile Networks (NGMN) Alliance: "Description of Network Slicing Concept," Version 1.0, Jan. 2016, Available Online:
http://www.ngmn.org/publications/technical.html

[NI] Nation Instruments, www.ni.com

[NMM14] N. Nikaein, M. Marina, S. Manickam, A. Dawson, R. Knopp and C. Bonnet, "OpenAirInterface: A flexible platform for 5G research," ACM SIGCOMM Computer Communication Review, pp. 33-38, Oct. 2014.

[NZS+16] B. Niu, Y. Zhou, H. Shah-Mansouri, V. WS Wong. "A Dynamic Resource Sharing Mechanism for Cloud Radio Access Networks," IEEE Transactions on Wireless Communications 15, no. 12, pp. 8325-8338, Dec. 2016.

[OAI] Open Air Interface, www.openairinterface.org

[OC17] J. Oueis and E. Calvanese Strinati, "Computation caching for local cloud computing," in Proc. IEEE WCNCW, TACNET workshop, May 2017.

[OCB15] J. Oueis, E. Calvanese Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in Proc. IEEE VTC Spring, May 2015.

[OCB16] J. Oueis, E. Calvanese Strinati, and S. Barbarossa, "Distributed mobile cloud computing: a multi-user clustering solution," in Proc. of IEEE ICC, May 2016.

[OCD+14] J. Oueis, E. Calvanese Strinati, A. De Domenico, and S. Barbarossa, "On the impact of backhaul network on distributed cloud computing," in Proc. IEEE WCNCW, April 2014.

[OCS+15] J. Oueis, E. Calvanese Strinati, S. Sardellitti, and S. Barbarossa, "Small cell clustering for efficient distributed fog computing: a multi-user case," in Proc. IEEE VTC Fall, Sept. 2015.

[PB] Protocol Buffers, https://developers.google.com/protocol-buffers/

[PGP+17] V. Persico, D. Grimaldi, A. Pescapè, A. Salvi, and S. Santini, "A fuzzy approach based on heterogeneous metrics for scaling out public clouds," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 8, pp. 2117–2130, Aug 2017.

[PHL+14] P. Padala, A. Holler, L. Lu, X. Zhu, A. Parikh, and M. Yechuri, "Scaling of cloud applications using machine learning," VMware Technical Journal, May 2014.

[PHT16] Pompili et al, "Elastic Resource Utilisation Framework for High Capacity and Energy Efficiency in Cloud RAN," IEEE Communications Magazine, 54(1), pp.26-32, Jan. 2016.

[Pok] Niroj Pokhrel, "Live Container Migration: Opportunities and Challenges," online https://wiki.aalto.fi/download/attachments/116662239/live-container-migration%20%281%29.pdf?version=1&modificationDate=1481801946073&api=v2

[Rap11] T. Rappaport, "Wireless Communications: Principles & Practices," Prentice Hall, 2011.

[RBB+16] P. Rost et al., "Mobile network architecture evolution toward 5G," IEEE Communications Magazine, vol. 54, no. 5, May 2016.

[RBD+14] P.Rost et al., "Cloud Technologies for Flexible 5G Radio Access Networks," vol. 52, no. 5, pp. 68-76, May 2014

[RedH] https://rhelblog.redhat.com/2016/09/26/from-checkpointrestore-to-container-migration/

[RSV15] P. Rost et al., "The complexity-rate tradeoff of centralised radio access networks," IEEE Transactions on Wireless Communications, vol. 14, no. 11, pp.6164-6176, Nov. 2015.

[SAC14] H. Shajaiah, A. Abdelhadi, and C. Clancy, "Multi-application resource allocation with users discrimination in cellular networks," In Proc. 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC), Sept. 2014.

[SB98] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," Cambridge, MA: MIT Press, 1998.

[SDK] https://docs.docker.com/develop/sdk/

[SKT14] B. Singh, K. Koufos, and O. Tirkkonen, "Co-primary inter-operator spectrum sharing using repeated games," in Proc. IEEE Int. Conf. Commun. System (ICCS'14), Nov. 2014.

[Sto05] A. Stolyar, "On the Asymptotic Optimality of the Gradient Scheduling Algorithm for Multiuser Throughput Allocation," Operations Research, vol. 53, no.1, Feb. 2005.

[SSH04] Q. H. Spencer, A. L. Swindlehurst, M. Haardt, "Zero-forcing methods for downlink spatial multiplexing in multiuser MIMO channels," IEEE Transactions on Signal Processing, vol. 52, no. 2, pp.461-471, Feb. 2004.

[UPS+05] B. Urgaonkar, G. Pacici, P. J. Shenoy, M. Spreitzer, and A. N. Tantawi. "An analytical model for multi-tier internet services and its applications," In Proc. International Conference on Measurements and Modeling of Computer Systems, June 2005.

[Wal12] B. Walder, "Cloud Architecture Patterns," O'Reilly Publications, 2012.

[WMC+17] Y. Wang, S. Meng, Y. Chen, R. Sun, X. Wang, K. Sun. "Multi-leader multi-follower Stackelberg game based dynamic resource allocation for mobile cloud computing environment," Wireless Personal Communications, vol. 93, no. 2, pp 461-480, Mar. 2017.

[WNW+16] Z. Wang, D. W. K. Ng, V. WS Wong, R. Schober, "Transmit beamforming for QoE improvement in C-RAN with mobile virtual network operators," In Proc. IEEE International Conference on Communications (ICC), May 2016.

[ZLT01] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," Eidgenössische Technische Hochschule Zurich (ETH),

Institut fur Technische Informatik und Kommunikationsnetze (TIK), Tech. Rep. TIK-Report 103, 2001.

[ZLB04] E. Zitzler, M. Laumanns, and S. Bleuler, "A tutorial on evolutionary multiobjective optimisation," Metaheurist. Multiobject. Optimis., pp. 3–37, 2004.

[ZScA] Z-Score Algorithm, Available at https://gist.github.com/ximeg/