UNIVERSITY OF OULU

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

DEGREE PROGRAMME IN ELECTRONICS AND COMMUNICATIONS ENGINEERING
DEGREE PROGRAMME IN WIRELESS COMMUNICATIONS ENGINEERING

MASTER'S THESIS

END-TO-END NETWORK SLICING ARCHITECTURE AND IMPLEMENTATION FOR 5G MICRO-OPERATOR LEVERAGING MULTI-DOMAIN AND MULTI-TENANCY

Author	Idris Badmus
Supervisor	Adjunct Prof. Marja Matinmikko-Blue
Second Examiner	Prof. Ari Pouttu

November 2019

Badmus I. (2019) Thesis Title in English. University of Oulu, Faculty of Information Technology and Electrical Engineering, Degree Programme in Electrical Engineering Degree Programme in Wireless Communications Engineering. Master's Thesis, 52 p.

ABSTRACT

Local 5G network are emerging as new form of 5G deployment targeted are service delivery for vertical specific purposes and other local users. As such, a well-defined network slicing architecture and implementation procedure is required for a local 5G network. A local 5G network also known as a 5G micro-operator network is targeted a network delivery for vertical-specific services. The aim of the micro-operator concept is to provide enough network flexibility and customization required by different vertical. Previous works on the micro-operator network have established different deployment scenarios that can exist, namely Closed, Open and Mixed Network. Thus, in order for any deployment of a micro-operator network to achieve the network flexibility, customization and privacy required by various vertical, it is essential to have a well-defined network slicing architecture and implementation procedure for local 5G networks. In this thesis, a sophisticated end-to-end network slicing architecture is proposed for different deployment scenarios of a local 5G micro-operator. The aim of the architecture is to address the unavailable description of network slicing for vertical specific network providers, leveraging multi-domain and multi tenancy. The proposed architecture incorporates a broad four-layer concept, leveraging a Multi-tenancy layer for different tenants and their end users, a descriptive Service layer, a multi-domain Slicing MANO layer and a Resource layer. A message sequence diagram is established based on the proposed architecture to describe the flow of information from when a tenant request a slice till the network slices are allocated as communication services to the various targeted user equipment. An actual implementation of network slicing is developed for specific layers of the proposed architecture. To do this, we used a softwarized network based on SDN/NFV, using OpenStack as a cloud infrastructure. On top of that, the network slicing implementation was done using the ETSI Open Source MANO. With these tools, different deployment scenarios' implementations are achieved. Performance analysis are made based on metrics such as CPU utilization, memory utilization, rate of packet sent and packet received between different network service. These metrics are used to compare shared and non-shared slices within a single or multiple domain slice implementation, which were used as basis for classification of network slice instantiation in 5G micro-operator deployment scenarios. The results from the thesis successfully support the end-to-end network slicing architecture for various deployment scenarios of a local 5G micro-operator network, proposes a slice formation sequence from the end users to the micro-operator network for each deployment scenarios, implement different part of the architecture using different open source tools and measure the performance metrics of different deployment scenarios based on CPU or memory utilization.

Key words: Network Slicing, Micro-Operator, SDN/NFV, Network Softwarization, Virtualization, Orchestration, Multi-Tenancy.

TABLE OF CONTENTS

FOREWORD

The primary focus of this thesis is to develop and end-to-end scalable network slicing architecture for 5G micro-operator deployment scenario. This work was partially supported by the European Union's Horizon 2020 Research and Innovation Program under the 5G!Drones project (Grant No. 857031), Business Finland in the MOSSAF Project and by the Academy of Finland 6Genesis Flagship (Grant No. 318927). First of all I would like to thank Allah for his countless blessing and mercies in my life. I would like to express my deep sense of gratitude and appreciation to my supervisor Dr. Marja Matinmikko-Blue for providing me the opportunity to learn and work under her patronage. Her constant support, guidance and advices from the inception of my research career till date are invaluable to the success of this work. I would like to also express my deepest gratitude to Prof. Ari Pouttu for the opportunity and trust he has given me to work under him and learn enormously. Special thanks to my friends and colleague Dr. Abdelquoddouss Laghrissi and Dr. Bidushi Barua for their advice and supports towards this thesis.
I would like to thank my parents Mr and Mrs Badmus, my siblings Fatimah, AbdulRahman, Ganiyat and Sodiq for their continuous supports and prayers in my life till date. I would also like to thank my friends and colleagues here in CWC, Mahmoud Goda, Mubarak Aminu, Muhammad Arif, and Nuru Huda,
Finally, I dedicate this work to my fiancé Barakat Adeyemo Adedamola for her unending support and love. Thanks for always dear, I love you.

Oulu, November 13 2019

Idris Badmus

LIST OF ABBREVIATIONS AND SYMBOLS

3GPP	3rd Generation Partnership Project
4G	Fourth mobile generation
5G	Fifth mobile generation
API	Application Programming Interface
AR	Augmented Reality
BSS	Business Support Systems
CCN	Common Core Network
CN	Core Network
CPU	Central Processing Unit
CSMF	Communication Service Management Function
DCN	Dedicated Core Network
EMBB	Enhanced Mobile Broadband
EPC	Evolve Packet Core
ETSI	European Telecommunications Standards Institute
HSS	Home Subscriber Server
IMT	International Mobile Telecommunication
InP	Infrastructure Providers
ISP	Internet Service Provider
IT	Information Technology
ITU	International Telecommunication Union
KPI	Key Performance Indicator
LCM	Life Cycle Manager
LTE	Long Term Evolution
MANO	Management and Network Orchestration
MDD	Multi-Dimensional Descriptor
MEC	Multi-access Edge Computing
MME	Mobility Management Entity
MMTC	Massive Machine Type Communication
MNO	Mobile Network Operator
MVNO	Mobile Virtual Network Operator
NFV	Network Function Virtualization
NFVI	Network Functions Virtualization Infrastructure
NFVO	Network Functions Virtualization Orchestrator
NGMN	Next Generation Mobile Networks
NS	Network Service
NSD	Network Service Descriptor
NST	Network Slice Template
NSI	Network Slice Instance
NSSI	Network Slice Subnet Instance
NSMF	Network Slice Management Function
NSSMF	Network Slice Subnet Management Function
ONAP	Open Network Automation Platform
ONF	Open Network Foundation
OSM	Open Source MANO
OSS	Operation Support System
PNF	Physical Network Function

QoS	Quality of Service
QoE	Quality of Experience
RAM	Random Access Memory
RAN	Radio Access Network
RO	Resource Orchestrator
SDD	Solid State Drive
SDK	Software Development Kit
SDM-C	Software Defined for Mobile Network Control
SDM-O	Software Defined for Mobile Network Orchestrator
SDM-X	Software Defined for Mobile Network Coordinator
SDN	Software Defined Network
SLA	Service Legal Agreement
SR-IOV	Single Root Input Output Virtualization
TN	Transport Network
TOSCA	Topology and Orchestration Specification for Cloud Application
URLLC	Ultra Reliable Low Latency Communication
VDU	Virtual Descriptor Unit
VIM	Virtualized Infrastructure Machine
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
VNFM	Virtualized Network Function Manager
VR	Virtual Reality
WIM	Wireless Infrastructure Machine

1 INTRODUCTION

The fifth mobile generation (5G) is expected to provide better network connectivity and reliability than the existing fourth mobile generation (4G) network (4g). Apart from the three key International Telecommunication Union (ITU) IMT-2020 [1] requirements for 5G which include Enhanced Mobile Broadband (eMBB), Ultra-Reliable Low Latency Communication (uRLLC) and Machine-to-Machine Type Communication (mMTC), 5G is also expected to pave ways for new enablers supporting local deployment of mobile connectivity for serving local needs without reliance on existing Mobile Network Operators (MNOs).

The concept of a 5G micro-operator involves having a completely new local 5G network away from the existing MNO network in terms of radio spectrum and infrastructure. The operations of the micro-operator network are targeted at specific vertical services [2], thus, providing the required network customization, flexibility, and isolation. These requirements have prompted various studies into how having a separate locally deployed 5G micro-operator network serves as an enabler for new business models [3], [4]. Micro-operator networks are expected to be deployed with 5G along network slicing. Network slicing defines the means of logically isolating network capabilities from a “one size fits all” to a set of different slices where each slice is responsible for specific network requirements [5], [6]. Since the micro-operator network is targeted at vertical specific services, each with different network requirements, it will be imperative to have a micro-operator network that can serve multiple verticals, and the only way to achieve that is network slicing in a micro-operator. Network slicing will make it possible for a single micro-operator to serve multiple verticals with different network requirement efficiently while providing the required network flexibility required by these verticals.

1.1 Background and motivation

Network slicing has shared benefits. A study carried out by [6], [7] , shows that while network slicing is available to increase network customization and provide tailored network services for various end users, it also proves to be a new form of revenue generation for operators, lowering Operational Expenses (OpEx) and increasing Capital Expenditures (CapEx) efficiency based on the number of slices that can be created. Thus, in light of creating/establishing a micro-operator network to ensure better network delivery for verticals specific services, if the deployment cost is considered, it won’t be economically feasible for stakeholders to deploy a separate end-to-end network for different verticals. Hence, having a micro-operator network with services of which are sliced over different tenants will ensure enough revenue for the micro-operator stakeholders and provide the required tailored network services for the verticals.

In lieu of this and to achieve an efficient network slicing for a micro-operator network, different deployment scenarios that can exist for a micro-operator are described in [8]–[10]. The deployment scenarios include the closed, open and mixed micro-operator networks. Each of the deployment scenarios of a micro-operator are described to cover all possible cases that can exist for a micro-operator. The Closed micro-operator, also described as vertical specific network providers [11] are targeted at the humans/machines present within a closed user group. The Open micro-operator network is targeted at serving subscribers from one or more MNOs within a locality while the mixed network combines both the capabilities of the open and closed micro-operator networks.

Generally, different network slicing architectures are proposed for many parameters, including applications, use cases, services implementation techniques etc. Since network slicing in a micro-operator network is targeted for vertical specific services, and each of these verticals composes specific use cases, it is important to have a descriptive architecture that will fulfill achieving network slicing for micro-operators. The proposed network slicing architecture should be able to properly implement all deployment scenarios considered for a micro-operator efficiently, while maintaining the required network reliability, and desired quality of service (QoS) /quality of experience (QoE).

However, existing network slicing architectures and implementation procedures have all been directed towards regular Mobile Network Operators (MNOs) [6], [12], [13] with little description for slicing in a micro-operator network. Thus, extensive research is required to properly define how a network slicing architecture can be achieved in a locally deployed 5G micro-operator network [14], [15].

1.2 Thesis contribution

To properly define network slicing for a micro-operator, it is paramount to leverage existing network softwarization concept such as Software Defined Networks and Network Function Virtualization (SDN/NFV) [16]–[18]. The network softwarization will be achieved while maintaining network resources across different administrative and technological domains [19]. In this light, the contributions in this thesis are as follows:

- Thesis defines the concept of achieving multi-tenancy and multi-domain for network slicing in a locally deployed 5G micro-operator network.
- End-to-end network slicing architecture is proposed for various deployment scenarios of a micro-operator network. The proposed architecture will proffer a solution in determining the slicing end product in terms of established communication service transmitted from tenants back to individual user equipment. Based on this, four broad layers are described in the proposed architecture and it include the multi-tenancy layer, the service layer, the slicing MANO layer and the Resource layer. The architecture also involves establishing separate orchestrators at different layer for, thereby ensuring proper integration of slicing procedures from a slice creation request to the slice allocation.
- The thesis also describes the messages formation sequence support through an interactive description to further explain how to achieve the proposed architecture for the slice creation.
- Network slicing implementation is developed based on the proposed architecture for different deployment scenarios of a micro-operator. The implementation was achieved leveraging network softwarization based on SDN/NFV. Open source tools such as OSM for management and network orchestration (MAN0), OpenStack for cloud infrastructure, Promethus for exposing the metrics etc.
- Performance of the proposed slicing architecture is evaluated using metrics such as CPU utilization, memory utilization, packet sent, packet received, and VM usage analysis between different network slice subnet instances, to compare shared and non-shared slices across single or multiple domain. Since the basis for classification of each deployment scenarios [8] were based on shared and non-shared constituents across

different domains, the conclusion of the performance analysis describes the different parameters that affect each deployment scenario of a 5G micro-operator regarding actual implementation.

1.3 Outline of thesis

Herein, we give the reader an outlook of the remaining of the thesis organization as follows:

- **Chapter 2:** discusses the basic literature review regarding existing work on micro-operator deployment scenarios, key terminologies, and existing work on network slicing architecture from the perspective of standard bodies, open sources and different projects.
- **Chapter 3:** introduces the idea of leveraging multi-tenancy and multi-domain form achieving network slicing in a micro-operator network.
- **Chapter 4:** presents the proposed end-to-end network slicing architecture for 5G micro-operator deployment scenarios and describes the slice formation sequence for the proposed architecture.
- **Chapter 5:** presents the real implementation procedure for achieving network slicing based on the proposed architecture and highlights the performance analysis based on the measured metrics from the implementation
- **Chapter 6:** consists of the conclusion section, the general summary and proposed future works.

2 EXISTING WORK

This chapter present existing work in this research performed on micro-operator network, network slicing architecture and network slicing implementation for micro-operator.

2.1 Key terminologies and their definations

To better describe a micro-operator network, it is important that we define some of the terms used frequently in this paper. These definitions are based on available standards and approaches that are used in describing a micro-operator network

- **Micro-operator network:** A locally deployed 5G network targeted at vertical specific services delivery [2], [20]. The micro-operator concept is the idea of having a separate local network different from the existing MNO network in terms of radio resources or infrastructure and are targeted at network delivery for verticals. The resources used in making up the micro-operator network can be own directly by the operator or lends from different infrastructure providers (InPs).
- **Deployment scenarios:** Deployment scenarios describes the classification types to which a micro-operator network can be deployed and maintained. Basically A micro-operator network can be Closed, Open or Mixed micro-operator and each of the micro-operator have their own deployment scenarios [8].
- **Verticals:** A vertical is a specialized term use to describe network services specific to customers with similar requirements, e.g. an industrial vertical, health vertical, transportation and logistics vertical etc. A micro-operator network might be responsible for a single vertical or multiple verticals within a locality or multiple localities [21].
- **Tenants:** In the context of a micro-operator, we defined tenants as entities within a vertical that requires communication network connectivity and services for their operation. For instance in a locality with an industrial vertical represented by an industrial area with multiple factories and workshops [8], [21]. Each workshop or factory within the vertical can be described as a tenant. From an MNO point of view, a Mobile Virtual Network Operator (MVNO) can be described as a Tenant.
- **End-Users:** End users are ultimate customers within a tenant that consumes the communication services provided by the micro-operator. For instance in an industrial vertical, mobile robots, Augmented Reality (AR), employees within a tenant subscribed to a classical communication services like voice and video etc. The end users are assigned the communication services within a network slice.
- **Network Resources:** A network resource is a basic element used in creation of a network. A network resources can be a Physical Network Resources (PNF) or Virtual Network Resources (VNF). A network resources can come in different types, this include connectivity resources, computing resources and storage resources.
- **Domain:** A domain is a topology of network resources and devices that are coordinated and administered as a single unit with common rules and procedure under a single administrator [19]. A micro-operator network resources can be gotten from a single domain or multiple domain managed by a single or multiple administrator respectively.
- **Internet Service Provider (ISP):** ISPs are organizations responsible for providing access to the internet for network operators or private firms.

To better grasp the above definitions, Figure 1 describes a general model of a micro-operator network.

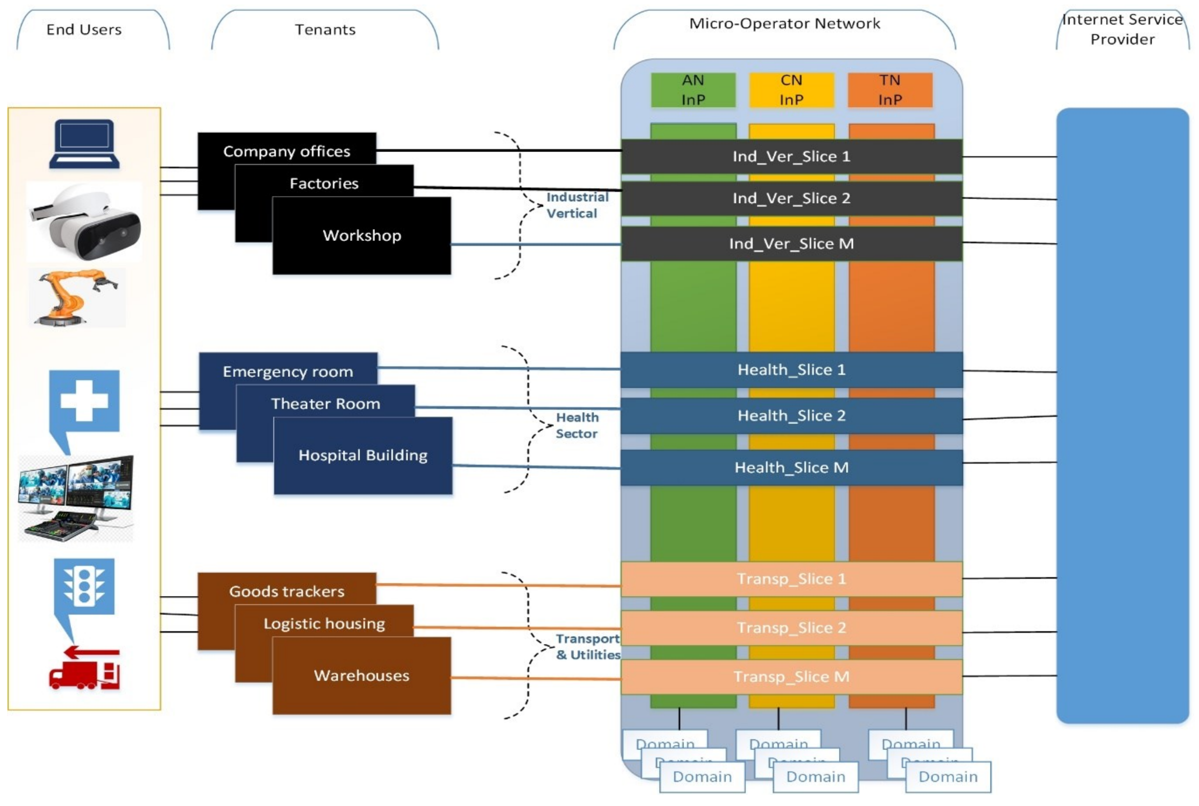


Figure 1. General micro-operator model.

2.2 Micro-operator network and it's deployment scenarios

The idea of having a completely new locally deployed network away from the regular MNO's network in terms of radio spectrum and infrastructure is the basis the micro-operator network [2]. Generally, the concept of micro-operators has been a growing topic amidst different stakeholders in the telecom industry for some years now [22] and many have shown enormous interest. The micro-operator concept will attract new stakeholders and bring about new business opportunities in the telecommunication industry, especially with the high degree of network flexibility, customization and privacy demanded by different verticals. With the recently concluded uO5G project [23], many opportunities have been presented in terms of the implementation and deployment of micro-operators networks and as described by authors in [15].

A micro-operator network can be deployed as Closed, Open or Mixed network [8]–[10]. The closed micro-operator network [8], [9] is targeted at providing communication services to vertical tenants whose human or machine are present within a closed user group which corresponds to a private network. The closed micro-operator network can be deployed in two scenarios, depending on the location of the tenants and the network. These include, closed Deployment A and Deployment B [8]. Closed Deployment A represent a micro-operator network whose tenants are at a single location while Closed Deployment B represent a micro-operator whose tenants are at different locations and their connectivity can be established via an external network.

An Open micro-operator network [8], [10] can be deployed as an MNO open or Public Open network. While the Public Open network is targeted at offering network service to unregistered public subscriber just like a public Wi-Fi, the MNO open is targeted specific MNO subscribers within a locality. The MNO open micro-operator is responsible for serving MNO subscribers within a micro-operator network in localities where the MNO aren't willing to deploy efficient network services. Financial analysis according to authors in [24], proves that MNO are reluctant to deploy their network over areas with little customers, hence a micro-operator network can be responsible for subscribers of different MNOs by allocating a separate slice for set of subscribers from each MNOs. This will be beneficial for both MNO and micro-operator. The number of MNO subscribers that will be covered under one slice of a micro-operator network will be determined by the SLA between the micro-operator and the MNO.

The Mixed micro-operator network [8] entails functionality of both the open and closed network by serving customer from both cases with a defined level of isolation between them. The mixed network can be deployed in a situation where a micro-operator is providing network service for a vertical and also responsible for MNO subscribers within the area. Due to the relationship between the micro-operator network and the MNO in the mixed network, two deployment scenarios can exist: Option A and Option B. Option A refers to a scenario where the micro-operator is in need of network resources from the MNO network for wide area access such as getting content from faraway servers, roaming scenarios for tenants at different locations (multi-site operations) and remote monitoring. Option B refers to the situation where the MNO network needs access to the micro-operator network to serve its subscribers within the micro-operator, very similar to the MNO open network. However, in this case the micro-operator is responsible for both vertical tenants and also MNO subscriber. So the MNO will be using the micro-operator broadband service to extent the indoor coverage within the vertical.

A descriptive diagram representing the three deployment scenarios that can exist for a micro-operator can be seen in Figure 2. Table I gives a detailed explanation about the deployment scenarios.

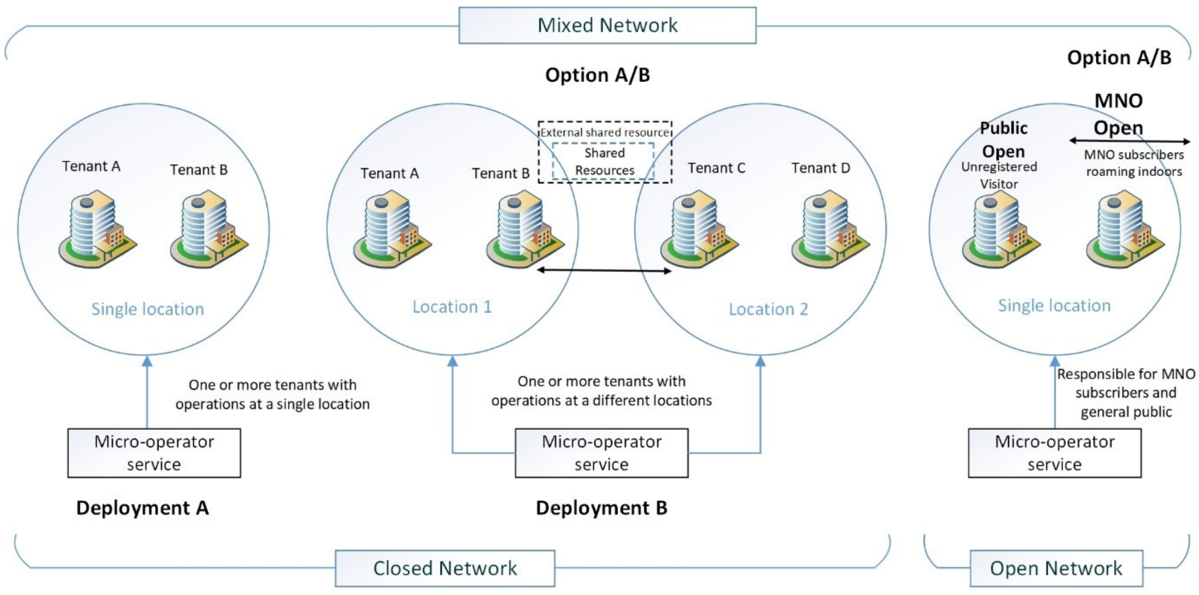


Figure 2. Micro-operator deployment scenarios.

Table 1. Micro-operator deployment scenarios

Micro-operator	**Deployment Scenario.**	**Deployment Description**
Closed	Deployment A	Serves a restricted customer set and responsible for one or more tenants at a single location.
	Deployment B	Serves a restricted customer set and responsible for a set of tenants at different locations.
Open	MNO Open	Targeted at MNO subscribers within a given locality based on the service agreement with the MNO. The micro-operator may be responsible for subscribers from one MNO or from multiple MNO.
	Public Open	Targeted at the general public.
Mixed	Option A	Micro-operator needs services from the MNO such as wide area access, remote monitoring, etc. thereby configuring tenant's slices with MNO resources.
	Option B	MNO needs services from the micro-operator network such as better service for its subscribers within the micro-operator's network and using the tenant's broadband slice for extending indoor coverage.

2.3 5G network slicing architecture

Network slicing is the mean to logically create network slices, such that each slice is responsible for specific network requirements. Over the years, several standard bodies and different commercial organizations have contributed to defining network slicing and numerous network slicing architectures have been proposed.

2.3.1 From the perspective of standard bodies

The Third Generation Partnership Project (3GPP)[1] is the main standard body for mobile communication network. 3GPP [5], [25] defines the required network slice functionalities to achieve an end-to-end network slice. 3GPP introduce the Network Slice Management Function (NSMF) which is responsible for creating the Network Slice Instance (NSI), the Network Slice Subnet Management Function (NSSMF) which is responsible for creating the Network Slice Subnet Instance (NSSI) and the Communication Service Management Function (CSMF) which is responsible for transmitting the communication service to the users. 3GPP also defined the concept of shared subnet either between two NSSIs or between two Network Functions (NFs). The NGNM alliance [12] proposes a three layer model in defining network slicing. The layers include the Service Instance Layer, the Network Slice Instance Layer and the Resource layer. NGMN further define the concept service instance, network slice instance, network slice blueprint, etc.

ITU-T-2020 network framework [1] proposes architecture to support diverse service requirement by ensuring enough end-to-end network softwarization leveraging existing tools such as SDN/NFV. ITU-T-2020 framework further introduces the concept of having multi

[1] https://www.3gpp.org/

access technologies with agonistic common core network that will separate the control plane and the user plane ensuring extensive and independent scalability and flexibility of the network slice.

ONF, [26] defines how the SDN architecture will enable a common infrastructure that will efficiently support multiple client network instances. ONF thus introduces the SDN controller client context that will provide an abstraction for network resources while supporting control logic for constituting a slice. ETSI [27] defines a network slice as a graph of network function (PNF and VNF) that are connected together to build the end-to-end network service with specific network requirement and capabilities. ETSI approach is based on how network slicing will be supported based on the ETSI NFV architecture [28].

2.3.2 Based on research projects

Different research projects have also proposed architectures and end-to-end implementations of network slicing. The 5G SONATA project[2] introduces flexible programmability of network software while developing orchestration framework for service developmental purposes. The SONATA project develops a Software Development Kit (SDK) component that supports service developers. The 5GEx project[3] expatiate more on the concept of multi-domain network service orchestration. The proposed architecture in the 5GEx project introduces a Multi-domain Orchestrator (MDO) that exposes the service specification APIs, thus allowing tenants to give their specific requirements under the same administrative domains. 5G Slicenet[4] [29] introduces the concept of cognitive multi-domain network slicing. The proposed architecture is divided into two broad domains, the advanced managed domain that will include the infrastructure, service and control layer which are based on existing descriptions and the innovative management domain which extensively describes the management and orchestration of the network slices based on cognitive operations.

The 5G PAGODA project [30] introduces a scalable and sophisticated architecture while extending the existing NFV architecture to suppose different slices across multiple vendor VNFs. This concept extends towards the proposing architecture to support multi-domain slicing and federated resource control introducing the concept of multi-domain service conductor stratum for managing network slice across federated domains [31]. The 5G Transformer project [32] proposes a 3 level architecture that supports federated slicing across multiple domains, the Vertical slicer, the Service orchestrator and the Mobile transport and computing platform. The 5GNORMA project [33], [34] introduces a reference architecture with four layers which are the Data layer, the control layer, the management and orchestration layer and the service layer. The 5GNORMA architecture further introduced the three SDMs. The Software Defined for Mobile Network control (SDM-C) in the Dedicated Control Layer Functions to manage resources assigned to a network slice. Thus, every network slice has a SDM-C in the control layer. The Software Defined for Mobile Network coordinator SDM-X in the Common Control Layer Functions to manage the shared resources across different network slices and the Software Defined for Mobile Network orchestrator (SDM-O) serving as the interface between the infrastructure and the business domains.

[2] http://sonata-nfv.eu
[3] http://www.5gex.eu/
[4] https://slicenet.eu.

2.3.3 From the perspective of open source bodies

Different open source management and orchestration frameworks are also available for the implementation of network slicing. The ETSI Open Source MANO (OSM) [35], which is used in this thesis, is a project that aims to produce a reference implementation technique leveraging the existing ETSI NFV-MANO architectural framework [28], and focusing on the Management and Orchestration (MANO) layer. OSM is open to supporting a wide range of Virtual Machines (VIMs), WIMs, NFVIs, VNFs, NSs and Network slices. Thus, OSM is a management and orchestration tool that manages the life-cycle or resources, virtual machines or network slices. OSM proposes two implementation techniques for network slicing, the Full E2E management also called the Integrated Modelling and the Standalone Management also called the Vanilla NFV/3GPP. The Full E2E management is done in such a way that the network slice can be treated as a meta-Network service. This is to say that the reference ETSI-NFVI architecture is leveraged, where the slicing management functionalities (i.e. CSMF, NSMF, and NSSMF) and the NFVO act like the full Operation and Business Support Systems (OSS/BSS) layer. With this method, the VIMs see these layer as just another service layer that gives information for resource selection. The Standalone Management on the other hand separates the Slicing MANO from the entire OSM ETSI NFV MANO. Thus, the network slice functionalities (i.e. CSMF, NSMF and NSSMF) form a separate standalone body that is connected with the ETSI OSM MANO using the Vanilla interface. The OSM approach is leveraged in this paper for the proposed architecture of network slicing in a micro-operator network with some enhancement. Figure 3 shows the ETSI-NFV reference architecture, the Full E2E Management and the Standalone Management.

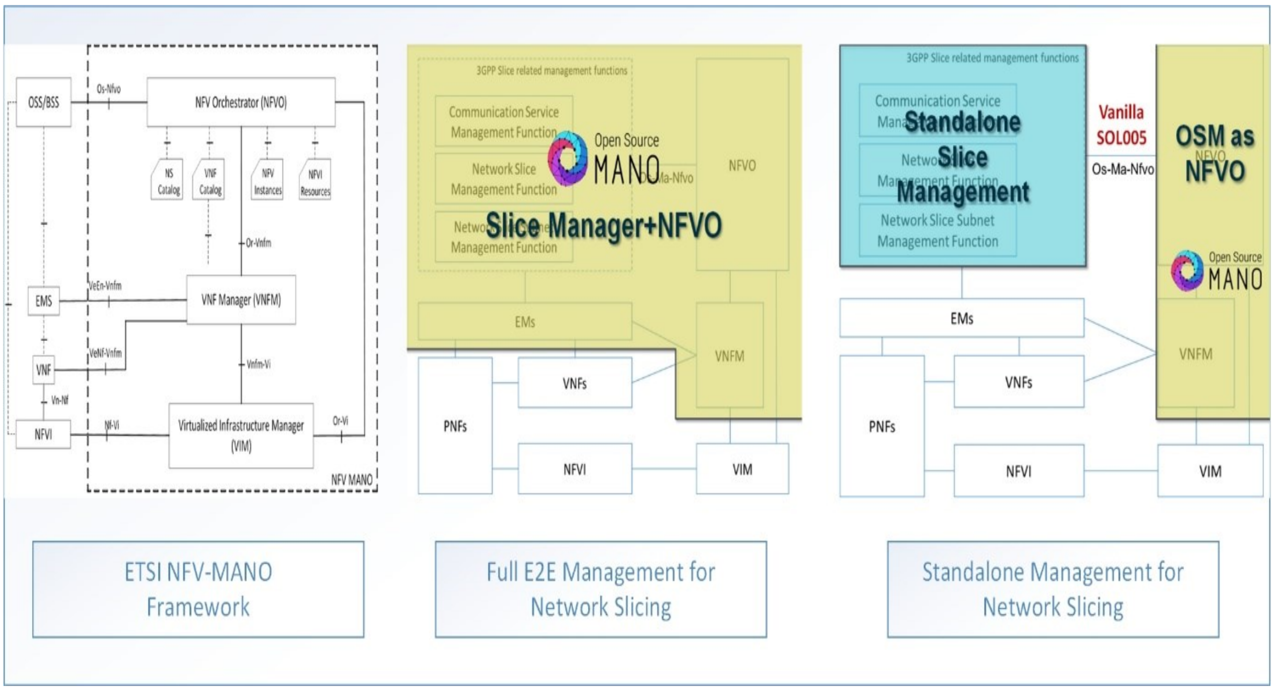

OSS/BSS
Os-Nfvo
NFV Orchestrator (NFVO)
NS Catalog
VNF Catalog
NFV Instances
NFVI Resources
Or-Vnfm
EMS
VeEn-Vnfm
VNF Manager (VNFM)
VNF
VeNf-Vnfm
Vnfm-Vi
Vn-Nf
NFVI
Nf-Vi
Virtualized Infrastructure Manager (VIM)
Or-Vi
NFV MANO
3GPP Slice related management functions
Communication Service Management Function
Network Slice Management Function
Network Slice Subnet Management Function
Open Source MANO
Os-Ma-Nfvo
NFVO
Slice Manager+NFVO
EMs
VNFM
VNFs
PNFs
NFVI
VIM
Standalone Slice Management
Vanilla SOL005
OSM as NFVO
ETSI NFV-MANO Framework
Full E2E Management for Network Slicing
Standalone Management for Network Slicing

Figure 3. ETSI NFV-MANO framework and OSM slicing implementation technique [35].

The Open Network Automation Platform (ONAP) project [36] by Linux foundation is aimed at building a comprehensive and sophisticated NFV implementation framework for a real-time automation of VNFs. Part of ONAP future work is to input network slicing automation into their framework. The OpenBaton project [37] is aimed at implementing the ETSI NFV-MANO framework. The architectural implementation supports network deployment over multiple cloud computing platforms. The NFV Orchestrator (NFVO) in the OpenBaton architecture which serves as the core part in resource instantiation and service creation, in relation with the Network Slice Engine, they can be used together for effective implementation of Network slicing. Finally the OpenStack Tacker project by Openstack [38] is focused on building an Open orchestrator for the management of VNFs. OpenStack Tacker uses the Topology and Orchestration Specification for Cloud Applications (TOSCA) language in the VNF catalogue for meta-date definition. The TOSCA language [39] is used to standardize how applications are described in terms of interaction between IT service developer and operators. The TOSCA language can be used in the service layer for the description of slice requirements.

2.4 Network slicing description for 5G micro-operator network

Even though the network slicing architecture for a micro-operator network hasn't been widely defined which is the aim of this thesis, some initial work has been laid down in terms of network slicing description for micro-operation deployment scenarios. These works serve as foundation to the proposed architecture in this thesis. The Network Slice Instance (NSI) configuration types that can exist for each deployment scenario of a micro-operator are defined in [8]. The NSI configuration types are based on the 3GPP network slice description [5], and they are used to describe how network slicing can be achieved for each deployment scenario of a micro-operator network. According to [8], the NSI configuration types (Type 1, Type 2 and Type 3 NSI configuration) are based on leveraging the concept of shared constituent between NSSIs and NFs within a micro-operator network or with an external network, to determine the type of services each deployment scenario can offer. Type 1 NSI configuration types are targeted at tenants with strictly low latency requirement and as such the slice instance is formed with no shared constituent between either the NFs or NSSIs. The Type 2 NSI configuration types are targeted at tenants with lesser latency requirement, such that the slice instance is formed with shared constituent of NSSIs or NFs between tenants of the same micro-operator network. Type 3 NSI configuration involves slice instances where there can be a shared constituent of NSSIs or NFs between the micro-operator network and the MNO network. Type 3 NSI configuration is available for tenants that requires external network resources for services such as wide area access, remote monitoring and roaming. Figure 4 describes the NSI configuration types for various deployment scenarios of a micro-operator network.

The result in [8] made it affirmative that for each deployment scenario of a micro-operator, network slicing can be practically defined based on the use cases and the tenants requirement within the vertical. This thus open new research interest in defining an end-to end architecture for each deployment scenario of a micro-operator.

Furthermore, to achieve network slicing for micro-operator deployment scenarios, [40] also proposed a management technique on how individual management functionalities defined by 3GPP [25] can be used in coordination to achieve network slicing for a micro-operator. The network slicing management functionalities considered include CSMF, NSMF and NSSMF. According to [40], to prove how the slicing management functions can coordinate together to achieve network slicing for each deployment scenario of a micro-operator network, a network slicing management technique is implemented with architectural blocks classified into three categories; the basic blocks, the management blocks, and the resource blocks. The basic blocks are made up of the Tenants block which describes the tenant's user equipment and serves as the communication service consumer, the Communication service provider block which represents a separate entity like a slice broker [41] and finally the Micro-operator block which represents the network operator. This can either be a Closed, Open or Mixed micro-operator network.

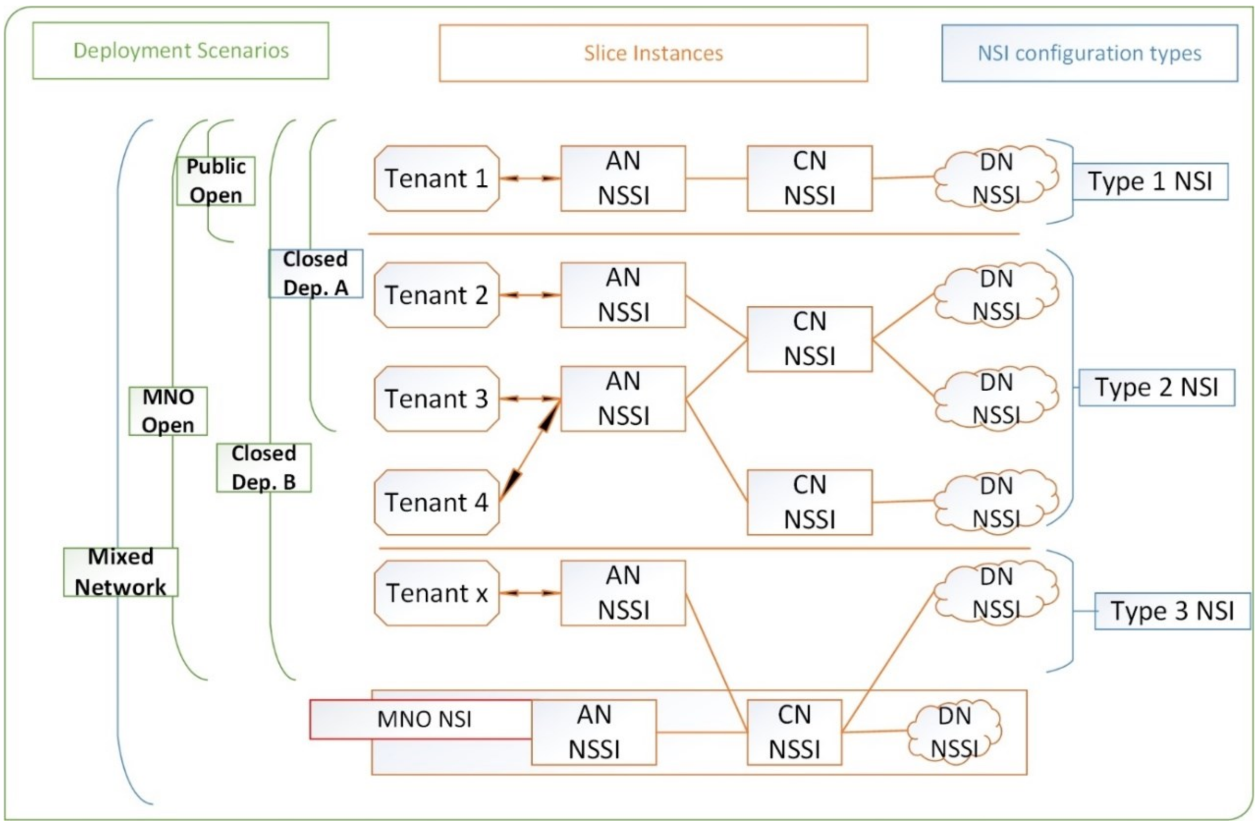


Figure 4. NSI configuration types for micro-operator deployment scenarios [8], [40]

The management blocks represents all management functionalities required to achieve an end-to-end network slice. The Management blocks are made up of the CSMF which manages and controls the activities of both tenants and communication service providers. The NSMF which is in some way the main management brain box that determines the deployment scenarios and creates slice instances based on the required services, and finally the NSSMF which is responsible for the creation of NSSIs based on the deployment scenarios and NSI configuration types. The NSSMF will manage shared constitutes within NSSIs.

The resource blocks are made up of the NFs, NSSIs and the communication services that are transmitted back to the tenants. Each of the defined management functionalities also make it more firm that it’s paramount to extend a simplified end-to end network slicing architecture for each deployment scenario of a micro-operator. Figure 5 describes the network slicing management technique for a general micro-operator network. It can thus be modified to fit any of the deployment scenarios

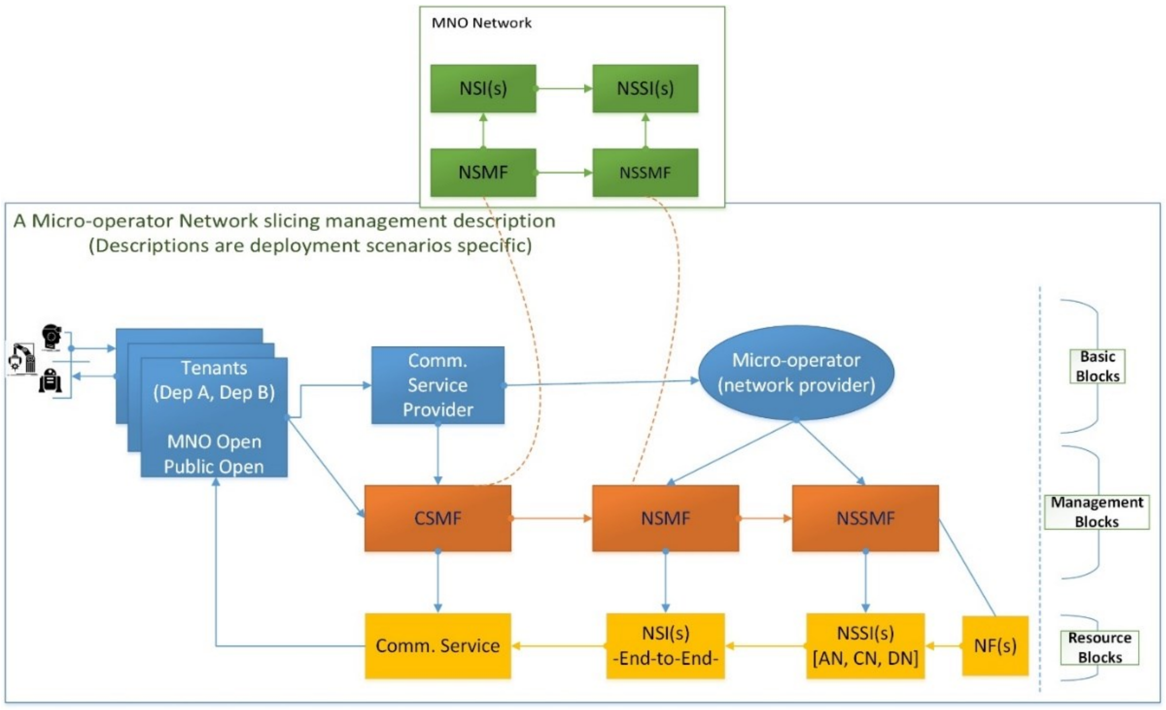


Figure 5.Network slice management technique for a micro-operator network [40].

3 LEVERAGING MULTI-TENANCY AND MULTI-DOMAIN FOR NETWORK SLICING IN A MICRO-OPERATOR

The idea of multi-tenancy and multi-domain in network slicing is an interesting aspect of research. Multi-tenancy describes how a single network resources can be sliced and maintained across multiple tenants with each tenant having the ability to determine its own network controlling ability. According to [41], achieving multi-tenancy in a network will involve full virtualization and softwarization of the network. This concept will fit perfectly into network slicing where each slices is responsible for specific communication requirement by different tenants. With multi-tenancy, different vertical tenants with specific requirement can be supported and this can be achieved by allowing dedicated core networks (DCNs), separation of the user and control plane while allowing serving chaining, and lastly leveraging multi-access edge computing (MEC) to move the network resources closer to the tenants. To achieve multi-tenancy, [41] further introduced the concept of 5G network slice broker to facilitate on-demand resource allocation to each tenant and perform admission control based on monitoring and forecasting. The 5G network slice broker resides with the network provider or the infrastructure provider, where all the required interfaces and functionalities are detailed. To further support multi-tenancy, [42] recently introduces the concept of having a multi-tenant NFV MANO that will be responsible for resources management and orchestration at the tenant level. The MANO as a Service (MANOasS) is proposed as an extension of the existing ETSI NFV-MANO model [28], by leveraging the virtualization abstraction of the ETSI NFV-MANO to the tenants layer. This is described as the Tenant MANO (t-MANO). Thus the t-MANO will ensure tenant have the capability to manage and orchestrate its own resources provided by the network operator. The source of the resources either within a single domain or multi-domain is not described, however the t-MANO is designed to manage and orchestrate resources for user equipment within a tenant.

However the approach of having a separate managers and orchestrator at the tenant layer presented in [42] will further raised more questions than the proposed solution it aimed at solving. The question include how to manage the complexity of separating the tenant's MANO, the NFV-MANO at the resource layer and slicing functionalities (in terms of slicing managers and orchestrator). Also, how to achieve the required orchestration if resources attributed to the tenant's slice are from multiple domain. This questions and more are what are intended to answer as part of our proposed approach in achieving proper management within individual tenant resources. Thus, In lieu of tenant managing their own resources for the user-equipment or customers within the tenants, our proposed architecture proffers another solution to achieve multi-tenancy in terms of a micro-operator network, we used the concept of Multi-Dimensional Descriptor (MDD) defined by 3GPP [43], which will be configured in the user-equipment to be attached to the network slice. Thus, if a tenant can have a specific MDD ID, each user-equipment within the tenant will have a similar MDD ID configured within them (see Figure. 7). Then the MDD at each tenant will determine the NSI that the end-user or user equipment will be connected to. The user-equipment attachment to certain tenant's NSI will be determined by the MDD of the user, which will be common to every user within a single tenant. For example, if there are 10 end users (10 VRs and mobile robots) within a tenant in an industrial vertical. The tenant's logical MDD ID can be described as MDD_ID_A, then every user within the tenant (in this case 10 users) will can have MDD_ID_A1 to MDD_ID_A10. Thus, after the CSMF allocate the communication service resources to tenant A, the users will connect to the NSI based on their MDD IDs, and in this case MDD_ID_A which corresponds to tenant A. This approach will simplify the resources allocation by each tenant to the user equipment within the tenant. Also at the initial stage of slice request that is being sent to the network operator, the MDD will be entailed in the slice request ID. According to 3GPP, the MDD will be provided by the user equipment during the signaling and attachment procedure, so leveraging this, all users within the same tenant will just be made to have the same MDD_ID and as such, they can be connected to the same NSI, thus achieving proper management within multi-tenancy

Meanwhile, the concept of network slicing across multiple domain involves having a network slice whose virtualized resources are administered across multiple infrastructure or across multiple providers and it can be achieved by proper softwarization of a network. Multi-domain resource allocation broaden the capabilities of a network slice and increases its flexibility. To achieve multi-domain network slicing, it's important to properly define how different virtualized resources at different layers will be created and instantiated across multiple administrator. For network slicing, multi-domain concept can be explained in the following ways ways. First of all, the multi-domain in the resource layer based on the NFV-MANO [28] architecture, this can be achieved as proposed by [19] where we have multiple technological domains, in either Radio Access Network (RAN) , Core Network (CN) or Transport Network (TN) . Each domain is a group of networks and devices that are administered as a single unit with common rules and procedures. Resources in different domain can be physical or virtualized depending on whether the tech domain is RAN, CN or TN. The resources from multiple domain can be aggregated together in the Aggregation Resource Orchestrator (RO) and each aggregated resources per tenant's slice can then be orchestrated by the NFV orchestrator followed by the domain-specific orchestrator. This approach proposed by [19] solves how resource orchestration can be done within this domain. Although multi-domain slice orchestrator were introduced above every administrative domain to create an end to end slice across multi-domain. As interesting and feasible this approach is, it didn't address the positions of each of the management functionalities defined by 3GPP across each domain. This aspect in multi-domain network slicing is tackled in the proposed architecture for micro-operator network. Secondly [44], proposes multi-domain management and orchestration framework for software defined infrastructure in terms of coordination and automation of cloud and networking resources. The approaches introduces orchestrators at different layers of a network infrastructure. This orchestrators are defined as SDN controllers since they will also be involved in continuous allocation of resources in an optimal manner. This approach thus, show how different orchestrators which can act as SDN controllers in terms of resource allocations can be deployed at different level to achieve the desired framework of automation and coordination of network resources. Thus with the approach in [44], it has been proven that different orchestrator at different infrastructural level can be feasible and the solution is also extended to achieve network slicing across multiple domain. Lastly in the terms of achieving multi-domain in the slicing layer i.e. achieving multi-domain based on the slicing management functionalities defined by 3GPP and how federated network slicing can be achieved, [31], proposed a sophisticated four strata divided architecture. The strata include the Multi-domain service conductor stratum, the domain-specific fully fledged orchestration stratum, the sub-domain MANO stratum, and the logical multi-domain slice instance stratum.

Apart from the multi-tenancy approach in our proposed architecture explained earlier, the architecture also simplify how the 3GPP network slicing management functionalities i.e. CSMF, NSMF, and NSSMF will fit into achieving multi-tenancy and multi-domain network slicing for micro-operator deployment scenarios. As described by [40], each of the deployment scenarios can be achieved with the 3GPP management functionalities, now our architecture shows how management functionalities will cut across multi-tenancy and different multi-domain.

Figure 6 shows our simplified description on how network slicing for different deployment scenario of a micro-operator will be achieved leveraging multi-domain and multi-tenancy.

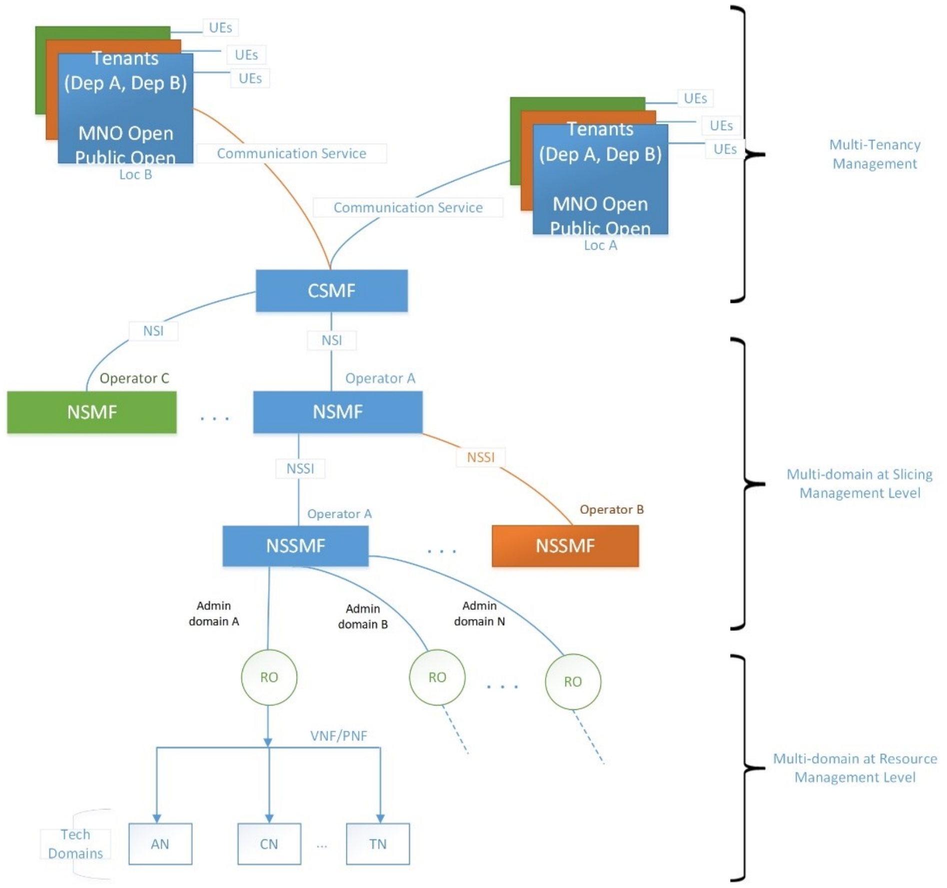


Figure 6. Simplified description of multi-domain using 3GPP functionalities.

As seen in Figure 6, by leveraging the network slicing management functionalities, we can achieve multi-domain and multi-tenancy across an end to end network slice. This approach is described so that network slicing can be achieved for each deployment scenario of a micro-operator network. Table II shows a tabulated description for each deployment scenario and how there slice formation will be achieved with multi-domain and multi-tenancy

With the approaches in Figure 6 and Table II, proposing an end-to-end network slicing architecture for different deployment scenarios of a micro-operator network is rather vital since one system architecture cannot fit all use cases.

Table 2. Achieving multi-domain and multi-tenancy for each deployment scenario of a micro-operator network

Deployment Scenario	**Multi-Tenancy level**	**Slicing Multi-Domain level**	**Resource Multi-Domain level**
Deployment A	CSMF from a single operator Tenants at a single location	NSMF and NSSMF from a single Operator	Resources from single or multiple tech domains
Deployment B	CSMF from a single operator Tenants at a single or multiple location	NSMF and NSSMF from a single or multiple operator	Resources from single or multiple admin domains
MNO Open	CSMF from a single operator Tenants within a single locality	NSMF from a single operator NSSMF from a single or multiple operator	Resources from single or multiple admin domains
Pubic Open	CSMF from a single operator Tenants within a single locality	NSMF and NSSMF from a single Operator	Resources from single or multiple admin domains
Option A	CSMF from a single operator Tenants at a single or multiple location	NSMF from a single or multiple operator NSSMF from a single or multiple operator	Resources from single or multiple admin domains
Option B	CSMF from a single or multiple operator Tenants at a single or multiple location	NSMF from a single or multiple operator NSSMF from a single or multiple operator	Resources from single or multiple admin domains

4 PROPOSED END-TO-END NETWORK SLICING ARCHITECTURE FOR MICRO-OPERATOR DEPLOYMENT SCENARIOS

Previous work on the merits and classification of network slicing for a micro-operator network have shown the importance of having a specialized end-to-end network slicing architecture for each deployment scenario for a micro-operator. Other reasons for establishing a new architecture for micro-operator network include the following:

- There are different deployment scenarios, and each deployment scenarios in a micro-operator network supports different use cases. As such implementation of network slices are deployment scenarios dependent, it is important that an individual architecture addresses the type of micro-operator network, be it Closed, Open and Mixed.
- Also, the proposed architecture is targeted at supporting multi-tenancy and describing extensively how the communication services from a tenant's allocated NSI will be transmitted or distributed across end users/user equipment within the multi-tenant network.
- The architecture also shows a better and simplified description of the 3GPP defined network slicing functionalities in terms of how orchestrators and managers be defined to achieve this functionalities within a single domain or multiple domains

The proposed network architecture for micro-operator deployment scenarios are designed leveraging multi-domain and multi-tenancy. The architecture is divided into four layers. These include the Multi-tenant layer, the Service layer, the Slicing MANO layer, and the Resource layer. Each of these layers will contribute to achieving end-to-end network slicing for a micro-operator network.

Figure 7 depicts the proposed end-to-end network slicing architecture for mixed micro-operator network. A mixed micro-operator network slice architecture is considered as it combines the capabilities of both closed and open micro-operator network. As such the explanation of all the layers within the mixed network might be enough to suffice the other micro-operator network, i.e. the closed and open. The remaining undescribed parts will be complemented for when explaining the closed and open micro-operator network.

4.1 Mixed micro-operator network slicing architecture

The mixed micro-operator network is proposed to entail the functionality of both open an closed micro-operator network with a defined level of isolation and privacy between them. Thus in the mixed network, the slicing architecture will fulfil the satisfaction of both open and closed subscriber and allocated network slice instances for them accordingly. The network slicing architecture for the mixed micro-operator network can be seen in figure 7.

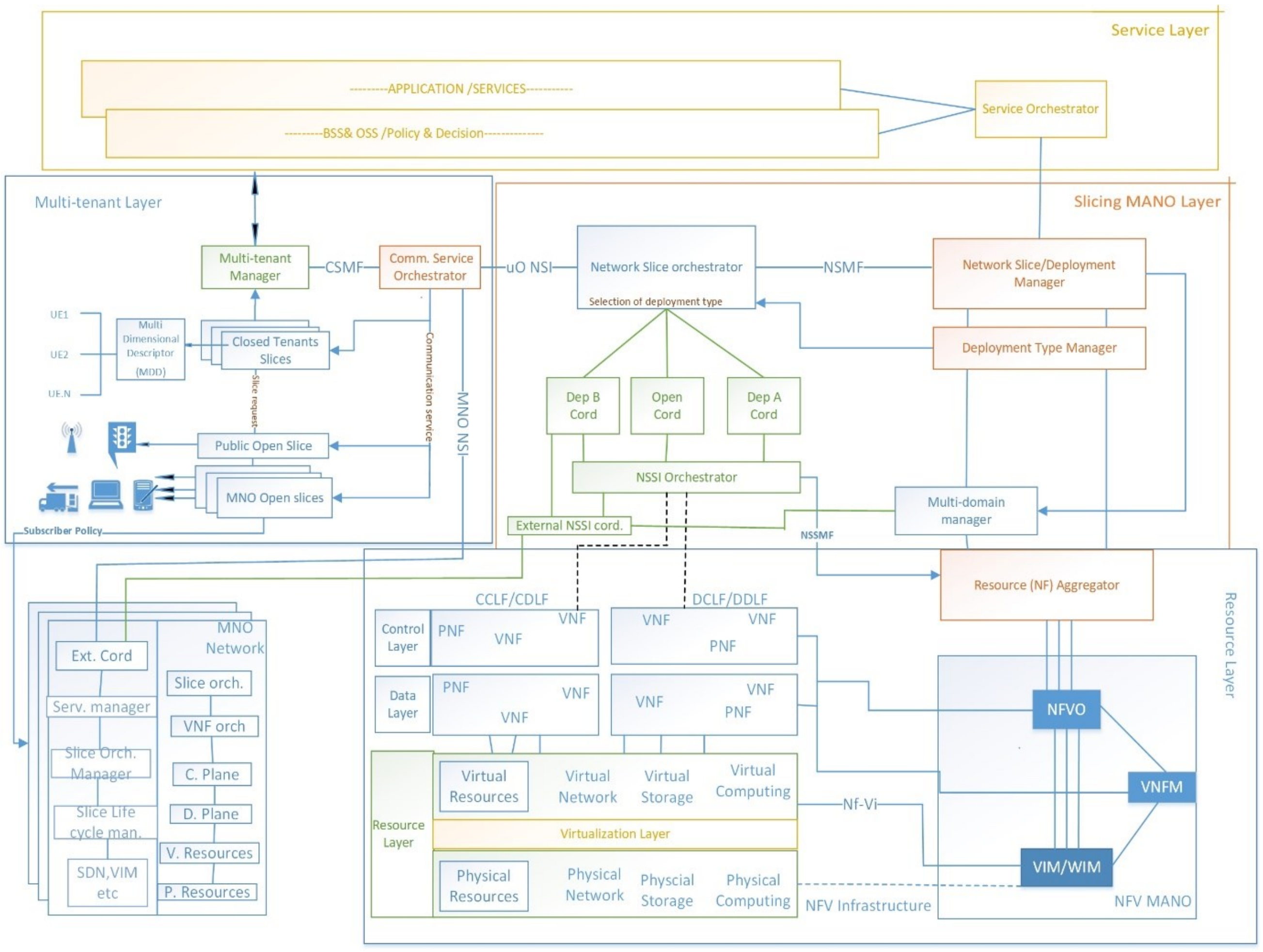


Figure 7. End-to-end Network slicing architecture for a mixed micro-operator network.

4.1.1 The multi-tenant layer

The multi-tenant layer is responsible for handling multi-tenancy within the micro-operator network. Its responsibilities extend to handling slice creation requests from each tenant and the allocation of communication services from the NSI to individual tenant slices. As seen in Figure 7, different blocks can be defined within the multi-tenant layer, and since it's a mixed micro-operator network, the blocks include tenants' slice blocks from both the open and closed micro-operator networks.

The CSMF capabilities are implemented using the multi-tenant manager and the comm. service orchestrator to ensure a proper division and allocation of the CSMF responsibility. **The multi-tenant manager** is responsible for collection of slice creation requests from individual tenants, and forwarding those requests to the service layer. The individual slice request such as the slice service type (eMBB, uRLLC, mMTC), for each tenant will come as a slice request ID that will be described in the Network Slice template (NST) during implementation phase. Every tenant's slice request will have different NST encoded in a TOSCA or YMAL descriptor so it can be easily use for future slice instantiation. The slice creation request in the NST will be dependent on the deployment scenarios and it will contain tenant information such as the deployment type, location of tenants, domain implementation, the slice service type (that will determine network requirement in terms of latency, throughput, bandwidth etc). It will also specify if the tenant slice will be created with shared constituent or if it will require external resources, and so on. All these parameters will be embedded in the tenant slice creation request at the multi-tenant manager, and transmitted to the service layer.

The comm. service orchestrator is responsible for allocation of the communication service to individual tenants. The comm. service orchestrator will handle the CSMF responsibilities of getting NSI from the micro-operator network's NSMF (i.e. the Network Slice Orchestrator) or from both micro-operator network and the MNO NSMFs in case mixed network option B, where the MNO needs access to the micro-operator network to expand its broadband services. The comm. service orchestrator will then transmit the NSI to individual tenants as a communication service that is enabled throughout the slice lifecycle. The comm. service orchestrator operation will ensure optimal allocation of resources (communication service) to different tenants based on their priority which will be determined by the tenant's SLA. Thus while the comm. service orchestrator is responsible for allocation of communication service orderly to individual tenants throughout the slice lifecycle, the multi-tenants manager is responsible for regularly updating slice creation requests and changes within tenants requests while transmitting them to the service layer.

In the mixed network, the tenant's slices include **the Closed Tenants slices** and **the Open tenants' slices.** The Open tenant's slices are further divided between the MNO open slices and the Public open slice. The **Closed Tenants slices** are connected to a logical MDD per tenants which determines how the individual user equipment within the tenant get the appropriate network slice allocation as explained in previous section. On the other hand, the MNO open slices at the multi-tenant layer of a mixed micro-operator network are responsible for MNO subscribers within the network. Individual slices are allocated for set of subscribers of the same MNO network whom the micro-operator is responsible for. The number of subscribers that can be accommodated within one open network slice will be determined by the SLA between the micro-operator and the MNO. The MNO open slices are always connected to the MNO network to determine individual **subscriber policy** with the MNO network. Thus, If the number of MNO subscribers connected to an open slice is up to the agreed number in the SLA, the micro-operator network will allocate another slice for the remaining subscribers of the MNO. This is important to maintain a desired level of quality of service for each subscribers within the network. The Public MNO slice will be reserved for general public (i.e. user equipment or end-users) not attributed to any MNO network. All slice creation requests within the open network are transmitted to the Multi-tenant Manager.

4.1.2 The service layer

As seen in Figure 7, the service layer will serve as both a slice service layer and business domain. It manages the slice operation and business of every tenant while checking the slice request conformity with the SLA. It also attributes the **network service and application** requirement for each tenant based on the received slice request. For a mixed micro-operator network, **the OSS&BSS block** will determine the priority at which different tenants request will be handled according to the mapped set of Key Performance Indicators (KPIs) attributed to each deployment scenario while **the policy and decision block** will determine the SLA between the micro-operator network and individual tenant in a closed tenant or make a confirmation of the subscriber policy in the MNO open case. The **application and service block** will handle the service and business implementation of the tenants' slice request, thus, approving the network slice formation by allocating each tenant's request a slide_ID. Different slice_ID forwarded to the network slice manager in the slice MANO layer are attributed to different tenant's slice request. However in this case the slice_ID represent real network requirements in terms of required, NSSI or Network Services, VNFs and PNFs to achieve the requested network slice.

4.1.3 The slicing MANO layer

The slicing MANO layer describes how network requirement for different tenant slice ID are being converted to network slicing requirement and ultimately network slice instances, which are being transmitted back as a communication service to the tenants. The slicing MANO layer implements the NSMF and NSSMF management capabilities of the network slice. Since the slicing MANO layer is acting based on the virtualized resources instantiated in the cloud, hence, to implement some of the blocks within the slicing MANO layer, a juju charm[5] Day 1 and Day 2 primitive can be used to achieve the blocks functionalities. All the blocks within the slicing MANO layer operate dependently and as such the layer can be explained as follows. Whenever a slice_ID in terms of network slicing requirement is received from the service layer, the **Network slice manager** which also serves as the slice Lifecycle Manager (LCM), will be responsible for handling the end to end life cycle of every tenant slice. To know the exact resource related to each deployment scenario, the network slice manager is connected with a **deployment type manage** which determines which deployment scenario a tenant's slice_ID belongs to, either a closed deployment A, closed Deployment B or Open Network. Thus, while the Network slice manager is in charge of a high-level slice_ID management, deployment type manager will be responsible for sorting out different deployment resource allocation. For proper management of multiple slice requests, the deployment type manager transmit the administrative control of different tenant's slice request to the **network slice orchestrator** which distribute the functionalities amidst the different coordinators.

Three coordinators are available to create a level of abstraction between the slice resource selection and the network slice orchestrator. These are **Dep. A cord, Dep. B cord** and the **Open cord**. Thus, while the deployment type manager is responsible for resource selection based on the tenant's deployment requirement received, the coordinators are responsible for handling different tenants within each deployment scenario. For example, all tenants belonging to deployment A will be logically coordinated by the Dep. A coordinator, while the resources will be selected by the deployment type manager and then individual NSI allocated for all tenants in Deployment A will be orchestrated by the Network slice orchestrator using the slice allocation ID which is will correspond to the slice request ID, and finally, the slice lifecycle is managed by the network slice manager. The communication between the Network slice orchestrator and the network slice manager will be formatted in a restful API[6].

The next stage after each tenant cases are separated to their deployment coordination, is the selection of network resources based on the tenant slice_ID. From Figure 7 it can be seen that the different deployment scenario coordinators are connected to the NSSI orchestrator. The **NSSI orchestrator** is responsible for aggregation of NFs from different domains within the micro-operator network. Since deployment B NSI will/can involve NSSI from external network, the Dep. B coordinator is also connected to an **external NSSI orchestrator** which serves as an abstraction layer between the micro-operator and the external network. The **Multi-domain manager** is responsible of aggregating network resources (NFs) from different domain

[5] https://jaas.ai/
[6] https://restfulapi.net/

of different operators, which is why it is connected to the external NSSI orchestrator. Thus the micro-operator NSSMF capabilities is implemented between the NSSI orchestrator and the Resource aggregator. The Resource aggregator serve as the layer of abstraction between the slicing MANO layer and the Resource layer and it is responsible for NF aggregation across multiple tech domain within the micro-operator network.

4.1.4 The resource layer

The Resource layer describes the implementation of the ETSI NFV-MANO architecture [28] with few modifications. As seen in Figure 7, the NFV infrastructure layer is exactly the same as the ETSI-NFV infrastructure with separation of the common control layer functions from the dedicated control layer functions in the control and data layer of the NFV architecture. The reason from this separation is to embed the individual user equipment MDD_ID in the dedicated control layer for proper differentiation of the resources for each layer. This approach is similar to the implementation technique in [34]. In the NFV orchestration at the resource layer is also similar for different Network functions across single or multiple domain.

The SDN functionality for data transmission amidst different VMs in the VIM is implemented with the Resource aggregator or Resource Orchestrator (RO). The Resource aggregator between the slicing MANO layer and the Resource layer is used in the proposed architecture to ease the implementation procedure and make it similar to the ETSI OSM-MANO i.e. Full end-to-end management (see Figure 3) approach for implementing network slicing. The approach here is such that at the resource layer, the management and decision of selected resources will be based on individual tenant slice request and slice ID, so the resource level handles the instantiation and management of VMs in form of VNFs and PNS using the ETSI-NFV-MANO architecture and the slicing MANO layer handles the slice management, while both layers can be connected via a resource aggregator.

According to OSM [35], the SDN-assist which implements the functionality of the Resource aggregator via the Single Root Input Output Virtualization (SR-IOV) or Passthrough interface, works in such as way that after the instantiation of a VNF in the resource layer, SDN-assist maps each instantiated VM interfaces to an Openflow port, and then creates a data plane network by talking to SDN controller and connecting appropriate Openflow ports. With this approach the slicing MANO layer which is implementation in this thesis is with OSM, will be able to connect and transfer packets between different VMs. This is similar to the approach proposed in [19] where the resource aggregation serves as the Aggregation RO and the NSSI orchestrator serve as the domain administrator.

4.2 Open and closed micro-operator network

Figure 8 depicts the network slicing architecture for the open and closed micro-operator. As seen in Figure 8, the Open micro-operator network is slightly different than the closed and mixed micro-operator network.

In the open micro-operator network, **the multi-tenant layer** operation described for the mixed network will be well applied to the open micro-operator network, however slice creation request will come from the MNO slice based on the subscriber policies. Also there is no MDD since each subscriber policy will be logically added by the corresponding MNO network. Other blocks will function as described in the mixed network

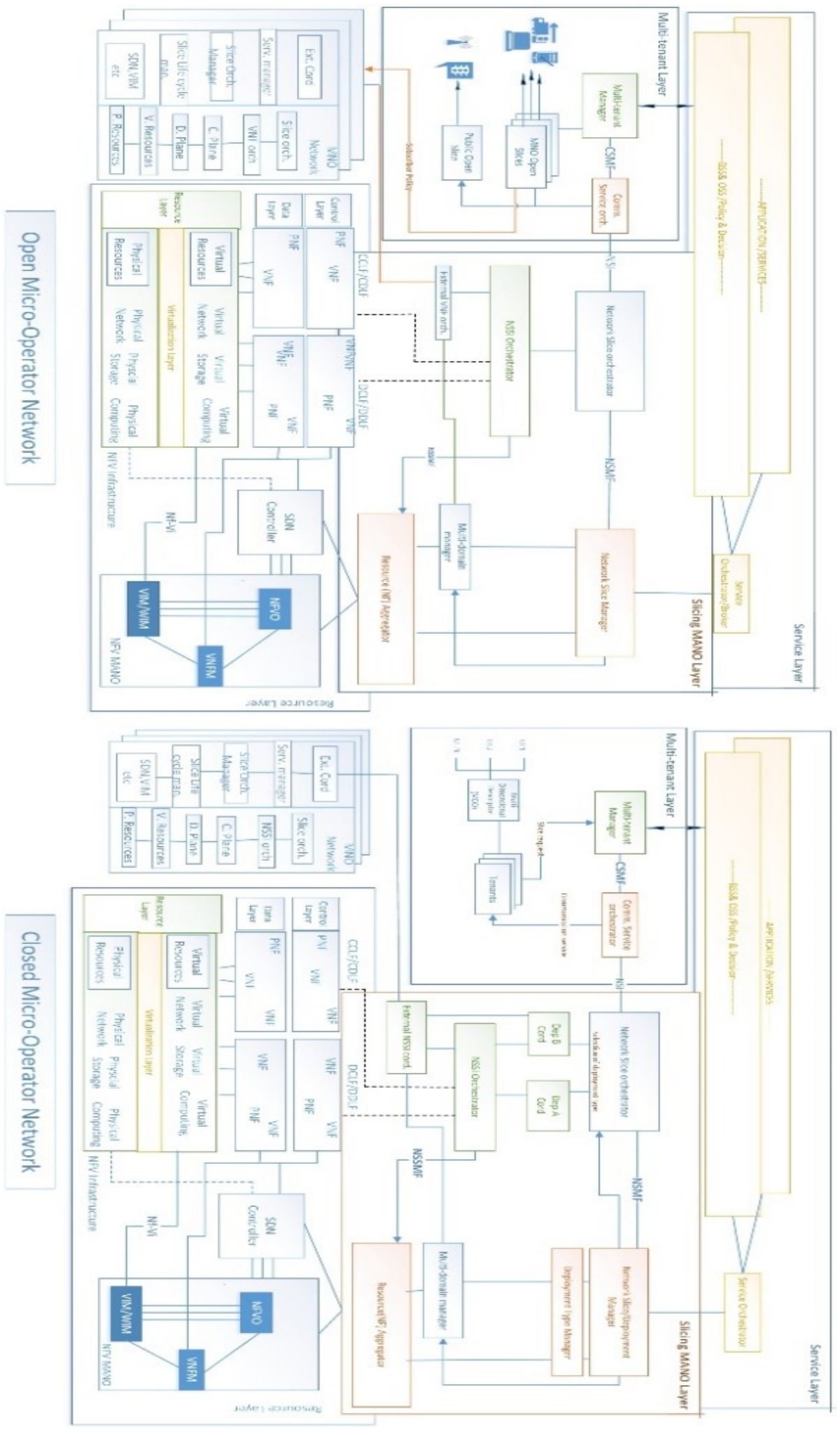


Figure 8. End-to-end Network slicing architecture for open and closed micro-operator network.

The basic difference in the operation of the **service layer** in approving abstracted slice creation requests to real network requirements based on the SLA will be described in the slice request descriptor. The service layer will also check if a subscriber is trying to connect with an MNO which it have service agreement with the micro-operator, if otherwise, the subscriber will be allocated resource under the public open slice.

The **slicing MANO layer** for the open micro-operator network does not have the deployment type manager, since the public open slice is declared in the multi-tenant layer, so only MNO tenants are treated in the slicing MANO layer. Every other blocks will function as described in the mixed micro-operator network and the **resource layer** is the same as the mixed network.

In the closed micro-operator network architecture as seen in figure 8 at every layer is similar to the mixed micro-operator network in figure 7. Except for the fact that the mixed network combines also open network slices.

4.3 Descriptive parameters used at different layers

Table III shows some parameters that can be used during the process of establishing an end to end network slice between different layers of the proposed architecture. These parameters will be defined and included in the network slice descriptor based on the micro-operator deployment scenario. Other exchange parameter may and will exist based on individual deployment scenarios. This will be highlighted during implementation phase at the network slice description.

Table 3. Exchange message parameters during network slicing at different layers of the proposed architecture

Layer	**Message type parameter**	**Description**	**Micro-operator Network depl.**
Multi-tenant Layer	MDD_ID_1	General descriptor for every tenant slice	Closed, Mixed
	MDD_ID_1A	MDD configured at the UE for tenant's NSI	Closed, Mixed
	Tenant_loc.	ID for sorting tenant request by deps.	Closed
	MNO_subcriber_ID	ID for each MNO network slices	Open
	Tenant_request_ID	ID per tenant's slice abstract requirement	Closed, Open, Mixed
	Tenant_comm_ID	Comm. service for each tenant slice	Closed, Open, Mixed
	uO_NSI_ID Ext_NSI_ID	Identify NSI from micro-operator network Identify NSI from MNO network	Closed, Open, Mixed Mixed
Service Layer	Slice_ID	ID for real network requeirement	Closed, Open, Mixed
	Tenant's policy	Responsible for closed tenant policy	Closed
	MNO policy	Responsible for open tenant policy	Open
	Service_ID	Attributing network service per slice ID	Closed, Open, Mixed

	Decision_meter	Deciding whether to forward slice ID to slice manager or not based on the SLA	Closed, Open, Mixed
Slicing MANO layer	Dep_type_ID	Dep. type manager use to sort slice ID based on deployments	Closed, Mixed
	Domain_ID	Identify tenant multi-domain requirement	Closed, Open, Mixed
	Micro_NSSI_ID	Identify NSSI from micro-operator	Closed, Open, Mixed
	External_NSSI_ID	Identify NSSI from MNO network	Closed, Open, Mixed
Resource Layer	Resource_ID	Select which tenant resources is next by the SDN controller	Closed, Open, Mixed
	Tech domain ID	Identify micro-operator network function domains	Closed, Open, Mixed

4.4 Message Sequence diagram for the proposed architecture

The message sequence diagram highlights the message flow from the user equipment slice creation request till when the slice communication service is established. Figure 9 shows the end to end sequence diagram for the mixed micro-operator network (covering both open and closed micro-operator).

Since the slicing MANO layer is one of the most important layers for achieving network slicing in a micro-operator network and for better understanding of the inter-block operation within this layer, a separate sequence diagram is shown in figure 10 to describe the slicing MANO layer. While the sequence diagram in figure 9 describes the slicing MANO layer using the network slice manager. The network slice orchestrator and multi-domain manager, the sequence diagram in figure 10 shows a better description of the operation at this layer.

To better explain the sequence information of the slice creation with the proposed architecture, a message exchange flow for each deployment scenario corresponding to the message sequence in figure 9 and 10 are shown for each micro-operator deployment scenario. This message exchange flow describes a summary of the slice creation procedure from the tenant layer till the resource layer.

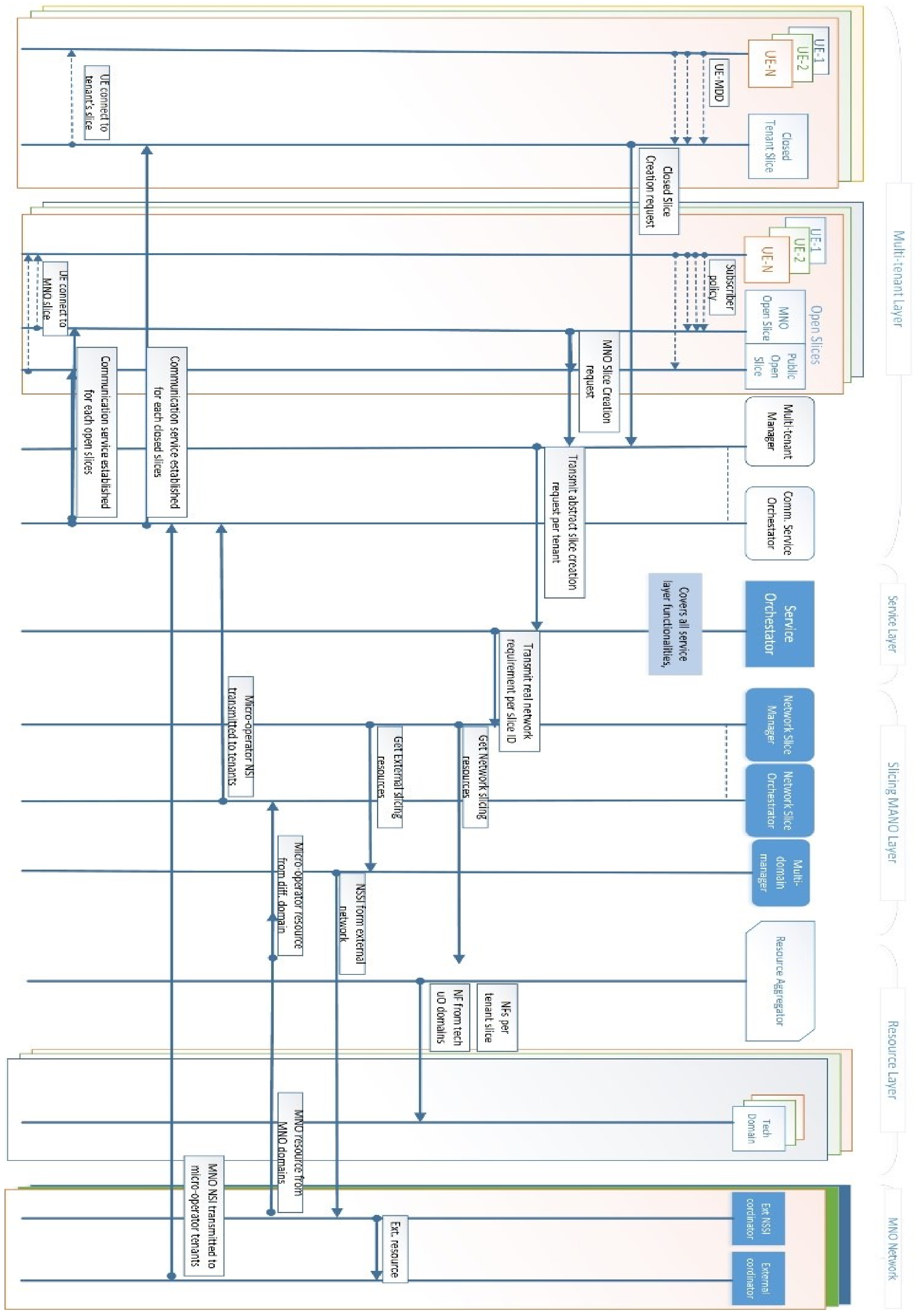

Multi-tenant Layer
Service Layer
Slicing MANO Layer
Resource Layer
MNO Network
UE-1
UE-2
UE-N
UE-MDD
Closed Tenant Slice
Closed Slice Creation request
UE connect to tenant's slice
Open Slices
Subscriber policy
MNO Open Slice
Public Open slice
UE connect to MNO slice
MNO Slice Creation request
Communication service established for each open slices
Communication service established for each closed slices
Multi-tenant Manager
Comm. Service Orchestator
Transmit abstract slice creation request per tenant
Service Orchestator
Covers all service layer functionalities,
Transmit real network requirement per slice ID
Network slice Manager
Network Slice Orchestrator
Multi-domain manager
Get External slicing resources
Get Network slicing resources
Micro-operator NSI transmitted to tenants
Micro-operator resource from diff. domain
NSSI form external network
Resource Aggregator
NFs per tenant slice
NF from tech uO domains
Tech Domain
MNO resource from MNO domains
MNO NSI transmitted to micro-operator tenants
Ext. resource
Ext NSSI cordinator
External cordinator

Figure 9.End to end message sequence diagram for a mixed micro-operator network.

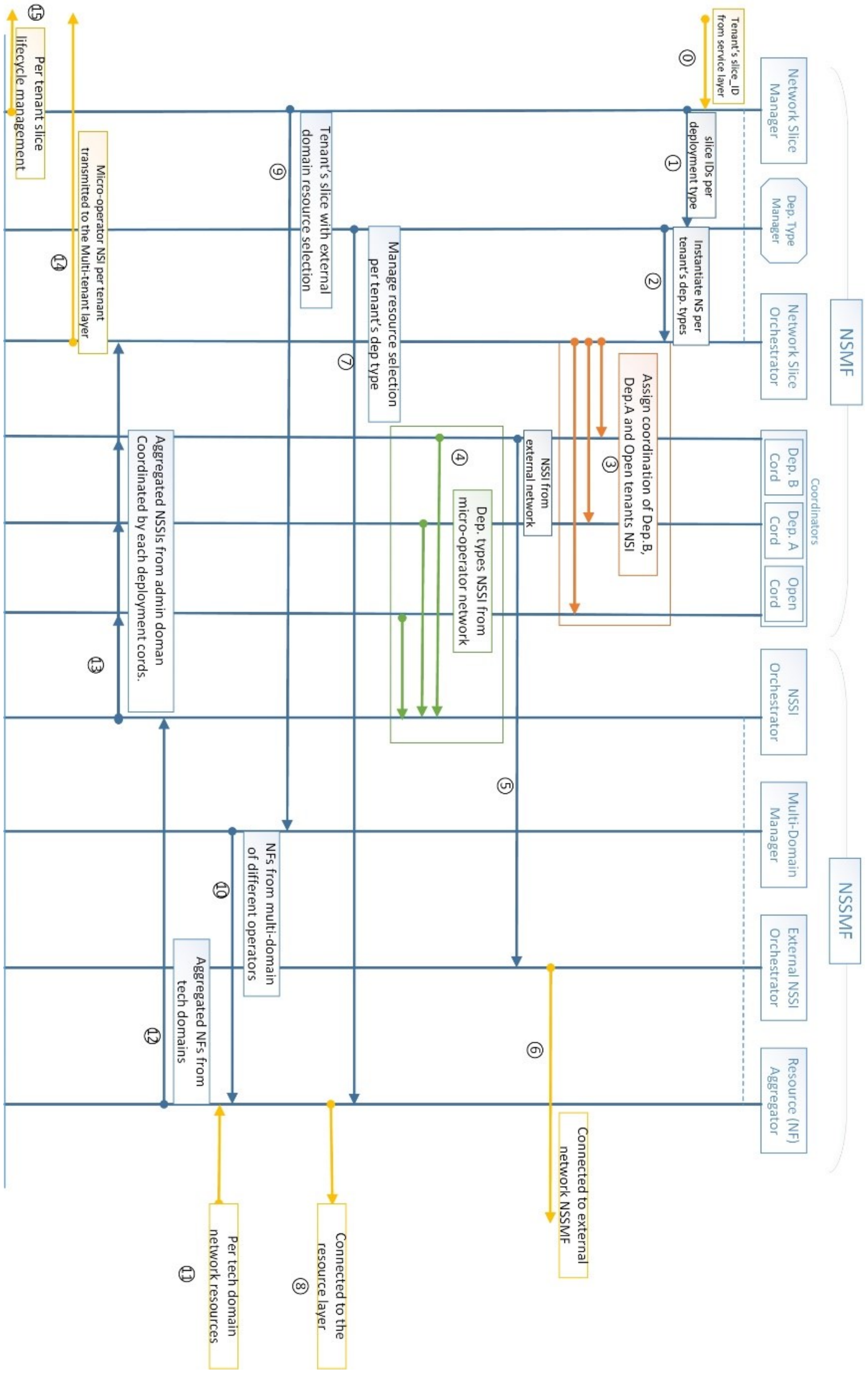

NSMF
NSSMF
Network Slice Manager
Dep. Type Manager
Network Slice Orchestrator
Coordinators
Dep. B Cord
Dep. A Cord
Open Cord
NSSI Orchestrator
Multi-Domain Manager
External NSSI Orchestrator
Resource (NF) Aggregator
⓪
Tenant's slice_ID from service layer
①
slice IDs per deployment type
②
Instantiate NS per tenant's dep. types
③
Assign coordination of Dep.B, Dep.A and Open tenants NSI
④
Dep. types NSSI from micro-operator network
NSSI from external network
⑤
⑥
Connected to external network NSSMF
⑦
Manage resource selection per tenant's dep type
⑧
Connected to the resource layer
⑨
Tenant's slice with external domain resource selection
⑩
NFs from multi-domain of different operators
⑪
Per tech domain network resources
⑫
Aggregated NFs from tech domains
⑬
Aggregated NSSIs from admin doman Coordinated by each deployment cords.
⑭
Micro-operator NSI per tenant transmitted to the Multi-tenant layer
⑮
Per tenant slice lifecycle management

Figure 10. Message sequence diagram for mixed micro-operator network slicing MANO layer.

5 IMPLEMENTATION AND PERFORMANCE ANALYSIS OF PROPOSED NETWORK SLICING ARCHITECURE

Implementation of a softwarized network requires some high level of hands-on experience, good familiarity with Linux and network commands. In this thesis, implementation was achieved by setting different environment right, deploying and installing the required tools. Although different programming language knowledge is required for different tools deployment, however the main orchestration of the network was achieved using the YMAL[7] descriptive language.

To discuss the performance analysis of a network slicing architecture, in this thesis, a real network slicing implementation is done and performance analysis was achieved based on the measured metrics such as CPU utilization, memory utilization packet sent and packet received between different NSSIs. Thus we considered the implementation environment, the procedure for achieving the proposed implementation, the slice instantiation commands used in OSM for the implementation and finally the performance analysis of the result of network slicing.

5.1 Implementation environment

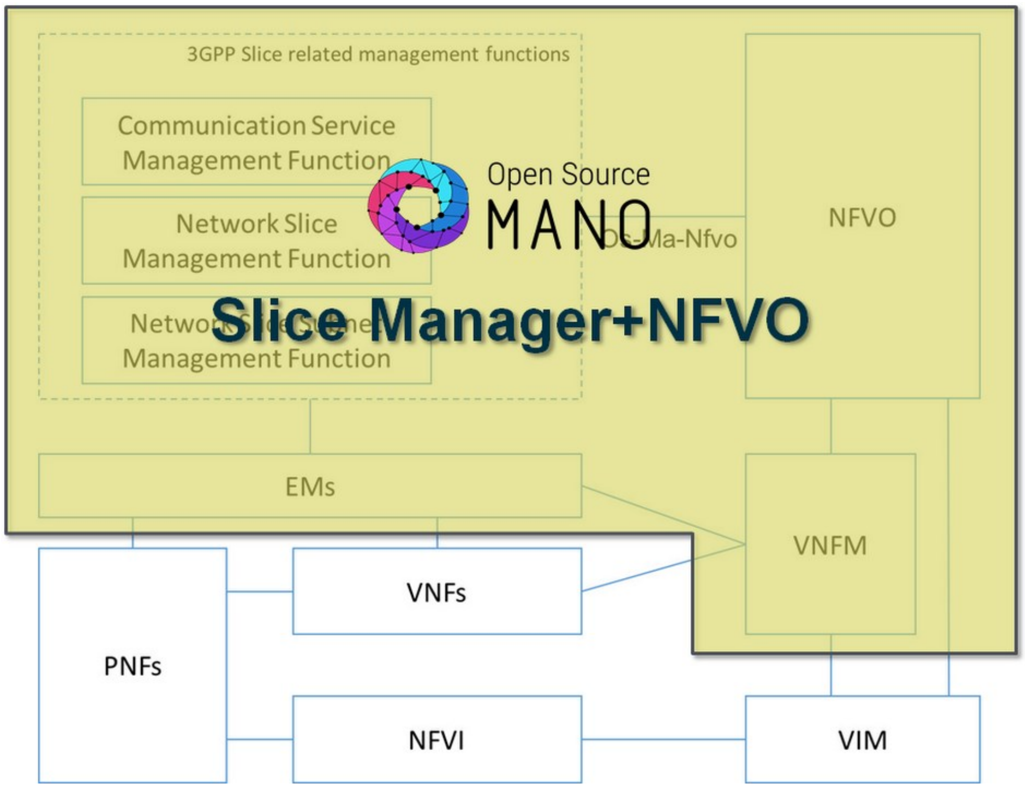
To implementation of network slicing, different environment need to be in place to achieve the final slice instantiation. The environment are generally categorized into Orchestration tools, Virtual Infrastructure Machine (VIM) or Cloud infrastructure and the Performance measurement tools. Orchestration tools such as OSM, ONAP are used for automatic configuration, management and orchestration of virtualized resources, and since our slicing implementation is based on a virtualized network, we need have this environment ready. The Cloud infrastructure is vital for serving as VIM where VNFs can be deployed. And finally the Performance measurement tools are important software, configuration, or commands made to the VIM and the orchestrator during their deployment to view the performance metrics such as CPU utilization, memory utilization in a graphical and readable forms. Different environment are considered as follows.

5.1.1 Orchestration tool

The orchestration tool used in this experiment was OSM [35]. OSM was deployed on a separate server different from the server where the Virtual Infrastructure Manager (VIM) was installed. OSM was used to orchestrate and instantiate the VNF, NSSI and NSI. In the description of network slice at the NST described in the multi-tenant layer, the slice type service such as uRLLC, mMTC and eMBB slices will be described. Also in the NST the status of every NSSI (NS) constituents forming the NSI will be described on whether they will be shared or non-shared slices.

However in order to achieve network slicing a lot of things need to be in place, thus this section discusses the procedure and open source tools used in achieving a network slicing. As seen in Figure 3, OSM proposes two methods of implementation for network slicing, in this thesis, the full end-to-end management for network slicing approach is followed where all the functionalities of the layers above the resource layer are treated as a single entity, managed and orchestrated by OSM as seen in Figure 11.

[7] https://yaml.org/


3GPP Slice related management functions
Communication Service Management Function
Network Slice Management Function
Network Slice Subnet Management Function
Open Source MANO
Os-Ma-Nfvo
NFVO
Slice Manager+NFVO
EMs
VNFM
VNFs
PNFs
NFVI
VIM
Figure 11. Full End-to-End management network slicing [36].

Thus, all the other layers including the slicing MANO layer, the service layer and the multi-tenant layer are implemented under a single umbrella, and the resource layer is implemented within a different umbrella.

The implementation of network slicing used in this thesis follows a hierarchical layered approach where one layer depends on the other as seen in figure 12. According to ETSI NFV-MANO, to implement a virtualized network, we need a VIM, a Network Function virtualization Infrastructure (NFVI) on top of where each VNFs will be deployed. ETSI NFV-MANO [28] also introduced other blocks like Virtual Network Function Manager (VNFM), and Network Function Virtualization Orchestrator (NFVO) for the management and orchestration of the VNFs.

Generally to have a virtualized network, the basis of the network are made of Virtual Machines (VMs) deployed on top of a NFVI in the cloud (VIM). The VM are described using a Virtual Descriptor Unit (VDU). The VDU depicts the basic information about instantiating the VM in the cloud, information such as the VM flavor, VM image, computing capacity etc. Multiple VDU are described together by a Virtual Network Function Descriptor (VNFD) to instantiate a VNF. Multiple VNFs are combined together to describe a Network Service (NS) which is the same as NSSI in the slicing MANO. The NS can be instantiated using a Network Service Descriptor (NSD) and managed by the NSSMF. Different NS/NSSIs are described together to form a NSI using a Network Slice Template (NST) and managed by the NSMF. Finally, an instantiated NSI is transmitted as a communication service down to the tenants and managed by the CSMF. It can be seen that the process above depicts the basis instantiating a network slice for our proposed architecture

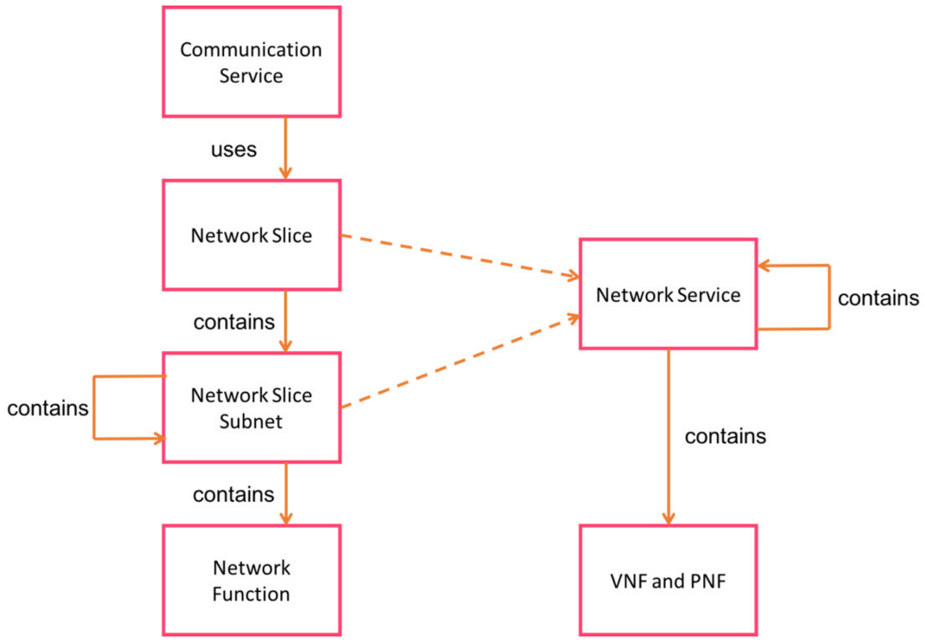


Figure 12. Hierarchical approach for network slicing [36].

5.1.2 VIM

In our implementation, openstack is used for the VIM. Openstack is a cloud infrastructure containing basic important nodes for virtualization. Basic nodes such as compute, storage and networking nodes are deployed. To achieve a highly efficient VIM, the free RAM after deploying openstack should be more than the RAM required for the image instantiation. Images are used to describe virtualized disk where operating system can be installed and it can be launched in the cloud. Hence, if the size of the images or VMs to be instantiated are large, the flavor for the images will be large and hence a much larger computing and storage resources will be required. For this experiment, the openstack was deployed on a 16GB RAM, 255GB SSD core i5 laptop and the free memory of 8GB after deploying openstack. Different VIM represent different domain of VNF instantiation. Generally a single openstack cloud infrastructure can have multiple administrator domains where different NSSIs or Network services can be instantiated. Thus we can either launch a NSI whose NSSIs are instantiated across a single domain i.e. under a single admin or project in the cloud, or under multiple project. We can also launch a single slice under multiple domain i.e. under different cloud (network) infrastructure, i.e. a single slice whose NSSI are launched across different networks, for example the CN NSSIs are instantiated in the micro-operator infrastructure and the AN NSSIs are instantiated in the MNO NSSIs as seen in Figure 4 where the NSI configuration types are described.

VM images are uploaded in the cloud for deploying the instances on them. However, due to the unavailability of virtualized Long Term Evolution (LTE) images at the time of this experiment, a set of dummy Ubuntu images that can accommodate multiple interfaces were used and uploaded to the cloud for instantiating the slices. The size of the image used is 400MB

5.1.3 Performance measurement tools

For performance measurement of different instantiated slices across different domain, the following open source tools are deployed to expose the metrics of the VMs from openstack and VNFs and NS from OSM.

The metrics measured in this experiment include CPU utilization which describes how each instantiated NSSI consumes the VIM's or cloud computation and processing resources. The consumption of resources can happen when a slice is just instantiated or after different services have started within the instantiated slice. However, to maintain a better processing power, the number of resources to be used can be scaled out or in to limit or allow the resource usage. Another Metrics measured is the Memory utilization which describes how the Random Access Memory (RAM) is used by every instantiated slice. Basically, the size of used memory by a VNF depends very much on the image flavor (i.e. configuration parameters of the images during instantiation), and also depends on the type of service the VNF is launching in the cloud. So all these summed up memory usage describes the memory utilization. The memory utilization can also be scaled in or out. The other two metrics measures the number or amount the packet sent and packet received between different NSSI instances. This is particularly useful in this experiment since we are having shared slices and we need to experiment the different amount of packets sent and packets received by a shared NSSI. However, the most general important metrics that fully determines performance of VNFs and equally performance of network slices are the CPU utilization and the memory utilization.

Thus, In order to expose this metrics from the cloud (VIM) where the resources are launched to the orchestrator and finally the graphical display, different performance measurement tools needs to be in place. This include the following:

Gnoochi and Ceilometer are the two open source tools that were deployed in the openstack form which the OSM is instantiating. Gnoochi and Ceilometer use an API to expose VM metrics. The metrics of the VMs whom slices are instantiated on are exposed to OSM and thus different performance analysis can be carried out.

Promethus and Graffana are used for graphic representation of the exposed metrics. Thus with Promethus and graffana, the analysis between shared and non-shared slice within a single and multiple domain can be seen. The Promethus displays the backend metrics, thus OSM exposes the metrics API to Promethus and the data can be transmitted to other display tools such as Graffana.

Network topology is a tool in openstack networking node called Neutron, The Network topology display a graphical relationship of every instance. The network topology thus shows how different slices are formed from the VM, to the VNF and the NS. The network topology shows the number of currently running instances in the cloud.

5.2 Implementation procedure

According to the environment above, our implementation of the network slice for a deployment scenario of a micro-operator is achieved as follows:

- Closed Deployment A's slice instantiation will be implemented across a single domain representing a micro-operator cloud infrastructure whose resources are launched at a single location and the NSSIs are instantiated as non-shared NSSI.

- Closed Deployment B's slices are instantiated with NSSI launched across multiple domain representing both micro-operator and MNO cloud infrastructure and their resources are at different locations.
- Open network is instantiated with different NSSI instantiated in the micro-operator cloud infrastructure for different MNOs subscribers.

Hence, this experiment covers some of the layers in the proposed architecture except slice distribution across different tenants at the multi-tenant layer and combining different NSIs from different operators for a single tenant. Research is currently still on going to achieved that.

In the network slicing experimental implementation performed in this thesis, as seen in Figure 13, 3 NSSIs are used to form the NSI for a single slice, the middle NSSI is made shared. Scripts for the Network slice template, the Network service descriptor for the NSSIs, and the VNFS are attached to appendices 1 to 6.

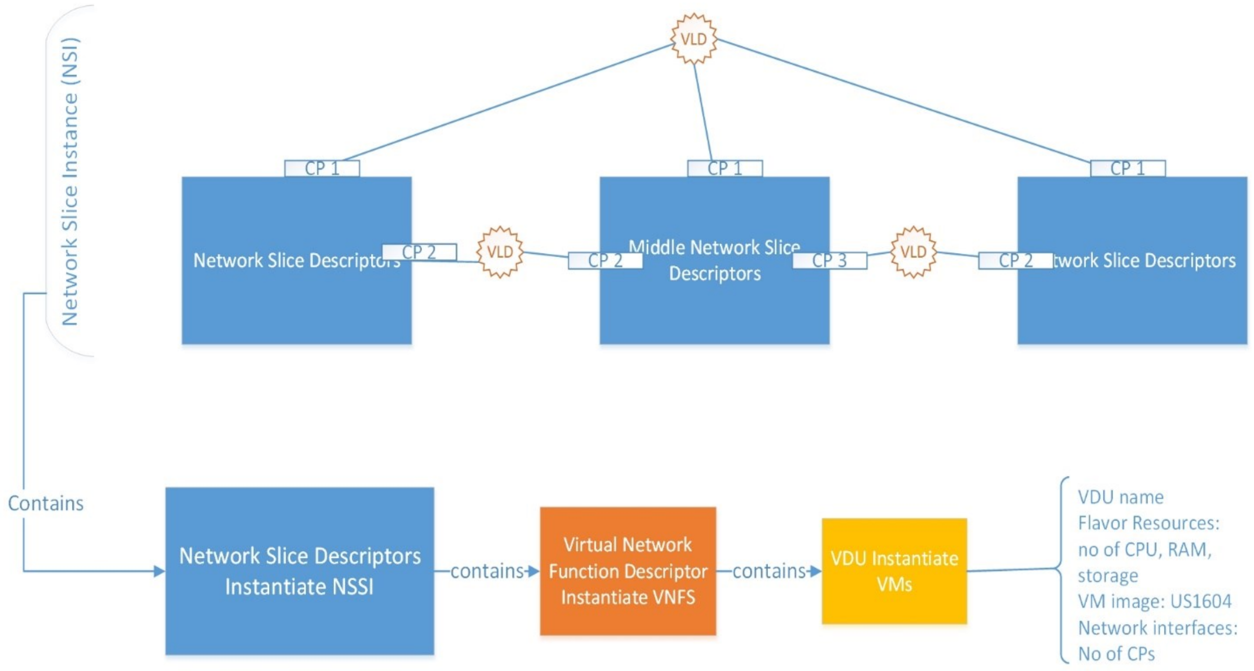

Network Slice Instance (NSI)
VLD
CP 1
CP 1
CP 1
Network Slice Descriptors
CP 2
VLD
CP 2
Middle Network Slice Descriptors
CP 3
VLD
CP 2
twork Slice Descriptors
Contains
Network Slice Descriptors Instantiate NSSI
contains
Virtual Network Function Descriptor Instantiate VNFS
contains
VDU Instantiate VMs
VDU name
Flavor Resources: no of CPU, RAM, storage
VM image: US1604
Network interfaces: No of CPs

Figure 13. Network slicing experimental setup for non-shared NSSI.

A single VNFD is used to form two end NSDs (NSD 1 and NSD 3) and a separate VNFD is used for the middle NSD (NSD 2). This is because even if they are instantiating the same images in the cloud, their interfaces are different. For a real LTE network, the NSD will be made up of VNFD representing different network components like Home Subscriber Server (HSS), Mobility Management Entity (MME), etc.

To achieved shared slicing, the middle NSD instantiating the middle NSSI is enabled to be shared, and a new NSI is created with a new separate NSD connecting to the middle NSD while the first slice is running as seen in Figure 14.

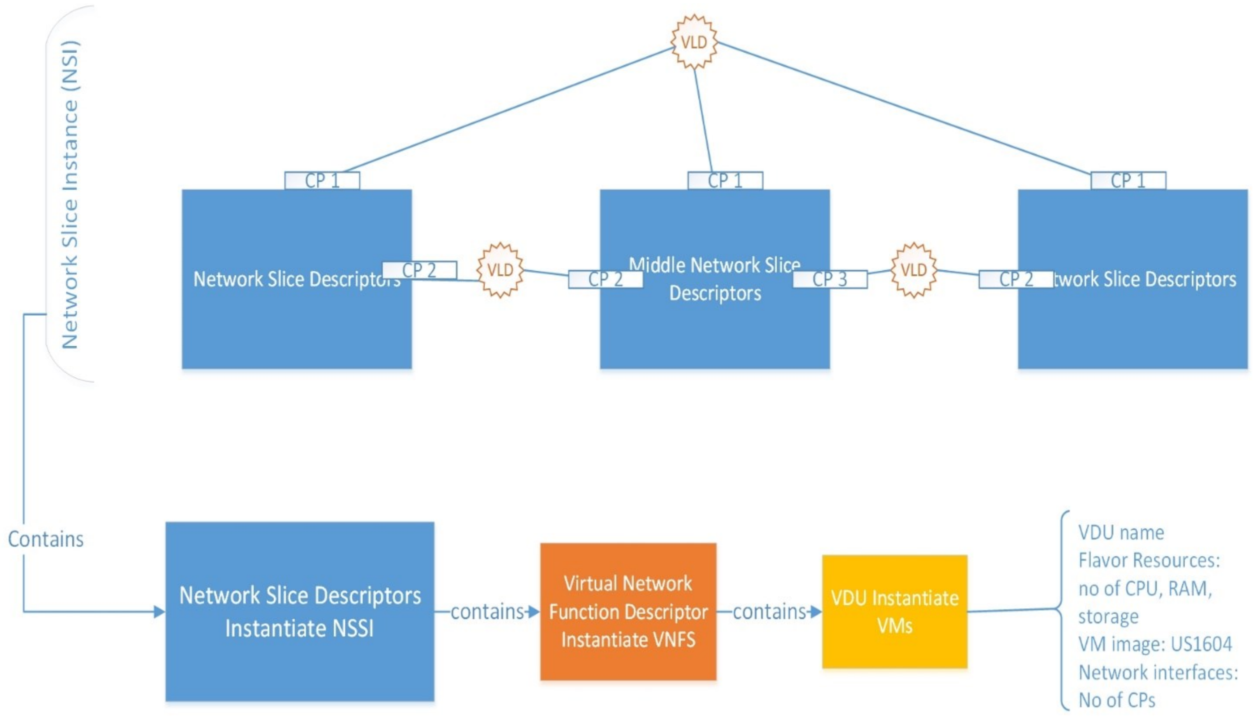

Network Slice Instance (NSI)
VLD
CP 1
CP 1
CP 1
Network Slice Descriptors
CP 2
VLD
CP 2
Middle Network Slice Descriptors
CP 3
VLD
CP 2
twork Slice Descriptors
Contains
Network Slice Descriptors Instantiate NSSI
contains
Virtual Network Function Descriptor Instantiate VNFS
contains
VDU Instantiate VMs
VDU name
Flavor Resources: no of CPU, RAM, storage
VM image: US1604
Network interfaces: No of CPs

Figure 14. Network slicing experimental setup for shared NSSI.

Basically, the advantage of the network slicing is that each NSI can be instantiated multiple times and multiple end-to-end networks can be created as many times as possible. So in a real network with real Evolved Packet Core (EPC), the multiple simultaneous networks under a single physical resource can be created and attributed to different tenants within a vertical, based on their network requirement.

The only drawback with that is that, whenever a network slice is instantiated, each NSSI that form the NSI will take new resources in terms of memory, CPU etc. and hence needs to be scaled down. In this implementation, we tested for available memory using command (*free –h)* from the first seconds of instantiation to the 20 seconds and observe the RAM usage within each instance. This was tested for both shared and non-shared slices. The graph in Figure 15 shows how the memory usage in the VIM reacts for every new slice instantiated.

Figure15 proves two things; first whenever a new instance is launching the number of memory used in the VIM will increase and the available memory will reduce. After the instance is launched, the memory become available afterwards except when there are functions or operations running within the instance. However, for an NSI with shared NSSI constituent, the memory usage is less than when there is no shared NSSI, and as such the available memory remains relatively the same even after a new NSI is launched. The reason why not so much memory is utilized for shared NSSI is because one of the NSSI (i.e. middle NSSI) is already launched, and it has already consume some of the memory and hence the newly attached NSSI (left NSSI) to the existing NSI 1 is forming a new NSI 2 will only occupy memory for the single NSSI (left NSSI) not a whole new NSI. Network topology depicting multiple shared and non-shared slices can be seen in figure 16.

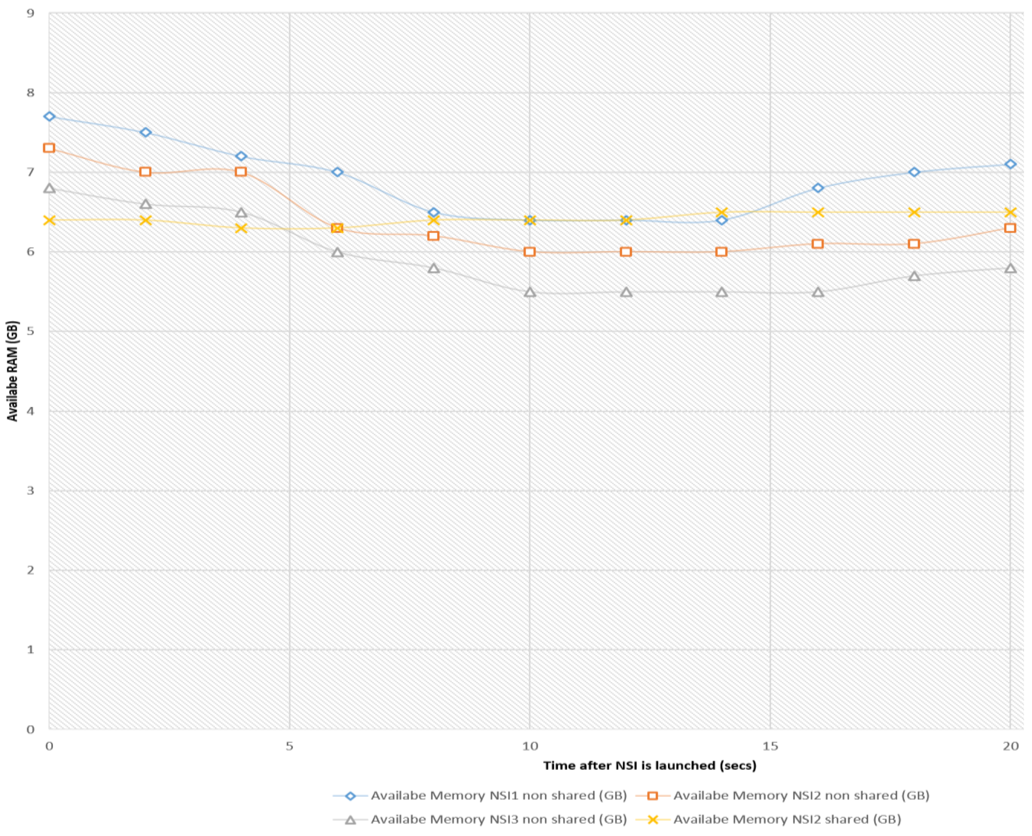


Figure 15. Graph of available RAM in VIM server after NSI is launched.

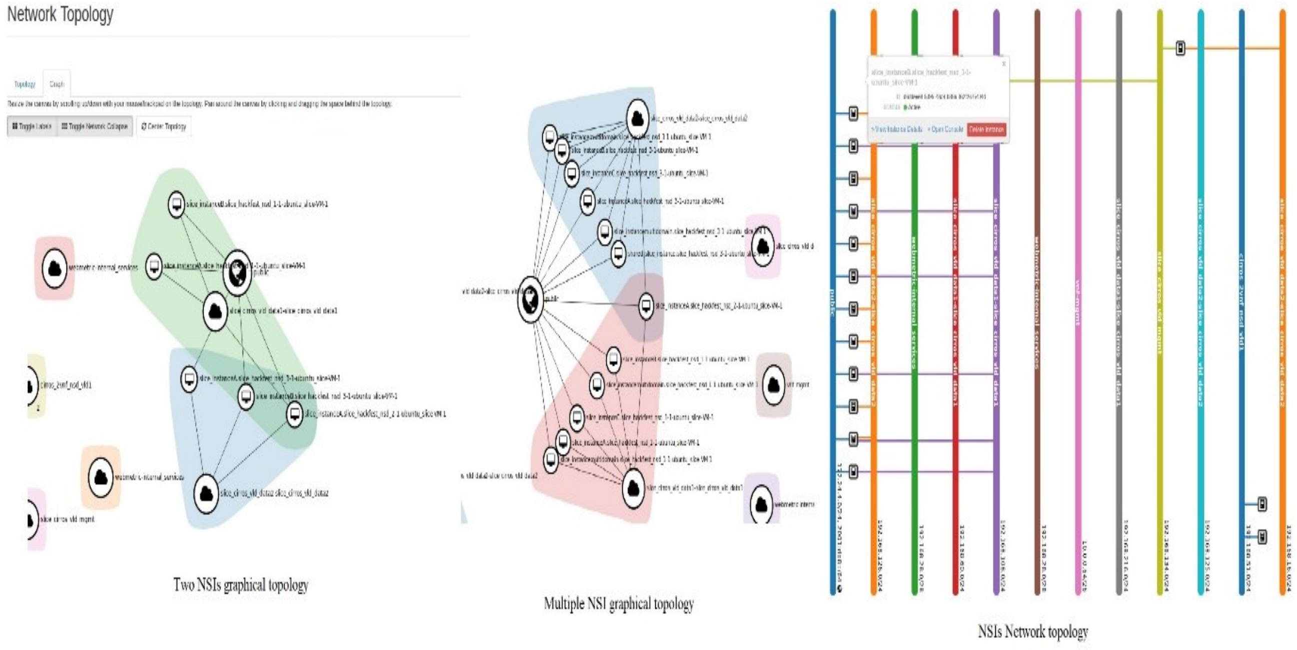


Figure 16. Network topology description for network slicing instances.

5.3 Network slice instantiation commands

In order to instantiate the network slices, the VNFD script **attached to this thesis as Appendix 1 and 2 for the left and middle VNFD ,** was written in a *ymal language* and the performance metrics enabler was also described in the script, afterwards, the VNFD packet were generated using the command:

devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t nsd -c slice_hackfest_vnfd

After the VNFD packet is generated, the package was created in OSM using the command:

osm vnfd-create slice_hackfest_vnfd.tar.gz
osm vnfd-create slice_hackfest_middle_vnfd.tar.gz

The NSD script attached to **this thesis as appendix b1 and b2 for the left and middle VNFD**, was also written in a *ymal language* , and packaged and the NS package was created in OSM using the command:

osm nsd-create slice_hackfest_nsd.tar.gz
osm nsd-create slice_hackfest_middle_nsd.tar.gz

Finally the NST scripts for both NSIs (with shared and Non-shared), attached to **this thesis as appendix c1 and c2 for both NSTs ,** was writtern in a *ymal language* , was also uploaded to OSM using the commands below:

osm nst-create slice_hackfest_nstd2.yaml
osm nst-create slice_hackfest_nstd.yaml

After all the scripts are created successfully in OSM, we need to instantiate the slice, we launched the instantiation commands directly from the command line. However, since we were planning to orchestrate the slices across different domain, the selection of VIM account was passed as configured when instantiating the slice.
Slice instantiation goes thus for the first Non-shared slice (NSI1) in a single domain using the command:

osm nsi-create --nsi_name my_first_slice --nst_name slice_hackfest_nstd --vim_account ***openstack*** *--config 'netslice-vld: [{name: slice_cirros_vld_mgmt, vim-network-name:* ***public****}]'*

Slice instantiation goes thus for the shared slice (NSI1) in a single domain using the command:

osm nsi-create --nsi_name my_first_slice --nst_name slice_hackfest_nstd2 --vim_account ***openstack*** *--config 'netslice-vld: [{name: slice_cirros_vld_mgmt, vim-network-name:* ***public****}]'*

Slice instantiation goes thus for the Non-shared slice (NSI1) in a multiple domain using the command:

*osm nsi-create --nsi_name my_first_slice --nst_name slice_hackfest_nstd2 --vim_account **openstack** --config 'netslice-vld: [{name: slice_cirros_vld_mgmt, vim-network-name: **public**}]' –config 'netslice-subnet: [{id:nssi_id, nsname: **nsd1,** vim-account: **openstack_new**}]'*

Slice instantiation goes thus for the shared slice (NSI1) in a multiple domain using the command:

*osm nsi-create --nsi_name my_first_slice --nst_name slice_hackfest_nstd2 --vim_account **openstack** --config 'netslice-vld: [{name: slice_cirros_vld_mgmt, vim-network-name: **public**}]' –config 'netslice-subnet: [{id:nssi_id, nsname: **nsd1,** vim-account: **openstack_new**}]'*

5.4 Performance analysis of Network slicing instantiation

The performance analysis of network slicing implemented in this thesis is based on the NSI configuration types, which forms the foundation for the classification of different the deployment scenarios of a micro-operator network.

Data from the instantiated slice's VNFs were generated, and exposed via the performance measurement tools. Afterwards, multiple slices were instantiated and the CPU utilization, memory utilization, packets send and packets received for each NSSI were plotted and generated via Promethus backend and exposed for graphical display on Graffana.

The metrics analysis from Promethus when a non-shared slice was instantiated can be seen in Figure 17

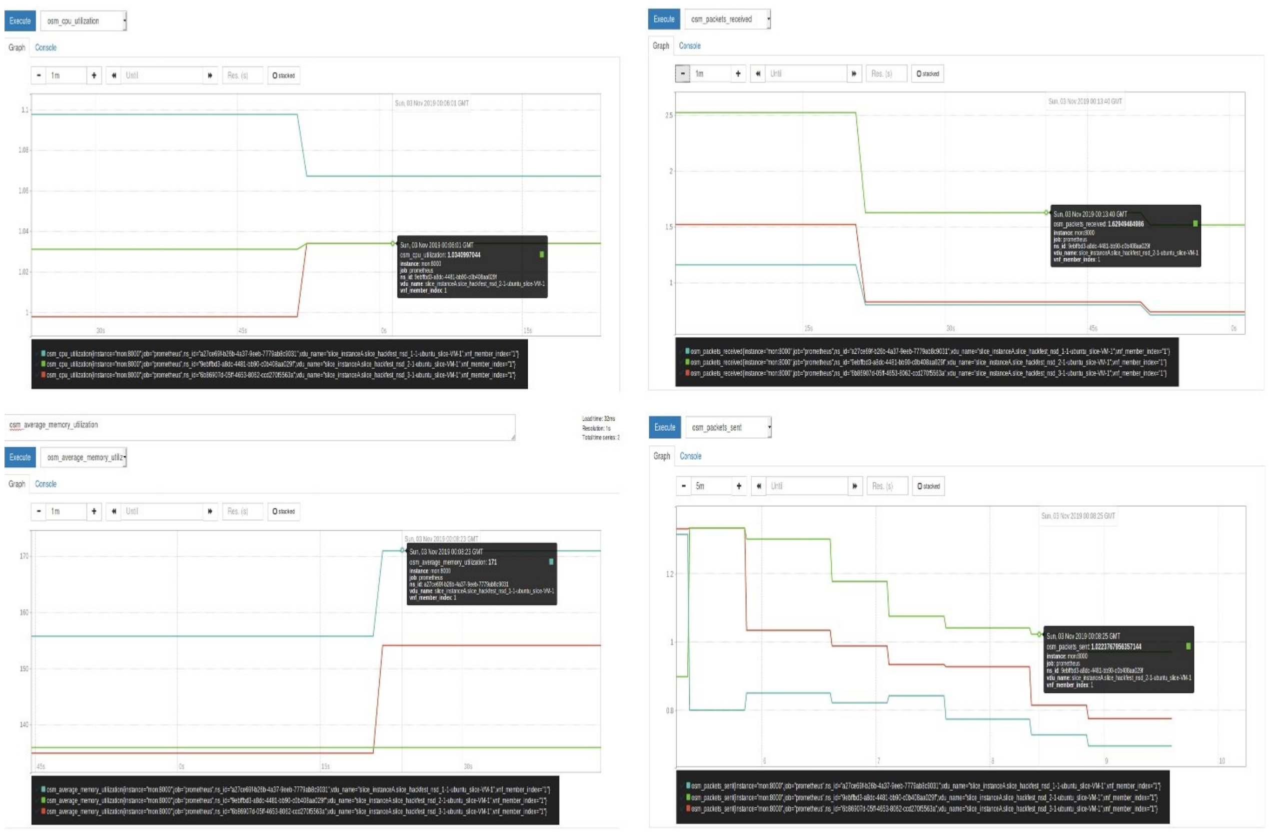


Figure 17. Performance metrics in Promethus for a non-shared slice.

The metrics display from Graffana when a non-shared slice was instantiated can be seen in Figure 18

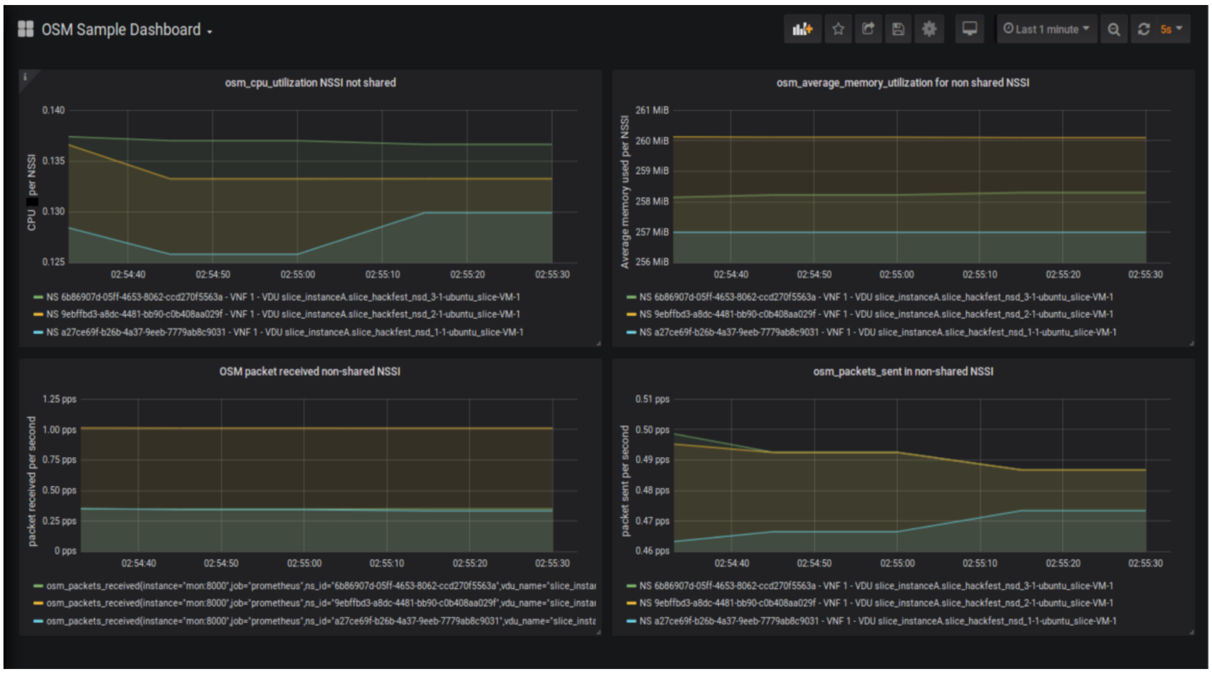


Figure 18. Performance metrics in Graffana for a non-shared slice.

The metrics analysis from Promethus when more non-shared slices were instantiated and when an already existing slice is running using the same slice template in the same VIM can be seen in Figure 19

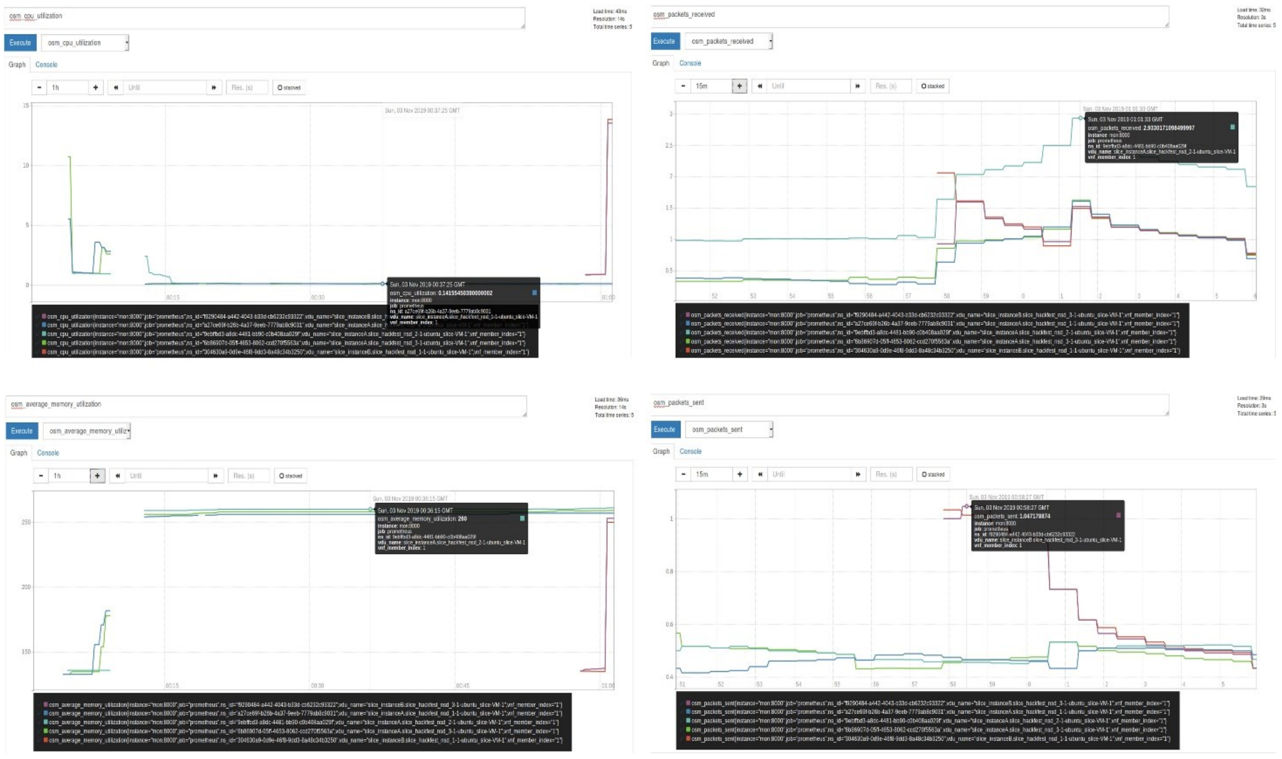
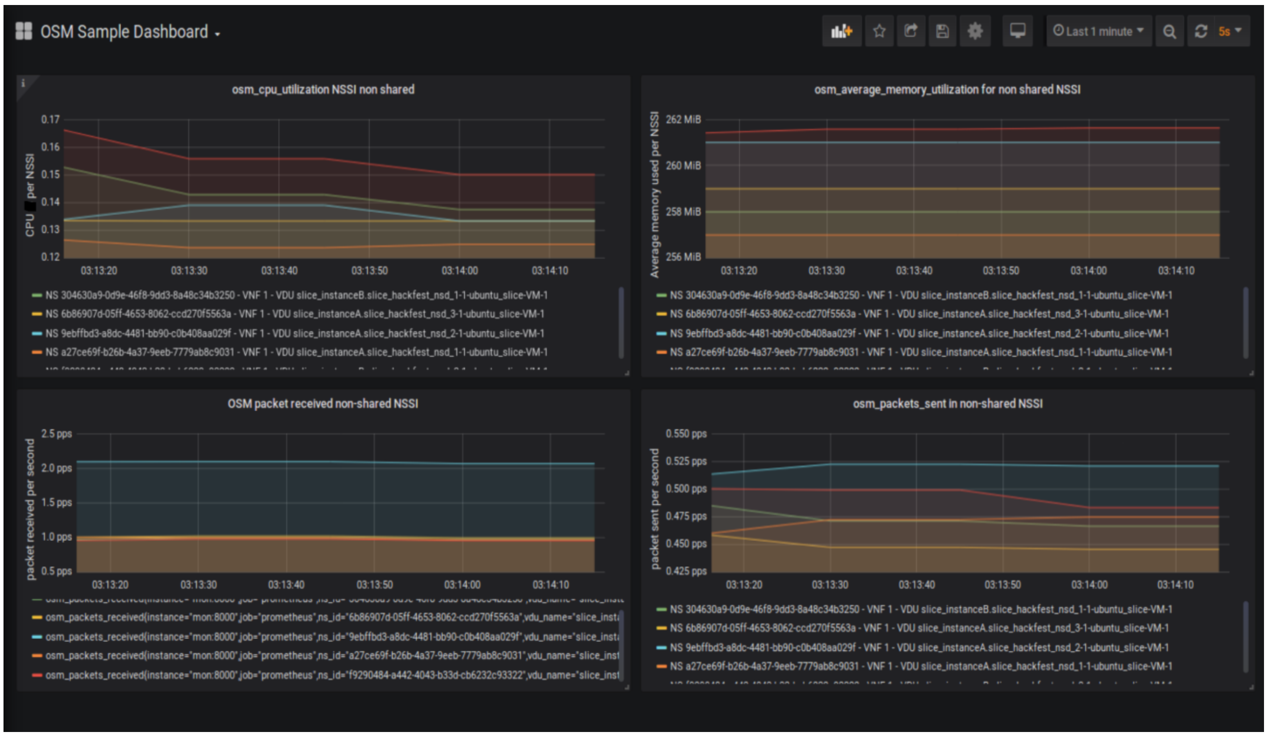


Figure 19. Performance metrics in Promethus when a new non-shared slice is launched.

The metrics display from Graffana when more non-shared slices were instantiated can be seen in Figure 20



NS 304630a9-0d9e-46f8-9dd3-8a48c34b3250 - VNF 1 - VDU slice_instanceB.slice_hackfest_nsd_1-1-ubuntu_slice-VM-1
NS 6b86907d-05ff-4653-8062-ccd270f5563a - VNF 1 - VDU slice_instanceA.slice_hackfest_nsd_3-1-ubuntu_slice-VM-1
NS 9ebffbd3-a8dc-4481-bb90-c0b408aa029f - VNF 1 - VDU slice_instanceA.slice_hackfest_nsd_2-1-ubuntu_slice-VM-1
NS a27ce69f-b26b-4a37-9eeb-7779ab8c9031 - VNF 1 - VDU slice_instanceA.slice_hackfest_nsd_1-1-ubuntu_slice-VM-1

osm_packets_received{instance="mon:8000",job="prometheus",ns_id="6b86907d-05ff-4653-8062-ccd270f5563a",vdu_name="slice_insta
osm_packets_received{instance="mon:8000",job="prometheus",ns_id="9ebffbd3-a8dc-4481-bb90-c0b408aa029f",vdu_name="slice_insta
osm_packets_received{instance="mon:8000",job="prometheus",ns_id="a27ce69f-b26b-4a37-9eeb-7779ab8c9031",vdu_name="slice_inst
osm_packets_received{instance="mon:8000",job="prometheus",ns_id="f9290484-a442-4043-b33d-cb6232c93322",vdu_name="slice_inst

Figure 20. Performance metrics in Graffana when a new non-shared slice is launched.

For shared slice, the following output results shows when a shared slice was instantiated on already running slice instance. In this case, a new nsd 3(new NSSI) is attached to the middle NSSI (nsd2). The performance output from promethus and graffana can be seen in Figures 21 and 22, respectively.

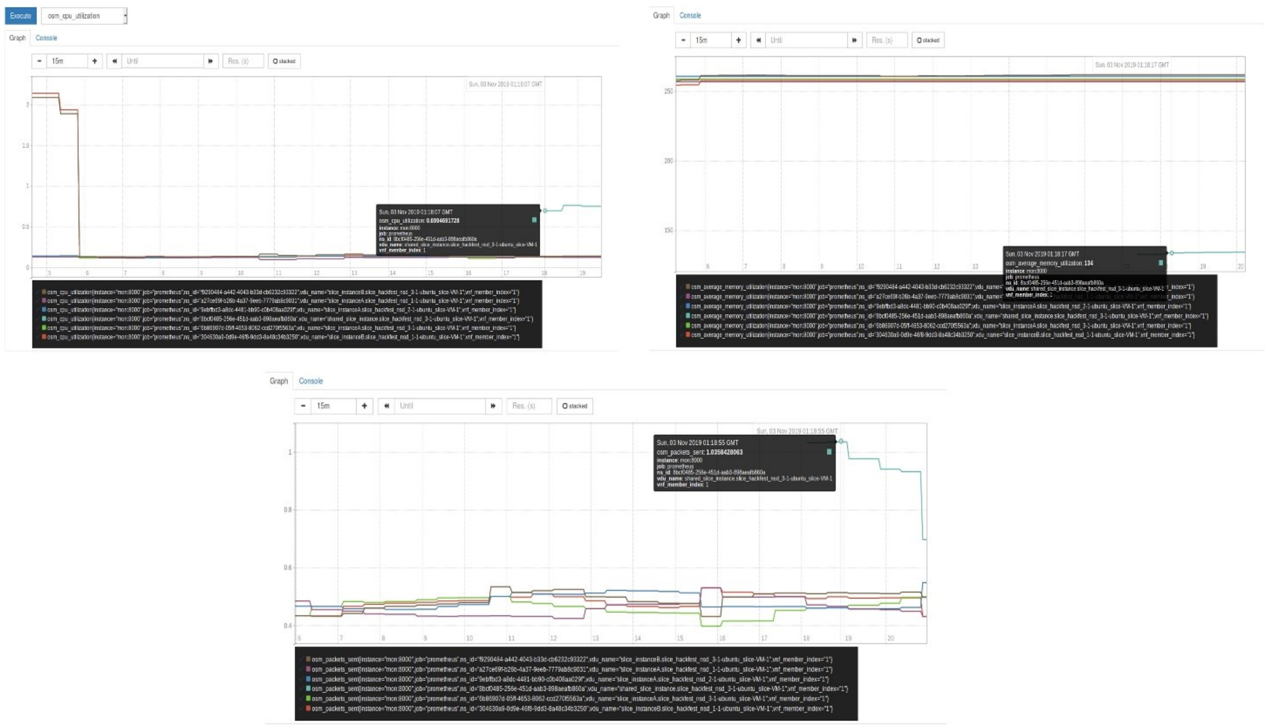
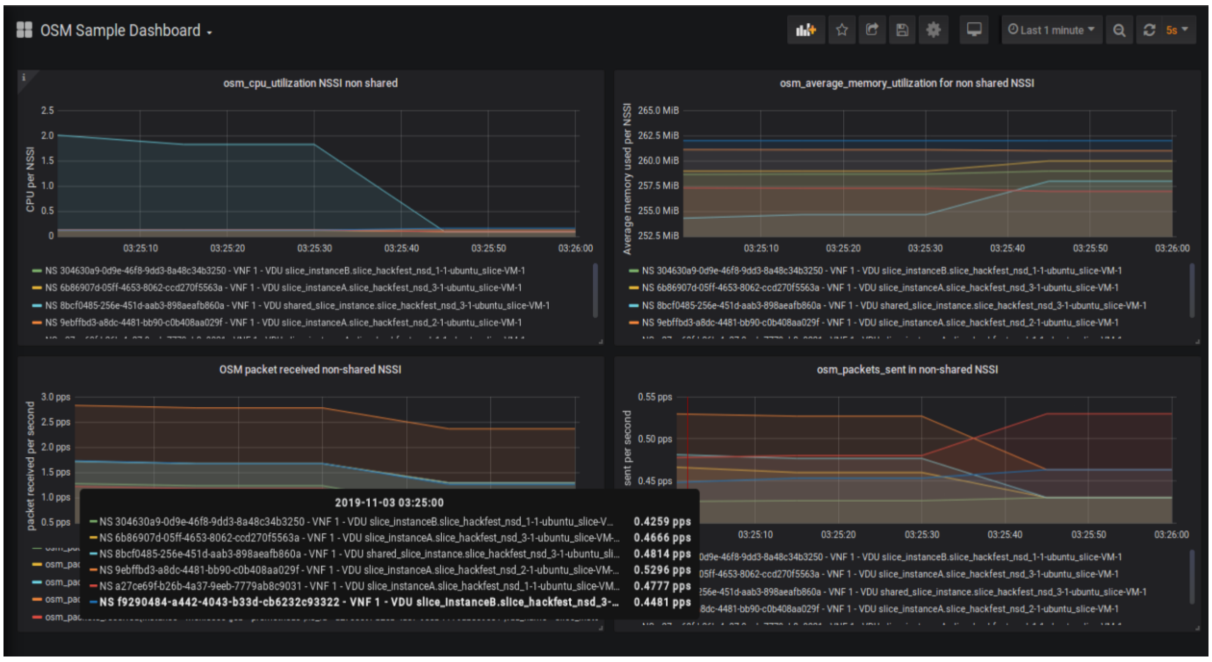


Figure 21. Performance metrics in Promethus when a new shared slice is launched.


OSM Sample Dashboard
Last 1 minute
5s
osm_cpu_utilization NSSI non shared
CPU per NSSI
osm_average_memory_utilization for non shared NSSI
Average memory used per NSSI
OSM packet received non-shared NSSI
packet received per second
osm_packets_sent in non-shared NSSI
sent per second
NS 304630a9-0d9e-46f8-9dd3-8a48c34b3250 - VNF 1 - VDU slice_instanceB.slice_hackfest_nsd_1-1-ubuntu_slice-VM-1
NS 6b86907d-05ff-4653-8062-ccd270f5563a - VNF 1 - VDU slice_instanceA.slice_hackfest_nsd_3-1-ubuntu_slice-VM-1
NS 8bcf0485-256e-451d-aab3-898aeafb860a - VNF 1 - VDU shared_slice_instance.slice_hackfest_nsd_3-1-ubuntu_slice-VM-1
NS 9ebffbd3-a8dc-4481-bb90-c0b408aa029f - VNF 1 - VDU slice_instanceA.slice_hackfest_nsd_2-1-ubuntu_slice-VM-1

```
2019-11-03 03:25:00
NS 304630a9-0d9e-46f8-9dd3-8a48c34b3250 - VNF 1 - VDU slice_instanceB.slice_hackfest_nsd_1-1-ubuntu_slice-V...   0.4259 pps
NS 6b86907d-05ff-4653-8062-ccd270f5563a - VNF 1 - VDU slice_instanceA.slice_hackfest_nsd_3-1-ubuntu_slice-VM-... 0.4666 pps
NS 8bcf0485-256e-451d-aab3-898aeafb860a - VNF 1 - VDU shared_slice_instance.slice_hackfest_nsd_3-1-ubuntu_sli... 0.4814 pps
NS 9ebffbd3-a8dc-4481-bb90-c0b408aa029f - VNF 1 - VDU slice_instanceA.slice_hackfest_nsd_2-1-ubuntu_slice-VM-... 0.5296 pps
NS a27ce69f-b26b-4a37-9eeb-7779ab8c9031 - VNF 1 - VDU slice_instanceA.slice_hackfest_nsd_1-1-ubuntu_slice-VM... 0.4777 pps
NS f9290484-a442-4043-b33d-cb6232c93322 - VNF 1 - VDU slice_instanceB.slice_hackfest_nsd_3-...                   0.4481 pps
```

Figure 22. Performance metrics in Graffana when a new shared slice is launched.

From Figure 17, 18, 19 and 20, it can be seen that memory usage increases when a slice is instantiated and then it becomes stable over time. The CPU utilization follows the same pattern. As seen from the result above, **nsd 1 2 and 3** represent the 3 NSSI that were instantiated for the non-shared NSI. One of the main result from the graph in Figures 19 and 20 is that the middle NSSI i.e. nsd2, which was declared as shared, tends to **have more memory usage and received more packets.** The reason for this is since the NSSI was declared shared, every new slice that was formed will just be connected to the existing middle NSSI. Also, the Packets received were also much for the middle slice whenever a new slices is connected to it as seen in Figure 20.

As shown in figure 21 and 22, only a new nsd 3 is launched since the shared slice nsd 2(middle NSSI) is already running. This makes the memory usage of the new nsd very low as confirmed in Figure 15. Also packet sent was higher for the new nsd 3 while packet received was increased for the nsd 3 (middle NSSI).

Generally, the performance analysis from the implementation of network slicing in this thesis was in two parts, the first part was checking the impact of memory usage on a shared and non-shared slices, and it can be concluded that even though shared slices might need better optimization of resources, it thus saves and uses less memory than non-shared slices when they are instantiated. The reason for that could basically be attributed to the fact that in non-shared slice all NSSIs are created newly and the NSI is dedicated for a specific end user, and as such they then to use more memory. Whereas the shared slices, some of the required NSSIs are already launched, so the shared slice will only launch new NSSIs that are currently not available, and with that it uses less memory compared to the non-shared slices. Results can be seen in figure 15

The second part of the performance analysis focus on testing implementation of shared and non-shared slices across different domain and across a single domain. The performance metrics measured during this implementation include the CPU utilization, memory utilization, packets sent and packets received. Figures 17 to 22 shows the result from both Promethus and Graffana. The output of the results affirms the conclusion made in the first experiment regarding memory usage for shared and non-shared slices. Meanwhile if a slice is instantiated across different domain, each domain will be responsible for memory utilization of the NSSI instantiated within them. That also depends on whether the domain are managed by a single administrator or separate administrator. For packet sent and packet received, it can be seen in figures 17 to 22, that the shared NSSI constituent, which in this case represented as NSD2 or the middle NSSI receives and send more packets since it’s responsible for communication between different slices.

6 DISCUSSION AND CONCLUSION

The research in this thesis has been able to establish reasons for the creation of specific network slicing architecture for micro-operator deployment scenarios leveraging multi-domain and multi-tenancy. The proposed architecture serves as basis for future works and development for network slicing in micro-operator. The thesis has been able to justify the proposed architecture with message sequence targeting individual blocks within the architecture. This thesis further described and implemented networks slicing using OSM as orchestration tool to achieve the proposed architecture for some deployment scenarios. Performance analysis in the thesis shows metrics such as CPU or memory utilization that show how non-shared slices use more resources than shared slices, however their performance is lower.

We highlighted different literature review in-terms of previous works that have been done before and during the course of this thesis. Different deployment scenario of a micro-operator Closed, Open and Mixed network were extensively described and how slicing can be achieved for each deployment scenarios were also proposed. We further describe the concept and application of multi-domain and multi-tenancy to achieving network slicing in a micro-operator network based on previously established work. We then introduce the proposed network slicing architecture for each deployment scenario. The architecture proposed is foreseen to cover all possible future use cases of a locally deployed 5G network while leveraging multi-tenancy and multi-domain. Finally the implementation of network slicing using open source tools show different performance metrics can be measured and hence how the network can be improved.

It can be deduced from the implementation performed in the thesis that extensive work is further needed to address some of the drawbacks of this current implementation. Drawback such as implementing the defined network slice service (eMBB, uRLLC and mMTC) in the NST, and transmitting the NSI to different tenants as communication services. Thus future works will involve proposing new implementation technique to achieve some of these challenged and address other deployment scenarios.

7 REFERENCES

[1] ITU-T, “ITU-T SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES,” *Futur. networks*, p. 26, 2018.

[2] M. Matinmikko, M. Latva-aho, P. Ahokangas, S. Yrjölä, and T. Koivumäki, “Micro Operators to Boost Local Service Delivery in 5G,” *Wirel. Pers. Commun.*, vol. 95, no. 1, pp. 69–82, 2017.

[3] M. Matinmikko-Blue and M. Latva-aho, “Micro operators accelerating 5G deployment,” in *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, 2017, pp. 1–5.

[4] A. Prasad, Z. Li, S. Holtmanns, and M. A. Uusitalo, “5G micro-operator networks — A key enabler for new verticals and markets,” in *2017 25th Telecommunication Forum (TELFOR)*, 2017, pp. 1–4.

[5] 3GPP TR 28.801, “Telecommunication management;Study on management and orchestration of network slicing for next generation network,” 2018.

[6] GSMA, “An Introduction to Network Slicing,” 2017. [Online]. Available: www.gsma.com. [Accessed: 25-Dec-2018].

[7] Ericsson, “Executive guide on Scalable network opportunities,” 2019.

[8] I. Badmus, M. Matinmikko-Blue, J. Singh, and W. T. Taleb, “Network Slice Instantiation for 5G Micro-Operator Deployment Scenario,” May 2019.

[9] Y. Siriwardhana, P. Porambage, M. Liyanage, J. S. Walia, M. Matinmikko-Blue, and M. Ylianttila, “Micro-Operator driven Local 5G Network Architecture for Industrial Internet,” Nov. 2018.

[10] J. S. Walia, H. Hammainen, and M. Matinmikko, “5G Micro-operators for the future campus: A techno-economic study,” *Jt. 13th CTTE 10th C. Conf. Internet Things - Bus. Model. Users, Networks*, vol. 2018-Janua, 2018.

[11] D. Schulz, “Intent-based automation networks: Toward a common reference model for the self-orchestration of industrial intranets,” in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 4657–4664.

[12] NGMN Alliance, “Description of Network Slicing Concept,” *white Pap.*, 2016.

[13] A. Devlic, A. Hamidian, D. Liang, M. Eriksson, A. Consoli, and J. Lundstedt, “NESMO: Network slicing management and orchestration framework,” in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2017, pp. 1202–1208.

[14] X. Zhou, R. Li, T. Chen, and H. Zhang, “Network slicing as a service: enabling enterprises’ own software-defined cellular networks,” *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 146–153, Jul. 2016.

[15] M. Matinmikko-Blue, S. Yrjoelae, and M. Latva-aho, “Micro Operators for Ultra-Dense Network Deployment with Network Slicing and Spectrum Micro Licensing,” in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, 2018, pp. 1–6.

[16] T. Taleb, B. Mada, M.-I. Corici, A. Nakao, and H. Flinck, “PERMIT: Network Slicing for Personalized 5G Mobile Telecommunications,” *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 88–93, May 2017.

[17] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, “Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions,” *IEEE Commun. Surv. Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[18] N. Zhang *et al.*, “Software Defined Networking Enabled Wireless Network Virtualization: Challenges and Solutions,” *IEEE Netw.*, vol. 31, no. 5, pp. 42–49, 2017.

[19] I. AFOLABI, A. KSENTINI, M. BAGAA, T. TALEB, M. CORICI, and A. NAKAO, “Towards 5G Network Slicing over Multiple-Domains,” *IEICE Trans. Commun.*, vol. E100.B, no. 11, pp. 1992–2006, 2017.

[20] M. Matinmikko, M. Latva-aho, P. Ahokangas, and V. Seppänen, “On regulations for 5G: Micro licensing for locally operated networks,” *Telecomm. Policy*, vol. 42, no. 8, pp. 622–635, Sep. 2018.

[21] J. S. Walia, H. Hämmäinen, K. Kilkki, and S. Yrjölä, “5G network slicing strategies for a smart factory,” *Comput. Ind.*, vol. 111, pp. 108–120, Oct. 2019.

[22] University of Oulu, “Interdisciplinary research showing the way for 5G micro-operators | Centre for Wireless Communications.” [Online]. Available: https://www.oulu.fi/cwc/node/53350. [Accessed: 25-Dec-2018].

[23] University of Oulu, “Micro operator concept for boosting local service delivery in 5G.” [Online]. Available: https://www.oulu.fi/uo5g/. [Accessed: 25-Dec-2018].

[24] Y. Chen, L. Duan, and Q. Zhang, “Financial analysis of 4G network deployment,” in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 1607–1615.

[25] 3GPP TS 23.501 v15.2.0, “System Architecture for the 5G System,” 2018.

[26] ONF TR-526, “Applying SDN Architecture to 5G Slicing Issue 1,” 2016.

[27] ETSI, “GR NFV-EVE 012 - V3.1.1 - Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework,” 2017.

[28] ETSI, “GR NFV-IFA 022 - V3.1.1 - Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on Management and Connectivity for Multi-Site Services,” 2018.

[29] Q. Wang *et al.*, “SliceNet: End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks,” in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2018, pp. 1–5.

[30] 5G!Pagoda, “A network slice for every service!,” 2016. .

[31] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, “On Multi-domain Network Slicing Orchestration Architecture and Federated Resource Control,” *IEEE Netw.*, pp. 1–11, 2019.

[32] 5GTransformer, “5G Platform for Vertical Actors.” [Online]. Available: http://5g-transformer.eu.

[33] 5GNORMA, “Novel Radio Multiservice adaptive network Architecture for 5G networks.” [Online]. Available: http://www.it.uc3m.es/wnl/5gnorma/.

[34] B. Sayadi *et al.*, “SDN for 5G Mobile Networks: NORMA Perspective,” Springer, Cham, 2016, pp. 741–753.

[35] ETSI, “OSM: Open-Source MANO,” 2016. [Online]. Available: https://osm.etsi.org/.

[36] Linux foundation, “ONAP: Open Network Automation Platform,” 2016. [Online]. Available: https://www.onap.org/.

[37] FOKUS, “OpenBaton,” 2015. [Online]. Available: https://openbaton.github.io/.

[38] OpenStack, “OpenStack Tacker,” 2015. [Online]. Available: https://wiki.openstack.org/wiki/Tacker.

[39] OASIS TOSCA TC, “Topology and Orchestration Specification for Cloud Applications Version 1.0 OASIS Standard,” 2013.

[40] I. Badmus, M. Matinmikko-Blue, and J. S. Walia, “Network slicing management technique for local 5G micro-operator deployments,” in *Proceedings of the International Symposium on Wireless Communication Systems*, 2019, vol. 2019-August, pp. 697–702.

[41] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, “From network sharing to multi-tenancy: The 5G network slice broker,” *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 32–39, 2016.

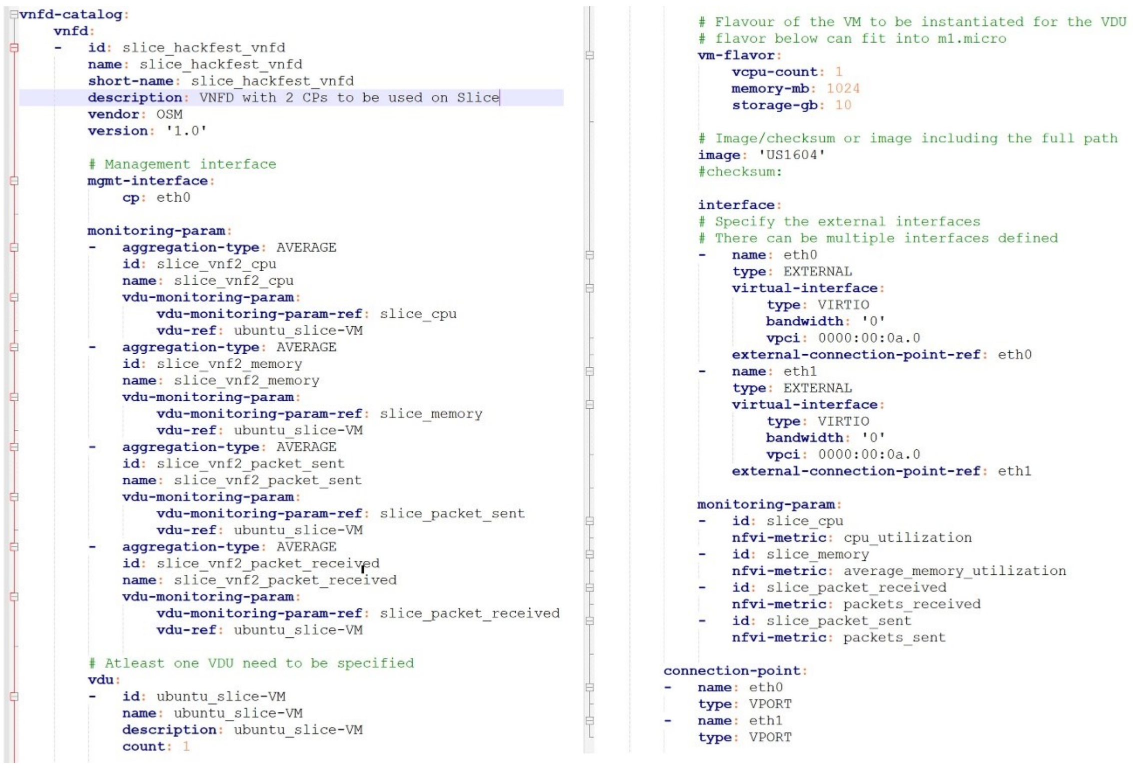
[42] F. Z. Yousaf, V. Sciancalepore, M. Liebsch, and X. Costa-Perez, “MANOaaS: A Multi-Tenant NFV MANO for 5G Network Slices,” *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 103–109, May 2019.

[43] 3GPP TR 23.799 v14.0.0, “Study on Architecture for Next Generation System,” 2016.

[44] R. Guerzoni *et al.*, “Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: an architectural survey,” *Trans. Emerg. Telecommun. Technol.*, vol. 28, no. 4, p. e3103, Apr. 2017.

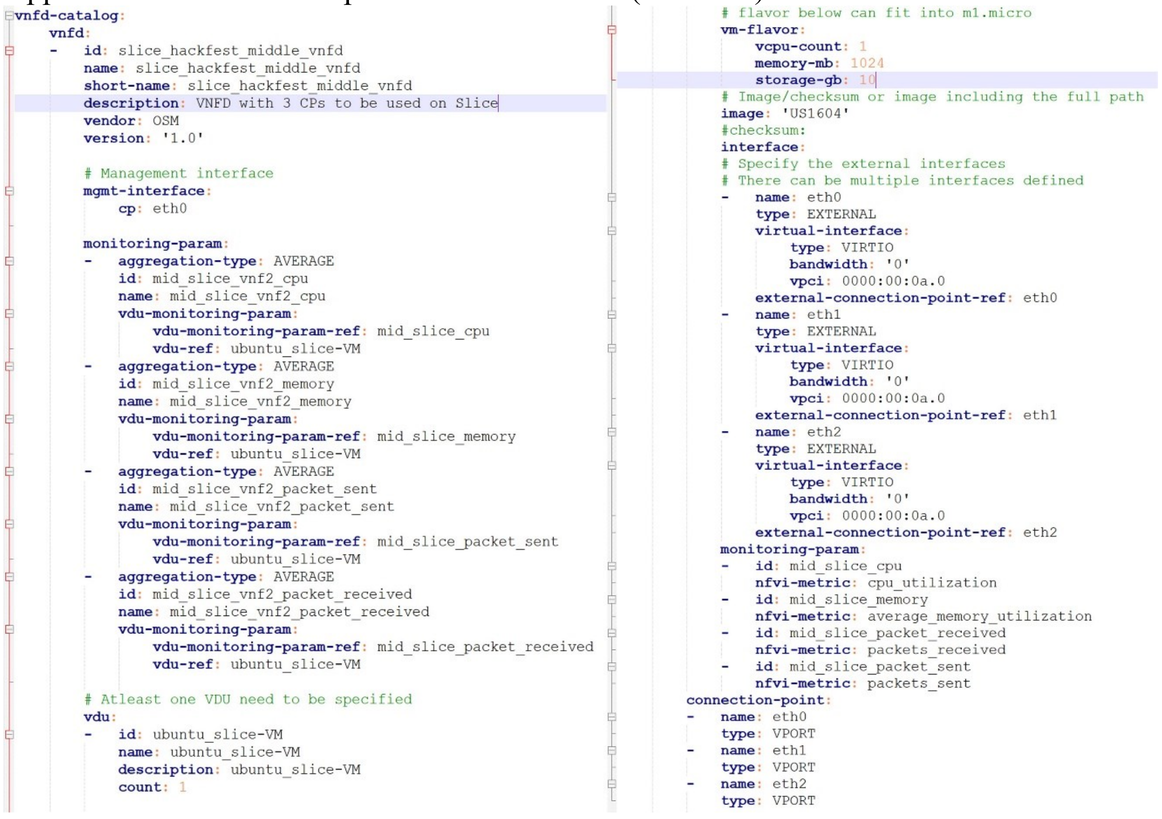
8 APPENDICES

Appendix 1 VNF descriptor for left VNFs (for nsd 1 and nsd 3)



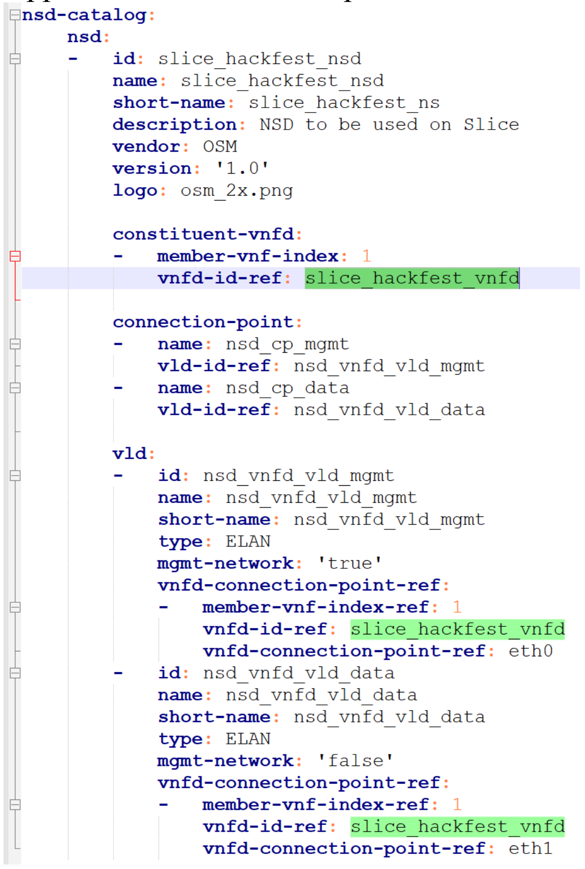
```yaml
vnfd-catalog:
    vnfd:
    -   id: slice_hackfest_vnfd
        name: slice_hackfest_vnfd
        short-name: slice_hackfest_vnfd
        description: VNFD with 2 CPs to be used on Slice
        vendor: OSM
        version: '1.0'

        # Management interface
        mgmt-interface:
            cp: eth0

        monitoring-param:
        -   aggregation-type: AVERAGE
            id: slice_vnf2_cpu
            name: slice_vnf2_cpu
            vdu-monitoring-param:
                vdu-monitoring-param-ref: slice_cpu
                vdu-ref: ubuntu_slice-VM
        -   aggregation-type: AVERAGE
            id: slice_vnf2_memory
            name: slice_vnf2_memory
            vdu-monitoring-param:
                vdu-monitoring-param-ref: slice_memory
                vdu-ref: ubuntu_slice-VM
        -   aggregation-type: AVERAGE
            id: slice_vnf2_packet_sent
            name: slice_vnf2_packet_sent
            vdu-monitoring-param:
                vdu-monitoring-param-ref: slice_packet_sent
                vdu-ref: ubuntu_slice-VM
        -   aggregation-type: AVERAGE
            id: slice_vnf2_packet_received
            name: slice_vnf2_packet_received
            vdu-monitoring-param:
                vdu-monitoring-param-ref: slice_packet_received
                vdu-ref: ubuntu_slice-VM

        # Atleast one VDU need to be specified
        vdu:
        -   id: ubuntu_slice-VM
            name: ubuntu_slice-VM
            description: ubuntu_slice-VM
            count: 1

            # Flavour of the VM to be instantiated for the VDU
            # flavor below can fit into m1.micro
            vm-flavor:
                vcpu-count: 1
                memory-mb: 1024
                storage-gb: 10

            # Image/checksum or image including the full path
            image: 'US1604'
            #checksum:

            interface:
            # Specify the external interfaces
            # There can be multiple interfaces defined
            -   name: eth0
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                    bandwidth: '0'
                    vpci: 0000:00:0a.0
                external-connection-point-ref: eth0
            -   name: eth1
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                    bandwidth: '0'
                    vpci: 0000:00:0a.0
                external-connection-point-ref: eth1

            monitoring-param:
            -   id: slice_cpu
                nfvi-metric: cpu_utilization
            -   id: slice_memory
                nfvi-metric: average_memory_utilization
            -   id: slice_packet_received
                nfvi-metric: packets_received
            -   id: slice_packet_sent
                nfvi-metric: packets_sent

        connection-point:
        -   name: eth0
            type: VPORT
        -   name: eth1
            type: VPORT
```

Appendix 2 VNF descriptor for middle VNFs (for nsd 2)



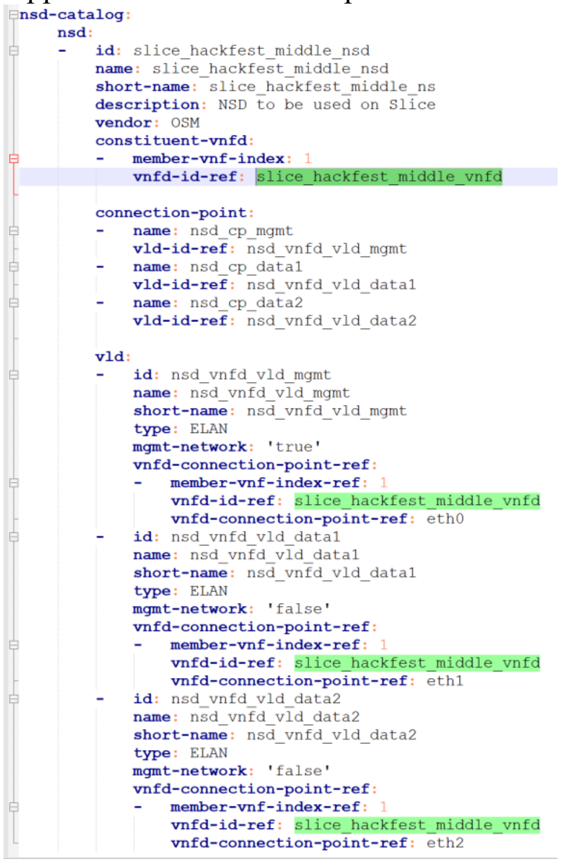
```yaml
vnfd-catalog:
    vnfd:
    -   id: slice_hackfest_middle_vnfd
        name: slice_hackfest_middle_vnfd
        short-name: slice_hackfest_middle_vnfd
        description: VNFD with 3 CPs to be used on Slice
        vendor: OSM
        version: '1.0'

        # Management interface
        mgmt-interface:
            cp: eth0

        monitoring-param:
        -   aggregation-type: AVERAGE
            id: mid_slice_vnf2_cpu
            name: mid_slice_vnf2_cpu
            vdu-monitoring-param:
                vdu-monitoring-param-ref: mid_slice_cpu
                vdu-ref: ubuntu_slice-VM
        -   aggregation-type: AVERAGE
            id: mid_slice_vnf2_memory
            name: mid_slice_vnf2_memory
            vdu-monitoring-param:
                vdu-monitoring-param-ref: mid_slice_memory
                vdu-ref: ubuntu_slice-VM
        -   aggregation-type: AVERAGE
            id: mid_slice_vnf2_packet_sent
            name: mid_slice_vnf2_packet_sent
            vdu-monitoring-param:
                vdu-monitoring-param-ref: mid_slice_packet_sent
                vdu-ref: ubuntu_slice-VM
        -   aggregation-type: AVERAGE
            id: mid_slice_vnf2_packet_received
            name: mid_slice_vnf2_packet_received
            vdu-monitoring-param:
                vdu-monitoring-param-ref: mid_slice_packet_received
                vdu-ref: ubuntu_slice-VM

        # Atleast one VDU need to be specified
        vdu:
        -   id: ubuntu_slice-VM
            name: ubuntu_slice-VM
            description: ubuntu_slice-VM
            count: 1
            # flavor below can fit into m1.micro
            vm-flavor:
                vcpu-count: 1
                memory-mb: 1024
                storage-gb: 10
            # Image/checksum or image including the full path
            image: 'US1604'
            #checksum:
            interface:
            # Specify the external interfaces
            # There can be multiple interfaces defined
            -   name: eth0
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                    bandwidth: '0'
                    vpci: 0000:00:0a.0
                external-connection-point-ref: eth0
            -   name: eth1
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                    bandwidth: '0'
                    vpci: 0000:00:0a.0
                external-connection-point-ref: eth1
            -   name: eth2
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                    bandwidth: '0'
                    vpci: 0000:00:0a.0
                external-connection-point-ref: eth2
            monitoring-param:
            -   id: mid_slice_cpu
                nfvi-metric: cpu_utilization
            -   id: mid_slice_memory
                nfvi-metric: average_memory_utilization
            -   id: mid_slice_packet_received
                nfvi-metric: packets_received
            -   id: mid_slice_packet_sent
                nfvi-metric: packets_sent
        connection-point:
        -   name: eth0
            type: VPORT
        -   name: eth1
            type: VPORT
        -   name: eth2
            type: VPORT
```

Appendix 3 NSD descriptor for nsd 1 and nsd 3



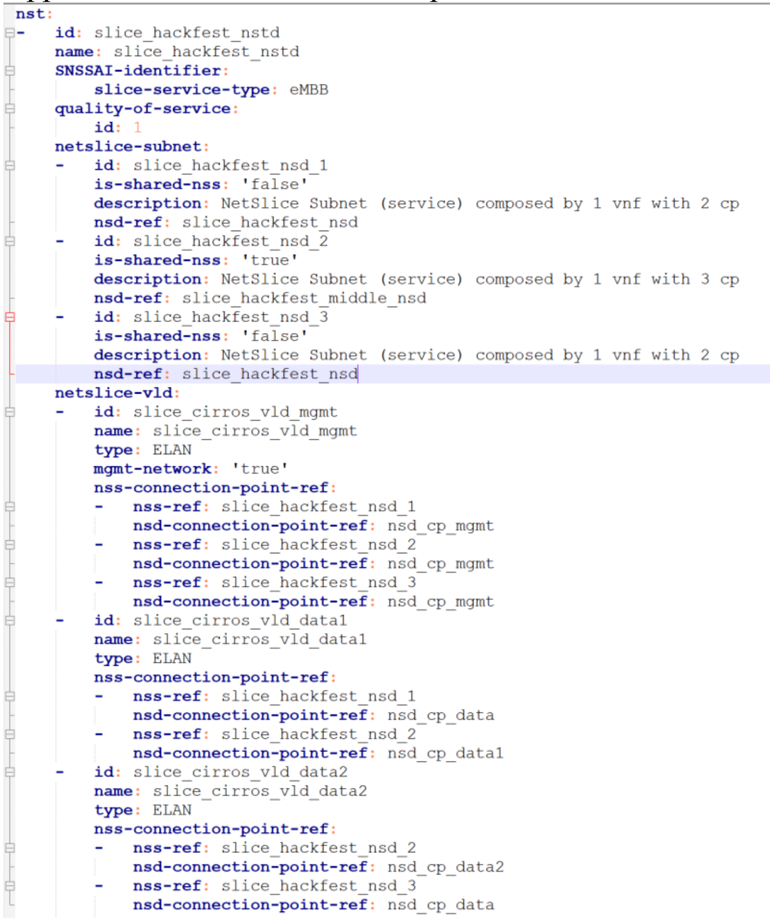
```
nsd-catalog:
    nsd:
    -   id: slice_hackfest_nsd
        name: slice_hackfest_nsd
        short-name: slice_hackfest_ns
        description: NSD to be used on Slice
        vendor: OSM
        version: '1.0'
        logo: osm_2x.png

        constituent-vnfd:
        -   member-vnf-index: 1
            vnfd-id-ref: slice_hackfest_vnfd

        connection-point:
        -   name: nsd_cp_mgmt
            vld-id-ref: nsd_vnfd_vld_mgmt
        -   name: nsd_cp_data
            vld-id-ref: nsd_vnfd_vld_data

        vld:
        -   id: nsd_vnfd_vld_mgmt
            name: nsd_vnfd_vld_mgmt
            short-name: nsd_vnfd_vld_mgmt
            type: ELAN
            mgmt-network: 'true'
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-id-ref: slice_hackfest_vnfd
                vnfd-connection-point-ref: eth0
        -   id: nsd_vnfd_vld_data
            name: nsd_vnfd_vld_data
            short-name: nsd_vnfd_vld_data
            type: ELAN
            mgmt-network: 'false'
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-id-ref: slice_hackfest_vnfd
                vnfd-connection-point-ref: eth1
```

Appendix 4 NSD descriptor for nsd 2



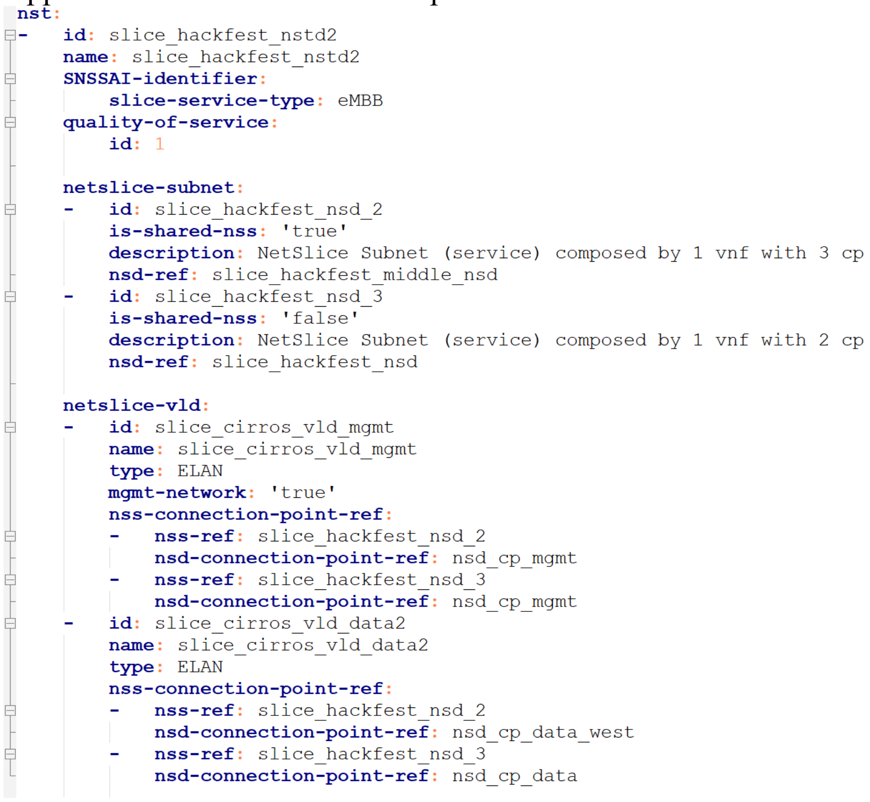
```
nsd-catalog:
    nsd:
    -   id: slice_hackfest_middle_nsd
        name: slice_hackfest_middle_nsd
        short-name: slice_hackfest_middle_ns
        description: NSD to be used on Slice
        vendor: OSM
        constituent-vnfd:
        -   member-vnf-index: 1
            vnfd-id-ref: slice_hackfest_middle_vnfd

        connection-point:
        -   name: nsd_cp_mgmt
            vld-id-ref: nsd_vnfd_vld_mgmt
        -   name: nsd_cp_data1
            vld-id-ref: nsd_vnfd_vld_data1
        -   name: nsd_cp_data2
            vld-id-ref: nsd_vnfd_vld_data2

        vld:
        -   id: nsd_vnfd_vld_mgmt
            name: nsd_vnfd_vld_mgmt
            short-name: nsd_vnfd_vld_mgmt
            type: ELAN
            mgmt-network: 'true'
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-id-ref: slice_hackfest_middle_vnfd
                vnfd-connection-point-ref: eth0
        -   id: nsd_vnfd_vld_data1
            name: nsd_vnfd_vld_data1
            short-name: nsd_vnfd_vld_data1
            type: ELAN
            mgmt-network: 'false'
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-id-ref: slice_hackfest_middle_vnfd
                vnfd-connection-point-ref: eth1
        -   id: nsd_vnfd_vld_data2
            name: nsd_vnfd_vld_data2
            short-name: nsd_vnfd_vld_data2
            type: ELAN
            mgmt-network: 'false'
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-id-ref: slice_hackfest_middle_vnfd
                vnfd-connection-point-ref: eth2
```

Appendix 5 Network slice template for non-shared slice



```
nst:
-   id: slice_hackfest_nstd
    name: slice_hackfest_nstd
    SNSSAI-identifier:
        slice-service-type: eMBB
    quality-of-service:
        id: 1
    netslice-subnet:
    -   id: slice_hackfest_nsd_1
        is-shared-nss: 'false'
        description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
        nsd-ref: slice_hackfest_nsd
    -   id: slice_hackfest_nsd_2
        is-shared-nss: 'true'
        description: NetSlice Subnet (service) composed by 1 vnf with 3 cp
        nsd-ref: slice_hackfest_middle_nsd
    -   id: slice_hackfest_nsd_3
        is-shared-nss: 'false'
        description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
        nsd-ref: slice_hackfest_nsd
    netslice-vld:
    -   id: slice_cirros_vld_mgmt
        name: slice_cirros_vld_mgmt
        type: ELAN
        mgmt-network: 'true'
        nss-connection-point-ref:
        -   nss-ref: slice_hackfest_nsd_1
            nsd-connection-point-ref: nsd_cp_mgmt
        -   nss-ref: slice_hackfest_nsd_2
            nsd-connection-point-ref: nsd_cp_mgmt
        -   nss-ref: slice_hackfest_nsd_3
            nsd-connection-point-ref: nsd_cp_mgmt
    -   id: slice_cirros_vld_data1
        name: slice_cirros_vld_data1
        type: ELAN
        nss-connection-point-ref:
        -   nss-ref: slice_hackfest_nsd_1
            nsd-connection-point-ref: nsd_cp_data
        -   nss-ref: slice_hackfest_nsd_2
            nsd-connection-point-ref: nsd_cp_data1
    -   id: slice_cirros_vld_data2
        name: slice_cirros_vld_data2
        type: ELAN
        nss-connection-point-ref:
        -   nss-ref: slice_hackfest_nsd_2
            nsd-connection-point-ref: nsd_cp_data2
        -   nss-ref: slice_hackfest_nsd_3
            nsd-connection-point-ref: nsd_cp_data
```

Appendix 6 Network slice template for shared slice



```
nst:
-   id: slice_hackfest_nstd2
    name: slice_hackfest_nstd2
    SNSSAI-identifier:
        slice-service-type: eMBB
    quality-of-service:
        id: 1

    netslice-subnet:
    -   id: slice_hackfest_nsd_2
        is-shared-nss: 'true'
        description: NetSlice Subnet (service) composed by 1 vnf with 3 cp
        nsd-ref: slice_hackfest_middle_nsd
    -   id: slice_hackfest_nsd_3
        is-shared-nss: 'false'
        description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
        nsd-ref: slice_hackfest_nsd

    netslice-vld:
    -   id: slice_cirros_vld_mgmt
        name: slice_cirros_vld_mgmt
        type: ELAN
        mgmt-network: 'true'
        nss-connection-point-ref:
        -   nss-ref: slice_hackfest_nsd_2
            nsd-connection-point-ref: nsd_cp_mgmt
        -   nss-ref: slice_hackfest_nsd_3
            nsd-connection-point-ref: nsd_cp_mgmt
    -   id: slice_cirros_vld_data2
        name: slice_cirros_vld_data2
        type: ELAN
        nss-connection-point-ref:
        -   nss-ref: slice_hackfest_nsd_2
            nsd-connection-point-ref: nsd_cp_data_west
        -   nss-ref: slice_hackfest_nsd_3
            nsd-connection-point-ref: nsd_cp_data
```