UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

MASTER THESIS

TITLE: Modelling and managing Service-Level Agreements in the context of 5G neutral hosting platforms

MASTER DEGREE: Master's Degree in Applied Telecommunications and Engineering Management

AUTHOR: Miguel Adrián Silva Peláez

ADVISORS: Apostolos Papageorgiou
Shuaib Siddiqui

TUTOR: David Rincón

DATE: October 22, 2019

Title : Modelling and managing Service-Level Agreements in the context og 5G neutral hosting platforms

Author: Miguel Adrián Silva Peláez

Advisors: Apostolos Papageorgiou
Shuaib Siddiqui

Tutor: David Rincón

Date: October 22, 2019

ABSTRACT

This document contains the study and development of Service-Level Agreement (SLA) management mechanisms in the context of a 5G neutral host platform. The infrastructure involved in a neutral host platform is evaluated by an SLA Manager that handles the database of agreements for all the users, and verifies if the monitored data complies with the thresholds stated in the Service-Level Objectives (SLO) agreed in the SLAs.

Neutral host is a platform that has different levels of virtualization over a 5G infrastructure. It starts from a sliced network infrastructure for logic separation between tenants, which in the next level of virtualization, can host 5G services with Network Functions Virtualization (NFV) techniques. This virtual platform runs on top of a physical infrastructure that not only covers data centres like in cloud platforms, but also includes access networks, edge computing and distributed cloud elements. Evaluating through all this infrastructure adds new levels of complexity for monitoring and obtaining an accurate value for any Key Performance Indicator, or high-level parameters for Quality of Service.

This challenge is faced with a software module, called SLA Manager, which identifies the different involved infrastructure elements and creates monitoring jobs according to high-level requirements described in each SLO to obtain low-level infrastructure data. This data is then computed to obtain a high-level value to compare latter with an SLO threshold and verify if there is a violation. Availability is the main KPI on which this study focuses.

A generic SLA template body is presented for being stored in a NoSQL database solution, able to adapt to any new service deployed over new technologies that may be deployed by the neutral host, and to add flexibility and scalability to the solution.

Results show that the accuracy and reliability of the high-level objectives stated in the SLOs obey the standards required for 5G applications. The system quickly detects any outage and gives feedback to the platform to recover and avoid any violation. Delay times for detection are observed in order to provide exact measurements for availability levels. The report ends with conclusions and future development lines, as well as ethical and sustainability considerations the study involves.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 761508 (5GCity project) and the Spanish national project 5GCity (TEC2016-76795-C6-1-R).

CONTENTS

Introduction 1

CHAPTER 1. STUDY FRAMEWORK 3

1.1. 5GCity Project Framework 3

1.2. Neutral host platform and its roles 3

1.3. 5GCity architecture 4

1.4. Concepts about virtualized infrastructure 5

1.4.1. Network Functions Virtualization (NFV) 5

1.4.2. Network Slicing 5

1.5. 5GCity Orchestrator 6

1.5.1. Slice Manager 6

1.5.2. NFV Orchestrator 7

1.5.3. Resource Placement 7

1.5.4. Infrastructure Abstraction 7

1.5.5. WAN Resource Manager 7

1.5.6. SLA Manager 8

1.6. Concepts 8

1.6.1. Service-Level Agreement (SLA) 8

1.6.2. Service-Level Objective (SLO) 8

1.7. Scope of the study 9

1.8. State of the art and contributions 10

CHAPTER 2. SLA MANAGER ARCHITECTURE 13

2.1. SLA Manager 13

2.1.1. SLA Dispatcher 14

2.1.2. SLA Evaluator 14

2.1.3. SLA Repository 14

2.1.4. Monitoring Driver 14

2.2. Computing a high-level parameter 14

2.2.1. High-level parameters offered 16

2.2.2. Availability 17

2.2.3. Applying the concepts to an example . . . 19

2.3. SLA Manager Sequences . . . 21
2.3.1. SLA contract life-cycle . . . 21
2.3.2. SLO evaluation . . . 23

CHAPTER 3. SLA Manager Implementation . . . 25

3.1. SLA Manager Interface . . . 25
3.1.1. Data models . . . 25
3.1.2. RESTful Interface . . . 28
3.1.3. API resources and REST operations . . . 29

3.2. Software Stack . . . 32
3.2.1. Flask . . . 32
3.2.2. Celery . . . 33
3.2.3. Postman . . . 33
3.2.4. MongoDB . . . 34

CHAPTER 4. TEST OF THE SLA MANAGER . . . 35

4.1. Overhead introduced by the solution . . . 35

4.2. Iteration execution times increasing stored SLAs. . . . 37

Conclusions . . . 39

4.3. Suggested work . . . 40

4.4. Sustainability considerations . . . 40

4.5. Ethical considerations . . . 41

Acronyms . . . 43

Bibliography . . . 45

LIST OF FIGURES

1.1 5GCity platform roles. From [4]. 4
1.2 5GCity framework architecture. From [2]. 4
1.3 NFV specification architecture. From [3]. 5
1.4 Network slices from physical infrastructure. 6
1.5 5GCity Orchestrator platform. From [4]. 7

2.1 SLA Manager architecture. 13
2.2 VNF Forwarding Graph (VNFFG) of an NS. 18
2.3 Example of two NS onboarded over a network slice. 20
2.4 SLA Manager repository life-cycle sequence diagram. 22
2.5 SLA verification sequence diagram. 24

3.1 SLA Manager software stack implementation. 33

4.1 Time to iterate through the SLAs contained into the repository 36
4.2 Iteration time of all SLAs stored in the repository vs network elements computed. 37

LIST OF TABLES

3.1 SLA repository management operations. 29
3.1 SLA repository management operations. 30
3.2 SLA status related operations. 30
3.2 SLA status related operations. 31
3.3 Parameter document operations. 32

INTRODUCTION

The decade of the 2020's will start by introducing improved ways of building and managing networks, bringing many new possibilities and fresh business opportunities, all possible with novel 5G technologies [1]. 5GCity is a European project that announces a new network slicing management platform ready to hold 5G network services, where different tenants will be able to have their own virtual infrastructure by creating a slice with desired computing, network and access resources, on top of a neutral host. This neutral host also provides a wide catalogue of Virtual Network Functions (VNFs), letting customers to program and structure their network services over the sliced virtual environment [2].

In this context, in which many different network services will rely on the platform and its slices, is mandatory to allow Service-Level Agreements (SLA) to ensure and maintain the level of service agreed between the tenants and the platform, for all their slices and services running, and for the tenants to lease high quality products to their end-users; for the business model of the new service to be sustainable over time; and to reflect customer relationships inside the platform. SLAs have been a key tool in establishing clear contractual terms and conditions between parties, and a lot of studies have been performed about them in different platforms and contexts.

A neutral host platform pushes the virtualization boundaries of cloud platforms to new physical devices, architectures, technologies and services over slicing and Network Function Virtualization (NFV) architectures, and unveils a new set of challenges that must be overcome. First of all, given Service Level Objectives (SLO) have to be based on indicators and monitoring metrics that are now in the virtual domain, as well as in the physical domain. These virtual resources now can also be shared on-demand with different customers, as in cloud platforms, but things get more complex involving now other components as network functions, virtual links, and network access elements. Secondly, we need to determine a generic method for having SLAs that support different types of services for different applications, and to propose universal SLOs that cover all the diversity of these possible 5G services. Finally, the Slice Manager, which is the entity that provisions and allocates the slices logically on the network, needs to utilize the SLA information, as constraints for optimizing the slice resources for the different tenants.

Here is proposed an SLA Manager solution that takes monitoring metrics data from different infrastructure domains, with algorithms designed to make accurate calculations for providing reliable measures of high-level objectives agreed with the customer. It is able to support new metrics coming from new devices technologies to ensure high quality for a wide range of possible incoming 5G services. The SLA Manager output information may be used by the Slice Manager to avoid conflicts between slices, and to foster an efficient operation over virtual and non-virtual resources. Is important to recall that the solution also provides basic management over the life-cycle of SLA contracts stored in the repository, with flexible body templates and data models, and reports true violation data.

The remainder of the document is organized as follows. The first chapter presents the 5GCity project framework, with some introduction on network slicing and NFV techniques, along with some related concepts. Chapter 2 introduces to SLA related definitions with a delimitation of the study and the related work in the field. Chapter 3 focuses on the architecture of the SLA Manager, describing the internal parts and its corresponding functionalities and sequence diagrams, with some definitions about QoS parameters offered and

how they are used in the current infrastructure platform. In chapter 4, the SLA Manager implementation is described, looking into the functionalities offered through its interfaces and the software stack used. Some experiments are done in chapter 5 over the SLA Manager to test the performance and the overhead of the complexity needed to overcome the translation of low-level metrics to compute high-level parameters, showing that works in acceptable level when deployed over commodity infrastructure.

CHAPTER 1. STUDY FRAMEWORK

In this chapter, the project framework of this study is introduced, along with some basic concepts and the architecture in which the solution is inserted. Next, concepts related with Service-Level Agreements (SLA) and monitoring are described. The chapter concludes with a definition of the scope of this study and a brief review of the related work done over the same technologies.

1.1. 5GCity Project Framework

5GCity is a European funded project that focus on designing a platform that works with Network Function Virtualization (NFV) technology, specified by European Telecommunications Standards Institute (ETSI)[3], and extending it with the adoption of Network Slicing, to enable Neutral Hosting, where 5G services of different applications can be hosted on shared infrastructure [2]. This platform provides an efficient usage and allocation of virtual infrastructure over physical resources through user-friendly management for running services of any kind with the help of virtualized functions. The platform is intended to use distributed cloud computing technologies along with edge virtualization.

1.2. Neutral host platform and its roles

Neutral host platform is presented as a business model, where an infrastructure owner and a network operator are abstracted into two different roles, in other words, it can be the infrastructure provider or the network operator, or a combination of the two[2]. This neutral host provides the end-to-end connectivity in the form of network slices, which are portions of the virtualized infrastructure. The customer of the slices is known also as the slice tenant. Another important role are the developers of the virtual functions that are offered in the 5G catalog, for the slice tenants to deploy novel services.

The main roles of 5GCity platform (Figure 1.1) are:

- **Neutral Host.** The owner of the infrastructure that is used for the creation of network slices.
- **Slice Tenant.** It is any content or service provider, including virtual network operators.
- **Service Developer.** Developers of the applications and network services to be offered in the 5GCity catalog.

For the Slice Tenant is important to have some service guarantees given by the neutral host, an agreed level of quality on which it can ensure quality to its end users. For that purpose, the SLA management is a fundamental part of the business, in which different neutral hosts can be more competitive, confirming that working over virtualized resources is feasible and reliable.

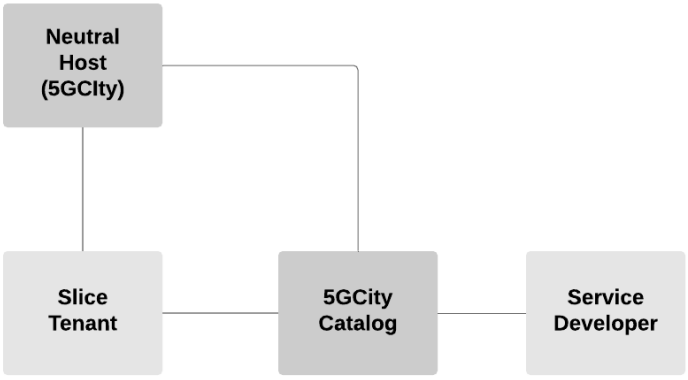


Figure 1.1 5GCity platform roles. From [4].

1.3. 5GCity architecture

5GCity platform connects the infrastructure with the users in three layers: infrastructure layer, that are the physical resources; the orchestrator and control layer, that is the management layer of all the virtualized resources; and the service layer, where the users deploy its services [2]. Inside the orchestrator and control layer, the core functionality is the orchestrator, where the slices are onboarded and the services are casted. In figure 1.2 we can see simplified view of the 5GCity platform.

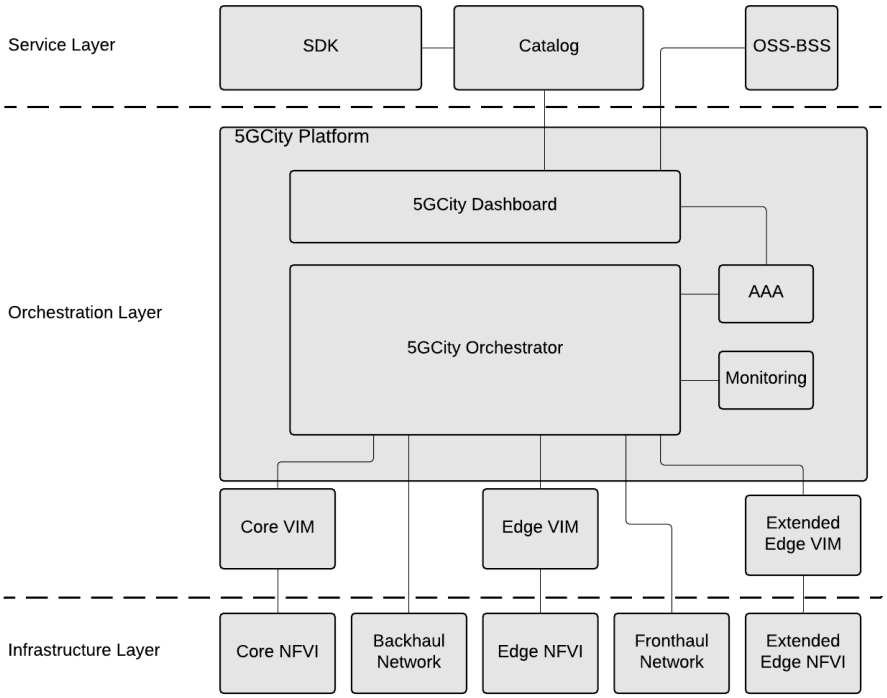


Figure 1.2 5GCity framework architecture. From [2].

1.4. Concepts about virtualized infrastructure

1.4.1. Network Functions Virtualization (NFV)

Network Functions Virtualization is an architectural framework concept that uses virtualization to take the physical appliances of the traditional network functions and implement them into software virtual machines, to group them into centralized data centres with commodity equipment [3]. This represents efficient usage of the physical resources, which involves energy, cost and space savings; network flexibility, that brings scalability, lets the network elements to be located where needed, and boosts fault resilience; and a standardized framework that simplifies operational procedures and eases maintenance.

NFV provides the users the capacity to dynamically compose network functions, that are not tied anymore to strict and static physical appliances and how they are chained. NFV enables the tools for build and manage new sets of these virtual functions, and also the implicit control and data flows. NFV will also help to deploy new services quickly and with less restriction that vendor specific hardware implies. Moreover, these functionalities are key to foster innovation, by developing new business opportunities based over new service applications that the wide possibilities of 5G will allow. In figure 1.3 is shown the architecture given by the ETSI specification [3].

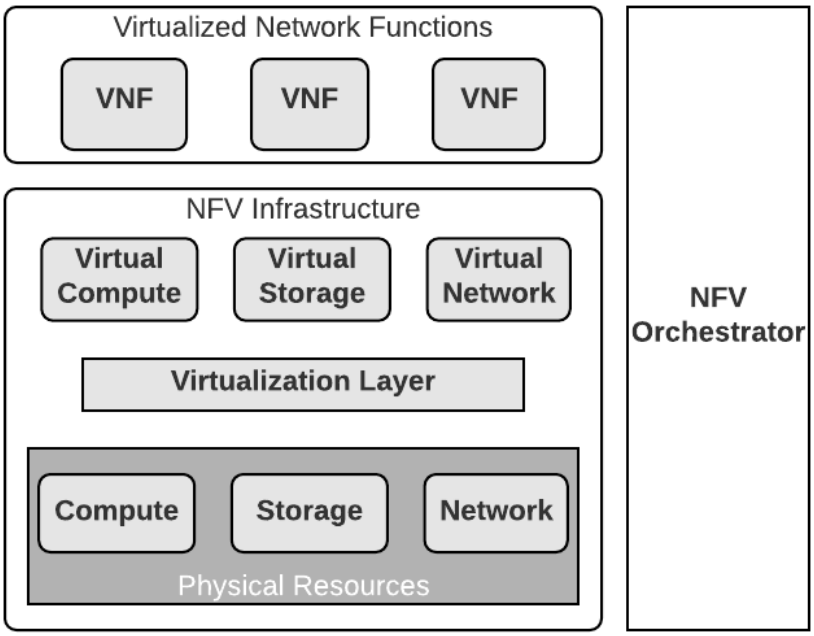


Figure 1.3 NFV specification architecture. From [3].

1.4.2. Network Slicing

A network slice is a set of virtualized resources that shares the same physical infrastructure with other slices, enabling multitenancy for service providers and virtual network operators

[2][4]. These resources are chosen by the tenant based on the required network and core capacity, number of users, geographical location, service coverage and QoS provided. A network slice is the key item provided by the neutral host operator of the 5GCity platform. Every network slice has its own network architecture, configuration, resources and security, in order to attend the -specific demand of a tenant.

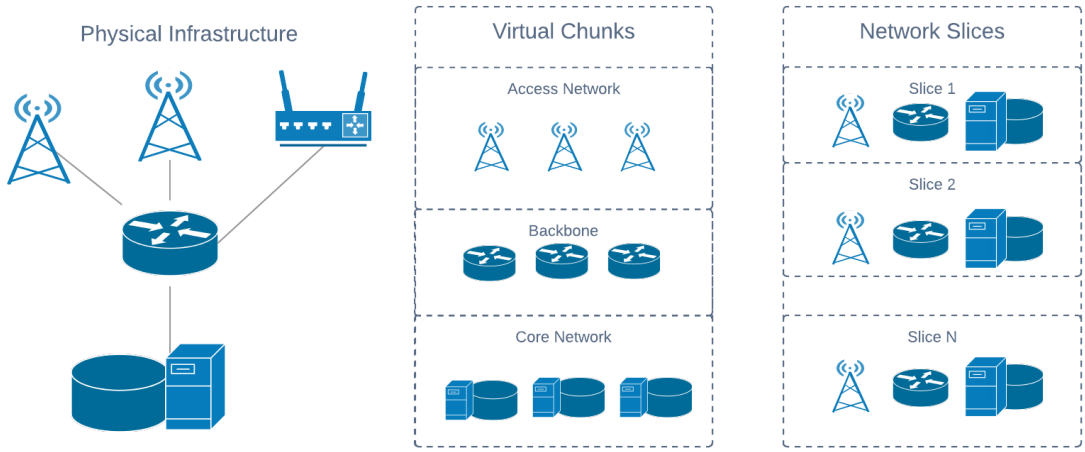


Figure 1.4 Network slices from physical infrastructure.

In figure 1.4 is shown the process of network slicing. A network slice takes the virtualized abstraction of all the physical infrastructure of a Telco operator and divide it into small virtual portions of infrastructure, or virtual chunks. These abstractions are done by multiple Virtual Infrastructure Managers (VIMs) that are interconnected and managed by an orchestrator.

1.5. 5GCity Orchestrator

In the 5GCity platform, the orchestration platform is the entity that takes care of deploy, control and manage 5G services over the virtual sliced infrastructure available [4]. It is the core component of the neutral host functionality. The orchestrator acts over the hardware resources, it abstracts virtual resources from them, controls the different slices for the different tenants and shape the new virtual end-to-end services, taking care of the network flows to connect them, the load capacity of the devices and multi-tenancy over the resources. Its architecture is illustrated in figure 1.5.

The 5GCity orchestration platform comply with ETSI NFV MANO specifications [3] [1] [5], and extends its base to adapt to the platform functionalities like neutral host and network slicing concepts.

1.5.1. Slice Manager

It allocates the network slices over the virtual and non-virtual resources, according to the desired architecture, configuration and mechanisms of the tenant. It manages the requirements during the life-cycle of the slice, keeping an efficient usage of the resources.

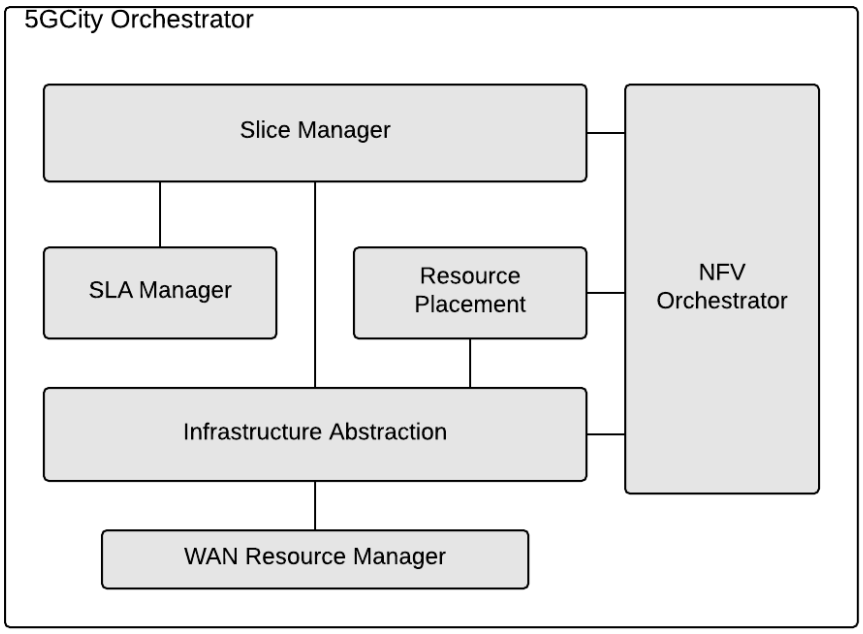


Figure 1.5 5GCity Orchestrator platform. From [4].

1.5.2. NFV Orchestrator

It is the responsible to create new Network Services (NS) and to manage the VNFs that compose them, running on cloud or edge infrastructures.

1.5.3. Resource Placement

It contains the intelligence to allocate the VNFs over the resources with optimization algorithms that tracks the usage of the resources and the traffic of the networks as input information.

1.5.4. Infrastructure Abstraction

It connects the orchestration platform with the underlying infrastructure through the different VIMs that virtualize the physical elements of the infrastructure (Core, Edge, Extended Edge) and the WAN network (Fronthaul, Backhaul) via the WAN Resource Managers

1.5.5. WAN Resource Manager

It manages the traffic of the lower layers, setting the flows to and SDN controller to the underlying physical infrastructure.

1.5.6. SLA Manager

Ensures the levels of QoS agreed between the provider and the slice owner. It is the object of this study, and will be explained through this document.

1.6. Concepts

1.6.1. Service-Level Agreement (SLA)

A Service-Level Agreement is a contract where the neutral host provider offers different levels of service for the network slices and virtual network services to the customers of the platform. There are two different types of SLA: one for network slices, and other for network services. Since network services run on top of network slices, we can say that an SLA for a network service is hierarchically below an SLA for network slice, and that an SLA for a network slice depends on the SLAs of the network services.

1.6.1.1. SLA for Network Slice

An SLA for network slices contains information about quality requirements for the infrastructure related to a slice, that is, to the virtual chunks and physical resources involved with the platform. However, since a slice is an assignation of the resources available and is not active if no service has been deployed, the SLAs for network slices are tightly related with the values generated from SLAs related with the virtual network services deployed on that slice.

1.6.1.2. SLA for Network Service (NS)

The SLA for a virtual network service contains information about the requirements of all network functions related with that network services. Those high-level requirements are translated to monitored metrics to compute a value, that will later be compared with the threshold contained in the SLO involves. The monitored infrastructure data needed is asked to the Monitoring module in monitoring jobs deployed to VNFs, PNFs and the related Forwarding Graphs (VNFFGs) and Forwarding Paths (VNFFPs) that connects them, information that is also consumed by the SLA Manager from the Network Service Descriptors (NSDs).

1.6.2. Service-Level Objective (SLO)

A Service-Level Objective (SLO) is one of the agreements of quality service that are contained in the SLA contract, in other words, is one of the service objectives that the service provider must respect and deliver to the customer. An SLA contract can have several SLOs, but it must have at least one.

An SLO is composed of three parts: the high- level parameter, the threshold and the rule. The high-level parameter is the name of the objective we are analyzing, such as availability,

latency, bandwidth or throughput. The threshold is the numerical value of reference, and it defines the level of service needed by the customer. And the rule, is the condition of comparison that, along with the threshold, determine if the measured value is within the agreed limits or not.

The SLO is also measured into a time window that is previously defined to consider the level of service and the violations raised. The time window is an important parameter since it is not the same to meet five nines of availability in a time window equal to a month than to a time window equal to a year. As an example, we have that in the following SLO:

$$\text{Availability} \geqslant 99.999\% \text{ of uptime, every month}$$

In this SLO statement, availability is the high-level parameter, the threshold is 99.999%, the rule is the condition "$\geqslant$" and time window is one month.

1.6.2.1. Monitoring job

Contains information about the infrastructure element to monitor and the monitoring metric from which to take the real time value and build a time series related with the health of that infrastructure element.

1.6.2.2. Monitoring metric

Represent the feature of the infrastructure element we are monitoring and on which we are retrieving a value from. Metrics can be numerical or labeled data and they are used to build a time series to analyze a high-level parameter contained in an SLO.

1.7. Scope of the study

In the SLA Manager studied here, we analyze the new features and particularities that the neutral host platform and business model impose on the SLA management life-cycle. However, since the study of SLA management in different platforms (such as cloud or web services) [6] [7] [8] has been performed for many different contexts and applications [9][10], it is important to define clearly the boundaries of the study. In the following list, there are some aspects of the field that are not included in this study.

- The SLA Manager is not a complete solution by itself, it relies on information provided by the Monitoring module for the metrics of the components. The Slice Manager is needed for the information related to composition and characteristics of each slice, as well as for information contained in the NSDs that allow the SLA Manager understand the deployment configuration of the NS.
- The solution of this thesis is envisioned as part of a neutral host platform business case, where the infrastructure owner is also the neutral host provider. Thus, multi-provider division or splitting of the SLA across multiple infrastructure owners as done in [11] [12] is out of the scope of this study.

- Automated negotiation of SLAs as in [13] [14] is out of the scope of this thesis. The neutral host platform provides a set of predefined high level- KPIs and boundaries from which the users can choose.
- Execution of corrective actions are not performed by the SLA Manager itself, but suggestions are sent to the Slice Manager to execute by redirecting the task to another entity of the platform. In other words, the SLA Manager does not implement any controller or actuator logic to reestablish the levels of the agreed service.
- The KPI selected for the evaluation of the SLA Manager of this study are quality related parameters. The most important is the availability parameter, but the solution can be adapted over other quality and performance related parameters such as latency, response time, throughput, and so on. Economic, energy and security are also important parameters to evaluate and needed in a neutral host, however they are left for further study.
- Although the resources and elements of the slice or network service related to an SLA that is being evaluated might change, renegotiation of terms of the SLA as in as in [15] is not supported in this design. A new SLA must be created in order to change the terms of a certain contract.
- Mechanisms of optimal levels for service offers according to infrastructure capacity as in [14] are also out of the scope of this study.

1.8. State of the art and contributions

The problem of SLA monitoring has been studied with an interesting approach in [7] where the authors develop a monitoring system to comply with the objectives in the SLA, with the help of some probe modules to measure some performance indicators, such as mean delay and packet loss rate. The study focuses on the accuracy of the data computed along a path, and although it takes into account only physical infrastructure, their study can be used for a basis for deploying such work in the virtual domains of the future networks. Translation of monitoring metrics has also been studied in [16].

The work done in [17] is deployed over SDN/NFV platforms and measures the performance metrics in a passive and active way. In a passive way, when they verify that there are not misconfigurations in the network and no suboptimal allocations of VNFs, and they do an active verification over the graph on the network. This work has been tested over more than a thousand nodes with a verification of the network below 20 milliseconds. This study contributes excellent information to the applications of SLAs in the context of neutral host platform.

SLAs over a sliced network and NFV platform has been applied in the 5G TANGO project in [18]. They propose a mechanism for template generations according to customers information and also a mapping of the low-level network related data to high-level performance parameters, that should be done by some set of predefined policies and rules that are supposed to be deployed according to the user need and, after all, are not clearly defined. In other studies of the same context [19] [19] [20], the mapping mechanism is defined

as an artificial neural network, where the inputs are the set of system policies, the high-level requirements and the low-level infrastructure data. All the knowledge collected by this mechanism will be stored into a mapping repository. This mechanism has two obvious drawbacks: first, an artificial neural network is a black box where the outputs need to be tested over reliable information that is the precise point of this study; and second, the neural network needs time to achieve optimal results, and this does not coincide with the need of onboarding new network services and a wide spectrum of 5G applications, in other words, it reduces the flexibility of the whole platform.

In the neutral host platform, to have a high-level quality requirement and to abstract it from monitored is not a straightforward task, and none of the previous studies addresses clearly. The challenge comes when there are monitoring jobs that must be deployed, retrieve the data and compute it to provide a reliable high-level value. This is a complex duty, since there is not only one domain of physical infrastructure, but a virtual one. Moreover, this virtual domain is also subdivided into sliced networking along with NFV platform, resulting to two different layers of virtualization. Besides this, the infrastructure consists not only of core cloud computing resources, but it also has edge computing, wireless access network, front-haul and back-haul inter-connectivity, which can influence the QoS of the top applications offered in the platform. In addition, obtaining an accurate and reliable value of an SLO parameter, such as availability, bandwidth or latency must be adapted to any kind of service the tenant wants to deploy, covering all the possibilities and applications that comes with 5G technologies.

We will focus our SLA Manager in the translation of the low-level monitoring metrics values to a reliable value of a high-level parameter contained in an SLO, with a generic and flexible approach. Some of our contributions are:

- Understand the infrastructure network configuration over the different layers of virtualization, leveraging on the NSD information, to decide which metrics to deploy over the NS elements running. The information is obtained to compute a high-level value that reflects the whole service.
- Develop a mathematical process to define a process for obtaining high-level parameters in function of low-level monitored data, such as it can adapt to any service configurations and be useful to apply to a wide range of high-level parameters and service applications
- Provide algorithms and endpoints that allow to compute high-level parameter values form the low-level information monitored from the infrastructure, focusing on the accuracy and reliability of the results.
- Propose data models and repositories that support flexible, generic and scalable mechanisms to adapt the SLA management services to a wide range of possible NS applications.

These contributions are aimed to lay the foundations of further studies in the field of SLA management in the context of virtualized network infrastructure, distributed cloud, edge computing and future networks in general.

CHAPTER 2. SLA MANAGER ARCHITECTURE

2.1. SLA Manager

The 5GCity SLA Manager is the module that measures the QoS levels for the different tenants. A Service-Level Agreement (SLA) is a document where the provider and the customer of a service agree the terms of the service that is being contracted.

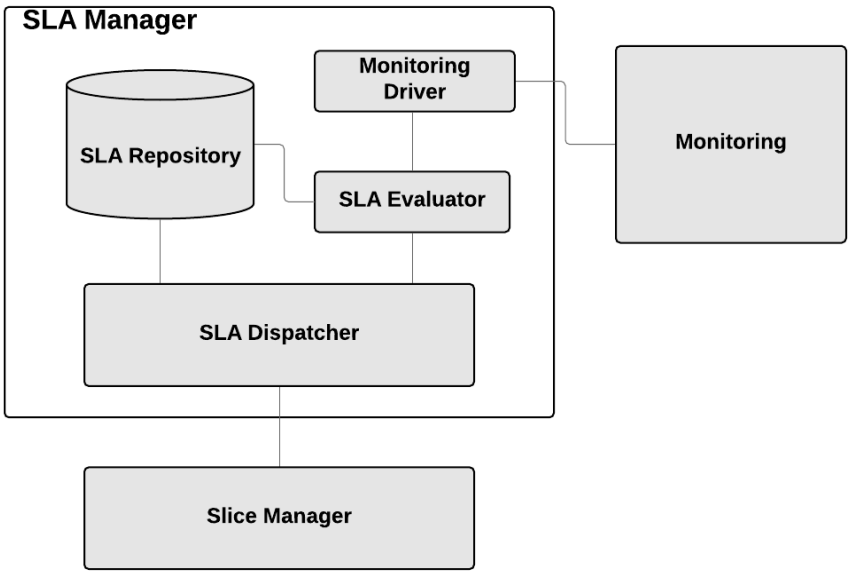


Figure 2.1 SLA Manager architecture.

In the case of the neutral host platform, the provider offers different levels of service, or Service-Level Objectives (SLOs), to the customer for their slices and network services that runs on the platform, and the customer can pick up which of those objectives adapts to the needs and deploy an SLA with the provider. If one of those objectives are not met, it is said that the service has been incurred a violation of the SLA, and this may carry some penalties that are also agreed between the parties. For the purpose of this thesis, the SLA may be viewed as a collection of SLOs, since the enforcement of reactions and the handling of penalties are out of scope.

The SLA Manager takes care of the following functionalities:

- Store and manage the life-cycle of the SLA for each customer of the neutral host.
- Deploy a periodical evaluation of the SLOs for each active SLA in the repository.
- Set up monitoring jobs to take values from the infrastructure related with the slices or network services deployed with the SLA.
- Compute a reliable value for the SLO high level parameter from the low level monitored values.

- Detect the source of the failure and suggest a proper action to be performed by the corresponding module of the orchestration platform.

Different software modules constitute the SLA Manager for performing the tasks involved, and can be observed in figure 2.1

2.1.1. SLA Dispatcher

It contains the API interface to connect with the other modules of the orchestration. Organizes the tasks between the modules of the SLA Manager, like storing the information, detecting breaches from the monitoring and notifying when there are breaches and violations to the service, as well as suggesting accurate actions to recover the health of the system.

2.1.2. SLA Evaluator

It evaluates the level of QoS of the all the SLOs of the active SLAs. Also, it contains the intelligence that relates the monitored low-level metrics with the high-level parameters contained in the SLOs.

2.1.3. SLA Repository

It stores the SLAs, the status and the violations of each one of them.

2.1.4. Monitoring Driver

It connects with the Monitoring module of the 5GCity platform to receive real time values of the deployed monitoring jobs.

2.2. Computing a high-level parameter

The neutral host platform will come to perform a key role in the deployment of many different services for vertical applications, leasing resources for any new business that wants to leverage its connectivity over the power of a 5G infrastructure. In that context, the SLA Manager is designed to cover an heterogeneous arrangement of many NFs deployed from the catalog of an SDK, placed to adapt to the needs of the service. Those NFs are related to different locations, technologies, and from virtualized and physical devices. The method is developed in order to ease the creation and management of SLAs for any new configuration, letting the tenant free to work with recent developed VNFs, advanced technologies and creative applications.

In the design proposed in this study, an SLA is seen as a collection of SLOs, where a tenant introduces each SLO with the required levels of service for a certain high-level parameter. The high-level parameter is introduced into the SLA Manager as an input to

compute a value for it. To correctly monitor a neutral host platform and to decompose the layers of virtualization of the neutral host, the SLA Manager enables two types of SLAs: one is related with the slice, and other is related with the deployed NSs over that slice. For a slice, computing a high-level value from monitored data is not a straightforward task, since a slice is a portion of infrastructure that is not active until an NS is on-boarded, so it is measured in terms of a formula that relates the high-level values computed for all NSs running over that slice. On the next level, for an NS, it is possible to establish a high-level parameter value from the monitored data, developing also a formula that relates it with the high-level value for that specific NS.

The formula to evaluate an SLO is specific for each high-level parameter, for the type of SLA it evaluates, and even for any application. For an NS, it is instantiated in terms of defined monitoring metrics taken from the different types of network elements that constitute the NS, such VNFs, PNFs, VLs, and so on. For a slice, the formula is instantiated in terms of the high-level values computed for the NSs related to that slice. The formula is saved on the SLA Repository in a data model called parameter document (see section 3.1.1.4.) and is stored in an independent process, with special functionalities available for developers (see section 3.1.3.3.).

Over a single network slice it runs N number of NSs. A slice then can be seen as a set of NSs:

$$Slice = \{NS_n : n \in N\} \tag{2.1}$$

Let x be the type of SLA the neutral host is offering. The SLA is then a set of SLO_x of a certain high-level parameter $i \in I$, where I is the set of all the high-level parameters described in the SLOs.

$$SLA_x = \{SLO_x(i) : i \in I\} \tag{2.2}$$

Let $SLO_x(i)$ be the value of the high-level parameter i for the SLA of type x. The high-level parameter value of the slice $SLO_{slice}(i)$ is then a function f of the high-level parameter values of all the deployed NS for the same SLO $SLO_{NS}(i)$. If we take the attention over a single SLO for a network slice, related with a high-level parameter i, and we take equation (2.1), we have:

$$SLO_{slice}(i) = f(\{SLO_{NS_n}(i) : i \in I, n \in N\} \tag{2.3}$$

For any $SLO_{NS}(i)$, its value is a function of the monitored data collected from the network elements that takes part into the topology of the NS. An NS is described in a document called Network Service Descriptor (NSD) that includes all the network elements that constitutes the NS (VNF, PNF, VL), how they are connected (VNFFG) and how the flows are directed (VNFFP) [5]. This document is used by the SLA Manager to extract information about the NS to deploy monitoring jobs with metrics that will collect the data from the network elements to compute a high-level value i for a given SLO of an NS. The formula to compute that high-level value requires different metrics to be deployed over different network elements.

Let NSD_n be the NSD related to NS_n, and $l \in L$ be the type of network element described in the NSD, being L the set of network element types (e.g. $L = [VNF, PNF, \ldots]$. Finally,

let Q_l the number of network elements of type l deployed on the NS. Thus, the extracted information from the NSD_n can be expressed as:

$$NSD_n = \{l_{q_{(NS_n)}} : q \in Q_{l(NS_n)},\ l \in L_n\} \tag{2.4}$$

The monitoring jobs collected for computing a high-level value are a list of metrics needed for extracting data for different elements. Let m be a metric included the set of M_l metrics collected from network element l.

$$J_n(i) = \{m(l_{q_{(NS_n)}}) : q \in Q_{l(NS_n)},\ l \in L_n\ \forall\ m \in M_l(i)\} \tag{2.5}$$

With all the monitored data collected. The value for the SLO of NS_n for high-level parameter i is a function g of that data. It is given by:

$$SLO_{NS_n}(i) = g(J_n(i)) \tag{2.6}$$

The formulas f and g from equation (2.3) and (2.6) respectively are stored in the parameter document (see section 3.1.1.4.) of each SLA type, and also the number of metrics M to collect data from network elements of type l from the NS. The information about the network elements like the amount Q_l of network elements of type l in the NS is available in its corresponding NSD.

After computing the value for $SLO_x(i)$, the system takes the rule and the threshold contained in the SLO to evaluate if there has been a violation of the agreement.

2.2.1. High-level parameters offered

According to the ETSI specification [21], the performance metrics can be categorized in Quality of Service (QoS) based metrics and Quality of Experience (QoE) based metrics. QoE based metrics are important to capture the subjective aspects of the service related to human experience, however, we need to maintain simplicity in the first stages of this project, we will start with QoS based metrics, letting QoE based metrics support to be added in further work. QoS metrics are the core network performance metrics based on network measurements to control the quality level of applications and services. Essentially the QoS metrics tell us the network impact on the quality of services, and is the first step to latter study other performance aspects that influence on the levels of service given to the customers.

According to the recommendations of the International Telecommunication Union (ITU), Quality of Service (QoS) is referred as the effect of service performance that affects the satisfaction of the user of a certain service [22]. According to the ETSI specification, the quality in the service area has different categories, each one with the respective measurement method. In the first one, we have the quality related to the reliability of the service; the second one, related to the service provision and is independent of the kind of service between the supplier and the customer; and, finally, we have the quality measured according to the subjective perception of the customer. In 5GCity, the first two categories mentioned are feasible to express in an explicit and numerical way for both the customer and supplier to agree and understand. The subjective measures of quality of service are

not taken into account in the development of this study, since they are estimated through surveys and polls applied to a large amount of people, that could be the end users of the service.

We will focus on the study of availability, since it is a key performance indicator that is relevant for most 5G services, including the services required for the Use Cases of the 5GCity project. For any tenant, having access to the platform that has been leased for deploying its services, and in the same way, give the service offered to an end-user at every moment is a basic feature for almost any kind of application. Availability has a different meaning for slices and for virtual NS, but they affect each other in particular ways that we will examine below.

2.2.2. Availability

In generic terms, availability is the proportion of time in which a system is capable of being operational and useful to its users, and is given as the percentage of uptime over the total time measured, i.e. the time window. According to [23], availability is the total uptime at service level in a given time window, and is expressed as follows:

$$Availability = \int_0^{\tau} a_x(t)dt \tag{2.7}$$

Where $a_x(t)$ is the instantaneous availability $a \in 0,1$ measured at time t for an SLA of type x. In the telecommunications industry, a common value for availability is the 5 nines percentage, that is, an uptime of 99,999% of the time window established [22]. This is a time that considers the downtime that are not related with a system outage, but with activities related with maintenance or migration of equipment.

Availability of a system is a parameter dependent on the availability of its constituent parts [21], so for having a reliable measure of that uptime as an automated task, it has to be evaluated as the availability of each of the parts that compose the system. There are some parts that are critical to the performance of the whole system, as there are some parts that are less important and any problem related to them is not so crucial to the whole system operation, so the root of failure has to be distinguished to give an optimal estimation of the availability of the system. In the SLA Manager, the availability related to any deployed NS will be estimated in a different way to the availability associated to a network slice, since they have different constitution and technology. However, both availability estimations are tightly related to each other.

2.2.2.1. Availability in Network Services

An NS is created by assembling a group of VNFs available at the catalog to compose a VNFFG that will fulfill the requirements of a certain application. The VNFFG can also be composed of PNFs in its chain. In the VNFFG the user can also define the VNFFPs to which the flows of information can be directed. In figure 2.2 there are three VNFFPs in which the information can flow. For VNFFP-1, VNF 4 is not a critical node, and its outage will not affect the performance of the service flowing through this path. The same goes for VNF 3, where the VNF is replicated for HA purposes and an outage of any of them will not

affect the availability of the NS. All of these considerations has to be taken into account to measure a value for instantaneous availability. In our study, however, this analysis are left for further study. In our case, the instantaneous availability is affected by the outage of any NF, in other words, all of them are critical points for the NS, assuming that any outage on any element of the NS will be considered as an outage of the whole chain.

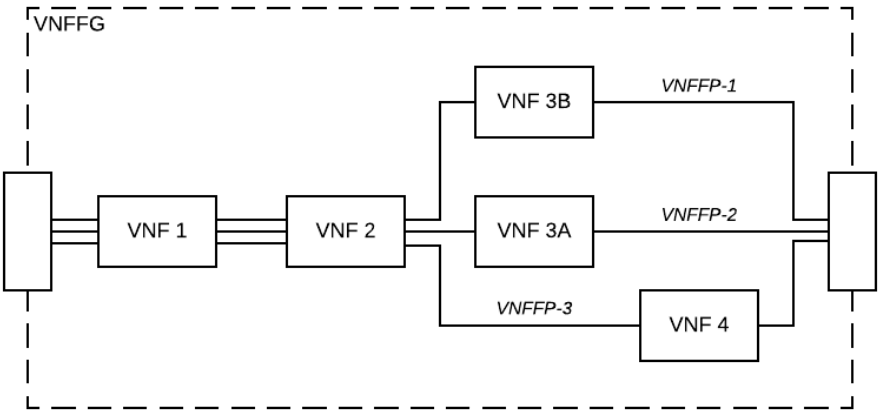


Figure 2.2 VNF Forwarding Graph (VNFFG) of an NS.

Let $a_{NS}(t)$ be the instantaneous availability of an NS at time t. For the availability compute we need a single metric to be deployed over the network elements, which is the up metric, a binary value that is 1 when the network element is healthy and 0 when it has an outage. The instantaneous availability for NS can be expressed as:

$$a_{NS}(t) = \prod_{0}^{Q_l} l_{q(NS_n)} \tag{2.8}$$

Recall that the value of instantaneous availability is a binary value that is 1 when all the network elements are up, and 0 if just one of those elements are down. To compute the value of availability over a time window, let A_{NS} be the availability or total uptime over a time window τ, and expressed as follows:

$$A_{NS} = \int_0^{\tau} \prod_{0}^{Q_l} l_{q(NS_n)} dt \tag{2.9}$$

Which will be a value between 0 and 1, representing the percentage of uptime of the NS over a predefined time window.

2.2.2.2. Availability in Network Slices

In addition to the network services and the other NFV elements, a network slice represents also the infrastructure in which the tenant may deploy any 5G service composed with the help of the VNFs from the catalog. In that way, we could say that the slice itself stays inactive if there are no deployed NSs on top of it. The availability of a network slice cannot be measured until an NS happens to be onboarded and running. Thus, we can compute

the availability of the network slice in terms of the NS running over it, and to not take any infrastructure monitored metric to compute it.

A downtime in any of the deployed NS over a network slice does not necessarily represent that the whole network slice has been out of service, because other NS can still be running over the slice. We need all the NS of a slice to be unavailable to state that there is a service breach at the slice level. Hence, there is a parallel composition of the NS values of availability to determine the value of the availability of a network slice, and its instantaneous value $a_{Slice}(t)$ can be expressed as follows:

$$a_{Slice}(t) = 1 - ((1 - a_{NS_1})x(1 - a_{NS_2})x \ldots (1 - a_{NS_n})) \tag{2.10}$$

Being the result a binary value that is equal to 1 when all the NS of that slice are down, and 1 when at least 1 of them is up at moment t. Let A_{Slice} be the availability measured over a time window τ, then:

$$A_{Slice} = \int_0^{\tau} (1 - ((1 - a_{NS_1})x(1 - a_{NS_2})x \ldots (1 - a_{NS_n}))) \, dt \tag{2.11}$$

Note that the availability of the slice will be quite high if there are multiple service instances with sufficiently high individual values of availability running on it, since it needs all NS to be out at least one time to decay the value of slice availability, and also that a downtime of the whole infrastructure of the slice will represent an outage on all of the NSs of the slice and be represented accurately by the equation.

2.2.3. Applying the concepts to an example

To validate the equations of the previous sections and see how the concept method is applied, we apply them to a simple example, illustrated in figure 2.3. We will suppose a slice with two NSs onboarded, NS_1 and NS_2, and we will create three SLA, one for each NS and one more for the slice with a single SLO where the $I = [availability]$. For NS the availability relies in only one metric, that is the availability of each NF. For NS_1 we have 2 VNFs and 2 PNFs and for NS_2 we have 3 VNFs.

First, we apply equation (2.1) to relate the NSs with the slice and we obtain:

$$Slice = \{NS_1, NS_2\}$$

Next, proceed to relate all the SLAs as a collection of their SLOs, according to equation (2.2). In this case, we have only one SLO per SLA, and the set I is included of just one high-level parameter i that is $Availability$ ($I = [Availability]$). Replacing all the values, we obtain the three following expressions:

$$SLA_{Slice} = \{SLO_{Slice}(Availability)\}$$

$$SLA_{NS_1} = \{SLO_{NS_1}(Availability)\}$$

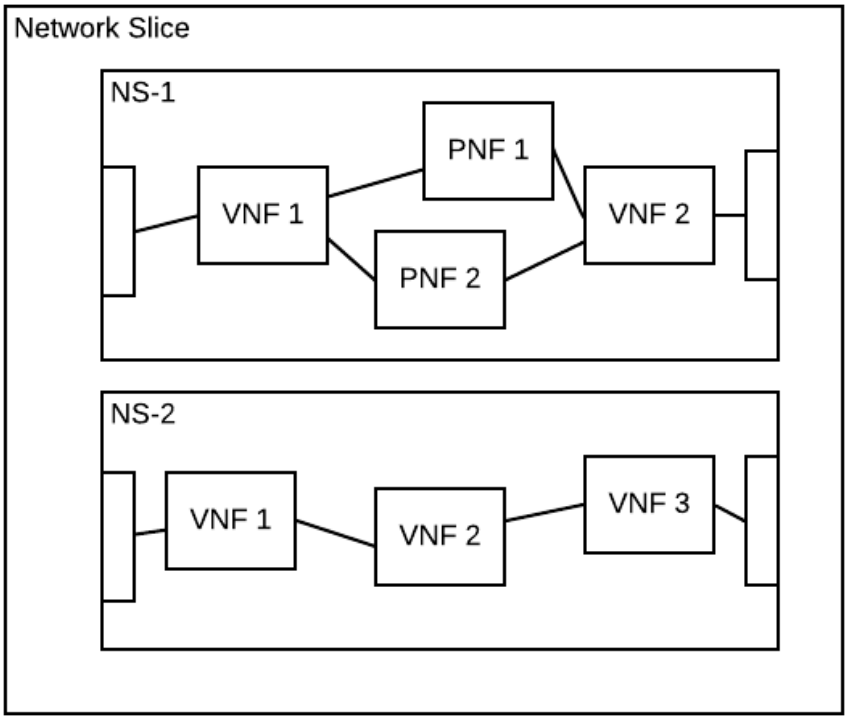


Figure 2.3 Example of two NS onboarded over a network slice.

$$SLA_{NS_2} = \{SLO_{NS_2}(Availability)\}$$

Going one step deeper, the SLO_{Slice} for availability parameter is defined in terms of the SLO of the running NSs for that same parameter, as stated in equation (2.3).

$$SLO_{Slice}(Availability) = f(\{SLO_{NS_1}(Availability), SLO_{NS_2}(Availability)\})$$

The SLA Manager then extract the data needed from the NSD about the network elements according to equation (2.4). Here we have two types of network elements for NSD_1, so $L_1 = [VNF, PNF]$, and its respective quantities $Q_{NS_1} = [2,2]$. For NSD_2 we have just one type of network element $L_2 = [VNF]$ and the number of network elements are $Q_{NS_2} = [3]$. Replacing all the values we obtain:

$$NSD_1 = \{VNF_{1(NS_1)}, VNF_{2(NS_1)}, PNF_{1(NS_1)}, PNF_{2(NS_1)}\}$$

.

$$NSD_2 = \{VNF_{1(NS_2)}, VNF_{2(NS_2)}, VNF_{3(NS_2)}\}$$

The data of the monitoring jobs for the single metric up to be extracted from the network elements is described applying equation (2.5) in the following sets:

$$J_1(Availability) = \{up(VNF_{1(NS_1)}), up(VNF_{2(NS_1)}), up(PNF_{1(NS_1)}), up(PNF_{2(NS_1)})\}$$

$$J_2(Availability) = \{up(VNF_{1(NS_1)}), up(VNF_{2(NS_1)}), up(VNF_{3(NS_2)})\}$$

Finally, we can compute the high level-value of availability for the SLOs of the NSs, according to equation (2.6), for any time window τ, with the following expressions:

$$SLO_{NS_1}(Availability) = \int_0^{\tau} (up(VNF_{1(NS_1)}), up(VNF_{2(NS_1)}), up(PNF_{1(NS_1)}), up(PNF_{2(NS_1)}))$$

$$SLO_{NS_2}(Availability) = \int_0^{\tau} (up(VNF_{1(NS_1)}), up(VNF_{2(NS_1)}), up(VNF_{3(NS_2)}))$$

As we can see, the composition can be done with any formula that relates a group of monitoring metrics collected from define network elements of a NS. That formula adapts to the particular configuration of the NS in terms of types of network elements and their respective quantities. This formula approach can also scale easily over NS with high amount of network elements.

2.3. SLA Manager Sequences

Here we use sequence diagrams to point the data flows between each entity of the SLA Manage, describing the steps for the processes of SLA life-cycle management and the SLO evaluation.

2.3.1. SLA contract life-cycle

When a slice is created, or a new NS is deployed, the user can create an SLA with the parameters chosen via the dashboard of the platform (figure 2.4). This information is sent to the Slice Manager and then passed to the SLA Manager. In the SLA Manager, the SLA Dispatcher checks if the SLA already has an SLA with the same identification fields to avoid duplication and validates the data sent, notifying when there is validation problem in the data fields. If the data is validated, it is stored in the repository, saving also the SLOs and the parameters contained in the SLA into their respective collections. There are a collection for SLOs for monitoring jobs onboarding processes, and a collection for parameters to store new mapping formulas related to new high-level performance parameters, or parameters applied to different infrastructure or service application.

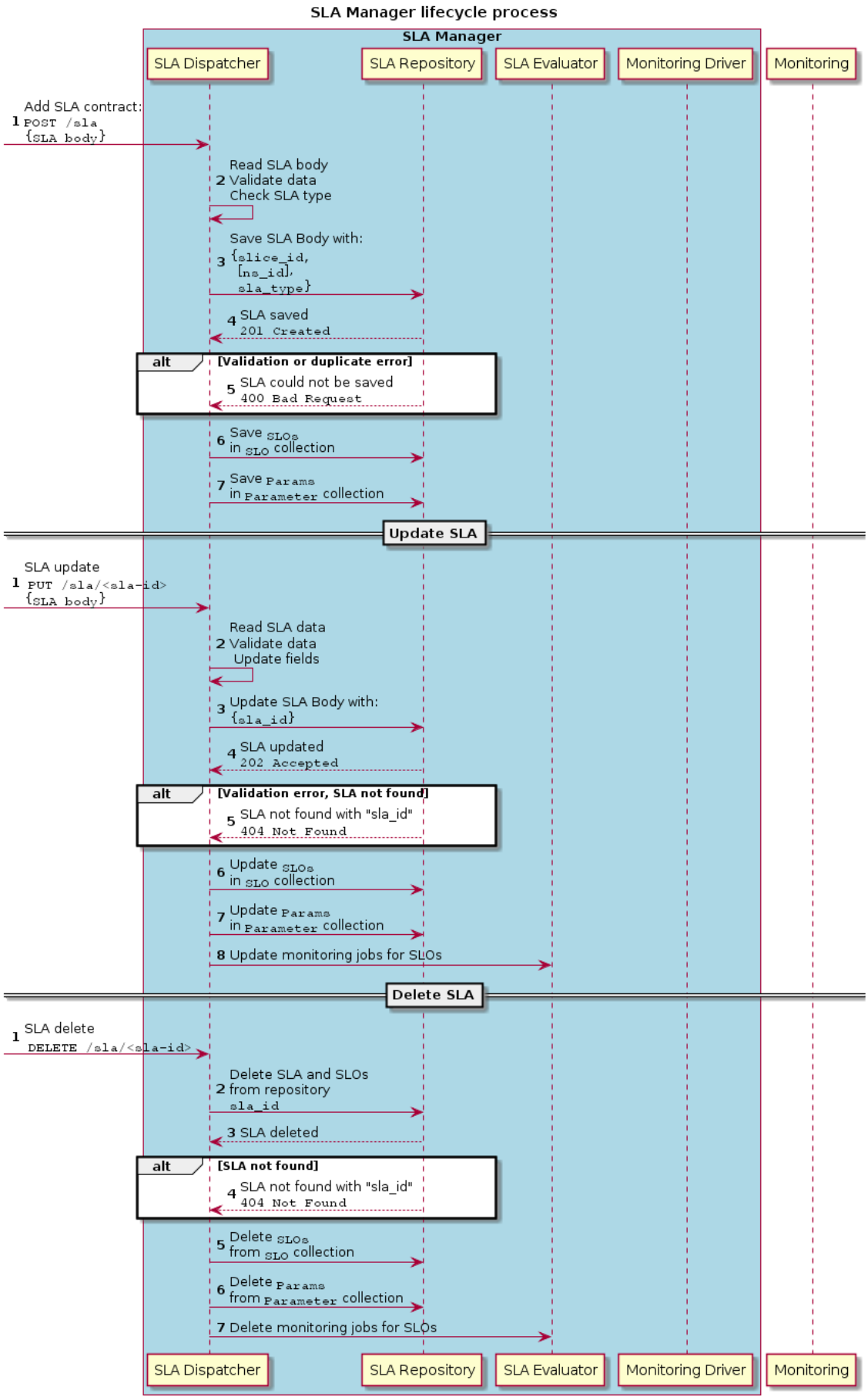


Figure 2.4 SLA Manager repository life-cycle sequence diagram.

When there is a need to update some of the information fields of the SLA contract, change some threshold levels in some SLO, or update some formula in a parameter document, the SLA Manager receives the information in the same body template, validates the data and compares the information that has changed for performing the update. The corresponding repository is then updated and the monitoring jobs that have changed are also updated, removing the old ones and creating new ones.

For deleting and SLA, the information is deleted from all the repository collections and the monitoring jobs removed.

2.3.2. SLO evaluation

SLO evaluation is the task referred to the monitoring of the infrastructure to observe its performance and quantify the levels of service the provider is giving. The result of this task can be available to both customer and provider in the form of the high-level parameters specified in the SLOs, whether or not there has been a violation.

The monitoring module of the 5GCity platform provides the information required by every SLO in every SLA contract. According to the high-level parameter and its corresponding mapping formula, the low-level metrics required are sent to the monitoring module and pointed to the infrastructure elements related with the SLA contract to create monitoring jobs that are periodically updated. Although this period has to be defined in the monitoring module, its value is desired to be minimum to reflect real-time values, but also balanced, for the monitoring task to not represent a high processing weight on the physical infrastructure and to not introduce overhead or interference on the main networking tasks.

The sequence diagram of the evaluation task is shown in Figure 2.5. A periodical task for verification of the SLA contracts stored in the repository is performed in the background by the SLA Evaluator. All the SLAs currently stored in the repository are queried to iterate and verify each SLO from every single SLA. The NSD is also queried from the repository to list all the infrastructure elements related to an SLA. In every SLO, the parameter document is queried to the parameter collection with the parameter name stored in the SLO in order to extract the mapping formula of that parameter. The low-level infrastructure metrics needed to compute the formula are applied to every infrastructure element listed in the NSD for the Monitoring Driver to check the monitoring jobs that will are instantiated in the Monitoring module of the platform. The system also checks if the monitoring jobs are created previously for newly created SLAs. The Monitoring Driver receives the monitored values of the jobs related with the SLO and compute the mapping formula to obtain a value of the high-level parameter of the SLO. This value is then used to be compared with the rule and threshold information in the SLO to check if there is a violation. The SLO Evaluator stores the violation along with the timestamp for that SLA and high-level parameter of the SLO in the violation's repository.

The SLO evaluator takes the data of all the SLA contracts and their SLOs, and reads again in every single iteration, to make sure that it takes all the new SLAs saved, removed or updated, as well as their respective SLOs and additional information, to proceed with the verification over recently updated data and keep track of the current events of the infrastructure.

Parameters collections has been created to store the mapping formula for a certain high-

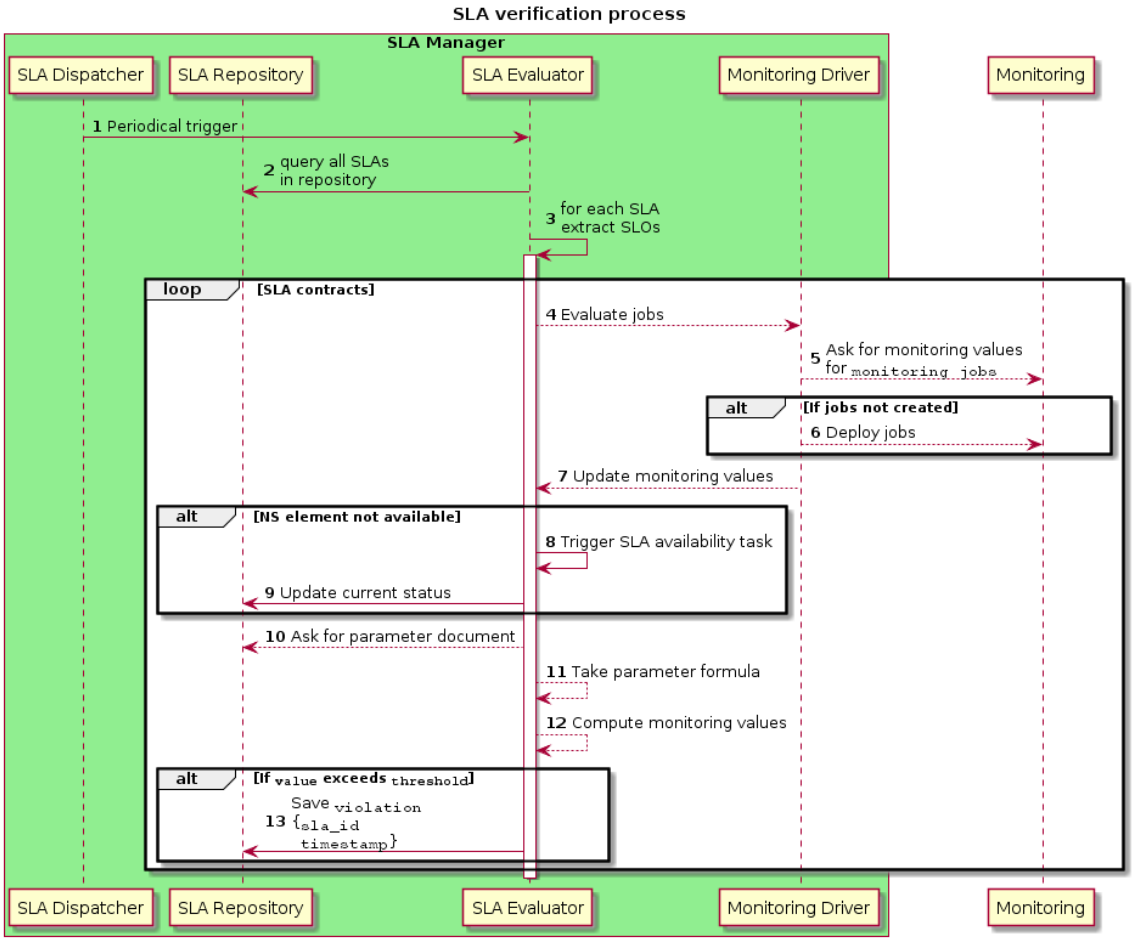


Figure 2.5 SLA verification sequence diagram.

level parameter with respect of the monitoring low-level metrics of the virtual and non-virtual infrastructure it runs over. This helps the SLA Manager to be more flexible because any future new service that runs over newer network designs, with new VNF configurations and over new hardware technology can still be handled.

Note that availability process verification will always be computing periodically in parallel with any other high-level performance parameter verification, because losing availability means losing also decreasing service levels on any other performance. The infrastructure first must be available in order to perform another task.

CHAPTER 3. SLA MANAGER IMPLEMENTATION

3.1. SLA Manager Interface

3.1.1. Data models

For this study, we will use JSON format to manage all the data that the SLA Manager will receive and yield. This format was chosen because it easily adapts to No-SQL database solution that we use to manage the repository of the SLA Manager (section 3.2.). The data that is inserted in the SLA Manager need to be validated previously to prevent any data corruption or error that can harm its normal performance. These data models are in the form of a JSON body. Each JSON body is mapped with its own database collection.

It is important to recall that JSON format and No-SQL database allow the data models to be extended and add new fields for new functionalities. In other words, can be extended to adapt to new services or templates in the future, as well as new functionalities related with SLA Management like negotiations, penalties, auto-templating and so on.

3.1.1.1. SLA contract

The SLA contract is posted and saved in the repository. The data model, shown in data model 3.1, maintains its simplicity by giving labelling the contract with an unique identifier, that can also be saved in the form of a string value, a field to state which type of SLA is the one that has been saved for further processing of its SLOs, and two fields to identify to which slice and NS it belongs. The NS field is optional and only used in NS type SLAs. The customer and provider information are given in a generic data model for company information. The field for SLOs represent a list, where the data model for SLOs are added according to the ones agreed. There is also a field that saves the timestamp when it was created.

```
"sla":
  {
    "sla_id": string,
    "sla_type": string,
    "slice_id": string,
    "ns_id": string,
    "valid_from": datetime,
    "valid_until": datetime,

    "slos": [slo_1, slo_2 ... slo_n],

    "customer_info": company_info,
    "provider_info": company_info,
    "date_created": timestamp
  }
```

Data Model 3.1 Template for SLA body.

3.1.1.2. Company information

The generic data of the parties involved in an SLA are saved in a data model for saving company related data. This generic model is proposed (data model 3.2) for being reused in other SLA contracts and to avoid duplication of company information and optimize database memory.

```
"company_info":
  {
    "company_id": string,
    "username": string,
    "first_name": string,
    "last_name": datetime,
    "email": mail,
    "phone": string,
  }
```

Data Model 3.2 Company information body.

3.1.1.3. SLO

The SLO data model stores the information related with any SLO and has its own repository to make easy the updates of the SLO information. As can be seen in data model 3.3, the SLO is identified with the SLA contract it belongs to and the high-level parameter is saved with the rule for comparison and the threshold value. A priority field is instantiated in the case of a quicker SLO evaluation for high reliability applications.

```
"slo":
  {
    "sla_id": string,
    "parameter": string,
    "rule": string,
    "threshold": float,
    "priority": string
  }
```

Data Model 3.3 Service-Level Objective body.

3.1.1.4. Parameter document

The parameter document is an independent data model and its template is presented on data model 3.4. It saves the low-level to high-level mapping information to be computed for obtaining an SLO value from the monitored infrastructure metrics. It has its own repository for being reused on all the SLAs that have the same SLA type and same application. The parameter ID field is there to differentiate the same parameter for different applications, with different infrastructure elements. The parameter is specified also for each SLA type available and the formula is a string value that is stored to relate the values provided by the

Monitoring Driver to the high-level parameter. There are some fields called "$\backslash monitor_to$" and "metrics" that stores information used to create the monitoring jobs needed to obtain the low-level data.

```
"parameter":
  {
    "parameter_id": string,
    "parameter_name": string,
    "sla_type": [ slice, ns],
    "monitor_to": string,
    "metrics": string,
    "formula": string,
    "unit": string
  }
```

Data Model 3.4 Parameter document body.

3.1.1.5. Status

To register the health of an SLA contract there is a data model called status (data model 3.5). It is labelled with the SLA identifier, the timestamp of its creation, and it lists the health of every SLO contained in the SLA contract. The health is described by the current status with respect to the current time window and a field of the action done to recover from the service level degradation.

```
"status":
  {
    "sla_id": string,
    "timestamp": timestamp,

    "slos": [
        "slo_1":
        {
            "parameter": string,
            "status": [NORMAL, WARNING, VIOLATION],
            "reaction": string
         },
         ....
        ]
  }
```

Data Model 3.5 Status document body.

3.1.1.6. Current

The SLA Manager does not push any information into the other entities of the platform, in order to prevent any bidirectionalities that can incur in conflicts and struggle the simplicity

of the modular paradigm. For that purpose, the SLA Manager continuously refresh a data model in where it notifies the Slice Manager with the events currently happening in the infrastructure, listing all the urgencies that need to be solved. If the list is empty, means that the whole virtual and non-virtual infrastructure is healthy. It is important to discern that the information offered here does not necessarily mean that there is a violation, but that there is a service degradation in some aspect that needs to be solved before the SLA incurs into a violation.

```
"current":
    {
    "last_refresh": timestamp,

    "breaches": [
            {
                "sla_id": string,
                "infrastructure": [string, ...],
                "reaction_advice": string
            },
            ....
        ]
    }
```

Data Model 3.6 Current data body to notify events .

The SLA Manager offers this data model as a return value in the /current endpoint of the API interface, and is structure is presented in data model 3.6.

3.1.2. RESTful Interface

The SLA Manager is connected to the platform in the form of a RESTful Application Programming Interface (API). REST stands for Representational State Transfer and is a software architecture designed by Roy Fielding in [24]. It is well known and used in the software industry, because it fosters horizontal, modular development of big projects and is platform and language independent.

A RESTful API is accessed through an HTTP interface to navigate through the processed data of the API module. The client can access all the resources offered by the module with URL addresses and it uses HTTP methods to define specific actions over them like GET, POST, PUT and DELETE methods that are the ones used in this document.

This software architecture is introduced in the SLA Manager because its simplicity at the moment of processing the transactions of data, having small overhead because it just addresses simple HTTP messages, and because it provides the developer the freedom to choose any programming language for the software implementation, and the client does not need to have external libraries to support the communication with the RESTful API. The SLA Manager then uses the RESTful architecture to easily adapt with other software modules in the orchestration platform and to give access to all the resources that are stored and computed by its internal composition and logic. In its RESTful interface, the SLA Manager has the following functionalities:

- Provides access to its repository, in the form of the SLA contracts, and its corresponding SLOs and status, as well as the violations happened in the time window defined. This information is available for both provider and customer of the platform to see at which level the service agreed has been met and establish the respective penalties or future credits for the customer when a violation happened, or in the opposite scenario, show statistical data of success that helps the provider grow good reputation over the industry.
- Receives information to store inside the repository, in the form of a new SLA contract or an update to the information, SLOs or parameter documents of an existing one.
- Shows the real time status of the infrastructure related with some SLA contract for the other entities to take the corresponding actions in case of a problem. It suggests also corrective actions in case of a service breach detected by the unavailability or degradation in the levels of a performance indicator to prevent a violation.

All these functionalities are offered to the other entities of the orchestration platform. More specifically, The Slice Manager is the entity that acts as a client of the SLA Manager RESTful interface and also communicates the SLA Manager with the other entities of the orchestration platform, by passing the information in, and spreading the useful data that is the result of the output of the SLA Manager. Is important to recall that the SLA Manager itself also acts as a client of the Monitoring module RESTful interface for processing and evaluating the SLO levels.

3.1.3. API resources and REST operations

We have defined some resources that are going to be explained in the following subsections form of REST operations, containing the URL resources with their specific HTTP methods and the expected result. In addition, there are some data parameters expected to be attached when inserting data to the SLA Manager that are also shown. The parameter can be any data body or the identifier written in the URL address (e.g. `sla_id`).

All the data that is inserted for processing and the results retrieved are in the form of JSON data body. This data form eases its processing for a wide range on programming languages, with fast object deserialization and concise format. For the SLA contracts, SLOs, parameter documents and status objects, a JSON body data model is designed for each one to be validated by the SLA Manager. These models are discussed in the section 3.1.1.

3.1.3.1. SLA contracts life-cycle

Are all the REST operations that are related with the management of the SLA Repository and are shown in table 3.1.

Table 3.1 SLA repository management operations.

Method	Resource	Parameters	Description	Response
SLA contracts lifecycle				

Continued on next page

Table 3.1 SLA repository management operations.

Method	Resource	Parameters	Description	Response
POST	/sla	sla, nds_id	Register an SLA into the repository.	*202 (Accepted). 400 (Bad Request)*: Validation Data Error. *409 (Conflict)*: Duplication Data Error.
GET	/sla	-	Retrieve all the SLAs in the repository.	*200 (Success)*
GET	/sla/<sla_id>	sla_id	Retrieve a single SLA identified by sla_id.	*200 (Success). 404 (Not Found)*: SLA not found.
PUT	/sla/<sla_id>	sla_id, nds_id	Update an SLA identified by sla_id.	*202 (Accepted). 400 (Bad Request)*: Validation Data Error. *409 (Conflict)*: Duplication Data Error.
DELETE	/sla/<sla_id>	sla_id	Delete an SLA identified by sla_id.	*200 (Success). 404 (Not Found)*: SLA not found.

The SLA Manager receives the information and notifies when all the data has been validated. In the $/sla$ posting action and when updating (PUT) in $/sla/ < sla\_id >$, the parameters has to be contained in the same data body, even when they are presented as two different ones. They just define that the data shall be mandatory included.

3.1.3.2. Status and violations

Are all the REST operation related with the querying of the status of the infrastructure. All these operations are simply GET operations since they are the outputs of the SLA Manager.

Table 3.2 SLA status related operations.

Method	Resource	Parameters	Description	Response
Status and violations				
GET	/status	-	Returns all the status data registered in the current time window.	*200 (Success)*. *409 (Conflict)*: Duplication Data Error.
GET	/status /<sla_id>	sla_id	Returns the historical status registered for the SLA identified by sla_id in the current time window.	*200 (Success). 404 (Not Found)*: status for SLA not found.

Continued on next page

Table 3.2 SLA status related operations.

Method	Resource	Parameters	Description	Response
GET	`/violations /<sla_id>`	sla_id	Returns only the violations registered for the SLA identified by sla_id in the current time window.	*200 (Success). 404 (Not Found)*: violations for SLA not found.
GET	`/current`	-	Shows the real time information of the infrastructure, like any service breach. It is supposed to be accessed periodically by the Slice Manager.	*200 (Success).*
Historical status data				
GET	`/historic`	-	Return all the status related data registered in the repository.	*200 (Success).*
GET	`/historic /<sla_id>`	sla_id	Return all the status related data registered in the repository for the SLA identified by sla_id.	*200 (Success). 404 (Not Found)*: data for SLA not found.

The operations are described in table 3.2.The endpoint resource $/current$ has been developed for the Slice Manager to take the information periodically, since the SLA Manager does not push any data or action, preventing any bidirectional communication that disturbs with the modular programming paradigm used in the platform. Here the Slice Manager has to repeat this operation very often to refresh all the notifications the SLA Manager creates and to check if there has been a problem in the infrastructure that requires immediate action to solve it.

3.1.3.3. Parameter document

The parameter documents can be stored, updated and deleted independently of the SLA contracts management. This is done because it represents the mapping intelligence of the SLA Manager and needs to be constantly growing and improving. The parameter documents, the infrastructure elements it monitors and the mapping formulas contained are submitted to the SLA Manager in its own process, to let the SLOs for new applications with new infrastructure and network configurations be evaluated. These parameters can also receive information resulting on artificial intelligence algorithms that relate the historical data and provide new or more accurate mapping formulas, for instance, a neural network that can find new relations over the infrastructure metrics and a high-level KPI, and moreover, to new technologies and applications, like Internet of Things (IoT), Vehicle to everything (V2x) and ony vertical use-case running on the slices.

Table 3.3 Parameter document operations.

Method	Resource	Parameters	Description	Response
Parameter documents management				
POST	`/parameter`	param_body	Stores a new parameter document into the repository.	*200 (Success).* *409 (Conflict)*: Duplication Data Error.
GET	`/parameter`	-	Returns all the parameter documents stored in the repository.	*200 (Success).* *404 (Not Found)*: status for SLA not found.
GET	`/parameter /<param_id>`	param_id	Returns a parameter document, identified by param_id	*200 (Success).* *404 (Not Found)*: violations for SLA not found.
UPDATE	`/parameter /<param_id>`	param_id, param_body	Updates an existing parameter document, identified by param_id.	*200 (Success).*
DELETE	`/parameter /<param_id>`	param_id	Deletes an existing parameter document, identified by param_id.	*200 (Success).*

Note: *Do not confuse the column parameter with the high-level parameter of an SLO. The former is the information to be added in the REST operation.*

3.2. Software Stack

The software implementation of the SLA Manager has been designed to be flexible, generic and scalable in order to support new functionalities related to SLA management like negotiation, auto-templating, multi-domain, multi-provider, and so on. One important feature that makes possible that flexibility is that the database repository has been onboarded over a No-SQL solution [25], that allows to add fields, relate with new data models and scales through distributed servers. For having the SLA Manager main tasks always available, verification of the SLAs and its contained SLOs needs to be done in asynchronous background tasks performed by Celery [26], which allows multi-threading and parallel processing. All the RESTful API and overall logic has been coding using python and its easy to use Flask library [27]. Here is presented a brief explanation of each of the software tools used to give birth to the SLA Manager proposed. Figure 3.1 illustrates the stack of the deployed solution.

3.2.1. Flask

Flask is a web framework library for python [27], and here is used to develop the API interface and uses the python programming power for the intelligence of the SLA Manager. In flask it is easy to adapt new libraries according to the needs of the applications, and that brings light weight to the develop of the software module. Flask can also be connected

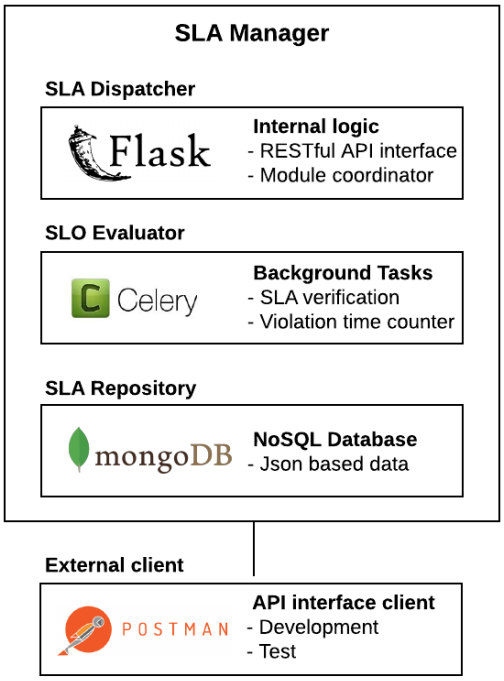


Figure 3.1 SLA Manager software stack implementation.

to any database solution, rather SQL or No-SQL, which brings flexibility at the moment of choosing the most efficient way of developing.

3.2.2. Celery

Celery [26] is a background task manager that helps to queue the jobs asynchronously, that is, without the need of interrupting the main processes to be available for the users. For the SLA Manager, Celery is used to perform the extraction of all the SLAs active in the database and make the SLO verification. In addition, it performs the actions needed when a service breach or even a violations happened, from the basic things like counting the time of a breach, notify the new events or reactions and store the SLA violations.

3.2.3. Postman

Postman [28] is an API client and is used as a testing tool for the development and testing of the solutions. It allows to have organized groups of calls, which helps saving the time at the moment of testing. It also allows to store data and parameters related with a call, that also is a plus at the moment of developing. It has a clear and graphical interface that make it easy and intuitive to use.

3.2.4. MongoDB

MongoDB [25] is a No-SQL solution that has a non-static data model. This helps the platform with the flexibility offered by the platform at the moment of deploy 5G services, you can deploy different models. Also helps in the further studies the feature that is schema-less, if you want to add extra features like automated negotiation, penalties and helps with the flexibility at the moment of giving shape to new templates for new applications. One last thing is that MongoDB and all the No-SQL solutions helps with the scalability of the platform by using auto balancing, that means that it fosters the SLA Manager to be location agnostic and also be distributed across different sites. One last feature is that is completely open source and has a big community to aid with the development of new functionalities and debugging.

One of the drawbacks is that it gives less flexibility at the moment of querying. With traditional relational databases, the query can contain any of their fields. In this case, we do not need to know all the fields to query, we just need to perform basic queries like just knowing the sla_id in the case of retrieving SLA data, its status and historical violations.

CHAPTER 4. TEST OF THE SLA MANAGER

The SLA Manager is built as a proof of concept that manages and enforces the levels of service given to the customers by the neutral host provider. To test the performance of the presented solution, here are presented some experiments that show how the system behaves in limited capacity. The goal is to see if the added complexity of the SLA Manager, to determine a high-level parameter value from the low-level infrastructure metrics, represents a low amount of overhead and the solution is feasible to scale. The added complexity is compared with a baseline solution, that is, an SLA management solutions that just queries the data from the repository. The experiments have been deployed over a virtual machine with 4 processors and 4 GB of RAM memory, which is a host with limited capacity only to have an initial insight about the potential of the software module.

In the experiment environment we will feed the SLA Manager with generic data, and only with NS type of SLAs, since that type is the one who is closer to the computation of monitored data and the deployment of monitoring jobs. Remember that each SLA for NS has related an NSD, where it is described the service configuration. Each NF in that NSD will be generically called network element hereafter. For instance, assume an NS formed by 5 VNFs and 5 PNFs, we say then that the related SLA has 10 network elements. We made this simplification because, at the end of the day, each NF is seen as a computational burden by the SLA Manager.

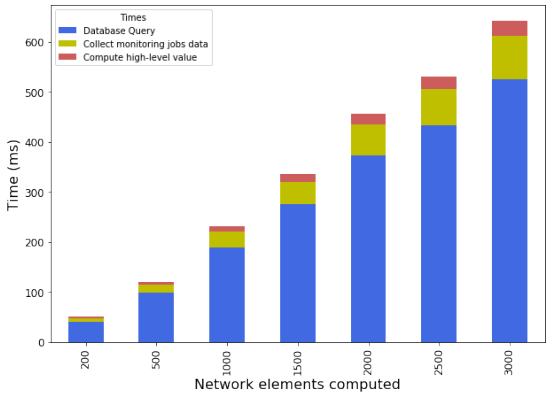
4.1. Overhead introduced by the solution

In this test we propose a baseline represented by an SLA management solution that does just the database query of all the SLAs stored in the repository, just as any simple SLA solution will face. For the experiment, we feed the SLA Repository with NS type SLAs and each SLA has related an NSD with 10 network elements. Remember that we name network element to any type of component that could be present in an NS, e.g. VNF, PNF, PL and so on. Consider the execution time of the database query as the baseline solution, and recall that this value is also part of our SLA Manager. The execution times that are introduced by our SLA Manager are the ones related with the collection of monitored data and the time related with computing of the high-level parameter value with the formula applied to the monitored data.

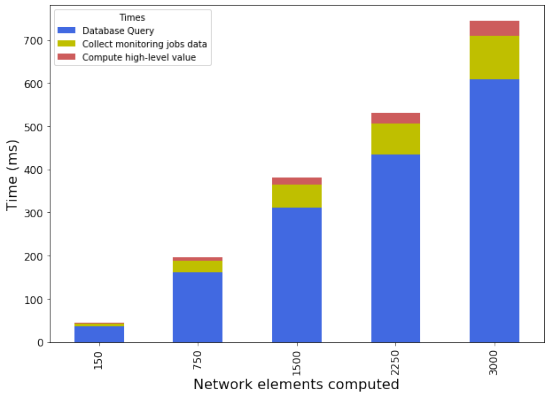
The process of collecting the monitoring data includes the looking into the subset needed in the monitoring jobs bucket. Here we deploy the monitoring jobs and simulate their monitored data with random values, that are available in a bucket of data. This is approximately similar to real environment conditions, since the Monitoring module will periodically load the bucket of monitored data of all deployed jobs in one call from the Monitoring Driver of the SLA Manager, and that single call represents an amount of time that we assume minimal.

The time it takes to compute the formula from all the collected data just represents the CPU load of the mathematical process to obtain the high-level value. Note that this value could be initially low, and that is why we feed the system with high number of network elements to see how this value scales.

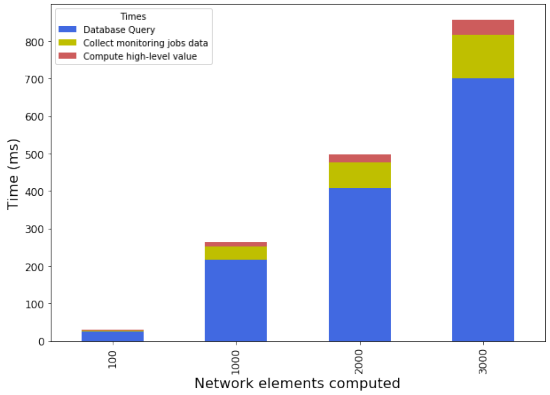
For all the execution times, we have established four scenarios where the number of mon-



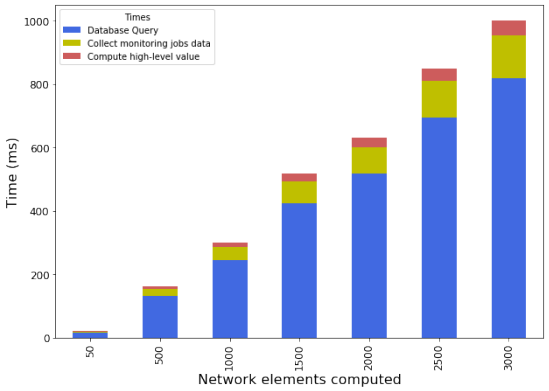
(a) Execution times for 5 metrics/network element



(b) Execution times for 10 metrics/network element



(c) Execution times for 15 metrics/network element



(d) Execution times for 20 metrics/network element

Figure 4.1 Time to iterate through the SLAs contained into the repository

itoring metrics measured for each network element is increased in order to arise the computational burden by involving more monitoring jobs per network element. Figure 4.1 represents the results of the four scenarios explained of testing. Note that the time for 3000 network elements arise to values near one second. However, this happens when quantities arise to unrealistic values for the application purposes of a neutral host platform. Moreover, the SLA Manager can be distributed across different nodes to split the load of SLAs and also could be deployed over more powerful resources if needed.

Note that the load of the system also affects the database query, that in theory must be the same in all scenarios, however, the execution times related with the added complexity of our SLA Manager represent a small percentage of the whole time measurement. This relation is observed to be approximately 20% of the total time in the 3000 nodes bar for all four scenarios

Going from figure 4.1(a) to 4.1(d) the database query execution time only is increased by a relatively small amount, which means that the database query is not affected by the operation computing a high-level parameter value and this processes are independent. The increase of the database query is caused by the increasing of SLAs stored in the SLA Repository.

4.2. Iteration execution times increasing stored SLAs.

The SLA Evaluator system works by doing an iteration through all the contracts in the repository, and analyzing each of their included SLOs. The execution time by which this happens is important because it establishes the period by which each SLA is being evaluated. This experiment has been set up to know if it takes more time to query an SLA or to compute a single network element. Thus, we will see how the system behaves in front of a network element scale. We are counting the number of VNFs that are being evaluated by the SLA Manager and see how much time it takes the iteration cycle. To simulate an overload in the SLA Manager, a measure on the iteration execution time is done after feeding more contracts to the repository and see how much time it takes to evaluate all the contracts in the system, until it gets to a considerable amount of contracts that is represent an environment of possible repository overload. For all network elements, only one metric is considered.

Besides measuring the execution time, different scenarios where SLAs are fed to the system with a specific number of network elements in the related NSD to see if it is more expensive to compute SLAs or nodes. First, the system is supplied with SLAs related with 5 network elements in its NSD, then with 10 network elements and so on until 20 network elements per each SLA. Notice that the total number of nodes is then the product of the number of SLAs times the number of nodes related per each one. In figure 4.2 we can see that iteration time grows steadily with the number of contracts stored in the repository.

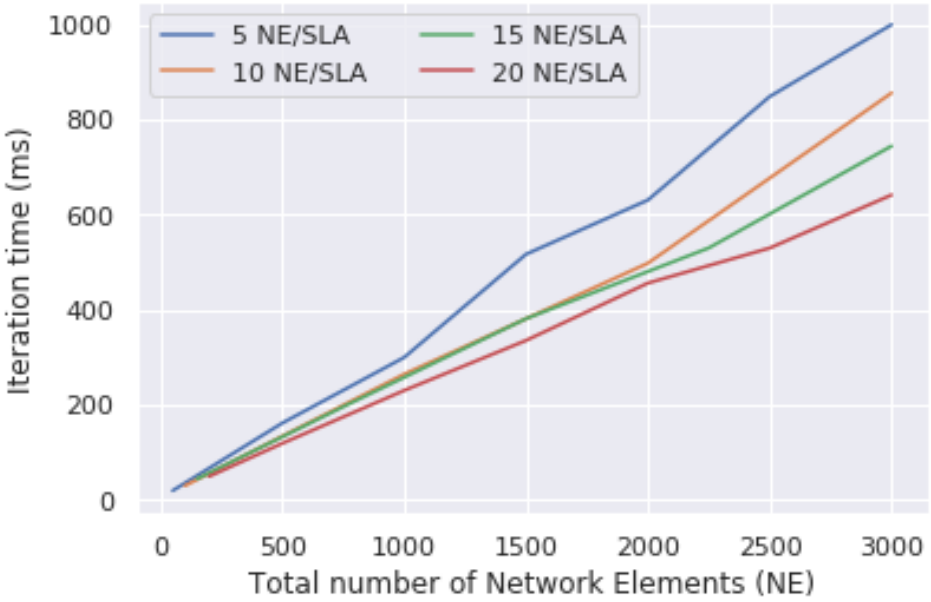


Figure 4.2 Iteration time of all SLAs stored in the repository vs network elements computed.

For all scenarios the execution time stays below one second and projects to grow for more contracts, which represents a situation that improbably happens in a real infrastructure. We can see also that iteration time is way lower in the scenario where we have more nodes per contract (20 NE/SLA), and this is a more realistic spot where each tenant has a limited amount of services deployed with a high number of nodes per each one. We

can clearly observe that most of the execution time of the SLA Manager tasks is gone over the database query task, and even the 20 network elements execution time is mostly composed by database query of SLAs.

In the 20 network elements graph-line, we see that for 1000 nodes the iteration time is maintained below 250 milliseconds, which is good to start the infrastructure with an SLA Manager deployed over a host with very-limited capacity. This means that SLA Manager does not take so much space in the infrastructure and it is cheap to compute. If the number of tenants grow and the services to be deployed arises, the SLA Manager can be migrated to a more powerful host to maintain the iteration times low.

For both experiments we can conclude that the SLA Manager solution proposed has a small overhead of time added to any SLA basic solution, compared with the big gain of having a computed value for a high-level parameter in the context of a generic and flexible solution, that gives reliable output and ease the creation and management of SLA over wide possibilities of NS configuration and applications.

CONCLUSIONS

This study focused on defining how an SLA Manager behaves in the context of a neutral host platform, where there is a physical infrastructure, and virtualization over different levels in the sense of slicing and virtualization of network services Moreover, the virtualization is done along an end-to-end telecommunication network infrastructure, taking all the access, networking and computing part, extending the cloud and web services domains where the SLA management topic has been largely studied.

Here we have made a difference between the slice that each tenant has, and also the services it can deploy based on a VNF catalog. The VNF catalog contains functions to be deployed over the virtual infrastructure.

Further, the study aimed at the monitoring and evaluation of Service Level Objectives (SLO) as part of the overall SLA management, because it is the most basic feature an SLA management solutions has to perform along with maintaining a repository of SLA. In this scope, we developed a mathematical process where a high-level parameter contained in an SLO is the value to be calculated in function of low-level monitored values. The process is developed from the NFV framework, taking the NSD that describes any NS onboarded over a slice, so it can be applied to any service configuration and adapt to new applications and technologies. Availability is studied one of the most important QoS indicators, where the infrastructure has to be available for the tenant to offer its services to its end-users. The SLA Manager can act as a virtual accountant for both parties to see how much time the system has been down and establish fair conditions, also as an indicator or performance for the provider. The SLA management solution has been designed to support also other performance indicators based on its flexible definition of QoS parameters (which are stored in NoSQL documents). Not only can other performance indicators be evaluated, but also new services with new VNF modules deployed on top of new physical technology that may come in the future can be supported. However, the details for managing other performance indicators have been left for future studies.

In the monitoring of the SLA Manager, it has been tested and observed that the load of the repository influences in the time the system executes its tasks. We not only need reliable measures but we also need that they are delivered in the shortest. The SLA Manager has exhibited a good performance over limited capacity of the hosting machine, meaning that it can perform excellent over a more powerful machine and also over real figures in terms of contracts and nodes that needs to be evaluated.

For the software design, the SLA Manager takes into account the wide possibilities of 5G, with a flexible, scalable and generic approach. We saw also that extra features to the SLA Manager can be easily added with the help of asynchronous tasks in the background. The NoSQL solution for hosting the database repository also facilitated this task and the support for new performance indicators, new services and new technologies where this SLA Manager can be useful. The RESTful application development interface will also help to the rest of the orchestration entities to develop new functionalities and to extend the SLA manager without harming the current performance. It also helps to modularize the SLA Manager over different neutral host providers and also inside the slices in case of subleasing of one of the tenants.

The SLA Manager consumes the monitoring data given by the Monitoring module, which

may use any monitoring technique available to retrieve the data related to the status and the performance of the infrastructure. For this reason, the Monitoring module has to develop and include in its logic a way to parse the messages from language-specific monitoring technique used, into a common notation used by all modules of 5GCity platform and offer this information through an API interface. Thus, the Monitoring Driver can be extended to obtain the data of any monitoring solution and we can state that the SLA Manager works independently of how the monitoring data is obtained. It is monitoring-agnostic. This feature will help to deploy the SLA Manager in a situation where the neutral host is no longer acting as the physical infrastructure provider, and will help to differentiate between the virtual data and physical data, helping the system to work on a multi-provider situation.

The SLA management is an essential feature to foster the neutral host platform business models, as it will provide its services to a wide range of customers that can now trust on virtual infrastructures to host their services. They can access the output data of all its deployed services on the dashboard and see how the system they are paying for is operating, to have a clean view of the health of the system that can be reported also to its end users. In virtual environments it is hard to detect from where the fault root is coming, could be from a wrong deployment of the customer, or a software problem in the catalog, also coming from maintenance tasks over the infrastructure. Having a clean view will also avoid misunderstandings between all the parties involved in the business and have a mutual cooperation industry.

4.3. Suggested work

This study is the basis for further studies where other aspects of SLA management can be added. An example aspect would be auto-templating according to the infrastructure for SLA negotiation, that is, to offer different threshold levels of service and defining conditions according to the provider's current capacity.

This SLA Manager can also be used to test and calibrate algorithms used in the study of infrastructure abstraction, and virtual machine allocation problem, like the studies deployed over other platforms done in [29] [30]. It can be used in deployments that need to monitor system availability. The data that this SLA Manager delivers is reliable enough to build data-sets for further algorithms with artificial intelligence and data science techniques to infer new connections and improve the execution times, new formulas for other KPIs for new services, and all the benefits that machine learning can contribute to the field.

With this work we seek to lay the foundations for the studies and implementations of SLA Management over future 5G networks and virtualized, slicing-related infrastructure.

4.4. Sustainability considerations

The work developed in this study is part of an infrastructure virtualization project, which one of its main goals is reducing energy consumption, with the associated reduced costs and CO2 emissions, and has a positive environmental impact.

4.5. Ethical considerations

The virtualization techniques offered in the neutral host context developed by 5GCity will allow new participants to provide new services. The quality of that service is ensured with trust mechanisms, like Service-Level Agreements, that allow the providers to offer to their end-users high quality and useful products. The SLA Manager has a positive impact on human relationships as it helps to target the source of the problems and avoids misunderstandings.

The data provided for SLA management mechanisms will also favor the fair competition for the service providers, as the customers can choose the ones who have historically and statistically the best performance for their needs.

ACRONYMS

ETSI	European Telecommunications Standards Institute
HA	High Availability
KPI	Key Performance Indicator
MANO	Management and Orchestration
NF	Network Function
NFV	Network Functions Virtualization
NFVI	Network Functions Virtualization Infrastructure
NS	Network Service
NSD	Network Service Descriptor
PNF	Physical Network Function
QOS	Quality of Service
SLA	Service-Level Agreement
SLO	Service-Level Objective
VIM	Virtual Infrastructure Manager
VL	Virtual Link
VNF	Virtual Network Function
VNFFG	VNF Forwarding Graph
VNFFP	VNF Forwarding Path

BIBLIOGRAPHY

[1] Network Functions Virtualisation NFV and Use Cases. ETSI GS NFV 001 V1. 1.1 (2013-10). 2013. 1, 6

[2] Viscardo Costa, Antonino Albanese, Viscardo Costa, Shuaib Siddiqui, August Betzler, Hamzeh Khalili, Apostolos Papageorgiou, Sergi Figuerola, Theodoros Rokkas, Ioannis Neokosmidis, David Pujals, Luis Moreno, Gabriele Baldoni, Nicola Ciulli, Paolo Cruscchelli, Elian Kraja, Elio Francesconi, Maria Rita Spada, Pedro Diogo, Ricardo Preto, Pedro Paolino, Daniel Raho, Simon Pyor, Antonio Garcia, Trevor Moore, Alexandre Ullisses, Pedro Santos, Mariano Lamarca, Jordi Cirera, and Gonzalo Cabezas. 5GCity Architecture & Interfaces Definition (D2.2), available at https://doi.org/10.5281/zenodo.2558243, February 2019. ix, 1, 3, 4, 6

[3] ETSI, NFVISG. GS NFV 002-V1. 1.1-Network Function Virtualisation (NFV)-Architectural Framework. *publishing October*, 2013. ix, 3, 5, 6

[4] Hazmeh Khalili, Papageorgiou, Shuaib Siddiqui, Julio Barrera, Felipe Huici, Kenichi Yasukata, Nicola Ciulli, Paolo Cruschelli, Elian Kraja, Elio Francesconi, Ricardo Preto, Antonino Albanese, Viscardo Costa, Carlos Colman, Gabrielle Baldoni, Teodora Sechkova, and Michele Paolino. Orchestrator design, service programming and machine learning models ((D4.1) available at https://doi.org/10.5281/zenodo.2558306, February 2019. ix, 4, 6, 7

[5] ETSI, NFVISG. Network Functions Virtualisation (NFV); Network Service Templates Specification. *ETSI GS NFV-IFA*, 14:V2. 6, 15

[6] Gongzhuang Peng, Jiaxin Zhao, Minghui Li, Baocun Hou, and Heming Zhang. A sla-based scheduling approach for multi-tenant cloud simulation. In *2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 600–605. IEEE, 2015. 9

[7] Saad Mustafa, Kashif Bilal, Saif Ur Rehman Malik, and Sajjad A Madani. Sla-aware energy efficient resource management for cloud environments. *IEEE Access*, 6:15004–15020, 2018. 9

[8] Lav Gupta, Mohammed Samaka, Raj Jain, Aiman Erbad, Deval Bhamare, and H Anthony Chan. Fault and performance management in multi-cloud based nfv using shallow and deep predictive structures. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8. IEEE, 2017. 9

[9] Saad Mubeen, Sara Abbaspour Asadollah, Alessandro Vittorio Papadopoulos, Mohammad Ashjaei, Hongyu Pei-Breivold, and Moris Behnam. Management of service level agreements for cloud services in iot: A systematic mapping study. *IEEE Access*, 6:30184–30207, 2017. 9

[10] Yahui Li, Xia Yin, Zhiliang Wang, Jiangyuan Yao, Xingang Shi, Jianping Wu, Han Zhang, and Qing Wang. A survey on network verification and testing with formal methods: Approaches and challenges. *IEEE Communications Surveys & Tutorials*, 21(1):940–969, 2018. 9

[11] Alexandru-Florian Antonescu and Torsten Braun. Simulation of sla-based vm-scaling algorithms for cloud-distributed applications. *Future Generation Computer Systems*, 54:260–273, 2016. 9

[12] Alexandru-Florian Antonescu, Philip Robinson, Luis Miguel Contreras-Murillo, José Aznar, Sébastien Soudan, Fabienne Anhalt, and Joan A García-Espín. Towards cross stratum sla management with the geysers architecture. In *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, pages 527–533. IEEE, 2012. 9

[13] Riccardo Guerzoni, David Perez-Caparros, Paolo Monti, Giovanni Giuliani, Javier Melian, and Gergely Biczók. Multi-domain orchestration and management of software defined infrastructures: A bottom-up approach. IEEE Communications Society. 2016. 10

[14] Kouessi Arafat Romaric Sagbo, Yénukunmè Pélagie Elyse Hounguè, and Ernesto Damiani. Sla negotiation and monitoring from simulation data. In *2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 766–772. IEEE, 2016. 10

[15] Taher Labidi, Achraf Mtibaa, Walid Gaaloul, and Faiez Gargouri. Ontology-based sla negotiation and re-negotiation for cloud computing. In *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 36–41. IEEE, 2017. 10

[16] Vincent C Emeakaroha, Ivona Brandic, Michael Maurer, and Schahram Dustdar. Low level metrics to high level slas-lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments. In *2010 International Conference on High Performance Computing & Simulation*, pages 48–54. IEEE, 2010. 10

[17] Ying Zhang, Wenfei Wu, Sujata Banerjee, Joon-Myung Kang, and Mario A Sanchez. Sla-verifier: Stateful and quantitative verification for service chaining. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017. 10

[18] Carlos Parada, José Bonnet, Eleni Fotopoulou, Anastasios Zafeiropoulos, Evgenia Kapassa, Marios Touloupou, Dimosthenis Kyriazis, Ricard Vilalta, Raul Muñoz, Ramon Casellas, et al. 5gtango: A beyond-mano service platform. In *2018 European Conference on Networks and Communications (EuCNC)*, pages 26–30. IEEE, 2018. 10

[19] Evgenia Kapassa, Marios Touloupou, Argyro Mavrogiorgou, and Dimosthenis Kyriazis. 5g & slas: Automated proposition and management of agreements towards qos enforcement. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 1–5. IEEE, 2018. 10

[20] Marios Touloupou, Evgenia Kapassa, Chrysostomos Symvoulidis, Panagiotis Stavrianos, and Dimosthenis Kyriazis. An integrated sla management framework in a 5g environment. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 233–235. IEEE, 2019. 10

[21] Julien Maisonneuve and Alcatel Lucent. ETSI NFV ISG:“GS NFV-INF 010-V1. 1.1-Network Functions Virtualisation (NFV); Service Quality Metrics,” . European Telecommunications Standards Institute. December 2014. 16, 17

[22] Sector, STANDARDIZATION and Itu, O. Series E: overall network operation telephone service service operation and human factors quality of telecommunication services: concepts models objectives and dependability planning-Use of quality of service objectives for planning of telecommunication networks. *ITU*, 2008. 16, 17

[23] Prof. Dr. Alessandro Birolini (auth.). *Reliability Engineering: Theory and Practice*. Springer-Verlag Berlin Heidelberg, 8 edition, 2017. 17

[24] Roy Thomas Fielding. Architectural styles and the design of network-based software architectures. university of california, irvine, 2000, 2016. 28

[25] Kristina Chodorow. *MongoDB: the definitive guide: powerful and scalable data storage*. ” O’Reilly Media, Inc.”, 2013. 32, 34

[26] Ask Solem. Celery: Distributed task queue. *Dostupné z: http://www.celeryproject.org/*, 2013. 32, 33

[27] Miguel Grinberg. *Flask web development: developing web applications with python*. ” O’Reilly Media, Inc.”, 2018. 32

[28] Postman — The Collaborative Platform for API Development. https://www.getpostman.com/. Accessed: 2019-09-30. 33

[29] George-Valentin Iordache, Florin Pop, Christian Esposito, and Aniello Castiglione. Selection-based scheduling algorithms under service level agreement constraints. In *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, pages 134–140. IEEE, 2017. 40

[30] Jaafar Bendriss, Imen Grida Ben Yahia, and Djamal Zeghlache. Forecasting and anticipating slo breaches in programmable networks. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, pages 127–134. IEEE, 2017. 40