NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY POLITEHNICA BUCHAREST

Doctoral School of Electronics, Telecommunications and Information Technology

Decision No. 93 from 05-10-2023

Ph.D. THESIS SUMMARY

Cosmin Andrei CONȚU

MANAGEMENTUL RESURSELOR ÎN SOLUȚII DE VIRTUALIZARE PENTRU REȚELE ȘI SERVICII

RESOURCE MANAGEMENT IN VIRTUALIZATION SOLUTIONS FOR NETWORKS AND SERVICES

BUCHAREST 2023

Content

Chapter 1

Introduction

In the telecommunications sector, there is a huge demand for mobile broadband, largely due to the need for high-quality video content delivery. 5G networks are addressing this problem while trying to support the development of several sectors by providing the infrastructure for the development and use of applications that require very low latency and high network reliability. In order to meet all these services and requirements, 5G networks need high flexibility at the architectural level [1].

Network Functions Virtualization (NFV) is an emerging architecture concept. It aims to address some of the challenges and limitations that exist in the telecommunication industry, such as the large number of proprietary hardware machines dedicated to specific services, high Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) costs, high power consumption and storage space issues of hardware machines as well as lack of flexibility or interoperability [2][3].

Nowadays, NFV is also a supporting technology in Cloud or Edge Computing. NFV decouples hardware machines from the network functions running on them, using generic hardware and running network functions through virtual machines installed on these generic machines. Based on virtualization technologies, NFV enables much faster development and deployment of services containing network functions that can be virtualized (compared to services composed of physical network functions). Virtualised Network Functions (VNFs) can be defined, instantiated, deployed or migrated using the same infrastructure. They can be created, modified or deleted without the need for a physical presence at the location of the hardware machines running these functions. Another advantage is the reduction of CAPEX and OPEX costs due to the growing IT industry enabling software development. It can also lead to a decrease in power consumption through a virtual machine management and migration plan.

Software Defined Networking (SDN) is a complementary technology to NFV. The concept of SDN is to separate the control plane from the data/user plane. This concept allows high flexibility, programmability and abstraction of the network, improving management and control functions. Although SDN and NFV are two completely independent technologies, they can be used together to create powerful and flexible telecommunications systems. From a general point of view, these two technologies can be considered as orthogonal, as SDN separates the control plane from the data plane and NFV can be used in both the data and control planes to implement various virtual network functions.

The 5G Network Slicing concept is based on virtualization and programmability, allowing for modularity in the provisioning of network resources, tailored to various system requirements in terms of bandwidth, latency, mobility, etc. [4-8]. From a general perspective, a Network Slice (NSL) is a managed logical group of subsets of resources, physical and/or virtualized network functions, architecturally placed in a Data Plane (DPl), a Control Plane (CPl) and a Management Plane (MPl). The slice is programmable and has the ability to expose its capabilities to users. NFV [9-11] and SDN technologies can be used together [12] to manage and control the sliced 5G environment in a flexible and programmable way.

In 5G systems, an operator can physically split network traffic, split a network, or split the resources of several networks combined. This flexibility allows operators to select the desired features that are required by a customer, such as traffic volume or connection density. As defined by the 3rd generation partnership project (3GPP) [13], there are three general types of network slices, each corresponding to a 5G use case. Enhanced mobile broadband (eMBB) slices, which can offer high data rates and moderate delay. Ultra-reliable low latency communications (URLLC) slices that provide very reliable transmissions with extremely low delay. Massive machine-type communications (mMTC) slices supporting high-density Internet of Things (IoT) connections.

1.1 Presentation of the field of the doctoral thesis

Considering the massive growth of the telecom industry and the deployment of 5G networks in recent years, paradigms such as SDN and NFV have been exploited, offering viable solutions to current industry problems. Using these technologies, the concept of network slicing can bring great benefits such as modularity in provisioning network resources and allocating subsets of them into logical groups, tailored to customer requirements in terms of latency, bandwidth, etc.

A very important phase in the virtualization of a network is its orchestration and management. For this reason, the SONATA and Open Source MANO frameworks have been studied in this paper and subsequently used in experiments.

Other technologies presented in this paper are web platform development (React library), REST API communication standard as well as machine learning algorithms.

1.2 Scope of the doctoral thesis

This thesis aims to present the NFV and Network Slicing paradigms and their various use cases. The study of two NFV frameworks such as SONATA and OSM complements the study of these paradigms. For a better understanding of how these frameworks work, various experiments have been conducted.

After analyzing the results obtained from these experiments, one of the two frameworks was chosen to be integrated into a stand-alone system that would allow intelligent configuration of new network slices using Machine Learning (ML) algorithms. The autonomous system was integrated into a platform through which a new service chain was built, allowing experiments to be carried out to validate this implementation. This autonomous system and the platform that includes it are original contributions of the author, both at the architectural level and at the implementation and testing level.

Using this automated platform, deploying a network slice through the Open Source MANO - Virtual Infrastructure Manager service chain becomes much faster and easier. Adding the intelligent configuration block optimizes network slices both in terms of deployment and installation on the right infrastructure, as well as adaptively changing alarm and scalability thresholds..

The solution proposed in this paper aims to improve the current network slice management workflow through automated mechanisms that can optimally allocate resources and adapt alarm and scaling thresholds to optimize network slice operation.

1.3 Content of the doctoral thesis

The work is divided into six chapters. The first chapter is an introduction to the field of the PhD thesis and the last chapter presents the conclusions of this work. The remaining chapters are presented below.

Chapter 2 contains an extensive overview of machine learning concepts and tools used in specific network slice cases. The concepts presented will form the basis of the following chapters. A comparison of optimization functions using an experiment as well as a review of other projects studying the use of machine learning in network slicing is also performed.

Chapter 3 presents the SONATA framework, starting with a review of other similar projects and continuing with its detailed architecture and a comparison with the ETSI NFV model. Furthermore, a series of experiments using the SONATA emulator are presented.

Chapter 4 covers the Open Source MANO framework starting with its architecture. A parallel between this framework and SONATA is reviewed to highlight the more complex architecture of Open Source MANO. A performance analysis of Open Source platforms is conducted and different use cases and projects using OSM are presented. Next, an architectural enhancement to the monitoring module is proposed by detecting an optimal alarm threshold using ML algorithms. Lastly, a series of implementations divided into various scenarios are presented.

Chapter 5 is dedicated to the Automation Platform. This is a new component added to an existing service chain (Open Source MANO and a virtual infrastructure manager) to improve the process of deploying and managing network slices. The architecture of this platform and its role in the new service chain is presented, as well as its graphical user interface along with its building blocks. A series of experiments covering various scenarios are presented in the last part of the chapter.

Chapter 6 contains the conclusions of this work, presenting the results of the multiple experiments performed, the author's original contributions and their role in improving existing systems. The author's list of papers and projects and the prospects for further development of his contributions are also presented.

Chapter 2

Using ML in Network Slicing

The objective of this chapter is to present and understand machine learning technologies and their applicability to network slicing. The conducted experiment facilitated the choice of an optimization function and the creation of a prediction model that was used in the following chapters.

ML is a branch of Artificial Intelligence (AI) that allows machines to learn and improve their performance over time using evolving algorithms and input data (training data, real-life data). While in traditional programming, the result will be obtained based on input data and already defined algorithm(s), ML algorithms are an adaptive process that is able to generate a result by learning from previous experience.

2.1 Concepts and tools

ML techniques are used today in a wide range of areas. A typical example of an ML application is image recognition, which is able to determine the type of an image based on a set of training data and can even recognize specific patterns such as objects or people. Prediction is another common use case for ML and is applied in various domains, from predicting real estate prices to predicting traffic patterns. Product recommendations, speech recognition, malware detection as well as natural language processing are other well-known use cases [14]. ML algorithms and techniques are increasingly used in network and service management and control operations (resource reservation and allocation, traffic forecasting, mobility management, etc.).

A classification of ML methods might be:

Supervised learning (SML) - these methods predict one or more dependent variables based on (initially) labeled data.

Semi-supervised learning (SSML) represents cases where not all data is labelled.

Unsupervised learning (UML) - the machine searches for structure in (unlabeled) datasets.

Reinforcement learning (RL) is the fourth major learning method in ML, alongside supervised, unsupervised and semi-supervised learning. However, the RL model does not require large amounts of data to train. It learns structures by rewarding desirable behaviours and punishing bad ones.

Deep learning (DL) - uses a series of layers of non-linear processing units to extract and transform features. DL models can be supervised, semi-supervised or unsupervised (or a combination of all three). DL is based on neural networks (NN), which mimic the way the human brain works [15].

RL has difficulties in cases with large state spaces because it has to go through each state and obtain a value function or pattern for each state-action pair in a direct and explicit way. Proposals have been developed to sample only parts of the states and then apply NN to train a sufficiently accurate function or model. The result is Deep Reinforcement Learning (DRL), which exposes sufficient performance stability. In short, DRL is a combination of RL and DL, where RL defines the goal and DL provides the mechanism.

2.2 Libraries and functions

ML is a branch of Artificial Intelligence (AI) that allows machines to learn and improve their performance over time using evolving algorithms and input data (training data, real-life data). While in traditional programming, the result will be obtained based on input data and already defined algorithm(s), ML algorithms are an adaptive process that is able to generate a result by learning from previous experience.

2.2.1 Open source ML libraries

There are many open source ML libraries available that are currently used in application development. These libraries are capable of providing predefined templates and extensive customization possibilities.

Scikit-learn [16] is one of the most popular and robust ML libraries offering a wide range of tools and templates for ML development in Python. PyTorch [17] is an open source framework that is frequently used in deep learning development, such as image recognition or language processing. Natural Language Toolkit (NLTK) [18] is an important platform for developing Python applications for natural language processing. It provides a wide range of resources for speech recognition, encoding, classification, etc. TensorFlow [19] is an open source ML library developed by Google, focused on deep learning and large-scale machine learning projects. TensorFlow provides a wide range of ML/DL models and algorithms implemented in different programming languages, such as Python or Javascript.

2.2.2 Loss and optimization functions

In order to start training a model, it must first be created by assembling one or more layers into a model. After its creation and assembly as a sequential model, the model must be compiled. In the compilation stage, the model is given two functions: a loss function that is responsible for measuring the difference between the predictions and the expected outcome, and an optimization function that aims to reduce the loss by adjusting the internal values until the most accurate form possible for the model is obtained. Both functions are called during the training stage to calculate the loss for each stage and to improve it.

While Mean Squared Error [20] is the most widely used loss function, there are several optimization functions that can be used.

The adaptive gradient algorithm (Adagrad) [21] is a small-batch gradient descent optimization method that adapts the learning rate by incorporating knowledge from prior experience. Adaptive Moment Estimation (Adam) [22] is a gradient descent optimization algorithm that uses the moment algorithm to speed up the gradient descent algorithm by using exponentially weighted averaging of the gradients and the adaptive root mean square propagation (RMSProp) learning algorithm that uses exponential moving average. RMSProp [23] works in the same way as Momentum by increasing the learning rate, using larger steps for directions that converge faster. The difference between the two optimizers lies in the way gradients are computed.

2.3 Optimization functions comparison experiment

In order to compare the optimization functions, the result of a short experiment performed in [24] was considered. Using a small data set with x and y values that can be described by linear equation 1.1:

$$y = 2x + 40 \tag{1.1}$$

Using an ML algorithm, the same equation should be found between the x and y values. The x values will represent the input data for the ML model, while the y values will be the labels or outcomes that the model should predict. After preparing the data (and splitting it into training and test data), the model will be created with a single layer and a single neuron (since a simple linear equation has to be solved). The model will be assembled and compiled with a loss function and an optimization function. In order to be able to compare the optimization functions, the model will be trained with the same loss function, the same training data and the same number of epochs, but with a different optimization function each time. As a final step, the accuracy of the model will be measured using a performance parameter (Figure 2.1).

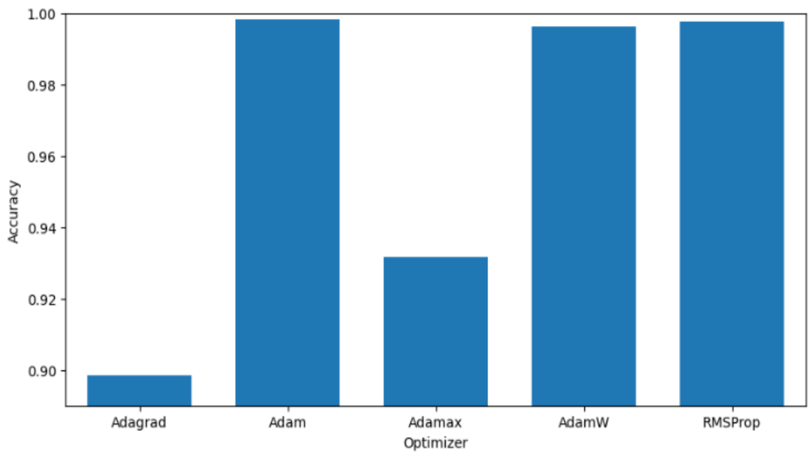


Figure 2.1 *Comparison results of optimization functions*

The results [25] indicate that the Adam optimization function had the best accuracy (0.9981), followed by RMSProps (0.9975) and AdamW (0.9963). Slightly less accurate were Adamax (0.9317) and Adagrad (0.8986). Based on these results and given that, among the optimization functions with a computed accuracy of over 99%, Adam has the lowest computation time, the Adam optimizer was chosen for further use in this paper.

2.4 Machine learning in network slicing

There are a lot of existing studies and implementations of machine learning techniques in network slicing. The studies cover several aspects of how to apply ML algorithms in network slicing use cases, such as predicting potential threats, resource allocation and traffic forecasting.

Chapter 3

SONATA Framework

In this chapter, the ability of the SONATA framework to orchestrate and manage virtual network functions in an ETSI-aligned NFV environment was studied. Performing the experiments in this chapter provided an important starting point for understanding the life cycle of a network service as well as the communication between an orchestrator and a virtual network manager.

The EU H2020 project SONATA: Service Programming and Orchestration for Virtualized Software Networks [26] aims to develop an NFV framework that provides external (third-party) developers with a programming model and a set of tools for developing virtualized services integrated with an orchestration system. SONATA offers the possibility for software developers to achieve a low time-to-market of a Network Service (NS), to optimize and reduce deployment costs and to shorten the integration time of software networks in the telecom industry.

From SONATA's perspective, OpenStack is a complementary platform. SONATA developers need access to a working instance of OpenStack to use its virtual infrastructure manager functionality to run services from the service platform.

Another option for SONATA service developers, is to use the SONATA emulator to develop and test various network services chained in different scenarios. The SONATA emulator provides OpenStack-like interfaces to allow an orchestrator (SONATA, Open Source MANO) to control the emulated virtual infrastructure manager.

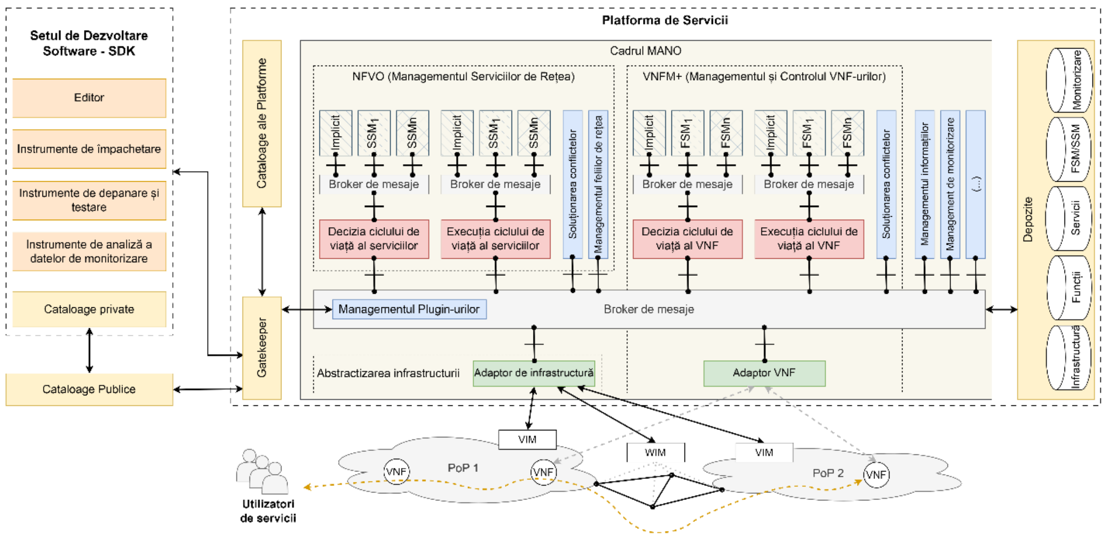
3.1 Architecture

The overall architecture of the SONATA framework (Figure 3.1), which is built upon the NFV MANO architecture model provided by ETSI, consists of the following components [27]:

- - Service Platform (SP)
- - Software Development Kit (SDK)
- - Catalogues

The Service Platform (Figure 3.2) receives the service packages created through the SDK and is responsible for placing, deploying, provisioning, scaling and managing services on existing cloud infrastructures. The component responsible for processing incoming and outgoing requests is the gatekeeper module. The service platform is also responsible for providing direct feedback to the software development set on deployed services, e.g. monitoring data of a service or its components. It has also been designed with the possibility of full customization in mind, thus giving both flexibility and control to operators and developers.

A more detailed view of the architecture of the SONATA framework and how the components communicate with each other can be seen in Figure 3.4.



Setul de Dezvoltare Software - SDK
Editor
Instrumente de împachetare
Instrumente de depanare și testare
Instrumente de analiză a datelor de monitorizare
Cataloage private
Cataloage Publice
Platforma de Servicii
Cataloage ale Platforme
Gatekeeper
Cadrul MANO
NFVO (Managementul Serviciilor de Rețea)
VNFM+ (Managementul și Controlul VNF-urilor)
Implicit
SSM1
SSMn
FSM1
FSMn
Broker de mesaje
Decizia ciclului de viață al serviciilor
Execuția ciclului de viață al serviciilor
Decizia ciclului de viață al VNF
Execuția ciclului de viață al VNF
Soluționarea conflictelor
Managementul feliilor de rețea
Managementul informațiilor
Management de monitorizare
(...)
Managementul Plugin-urilor
Abstractizarea infrastructurii
Adaptor de infrastructură
Adaptor VNF
Depozite
Monitorizare
FSM/SSM
Servicii
Funcții
Infrastructură
VIM
WIM
PoP 1
PoP 2
VNF
Utilizatori de servicii

Figure 3.4 Detailed architecture of the SONATA framework [28].

3.2 ETSI NFV model comparison

The MANO framework is the core element of the platform and provides management for the entire lifecycle of complex network services. It has the same building blocks as the ETSI-MANO block (NFV orchestrator and VNF manager).

Both in SONATA and in the ETSI model, the virtual infrastructure manager controls and manages virtualized compute, storage and network resources in the operator infrastructure domain. A virtual infrastructure manager can control one or more types of resources within the NFV infrastructure.

Also, compared to the ETSI-MANO model, in SONATA there is also the gatekeeper component which is responsible for validating the network services uploaded to the platform as packets, by mediating between development and operational actions [29].

Therefore, the service platform follows the ETSI NFV model but, at the same time, adds new proprietary extensions that make it easier to support multiple clients by slicing resources that can be allocated exclusively to a specific client.

Compared to the ETSI model, SONATA has added the SDK as a very important component of the proposed architecture. This set supports third-party developers who want to develop complex services containing multiple virtualized network functions, with a variety of software tools but also with support for deployment and management of network services created on different SONATA service platforms..

3.3 SONATA emulator experiments

These experiments were a first step in understanding how an orchestrator works and the lifecycle of a virtual network function. Starting from simple topologies such as communication between two client machines, to topologies with chained functions (client, router, firewall, etc.), these experiments have provided a good understanding of virtualization and management of network functions in an NFV environment. These contributions formed the basis for further experiments in this work.

The topologies used in these experiments are represented as emulated networks using Docker containers [30] as instances on which to run virtual network functions.

Experiments performed:

- **Topology with simple client virtual machines**: The objective of this experiment is to create two client virtual machines and test the communication between them.
- **Topology with HTTP server**: The objective of this experiment is to create an HTTP server virtual machine and a client virtual machine to test the functionality of the HTTP virtual server.
- **Topology with virtual firewall**: The objective of this experiment is to create a firewall virtual machine and two client virtual machines to test the functionality of the firewall virtual machine, which will allow or block a certain type of traffic between the two client machines.
- **Topology with virtual routers**: The objective of this experiment is to create a network of router virtual machines that will route traffic between three client virtual machines in three different networks.
- **Topology with chained virtualized network functions**: The objective of this experiment is to create a topology that instantiates a chaining of various virtualized network functions. The virtual machines used in this topology will be client, router, firewall, proxy server and HTTP server.

Chapter 4

Open Source MANO Framework

In this chapter, the architecture of the OSM framework compared to the SONATA framework and its performance was studied. For a better understanding of how the OSM works, a series of implementations with various scenarios were performed. A proposal for improving the architecture using ML algorithms to detect an optimal alarm threshold was also presented.

Open Source MANO (OSM) is an ETSI-supported project that aims to develop a management and orchestration software package aligned with the NFV architecture presented by ETSI. OSM is based on a multi-layer model, where each layer contains a service composed of the services of the lower layers. Using this approach, OSM aims to provide better service integration with minimal effort. This is achieved by using an information model (Figure 4.1) based on the NFV model provided by ETSI, which is able to model and automate the life cycle of network functions..

4.1 Architecture

From an architectural point of view [31], OSM consists of several modules. Each of these has specific roles and is decoupled from each other (Figure 4.2). NBI is an important module that consumes information from other modules. NBI provides communication via Representational State Transfer (REST) APIs, which are consumed by various clients, such as the CLI client. The CLI is an OSM client provided as part of the installation. In addition, there is also a GUI client that allows direct use of OSM. Other types of clients that can interact with the NBI are Operations and Business Support Systems (OSS/BSS). The VIM client will be responsible for managing the resources (compute, storage and network) of the NFV infrastructure.

.

4.1.1 Functional components

The main components of the system are the LCM, the VNF Configuration and Abstraction (VCA) module and the Resource Orchestrator (RO). When a new instantiation request is made, the NBI records an entry in the MongoDB database stating that a new instance is to be created and passes control to the LCM, which is responsible for the instantiation operations and provides end-to-end orchestration. First, if deployed on an infrastructure manager, the LCM will interact with that VIM to create networks and provide connectivity between virtual machines in that VIM. It also interacts with SDN controllers which are, like the VIM, external components that the OSM interacts with via the RO module, and deploys those virtual machines.

After installing the OSM software on a local or remote server, a connection must be established with a VIM.

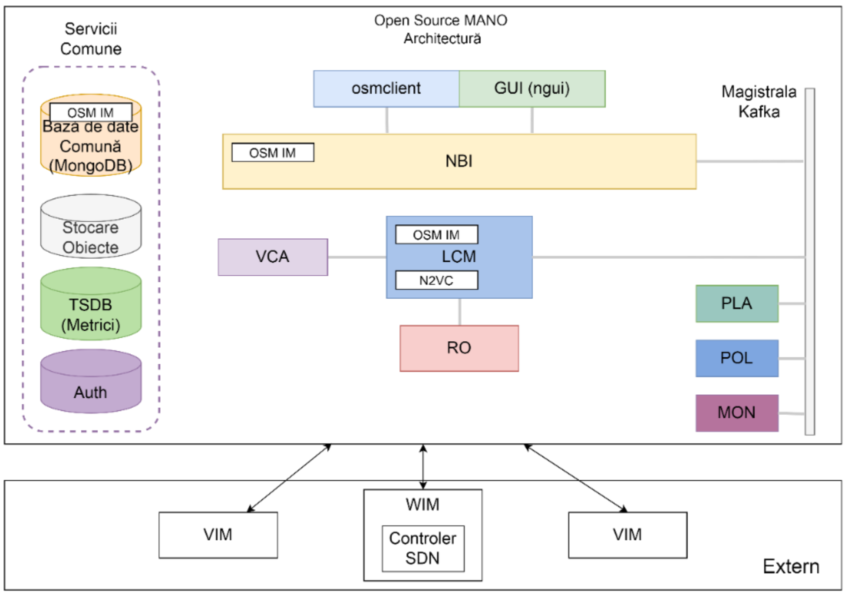


Figure 4.2 *OSM Architecture. [31].*

4.2 SONATA parallel

This section presents a selective overview of related work on the development and orchestration of services in virtualized networks and their relationship to the OSM architecture. It also presents the SONATA framework which is part of the OSM framework.

SONATA is in a direct relationship with OSM thanks to the integration of the SONATA emulator from the SDK module into OSM DevOps, which was part of the third release of OSM. This emulator provides easier integration with MANO stacks thanks to its APIs, which resemble the APIs provided by Openstack [32].

The SP component of SONATA can be seen as an alternative to OSM. It contains elements such as a gateway module, a MANO framework, a network slice manager, infrastructure abstraction, catalogues, repositories, a policy manager, a Service Level Agreement (SLA) manager, a monitoring manager and a portal.

SONATA is rather an OSM collaborator than a competitor, with various components that are already compatible or easily integrated with the OSM framework. SONATA is also mentioned in various scenarios in published OSM papers. One type of scenario is where the operator wants to cooperate with another operator to provide network service. For example, it may transfer the provision of infrastructure or a specialized VNF to another operator. The implication is that OSM needs an architecture that allows the orchestration of orchestrators, and the SONATA project contributes to this through its MANO framework [33].

4.3 Open source NFV MANO systems performance comparison

Open source MANO projects such as Open Network Automation Platform (ONAP) [34], OSM, Open Baton [35], Cloudify [36], OPNFV [37], are in various stages of constant development. Among these more prominent projects are Open Network Automation Platform (ONAP) and Open Source MANO (OSM), which have gained a lot of attention from the operator community, mainly due to the patronage of some of the large operators behind their development. For example, ONAP, which is being developed under the Linux Foundation umbrella, is mainly supported by AT&T, while OSM is led by Telefonica and is being developed by ETSI's recently established Open Source Group (OSG)..

Both ONAP and OSM are in various stages of deployment, but are far from complete or stable. Both aim to provide an integrated NFV MANO framework, but follow very different directions in terms of architecture and implementation. Due to their relatively recent development, there is very little information and experience available on the functional and operational capabilities of these platforms and the level of technological readiness. In addition, benchmarking the performance of MANO systems is itself a challenge. This is because, unlike other traditional network entities, which have well-defined Key Performance Indicators (KPIs) to assess performance, there are no established and well-defined KPIs against which the performance of a MANO system can be assessed..

4.4 Use cases

The OSM community brings together a lot of research projects implementing NFV orchestration using the OSM stack. Most of these projects are 5G oriented and use Openstack as VIM.

The 5GCity project [38] is building a platform that enables the use of a city's information and communication technology infrastructure in cloud-to-edge environments. 5GTango [39] is a 5GPPP project that provides flexible programmability of a 5G network with several components such as an NFV-oriented SDK, a VNF/NS storage platform with validation and verification mechanisms, and a modular service platform. The Metro-Haul project aims to design metro networks (agile, programmable and low power and cost) that should be 5G-enabled, including the design of fully optical metro nodes, including storage and compute capabilities [40]. The MATILDA project implements a 5G E2E service operational framework, focusing on the lifecycle of 5G-ready applications and 5G network services (design, development and orchestration) on both virtual and physical infrastructure using a unified programmability model [41].

4.5 Optimal alarm threshold detection using machine learning

The monitoring component (Figure 4.4) is responsible for collecting Virtual Deployment Unit (VDU) metrics provided by VIM, VNF-specific metrics via Juju charms and infrastructure metrics. It also takes care of storing the values of these metrics in the TSDB in Prometheus and manages and evaluates alarms. The MON module contains three modules: a server, a collector and an evaluator [42].

The POL module (Figure 4.11) has been designed for the autoscaling process and handles listening or configuring alarms, sending scaling messages and calling webhooks when alarm policies are met. [42].

The MON evaluator will assess the specific measurement thresholds and then POL will take the necessary actions, such as self-calibration. Whenever an alarm is triggered, MON will generate a notification and send it via the Kafka bus so that other components, such as POL, can consume that message.

4.5.1 Alarm generation

The alarm descriptor is also part of a VNF descriptor and specifies which metrics should be monitored, which metrics should already be defined in the monitoring list, the thresholds to be monitored and the webhook method to be invoked..

The current architecture allows OSM users to choose the alarm thresholds themselves, without any information or knowledge of what the appropriate threshold value should be. This should not be a problem for experienced users, but for new users or new use cases, this could cause some difficulties in choosing an appropriate alarm threshold.

In order to overcome this shortcoming, an improvement of the architecture has been proposed in this paper. Specifically, a fourth component called Predictor (shown in Figure 4.13) is introduced in the monitoring module, which applies a machine learning algorithm to predict an appropriate alarm threshold based on existing data from the common database (MongoDB).

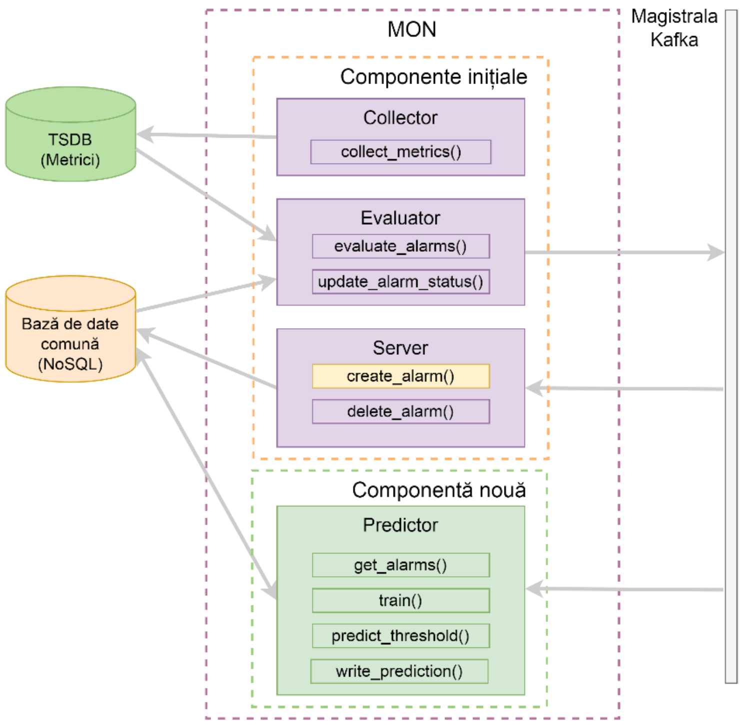


Figure 4.13 *Process proposal with integrated Predictor component.*

4.5.2 Predictor component

The Predictor component will be responsible for generating alarm threshold suggestions and predicting the best threshold value for specific use cases.

This new component will contain four main processes:

- get_alarms() which will query the common database and retrieve all stored alarms.
- train() which will be a background process scheduled to train the prediction model at specific time intervals.
- predict_threshold() is the process that will call the model prediction function to get the predicted value.
- write_prediction() is the final process that is responsible for writing the predicted threshold to the common database.

The Predictor component will listen on the Kafka bus for the new alarm creation request (topic: alarm_request; key: create_alarm_request) and, whenever this message is read, will trigger the start of the predict_threshold() process.

Since the predictor adds a predicted threshold value to the newly created alarm record, some modifications to the common database model are required. The alarm template will contain an additional field named "predicted_threshold" of type float. This change is also reflected in the message bus schema where the alarm structure is involved. In this way, an improved alarm threshold can be set automatically, independent of the OSM user's previous experience, improving other OSM processes such as auto-healing and ensuring better reliability.

4.6 Implementations

When creating new NSs containing custom VNFs, certain steps need to be followed to get a proper and functional network service ready for deployment. A good practice when creating new network services and virtual network functions is to create diagrams for both before starting to create their associated descriptor.

A series of implementations have been presented covering the following scenarios:

- NS with a single VNF
- NS with multiple VNFs and VDUs
- NS with Day0 configuration
- Network slice

Chapter 5

The automation platform

In this chapter, an automation platform has been presented to improve and simplify the process of deploying network slices through OSM. Its architecture, graphical interface and operating mode were reviewed as well as experiments covering various scenarios, highlighting the platform's functionalities.

As web applications have evolved from simple web pages to highly complex systems, libraries and frameworks have been improved to provide web developers with the best possible solutions. Modern web applications typically place logic on the client side instead of the server and dynamically retrieve data from a server API [43].

React este o bibliotecă JavaScript cu sursă deschisă dezvoltată de către Facebook şi utilizată pentru proiectarea de interfețe pentru utilizator [44]. Biblioteca a fost publicată în 2013 şi astăzi se numără printre cele mai populare instrumente utilizate pentru dezvoltarea web de front-end.

5.1 Architecture

OSM offers the ability to create network slices by using their software package which requires a VIM to create the necessary virtual machines. Communication between OSM and the VIM is achieved by sending messages via Hypertext Transfer Protocol (HTTP) to the VIM's APIs. In terms of the process of creating network slices using OSM, there are three different ways:

- UI,
- CLI commands,
- OSM NBI API.

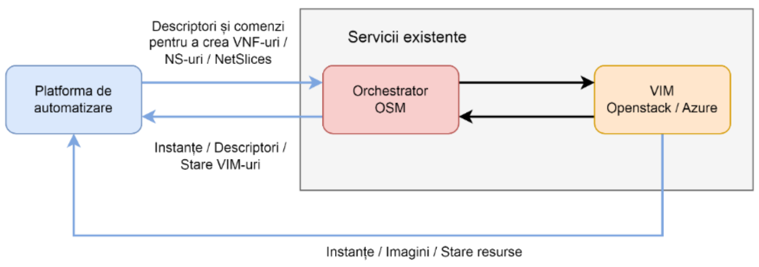

Descriptori şi comenzi pentru a crea VNF-uri / NS-uri / NetSlices
Servicii existente
Platforma de automatizare
Orchestrator OSM
VIM Openstack / Azure
Instanțe / Descriptori / Stare VIM-uri
Instanțe / Imagini / Stare resurse

Figure 5.1 *Automation platform workflow.*

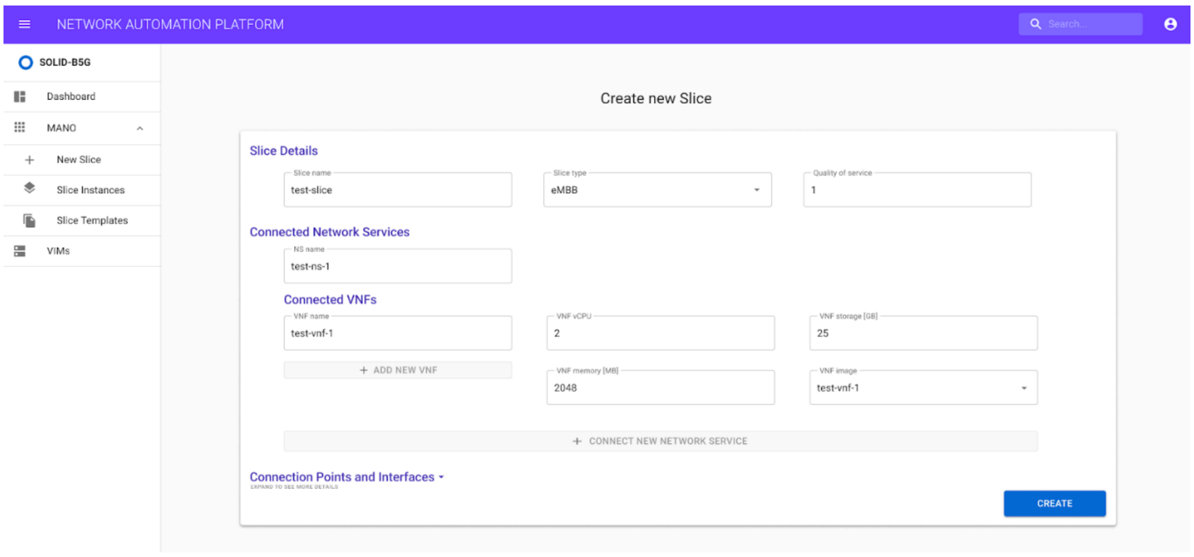
The automation platform is a web application developed with the ReactJS library, which uses HTTP calls to communicate with OSM and Openstack. Its interface is styled using the Material UI library.

The architecture (Figure 5.2) of this web application contains several components and services. Each component represents the graphical implementation of each page displayed in the user interface. The services have the following roles:

- Authentication: initiates initial communication with both OSM and Openstack to obtain access tokens that will be used by the other two services when they request different information.
- OSM service: handles all requests to the OSM NBI to retrieve all necessary data.
- Openstack service: handles all requests to Openstack API, which will forward the requested data such as available image list etc..

5.2 Graphic interface

In the following, the use and operation of the automation platform for different scenarios is presented. This interface provides the user with a number of actions such as: creating a new network slice, displaying network slice instances, displaying network slice templates or displaying available VIMs.



Create new Slice

Slice Details

Slice name: test-slice
Slice type: eMBB
Quality of service: 1

Connected Network Services

NS name: test-ns-1

Connected VNFs

VNF name: test-vnf-1
VNF vCPU: 2
VNF storage [GB]: 25
VNF memory [MB]: 2048
VNF image: test-vnf-1

Connection Points and Interfaces

Figure 5.3 *Automation Platform New Slice Page.*

5.3 Experiments

In order to achieve automatic network slice deployment, it is necessary to configure several parameters (processing, storage, memory, image, etc.) for each VNF and for each NS contained in this slice. This is an important step to develop a platform capable of automating and simplifying the process of deploying a network slice through an Open Source MANO (OSM) system [45].

A network slice can also be created from a generic use case. Each network service in the slice will be preconfigured using cloud-init scripts [46] on the instantiated machines. Instantiation will take place on a specific VIM, suited for the respective network slice type.

A new feature has been integrated into the automation platform to provide better resource allocation. The Resource Prediction (RP) component is able to optimize a network slice by providing predicted values for certain parameters such as CPU, memory, storage. The model used was the one presented in Section 2.3, with a single neuron dense layer that was compiled with a mean-square error loss function and the Adam optimizer with a learning rate of 0.1, the model was trained for 1000 epochs. The dataset was built using previous slice data previously implemented through the automation platform. Even though the dataset size is not very large, each new network slice implementation will add new data to the dataset, improving it. The dataset headers are Slice_Type, Use_Case, NS_Name, VNF_Name, vCPU, Memory and Storage. Currently, training is initiated manually, but a scheduled training function will be added as a next step.

Three different experiments were conducted to test different functionalities of the platform:

- **Predefined network slice for high-definition video streaming**: the slice was created from a use case, so for VNFs, the useful content will also contain the cloud-init script that will install the necessary media server in the case of ultra HD streaming.
- **Dedicated network slice for high performance computing**: a dedicated network slice to support parallel simulations for large-scale problems at scale and speed, providing high performance computing.
- **eMBB network slice with ML-optimized parameters**: to test the RP component, the general eMBB network slice was chosen as a scenario because the current dataset contains mostly eMBB slice implementation data, and the model should produce more accurate values.

Chapter 6

Conclusions

In this thesis, NFV and Network Slicing paradigms and their various use cases were studied. This study continued by using and evaluating two MANO frameworks with which various experiments were carried out. Following their evaluation, the OSM framework was chosen to continue the study and integrate it into a new network slicing automation platform.

The advantages of ML techniques and their usability in network slicing scenarios were also presented. On this basis, an architectural contribution to the OSM monitoring module was proposed, which will increase its efficiency and flexibility. The proposed new sub-module will be involved in the alarm workflow, generating an alarm threshold value recommendation for each parameter for which an alarm is configured. The recommended alarm threshold value is generated by using a prediction algorithm ML.

The same type of algorithm is used in a resource prediction block that is implemented and tested in an automation platform. With this new block, whenever a user starts the process of creating a new network slice through the automation platform, for each slice type or use case, the recommended values will be displayed as input fields, allowing the user to configure better resource values based on previous slice deployment data.

Using this automated platform, deploying a network slice through the MANO - VIM open source service chain becomes much faster and easier. Adding the resource prediction block optimizes network slices both at the deployment and installation level on the right infrastructure, as well as adaptively changing alarm and scalability thresholds. Thanks to the platform's ability to retrieve data about instantiated slices and services from OSM and Openstack, service operators will have a better view of these slices and services before slice deployment.

The solutions proposed in this paper aim to improve the current network slice management workflow through automated mechanisms that can optimally allocate resources and adapt alarm and scaling thresholds to optimize network slice operation.

6.1 Obtained results

The first chapter is an introduction to the scope of the thesis and its purpose and content.

Chapter 2 contains a broad overview of machine learning concepts and tools used in specific network slicing cases, a comparison of optimization functions using an experiment as well as a review of other projects studying the use of machine learning in network slicing. This chapter has led to a better understanding of machine learning technologies and their applicability to network slicing. The experimental results facilitated the choice of an optimization function and the development of a prediction model that was used in the following chapters.

Chapter 3 presents the SONATA framework, starting with a review of other similar projects and continuing with its detailed architecture and a comparison with the ETSI NFV model. Furthermore, a series of experiments using the SONATA emulator are presented. The results of this chapter were the understanding of the life cycle of a network service as well as the communication between an orchestrator and a virtual network manager through the experiments, representing an important starting point for understanding how a MANO framework works..

Chapter 4 discusses the Open Source MANO framework starting with its architecture. A parallel between this framework and SONATA is reviewed to highlight the more complex architecture of Open Source MANO. An analysis of the performance of Open Source platforms is performed and different use cases and projects using OSM are presented. An improvement of the architecture at the monitoring module level by detecting an optimal alarm threshold using ML algorithms has been proposed and, in the end, implementations divided into different scenarios have been realized. As an outcome of this chapter, it can be mentioned the understanding of the OSM architecture, which led to a proposal to improve the architecture using ML algorithms in order to detect an optimal alarm threshold.

Chapter 5 introduces the automation platform. This is a new component added to an existing service chain (Open Source MANO and a virtual infrastructure manager) to improve the process of deploying and managing network slices. The architecture of this platform and its role in the new service chain is presented, as well as its graphical user interface along with its building blocks. A series of experiments covering various scenarios are presented in the last part of the chapter. The results of the chapter are the definition of the automation platform architecture, its implementation and the addition of automation and resource prediction blocks. The implementation of the platform was validated by performing experiments covering different scenarios, highlighting its functionalities.

6.2 Original contributions

The original contributions made during the research activity within the doctoral stage are the following:

1. Summary study of NFV and Network Slicing paradigms [6.3 – 2];

2. Summary study of ML concepts and tools applicable to network slicing [6.3 – 1];

3. Running the optimization function comparison experiment and interpreting its results [6.3 – 1];

4. Summary study of the architectural components of the SONATA framework [6.3 – 6,7,10];

5. Parallel between the SONATA framework and the ETSI NFV mode [6.3 – 6,7,10];

6. Performing various experiments using the SONATA emulator [6.3 – 6,7,10];

7. Summary study of the architectural components of the OSM framework [6.3 – 19];

8. Comparison of the OSM framework with SONATA at architecture level and with ONAP at performance level [6.3 – 19];

9. Proposal of a new sub-module for optimal alarm threshold detection using ML algorithms within the OSM monitoring module architecture [6.3 – 1];

10. Defining the architecture of the automation platform [6.3 – 3];

11. Design of the graphical user interface of the automation platform [6.3 – 3];

12. Implementation of the automation platform as a web application [6.3 – 2];

13. Implementation and integration of a resource prediction block in the automation platform [6.3 – 1];

14. Validation of automation platform functionalities through experiments [6.3 – 1,2];

6.3 List of original publications

During the research activity within the doctoral stage, the author has published a total of 14 scientific papers related to the PhD thesis field, 6 as first author and 6 indexed in IEEEXplore. 5 scientific reports have also been added to the list below. During the PhD training, the author was also involved in two research projects.

1. **Cosmin Conțu**, Eugen Borcoci, Marius-Constantin Vochin, Frank Y. Li and Alexandru Aloman, *Machine Learning-based Monitoring in Network Slice Creation and Resource Management*, IEEE Access, SUBMITTED (**ISI)**

2. **Cosmin Conțu**, Andra Ciobanu, Eugen Borcoci, Marius-Constantin Vochin, Indika A. M. Balapuwaduge, Silviu Topoloi and Razvan-Florentin Trifan, *Deploying Use Case Specific Network Slices Using An OSM Automation Platform*, 25th International Symposium on Wireless Personal Multimedia Communications (WPMC) 2022, Herning, Denmark, October 30 – November 2, 2022 (**IEEEXplore**);

3. **Cosmin Conțu**, Andra Ciobanu, Eugen Borcoci, Marius-Constantin Vochin, Frank Y. Li, *An Automation Platform for Slice Creation using Open Source MANO*, Proceeding of The 14th International Conference on

COMMUNICATIONS, COMM 2022, Bucharest, ROMANIA, June 16-18, 2022, ISBN: 978-1-6654-9485-4 (**ISI, IEEEXplore**);

4. **C. Conțu**, I. Cioarcă, M. Ene, L. Nichifor, *Study on Optimal Convolutional Neural Networks Architecture for Traffic Sign Classification Using Augmented Dataset*, Proceeding of The 13th International Conference on COMMUNICATIONS, COMM 2020, Bucharest, ROMANIA, June 18-20, 2020, ISBN: 978-1-7281-5611-8 (**ISI, IEEEXplore**);

5. Andra-Isabela-Elena Ciobanu, **Cosmin Conțu**, Eugen Borcoci, Marius-Constantin Vochin, and Frank Y. Li, *Optimal Service Placement with QoS Monitoring in NFV and Slicing Enabled 5G IoT Networks*, 2021 IEEE Globecom Workshops (GC Wkshps): IEEE IoST-5G&B: 2nd Workshop on Recent Trends of Internet of Softwarized Things - 5G & B, Madrid, Spain, 7–11 December 2021 (**IEEEXplore**);

6. A. Țapu, **C. Conțu**, E. Borcoci, *Multiple Chained Virtual Network Functions Experiments with SONATA Emulator*, Proceeding of The 12th International Conference on COMMUNICATIONS, COMM 2018, Bucharest, ROMANIA, June 14-16, 2018, ISBN: 978-1-5386-2350-3 (**ISI, IEEEXplore**);

7. A. Țapu, **C. Conțu**, E. Borcoci, *Network Function Virtualization Experiments using SONATA Framework*, Proceeding of The International Symposium on Advances in Software Defined Networking and Network Functions Virtualization SOFTNETWORKING 2018, Athens, GREECE, April 22-26, 2018, ISBN: 978-1-61208-625-5 (**Best Paper Award**);

8. A. Ciobanu, **C. Conțu**, E. Borcoci, *Study on Use-Cases of Open Source Management and Orchestration Framework in 5G Projects*, Proceeding of The Nineteenth International Conference on Networks 2020, Lisbon, Portugal, February 23-27, 2020, ISBN: 978-1-61208-770-2 (**Best Paper Award**);

9. A. Ciobanu, **C. Conțu**, E. Borcoci, *Charms and Virtual Network Functions Primitives Experiments using Open Source MANO Framework*, Proceeding of The Sixteenth Advanced International Conference on Telecommunications AICT 2020, Lisbon, Portugal, September 27 – October 1, 2020, ISBN: 978-1-61208-802-0 (**Best Paper Award**);

10. **C. Conțu**, A. Țapu, E. Borcoci, *Virtual Network Function Use Cases Implemented on SONATA Framework*, International Journal on Advances in Networks and Services, issn 1942-2644 vol. 11, no. 3 & 4, year 2018, 103:112;

11. E. Borcoci, A. Ciobanu, **C. Conțu**, *Layered Network Domain Resource Management in Multi-domain 5G Slicing Environment*, Proceeding of Advanced Information and Communication Technologies-2019, Nice, France, July 28-August 2, 2019, ISBN: 978-1-61208-727-6;

12. E. Borcoci, **C. Conțu**, A. Ciobanu, *5G Slicing Management and Orchestration Architectures - Any Convergence?,* Proceeding of The Eleventh International Conference on Advances in Future Internet AFIN 2019, Nice, France, October 27-31, 2019, ISBN: 978-1-61208-747-4;

13. E. Borcoci, **C. Conțu**, A. Ciobanu, *On Heterogeneity of Management and Orchestration Functional Architectures in 5G Slicing*, International Journal on Advances in Internet Technology, vol 13 no 1 & 2, year 2020, 83-96;

14. **Cosmin Conțu**, Eugen Borcoci, Marius-Constantin Vochin and Frank Y. Li, *Automating Network Slices in Open Source MANO*, Proceedings of the Doctoral Symposium on Electronics, Telecommunications, & Information Technology SD-ETTI 2023, October 4-6, 2023, Bucharest **(DBLP)**

15. Eugen Borcoci, Andra Țapu, **Cosmin Conțu**, *D1.2 Virtual Evolved Packet Core*, ORANGE Romania – UPB cooperation project “5G Technologies”, 2018;

16. **C. Conțu**, *Experimente de Virtualizare a Funcțiilor de Rețea utilizând Framework-ul SONATA*, Raport Științific Nr. 1 (nepublicat), Universitatea Politehnica din Bucureşti, Romania, Iunie 2018;

17. **C. Conțu**, *Cazuri de Funcții Virtualizate de Rețea Înlănțuite Implementate utilizând Framework-ul SONATA*, Raport Științific Nr. 2 (nepublicat), Universitatea Politehnica din Bucureşti, Romania, Decembrie 2018;

18. **C. Conțu**, *Cazuri de Funcții Virtualizate de Rețea Implementate pe Infrastructură Fizică utilizând Framework-ul Open Source Management and Orchestration (MANO),* Raport Științific Nr. 3 (nepublicat), Universitatea Politehnica din Bucureşti, Romania, Iunie 2019;

19. **C. Conțu**, *Managementul Funcțiilor Virtualizate de Rețea și Implementarea unui “Slice” de Rețea Utilizând Framework-ul Open Source Management and Orchestration (MANO) – Studii de caz*, Raport Științific Nr. 4 (nepublicat), Universitatea Politehnica din Bucureşti, Romania, Decembrie 2019;

20. **C. Conțu**, *Despre Eterogenitatea Administrării și Orchestrării a Arhitecturilor Funcționale în Felierea Rețelelor 5G*, Raport Științific Nr. 5 (nepublicat), Universitatea Politehnica din Bucureşti, Romania, Iunie 2020;

6.3.1 List of research projects

1. **Februarie – Mai 2018**: *SLICENET: End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks* (collaboration between Universitatea Politehnica of Bucharest and Orange Romania);

2. **Noiembrie 2020 – Decembrie 2023**: *A Massive MIMO Enabled IoT Platform with Networking Slicing for Beyond 5G IoV/V2X and Maritime Services*;

6.4 Perspectives for further developments

The automation platform presented in this paper has succeeded in simplifying the process of deploying a network slice and giving the operator better control and insight into the lifecycle of the network slice.

As perspectives for further development, both the quality and quantity of the datasets will need improvement, the training phase of the resource prediction block will be automated, and the automation platform will be extended with more features. Regular updates will also be required to maintain compatibility with new OSM releases through collaboration with the OSM community, where the author's organization is registered as an official Participant.

Another direction of development of the platform is to increase its compatibility with several types of VIM already compatible with OSM (Amazon Web Services, Microsoft Azure and Google Cloud Platform).

Nevertheless, new use cases for ML algorithms will be studied within the platform and new blocks will be implemented to support these cases.

Bibliography

[1] 5G PPP Architecture Working Group, *View on 5G Architecture*, Version 1.0, July 2016. Available: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-July-2016.pdf [Online]

[2] NFV White paper: *Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action. Issue 1*. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf, 2018 [Online]

[3] R. Mijumbi et al., *Network function virtualization: State-of-the-art and research challenges*, IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 236-262, 1st Quart. 2016

[4] 5GPPP Architecture Working Group, *View on 5G Architecture*, Version 3.0, June, 2019, Available: https://5gppp.eu/wp-content/uploads/2019/07/5G-PPP-5GArchitecture-White-Paper_v3.0_PublicConsultation.pdf, [Online].

[5] J. Ordonez-Lucena et al., *Network Slicing for 5G with SDN/NFV: Concepts, Architectures and Challenges*, IEEE Communications Magazine, 2017, pp. 80-87, Citation information: DOI 10.1109/MCOM.2017.1600935.

[6] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, *Network Slicing in 5G: Survey and Challenges*, IEEE Communications Magazine, May 2017, pp. 94-100

[7] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, *Network Slicing & Softwarization: A Survey on Principles, Enabling Technologies & Solutions*, IEEE Communications Surveys & Tutorials, March 2018, pp. 2429-2453.

[8] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, *On Multi-domain Network Slicing Orchestration Architecture & Federated Resource Control*, Available: http://mosaic-lab.org/uploads/papers/3f772f2d-9e0f-4329-9298-aae4ef8ded65.pdf, 2019 [Online].

[9] ETSI GS NFV 002, *NFV Architectural Framework*, V1.2.1, December, 2014

[10] ETSI GS NFV-IFA 009, *Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options*, Technical Report, V1.1.1, July, 2016

[11] ETSI GR NFV-IFA 028, *Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on architecture options to support multiple administrative domains*, Technical Report, V3.1.1, January, 2018.

[12] ONF TR-526, *Applying SDN Architecture to 5G Slicing*, April 2016

[13] 3GPP, "System architecture for the 5G System (5GS)," TS 23.501, R17, v17.5.0, Jun. 2022.

[14] *An Introduction to Machine Learning*,. Available: https://monkeylearn.com/machine-learning/ 2023. [Online]

[15] *Deep learning vs. Machine learning vs. Artificial Intelligence* Available: https://www.javatpoint.com/deep-learning-vs-machine-learning-vs-artificial-intelligence/, 2023. [Online].

[16] *scikit-learn Machine Learning in Python,* Available: https://scikit-learn.org/stable/, 2023. [Online].

[17] *PyTorch,* Available from: https://pytorch.org/, 2023. [Online].

[18] *NLTK :: Natural Language Toolkit,* Available: https://www.nltk.org/, 2023. [Online].

[19] *TensorFlow,* Available: https://www.tensorflow.org, 2023. [Online].

[20] *Mean squared error,* Available: https://en.wikipedia.org/wiki/Mean_squared_error, 2023. [Online].

[21] *AdaGrad,* Available: https://www.databricks.com/glossary/adagrad, 2023. [Online].

[22] *Intuition of Adam Optimizer,* Available: https://www.geeksforgeeks.org/intuition-of-adam-optimizer/, 2023. [Online].

[23] *A Look at Gradient Descent and RMSprop Optimizers,* Available: https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b/, 2023. [Online].

[24] *Build your first Machine Learning Model using TensorFlow,* Available: https://techwithshadab.medium.com/build-your-first-machine-learning-model-using-tensorflow-d61b9b2b7d5e/, 2023. [Online].

[25] C. Conțu, I. Cioarcă, M. Ene, L. Nichifor, *Study on Optimal Convolutional Neural Networks Architecture for Traffic Sign Classification Using Augmented Dataset*, Proceeding of The 13th International Conference on COMMUNICATIONS, COMM 2020, Bucharest, ROMANIA, June 18-20, 2020, ISBN: 978-1-7281-5611-8

[26] S. Dräxler, H. Karl, M. Peuster, H. R. Kouchaksaraei, M. Bredel, J. Lessmann, T. Soenen, W. Tavernier, S. Mendel-Brin, and G. Xilouris, *Sonata: Service programming and orchestration for virtualized software networks*, in 2017 IEEE International Conference on Communications Workshops (ICC Workshops), May 2017, pp. 973–978

[27] Sevil Draxle et al., *SONATA: Service Programming and Orchestration for Virtualized Software Networks*, 2017 IEEE International Conference on Communications Workshops (ICC Workshops).

[28] SONATA. *D2.2 Architecture Design*. Available: http://sonata-nfv.eu/sites/default/files/sonata/public/contentfiles/pages/SONATA_D2.2_Architecture_and_Design.pdf, [Online].

[29] *The SONATA Gatekeeper*, Available: http://sonata-nfv.eu/sites/default/files/sonata/public/content-files/article/SONATA_Gatekeeper_SDNWorld_3.pdf, 2018 [Online].

[30] *Docker - Build, Ship, and Run Any App, Anywhere*. Available: https://www.docker.com/, 2018 [Online].

[31] Gerardo García de Blas, *OSM architecture*, Available: https://osm-download.etsi.org/ftp/osm-11.0-eleven/OSM12-hackfest/presentations/OSM%2312%20Hackfest%20-%20OSM%20architecture.pdf, 2023. [Online].

[32] *Research – OSM Public Wiki*, Available from: https://osm.etsi.org/wikipub/index.php/Research#Sonata, 2019, [Online].

[33] A. Ciobanu, C. Conțu, E. Borcoci, *Study on Use-Cases of Open Source Management and Orchestration Framework in 5G Projects*, Proceeding of The Nineteenth International Conference on Networks 2020, Lisbon, Portugal, February 23-27, 2020, ISBN: 978-1-61208-770-2

[34] ONAP, *ONAP Architecture Overview*, Available from: https://www.onap.org/architecture, 2019, [Online].

[35] G. A. Carella and T. Magedanz, *Open baton: A framework for virtual network function management and orchestration for emerging software-based 5G networks*, IEEE Softwarization, July 2016.

[36] Cloudify, *Cloudify Orchestration Project Portal*, Available from: https://cloudify.co/, 2019, [Online].

[37] OPNFV, *Open Platform for NFV (OPNFV) Project Portal*, Available from: https://www.opnfv.org/, 2019, [Online].

[38] *5GCity – A distributed cloud & radio platform for 5G Neutral Hosts*, Available: https://www.5gcity.eu/, 2022, [Online].

[39] *5GTANGO*, Available: https://www.5gtango.eu/ 2022, [Online].

[40] *METRO-HAUL 5G Project*, Available: https://metrohaul.eu/ 2022, [Online].

[41] *MATILDA*, Available: https://www.matilda-5g.eu/ 2022, [Online].

[42] Subhankar Pal, *HD4.3 Closed-Loop Operations: Adding Auto-Scaling & Alerting to VNFs*, Available: https://osm-download.etsi.org/ftp/osm-8.0-eight/OSM-MR9-hackfest/presentations/OSM-MR%239%20Hackfest%20-%20HD4.3%20-%20Closed-Loop%20Operations.pdf, 2023. [Online].

[43] D. Johansson, *Building maintainable web applications using React*, Available: https://www.divaportal.org/smash/get/diva2:1415320/FULLTEXT01.pdf, 2022, [Online].

[44] *React – A JavaScript library for building user interfaces*, Available: https://reactjs.org/ 2022, [Online].

[45] ETSI, *Open source MANO*, Available: https://osm.etsi.org/, 2022. [Online].

[46] Canonical, *Cloud init – The standard for customising cloud instances,* Available: https://cloud-init.io/, 2022. [Online].