D3.2: NECOS System Architecture and Platform Specification. V2

Deliverable

Document ID	NECOS-D3.2
Status	Final
Version	4.11
Editors(s)	Stuart Clayman (UCL) and Alex Galis (UCL)
Due	30/04/2019
Submitted	30/04/2019

Abstract

This deliverable presents a final version of the NECOS System Architecture and the Platform Specification. After an analysis of slicing concepts in general, it provides an overview of the NECOS architecture, a more detailed functional description and the essential architectural artefacts. This is followed by a section which presents the interactions between the functional blocks within the NECOS architecture. As concluding remarks, the main envisaged remaining challenges in cloud network slicing together with a positioning of NECOS architecture within the slicing characteristics are presented.

EUB-01-2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CONTRIBUTORS

Contributor	Institution
Stuart Clayman	University College London (UCL)
Francesco Tusa	University College London (UCL)
Alex Galis	University College London (UCL)
Christian Esteve Rothenberg	University of Campinas (UNICAMP)
David Fernandes Cruz Moura	University of Campinas (UNICAMP)
Asma Swapna	University of Campinas (UNICAMP)
Antonio Jorge Gomes Abelém	Federal University of Pará (UFPA)
Billy Anderson Pinheiro	Federal University of Pará (UFPA)
Joan Serrat	Universitat Politècnica de Catalunya (UPC)
Javier Baliosian	Universitat Politècnica de Catalunya (UPC)
Ilias Sakellariou	University of Macedonia (UOM)
Panagiotis Papadimitriou	University of Macedonia (UOM)
Lefteris Mamatas	University of Macedonia (UOM)
Polychronis Valsamas	University of Macedonia (UOM)
Antonis Tsioukas	University of Macedonia (UOM)
Luis M. Contreras	Telefónica Investigación y Desarrollo (TID)
Augusto Neto	Federal University of Rio Grande do Norte (UFRN)
Marcilio Lemos	Federal University of Rio Grande do Norte (UFRN)
Silvio Sampaio	Federal University of Rio Grande do Norte (UFRN)
Marcelo Ribeiro Nascimento	CPqD Telecom Research and Development Center
Douglas Salles Viroel	CPqD Telecom Research and Development Center
Alisson Medeiros	Federal University of Rio Grande do Norte (UFRN)
Sand Luz Correa	Federal University of Goias (UFG)
Leandro Alexandre Freitas	Federal University of Goias (UFG)
Paulo Ditarso Maciel Jr.	Federal University of São Carlos (UFSCar)
Fábio Luciano Verdi	Federal University of São Carlos (UFSCar)
André Beltrami	Federal University of São Carlos (UFSCar)
Rafael Pasquini	Federal University of Uberlândia (UFU)

REVIEWERS

Reviewer	Institution
Augusto Neto	Federal University of Rio Grande do Norte (UFRN)
Joan Serrat	Universitat Politècnica de Catalunya (UPC)
Alex Galis	University College London (UCL)
Stuart Clayman	University College London (UCL)
Christian Esteve Rothenberg	University of Campinas (UNICAMP)

ACRONYMS

API	Application Programming Interface
CDN	Content Distribution Network
DC	Data Center
eMBB	enhanced Mobile BroadBand
GRE	Generis Routing Encapsulation
IMA	Infrastructure and Monitoring Abstraction
YAML	YAML Ain't Markup Language
KPI	Key Performance Indicator
LSDC	Lightweight Software Defined Cloud
MEC	Multi-Access Edge Computing
MPLS	Multiprotocol Label Switching
mMTC	massive Machine Type Communication
NFV	Network Functions Virtualization
QoS	Quality of Service
OTT	Over The Top
PoC	Proof of Concept
PDT	Partially Defined Template
SDN	Software Defined Networking
SLA	Service Level Agreement
SLO	Service Level Objective
SRO	Slice Resource Orchestrator
URLLC	Ultra Reliable and Low Latency Communication
VDU	Visual Display Unit
VNFM	Virtual Network Function Manager
VM	Virtual Machine
VNF	Virtual Network Function
VIM	Virtual Infrastructure Manager
WAN	Wide-Area Network
WIM	Wide-area network Infrastructure Manager

Executive Summary

In NECOS a Cloud Network Slice is a set of infrastructures (network, cloud, data centre) components/network functions, infrastructure resources (i.e., managed connectivity, compute, and storage resources) and service functions that have attributes specifically designed to meet the needs of an industry vertical or a service. As such a Cloud Network Slice is a managed group of subsets of resources, network functions/network virtual functions at the data, control, management/orchestration, and service planes at any given time. The behaviour of the Cloud Network Slice is realized via network slice instances (i.e., activated slices, dynamically and non-disruptively re-provisioned).

The NECOS architecture is designed to provide slices as a service. Each of these slices is a set of computational and networking resources created with the purpose that customer users (tenants) can run one or several application services on top.

To ensure that the supported services can be deployed appropriately, these slices are orchestrated to fulfil on-demand end-to-end quality requirements, independently each one from the others, and spanning across a shared infrastructure of several administrative domains.

The slicing in NECOS can be performed either at the resource level or the virtual infrastructure management level of each of the administrative domains contributing to a slice. The high-level NECOS architecture can be described in terms of its constituting functions, and from that point of view, four functional domains can be distinguished, namely the Tenant's Domain, the Resource Domain, the Resource Marketplace, and the Slice Provider.

The ***Tenant's Domain*** contains the functionality needed by the slice users to request the creation/teardown of slices and to order the deployment of services. The Tenant's Domain interacts with the Slice Provider through an API named Client-to-Cloud Interface. The way by means of which the interaction between these two functional domains is produced depends on the tenant's preferences, and it can range from the detailed specification of the required slice resources to the specification of the service attributes, leaving, in that case, the Slice Provider the responsibility to create the appropriate slice.

The ***Resource Domains*** contain the physical and virtual resources that are parts of a given slice. Each Resource Domain may rely on infrastructures of edge data centres, central data centres, wide area networks or any combination of the above. Whatever the type of resources it may have, an entity wishing to participate in NECOS must:

(i) exhibit its own capabilities for the on-demand creation of virtual resources with a given capacity as well as the virtual managers associated with them, and

(ii) the capability to offer the control of those virtual resources to the Marketplace and their management to the Slice Provider by means of well-defined APIs. In particular, the Resources Domain interacts with the Resource Marketplace through the Slice Marketplace Interface. In addition, the Resources Domain interacts with the Slice Provider through four types of interfaces, namely the Slice Instantiation Interface, the Slice Runtime Interface, the VIM/WIM Control Interface and the VIM/WIM Monitoring Interface.

The ***Resource Marketplace*** is the entity that will facilitate the Resource Domains to offer their resources to participate in slices and the Slice Providers to discover such resource offers. Three types of marketplaces have been foreseen in NECOS. With no particular order of preference, the first one is between telco providers, the second is within pre-established trusted parties, and the third one is a public one, open to everybody. A given Resources Domain can participate in any of the above marketplace types with an appropriate resource filtered offer. The Slice Broker stands as the key element in the Resource Marketplace. The Brokerage can be done either in push-mode, where the Slice Broker contact the Resource Domains to get specific resources or in pull-mode, where the Slice Broker searches for resources in a pre-established resource catalogue. The Resource Marketplace

interacts with the Resource Domains through the Slice Marketplace Interface. In addition, the Resource Marketplace interacts with the Slice Provider through the Slice Request Interface.

The ***Slice Provider*** is the core of the architecture as it constitutes the Lightweight Software Defined Cloud (LSDC) platform. The functionality of the LSDC is meant to receive tenant's requests for the creation of end-to-end slices, take care of the whole slice lifecycle (creation, runtime monitoring and control, and decommission) and facilitate the deployment of the service components (i.e., virtual machines) on top of the slice infrastructure. The LSDC may receive slice creation requests in different granularity and formats. For that reason, it has to process such slice requests to make a common slice specification. Once this has been achieved, the LSDC has to interact with the Resource Marketplace Domain to get Slice Parts from different Resource Domains. These Slice Parts have to be then assembled to create the end-to-end slice. A model of the particular topology of each slice is kept in a database for the whole lifetime of the slice. The LSDC will have to track the behaviour of the slice in terms of a set of KPIs through a monitoring module and trigger a vertical/horizontal escalation when needed. Once the slice is already operational, the LSDC will be able to receive the request from the Tenant's Domain to deploy the service components. These requests can also be of different formats, and for that reason, the LSDC will have to include appropriate adaptors. The coordination of all the above described functions is conducted in the framework of the slice orchestration function.

1. Introduction and Context

Currently, evolutionary alternatives for network operators and service providers such as the Telco Cloud and the Multi-Access Edge Computing (MEC) have emerged to extend the capillarity of the existing centralized data centres, enabling new environments of geographically spread cloud capabilities. The availability of computation, storage, and network resources as well as the kind of workloads and services to be supported differs from the traditional ones leveraging on the large centralized Data Centres (DCs). This fact imposes the need for new mechanisms to manage and control the computing infrastructure for this new kind of demands.

The concept of multi-tenancy is key to the evolution of the networks and cloud environments, enabling novel network slicing ideas on top of the existing telecom networks. The paradigms of network virtualization, mainly based on the Network Functions Virtualization (NFV) [RFC8172] approach, and network programmability through Software Defined Networking (SDN) [RFC7149], have tremendously fostered this evolutionary view, appearing as tools leveraging the implementation of slicing. However, these two paradigms are not sufficient for network slicing. Especially for the former, there is a prevalent NFV-centric view in the state-of-the-art related research projects and in the industry and which does not leverage a native integration of cloud computing and advanced networking. Overloading the NFV orchestration and management artefacts with slicing mechanisms without taking into account the cloud and the network perspective could lead to inefficient partitioning. This problem tends to become further exacerbated as new telecom services (5G services) are expected to be provisioned over multiple-technologies and spanning across multiple operators.

In light to help fulfilling the requirement of making such an innovation for 5G services more reliable, faster, and straightforward, slicing is associated with the partitioning of resources, and featuring capabilities to create and redefine these partitions as needed. A slice is a grouping of physical or virtual (network, compute, storage) resources, which can act as a seemingly independent sub-cloud, sub-network and can accommodate service components.

The NECOS project aims to address this innovation in a lightweight manner by means of a service and infrastructure management platform, called the Lightweight Software Defined Cloud (LSDC) platform [NC1] [NC2]. The LSDC platform exhibits the following three properties:

(i) provisioning of service-agnostic slices but adapting the slices to the desired service characteristics;

(ii) automating the process of slice configuration in multi-cloud environments; and

(iii) providing a uniform management with a high-level of autonomicity.

The first property (i), will be achieved by developing adaptive orchestrators in charge of ensuring that the attributes prescribed to the network are also propagated into the (possibly heterogeneous) data centres that host the services. Tools such as Cloudify [IN1], Terraform [IN2] and Scalr [IN3] are currently used to orchestrate multi-cloud environments However, they lack essential features such as the creation of separate cloud slices to be used in isolation by different customers (tenants) to provide effective service elasticity. Similarly, research projects such as 5G Exchange (5GEx) [IN4] and UNIFY [IN5] offer capabilities to orchestrate network services over multiple administrative domains (multi-operator) so as provisioning network slices, but they are too NFV-centric.

NECOS will achieve the second property (ii) by designing a mechanism that partitions all the underlying distributed infrastructure into constituent slice parts, combines these parts into a full cloud slice, and sets network slices for highly-isolated networking service perspective, thus provisioning end-to-end cloud network slices.

Finally, the third property (iii) will be achieved by extending existing management functionalities to: (a) discover suitable slice parts from a set of participating resource domains; (b) deploy virtual

elements over the end-to-end cloud network slice; and (c) monitor the resources and services inside the end-to-end cloud network slice.

The NECOS LSDC has the following important factors:

1. It empowers a new service model – the Slice as a Service, by dynamically mapping service components to a slice. The enhanced management for the infrastructure creates slices on demand and slice management takes over the control of all the service components, virtualized network functions and system programmability functions assigned to the slice, and (re)configure them as appropriate to provide the end-to-end service.
2. It enables easy reconfiguration and adaptation of logical resources in a cloud networking infrastructure, to better accommodate the QoS demand of the Slice, through using software that can describe and manage various aspects that comprise the cloud environment.
3. It allows each aspect of the cloud environment – from the networking between virtual machines to the SLAs of the hosted applications – to be managed via software. This reduces the complexity related to configuring and operating the infrastructure, which in turn eases the management of the cloud infrastructure.
4. The LSDC platform will offer the ability to a specific cloud provider to federate his own infrastructure with other cloud providers featuring different configurations in order to realize virtualized services through the use of the Slice as a Service concept. The users of the LSDC APIs and platform will be able to create virtual services that can span the merged cloud infrastructure offered by different cloud providers. This concept is not purely technical, and it can also encompass business, cultural, geographical or in any other domain.

NECOS is a research project. Being a research activity means that there is not a pre-established path to follow to reach the final target. New paradigms or algorithms have to be pursued. Uncertainty is then consubstantial. The methodology relies on the Scientific Method, which starts identifying an observation domain where a given problem is identified, and the subsequent hypotheses are established to solve it. Those hypotheses are then validated or reformulated again through an appropriate campaign of experiments. At the same time, NECOS is an engineering project, more specifically a software engineering project. Therefore, it makes sense to establish a set of requirements that will have to be analysed, specified and validated.

To better understand the type of requirements to be considered, these will be coined in the context of a set of specific service scenarios. We undertake an analysis process to confer the LSDC with the main requirements based on its service agnostic properties. Immediately after the requirements elicitation, we start the design phase that entails the system architecture, its fundamental components, and interfaces. The system architecture is a critical step in the project lifecycle. Flexibility is a must in order to avoid decisions that can jeopardize the project outcome. For that reason, the system architecture will be conceived in two cycles. The first one is the overall architecture that leaves some aspects open to being fixed in the second one, which is taking into account the feedback from the implementation and the initial validation. These two last phases of the engineering process have to be carefully selected to provide enough information within the limitations of the available resources.

1.1 Structure of this document

This document describes the second and final cycle overall architecture of the NECOS platform, its main components, and their functionalities and is structured as follows. After this introduction, Section 2 summarizes the most important terms and concepts and an overview of the different levels at which cloud-networking slicing could be performed. Section 3 presents the NECOS view on slicing definitions and key terms. Section 4 presents the NECOS architecture at its highest level of abstraction. It is constituted by the LSDC, the tenant of NECOS, the marketplace and the resource providers. The functionality of each one of these stakeholders and their mutual high-level interactions are presented. In addition, this section explains how the requirements elicited from several scenarios have been considered. Section 5 describes with more granularity the functionality and interfaces

between the functional blocks (Slicing Orchestrator, Infrastructure & Monitoring Abstraction, Resource Domain and Resource Marketplace) characterizing the NECOS architecture. Section 5 is devoted to present the concept of roles in NECOS and how the functionality of these roles is mapped to the components of the architecture. In particular, this shows how a given entity can participate in the NECOS ecosystem. Section 6 presents the primary architectural artefacts, which define the NECOS approach: Service slice control loops, End to End Slicing, VIM on demand, WIM on demand, Resource Marketplace, Elasticity and Scaling, Infrastructure Monitoring. Section 7 describes the slice lifecycle in terms of the interaction of the functional components of the architecture and focuses on the most common of its workflows. The main part of the document concludes with Section 8, which includes a positioning of NECOS in Slicing Ecosystems and an elicitation of further challenges in fully realising and utilising slice capabilities. Finally, one appendix is provided (i.e. Appendix A) where we analyse the prioritisation of requirements elicited in Deliverable D2.1, which are supported by the NECOS architecture.

1.2 Contribution of this deliverable to the project and relation with other deliverables

Deliverable D3.2 is the evolution of its preceding counterpart D3.1 as foreseen in the workplan. As soon as D3.1 was available, the design and development of the NECOS architecture was started within WP5. In particular, four design and development directions were undertaken: a) to provide the appropriate functionality to create end-to-end slices; b) to conceive the marketplace to choose the necessary slice parts supported by different infrastructure providers; c) to design and implement the elasticity functions that should characterize the slice during its lifecycle; d) and finally, to design and implement the monitoring functionality needed in the corresponding slice lifecycle workflows. These four design and development directions gave us a better understanding of aspects that were left open or simply not considered in the initial architecture proposal. This knowhow has been feed backed into the initial architecture of D3.1 ending up in this document. Therefore D3.2 is a self-contained document describing at the moment of its release the most accurate understanding of the NECOS architecture. This deliverable marks also the end of WP3. Any further update on the architecture that is dictated by the project evolution will be reflected in deliverable D5.2, which is foreseen at the end of the project..

2. Slicing concepts

Slicing is a move towards segmentation of resources and deployment of virtual elements for the purpose of enhanced services and applications on a shared infrastructure. Many groups including ITU, ETSI, IETF and more are currently considering it.

Slices are expected to considerably transform the networking perspective and enhance software-defined architectures (e.g., SDN, NFV, etc.) by:

- Abstracting away the lower level elements, in various ways.
- Isolating connectivity at a sub-network level.
- Separating logical network behaviours from the underlying physical network resources.
- Allowing dynamic management of network resources by managing resource-relevant slice configuration.
- Simplifying and reducing the expenditure of operations.
- Support for rapid service provisioning.
- Support for NFV deployment.

Key characteristics of the Cloud Network Slicing include:

- The *concurrent deployment* of multiple logical, self-contained and independent, shared or partitioned slices on a common infrastructure platform.
- *Dynamic multi-service support, multi-tenancy* and the integration means for vertical market players.
- The *separation of functions*, simplifying the provisioning of services, the manageability of networks, and integration and operational challenges especially for supporting communication services.
- The means for Network /Cloud operators/ ISP and infrastructure owners to reduce operations expenditure, allowing programmability and innovation necessary to enrich the offered services, for providing tailored services, and allowing network programmability to OTT providers and other market players without changing the physical infrastructure.
- *Hosting applications*, offering the capability of hosting virtualized versions of network functions or applications, including the activation of the necessary monitoring information for those functions.
- *Hosting on-demand 3rd parties*/OTTs, empowering partners (3rd parties / OTTs) to directly make offers to the end customers augmenting operator network or other value creation capabilities.

Slicing itself is not new, it has been considered in the past and it is progressively being included in the 5G standards, as discussed below. To clarify how NECOS aims at filling gaps towards the realization of cloud network slicing, in the following we survey the state of the art[1] in standardization and research projects and conclude with a summarizing discussion on which key features of the cloud network slicing are being covered.

2.1 Early Definitions of Slicing

The followings are early definitions of slicing:

[1] For a detailed state-of-the art review of slicing (history, terms & concepts, use cases, SDOs, research projects, remaining challenges, references), interested readers are pointed to [AG2], the Network Slicing Tutorial at IEEE CNSM, Rome, 5-9 November 2018, publicly available at http://cnsm-conf.org/2018/files/CNSM18_SlicingTutorial_AlexGalis_5-10-2018.pdf

i) **Active / Programmable Networks research:** node operating systems & resource control frameworks (1995 -2005) produced the text Programmable Networks for IP Service Deployment [SC1].

ii) **Federated Testbed research:** Planet Lab USA (2002), PlanetLab EU (2005), OneLab EU (2007), PlanetLab Japan (2005), OpenLab EU (2012).

iii) **GENI Slice** (2008-): "GENI [SC5] is a shared network testbed, i.e., multiple experimenters may be running multiple experiments at the same time. A GENI slice is:

- The unit of isolation for experiments.
- A container for resources that are used in an experiment. GENI experimenters add GENI resources (compute resources, network links, etc..) to slices, and run experiments that use these resources.
- A unit of access control. The experimenter that creates a slice can determine which project members have access to the slice, i.e., are members of the slice.

iv) **Slice capabilities** (2009) [SC2]

- 3 Slices Capabilities: "Resource allocation to virtual infrastructures or slices of virtual infrastructure."; "Dynamic creation and management of virtual infrastructures/slices of virtual infrastructure across diverse resources."; "Dynamic mapping and deployment of a service on a virtual infrastructure/slices of virtual infrastructure."
- 17 Orchestration capabilities.
- 19 Self-functionality mechanisms.
- 14 Self-functionality infrastructure capabilities.

v) **Cloud manifest (2009)** as defined in the RESERVOIR federated cloud environment [SC3] - The paper is ranked 17 out of approx. 25,000 papers published in the last 7 years in Cloud Computing as far as citation is concerned [i.e. August 2018 number of citation for [SC3] is 900] as stated in the review paper [SC9]. The cloud manifest specifies the structure of the service application in terms of component types that are to be deployed as virtual elements. The manifest also specifies the grouping of components into virtual networks and tiers that form the service applications.

vi) **ITU-T Slicing (2011)** as defined in [SC6], it is the basic concept of the Network Softwarization. Slicing allows Logically Isolated Network Partitions (LINP) with a slice being considered as a unit of programmable resources such as network, computation, and storage.

vii) **NGMN Slice capabilities (2016) -** consist of 3 layers: 1) Service Instance Layer, 2) Network Slice Instance Layer, and 3) Resource layer.

- The Service Instance Layer represents the services (end-user service or business services), which are to be supported. Each service is represented by a Service Instance. Typically, services can be provided by the network operator or by 3rd parties.
- A Network Slice Instance provides the network characteristics, which are required by a Service Instance. A Network Slice Instance may also be shared across multiple Service Instances provided by the network operator.
- The Network Slice Instance may be composed by none, one or more Sub-network Instances, which may be shared by another Network Slice Instance.

viii) **IETF** (2017) Network Slicing is defined in [AG1] as managed partitions of physical and/or virtual network and computation resources, network physical/virtual and service functions that can act as an independent instance of a connectivity network and/or as a network cloud. Network resources include connectivity, compute, and storage resources. As such Network Slices considerably transform the networking and servicing perspectives by abstracting, isolating,

orchestrating, softwarising, and separating logical network components from the underlying physical network resources and as such they enhance Internet architecture principles.

ix) **GPP TR23.799** Study Item "Network Slicing' 2016.

x) **ONF Recommendation TR-526** "Applying SDN architecture to Network Slicing" 2016.

xi) **EU 5GPPP**

- 15 Large Scale Research projects – all based on Network Slicing (https://5g-ppp.eu) (2015-2018+).
- White Papers on 5G Architecture centred on network slicing (mark 1 - (2016) [SC7]) (mark 2 - (2018) [SC8].

xii) Additional characteristics, standard and research activities on Infrastructure slicing and references are presented in the tutorial [AG2]

2.2 ITU-T Slicing

References:

- ITU-T Y.3011- http://www.itu.int/rec/T-REC-Y.3001-201105-I
- ITU-T IMT-2020 - https://www.itu.int/en/publications/Documents/tsb/2017-IMT2020-deliverables/mobile/index.html#p=49

According to ITU-T Y.3011 (2011) slicing allows Logically Isolated Network Partitions (LINP) with a slice being considered as a unit of programmable resources such as network, computation and storage. LINPs are isolated from each other, and when combined with programmability in virtual resources, users of LINPs can program the virtual resources on above the virtualization layer. In other words, each LINP can provide the corresponding users with services similar to those provided by traditional networks without network virtualization. The users of LINPs are not limited to the users of services or applications but can include service providers. For example, a service provider can lease a LINP and can provide emerging services or technologies such as cloud computing service. The service providers can realize the emerging services as if they own a dedicated physical network. In order to facilitate the deployment of network virtualization, it is necessary to provide control and management procedures such as creating, monitoring, and measuring the status of LINPs.

Figure 1 represents the conceptual architecture of network virtualization, which consists of LINPs over physical resources supporting network virtualization.

A single physical resource can be shared among multiple virtual resources and each LINP consists of multiple virtual resources. Each LINP is managed by individual LINP manager.

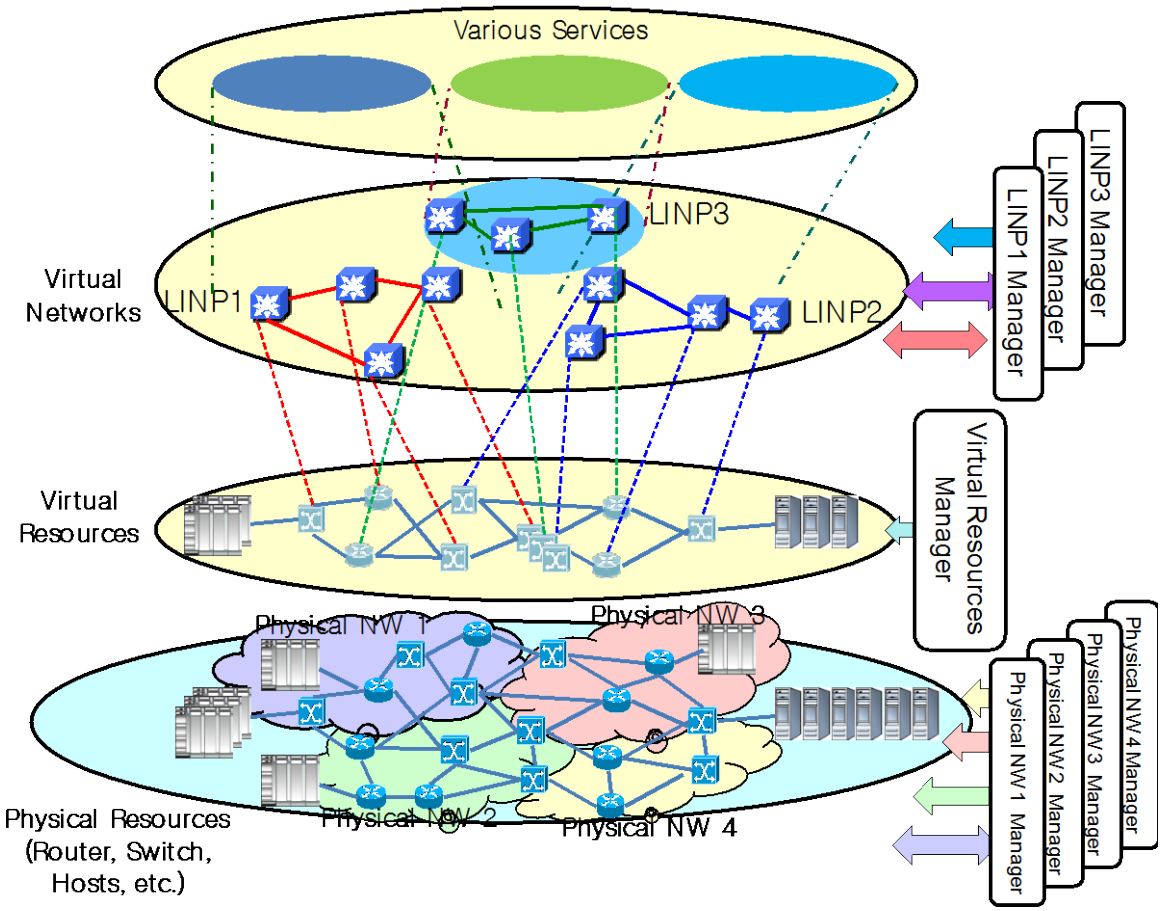


Figure 1: Conceptual architecture of network virtualization

2.3 NGMN Slicing

References:

- White Paper on 5G (2016) - https://www.ngmn.org/5g-white-paper/5g-white-paper.html
- NGMN Alliance "Description of Network Slicing Concept" - https://www.ngmn.org/fileadmin/user_upload/160113_Network_Slicing_v1_0.pdf.

According to NGMN slicing consists of 3 layers (1) Service Instance Layer, (2) Network Slice Instance Layer, and (3) Resource layer:

- The Service Instance Layer represents the services (end-user service or business services) which are to be supported. Each service is represented by a Service Instance. Typically, services can be provided by the network operator or by 3rd parties.
- A Network Slice Instance provides the network characteristics which are required by a Service Instance. A Network Slice Instance may also be shared across multiple Service Instances provided by the network operator.
- The Network Slice Instance may be composed by none, one or more Sub-network Instances, which may be shared by another Network Slice Instance.

The Network Slice Instance may be composed by none, one or more Sub-network Instances, which may be shared by another Network Slice Instance. Similarly, the Sub-network Blueprint is used to create a Sub-network Instance to form a set of Network Functions, which run on the physical/logical

resources.

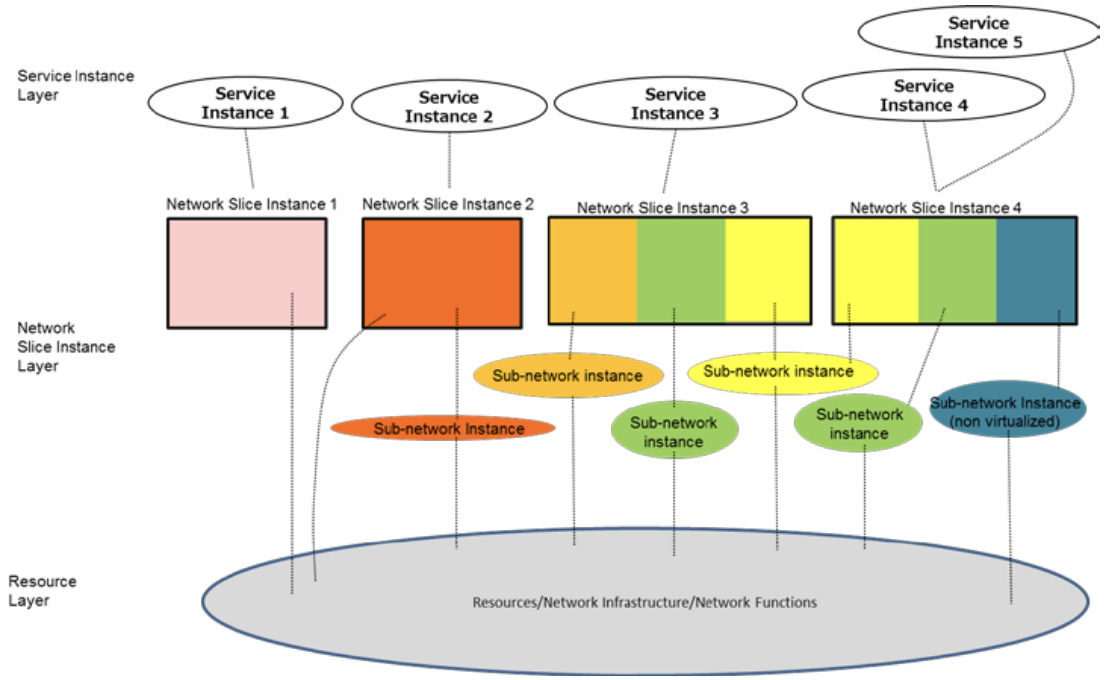


Figure 2: Network slicing conceptual outline

2.4 IETF Slicing

References (including work in progress):

- IETF draft "Network Slicing" Galis., A, Dong., J, Makhijani, K, Bryant, S., Boucadair, M, Martinez-Julia, P. https://tools.ietf.org/html/draft-gdmb-netslices-intro-and-ps-01
- NetSlices Architecture draft-geng-netslices-architecture-02
- General Requirements for Network Slicing: https://www.ietf.org/id/draft-homma-slice-provision-models-00.txt
- Network Slicing - Revised Problem Statement draft-galis-netslices-revised-problem-statement-03
- NetSlices Management Architecture draft-geng-coms-architecture-01
- NetSlices Use Cases draft-netslices-usecases-01
- NetSlices Management Use cases draft-qiang-coms-use-cases-00
- NetSlices Information Model draft-qiang-coms-netslicing-information-model-02
- Autonomic NetSlicing draft-galis-anima-autonomic-slice-networking-04
- NetSlices Interconnections https://tools.ietf.org/html/draft-defoy-coms-subnet-interconnection-03

According to the IETF network slicing architecture the following terms are defined:

- Resource Slice - A grouping of physical or virtual (network, compute, storage) resources. It inherits the characteristics of the resources which are also bound to the capability of the resource. A resource slice could be one of the components of Network Slice, however on its own does not represent a Network Slice fully.
- Network Slice - A Network slice is a managed group of subsets of resources, network functions / network virtual functions at the data, control, management/orchestration planes

and services at a given time. Network slice is programmable and has the ability to expose its capabilities. The behaviour of the network slice is realized via network slice instance(s).

- End-to-end Network Slice - A cross-domain network slice which may consist of access network (fixed or cellular), transport network, (mobile) core network and etc. End-to-end network slice can be customized according to the requirements of network slice tenants
- Network Slice Instance - An activated network slice. It is created based on a network template. A set of managed run-time network functions, and resources to run these network functions, forming a complete instantiated logical network to meet specific network characteristics required by the service instance(s). It provides the network characteristics that are required by a service instance. A network slice instance may also be shared across multiple service instances provided by the network operator.

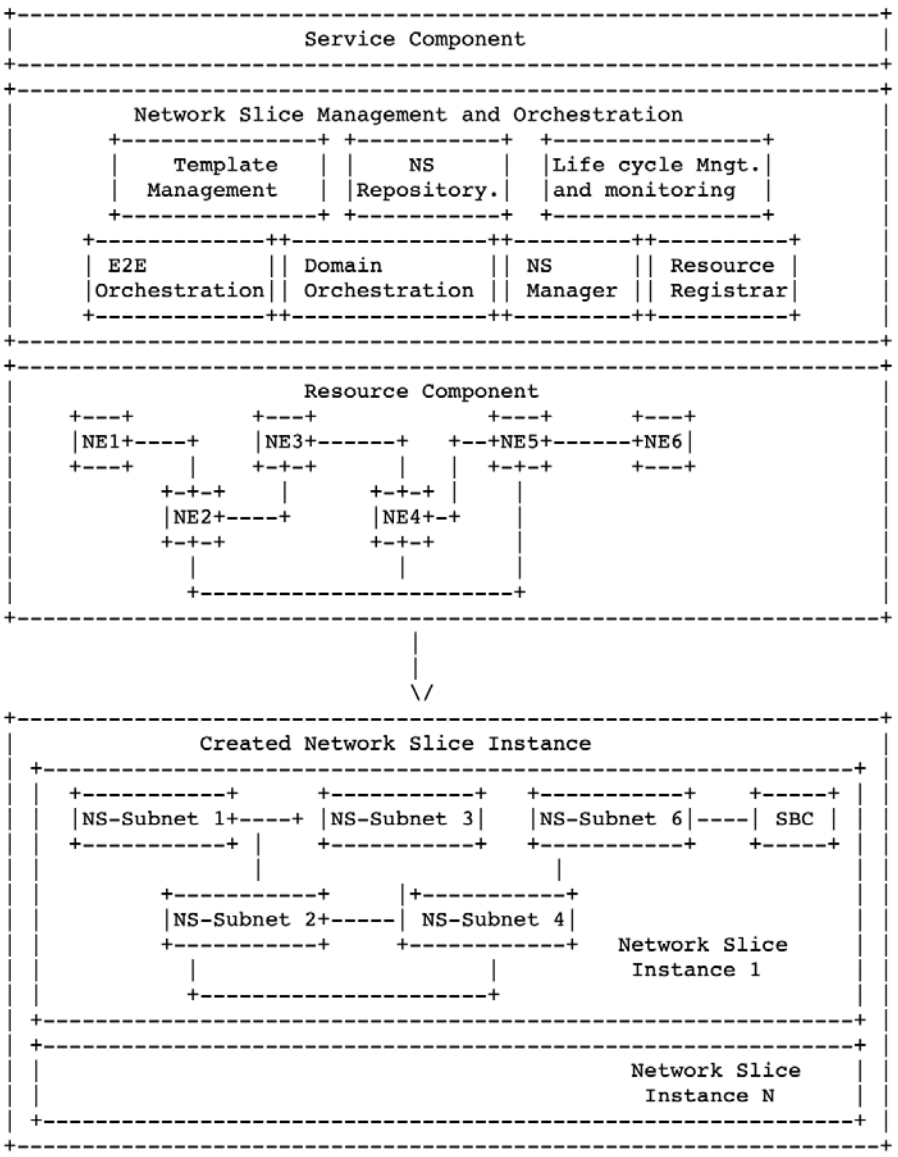


Figure 3: IETF Network slicing architecture

2.5 3GPP Slicing

References: 3GPP: www.3gpp.org/ Published & Work In progress on slicing includes:

- 3GPP SA2 - Study on Architecture for Next Generation System /Network slice related functionality (3GPP TR 23.799)
- 3GPP SA2 - System Architecture for the 5G System /Network slice related functionality (3GPP TS 23.501)
- 3GPP SA2 - Procedures for the 5G System: Procedures and flows of the architectural elements/ Network slice related procedures (3GPP TS 23.502)
- 3GPP SA3 - Study on the security aspects of the next generation system/ Network slice related security (3GPP TR 33.899)
- 3GPP SA5 - Study on management and orchestration of network slicing/ Network slice management (3GPP TR 28.801)
- 3GPP SA5 - Provisioning of network slicing for 5G networks and services: Detailed specification of network slice provisioning/ Network slice management (3GPP TS 28.531)
- 3GPP SA5 - Management of network slicing in mobile networks - concepts, use cases and requirements (3GPPTS 28.530)
- 3GPP TR 28.801 - Study on management and orchestration of network slicing for next-generation network

According to 3GPP TR 28.801 the network slice concept includes the following aspects:

1. Completeness of an NSI (network slice instance): An NSI is complete in the sense that it includes all functionalities and resources necessary to support a certain set of communication services thus serving a certain business purpose.
2. Components of an NSI:
3. The NSI contains NFs (e.g. belonging to AN and CN). If the NFs are interconnected, the 3GPP management system contains the information relevant to the connections between these NFs such as topology of connections, individual link requirements (e.g. QoS attributes), etc. For the part of the TN (Transport Network) supporting connectivity between the NFs, the 3GPP management system provides link requirements (e.g. topology, QoS attributes) to the management system that handles the part of the TN supporting connectivity between the NFs.
4. Resources used by the NSI: The NSI is realized via the required physical and logical resources.
5. Network Slice Template: The network slice is described by a Network Slice Template (NST). The NSI is created using the NST and instance-specific information.
6. NSI policies and configurations:
7. Instance-specific policies and configurations are required when creating an NSI Network characteristics examples are ultra-low-latency, ultra-reliability, etc. NSI contains a Core Network part and an Access Network part.
8. Isolation of NSIs: A NSI may be fully or partly, logically and/or physically, isolated from another NSI

The network slice stakeholders involved in 3GPP network slice management are depicted in the following figure:

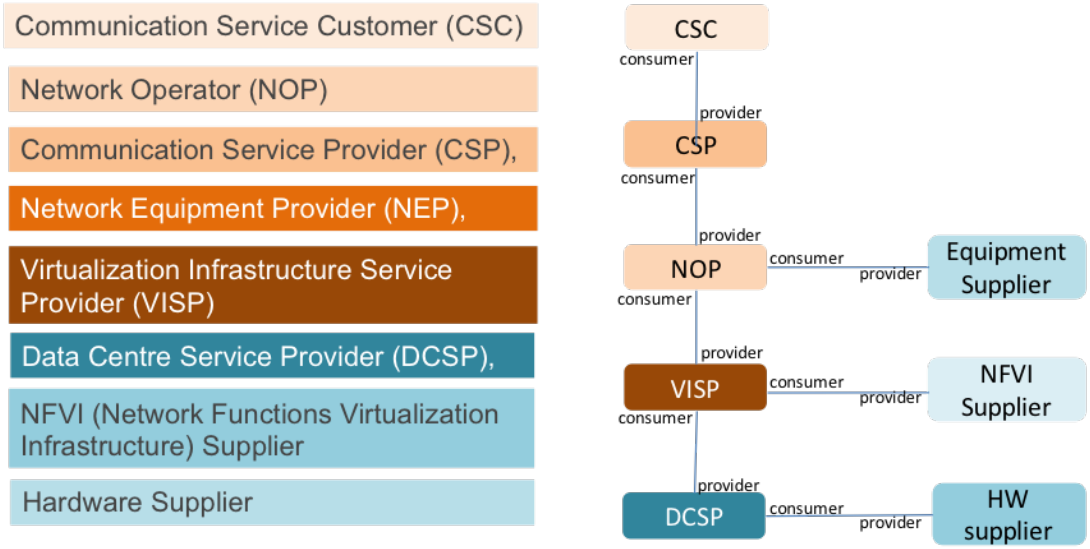


Figure 4: 3GPP network slice stakeholders

There is also a collaboration between IETF and 3GPP, where it has been suggested that the best approach is to ensure that the 3GPP engineers are involved in the IETF work they are interested in, and that 3GPP states clearly what their requirements (rather than solutions) are. IETF also noted that the work in 3GPP is ongoing.

2.6 ETSI Slicing

References:

- Network Operator Perspectives on NFV priorities for 5G- https://portal.etsi.org/NFV/NFV_White_Paper_5G.pdf
- Report on Net Slicing Support with ETSI NFV Architecture Framework http://www.etsi.org/deliver/etsi_gr/NFV-EVE/001_099/012/03.01.01_60/gr_NFV-EVE012v030101p.pdf
- "E2E Network Slicing Reference Framework and Information Model" - Kiran, Makhijani- Huawei Technologies, Kevin Smith-Vodafone John Grant - Ninetiles Alex Galis- UCL, Xavier Defoy- Interdigital - approved & published in October 2018 by ETSI as a standard specification document- https://www.etsi.org/deliver/etsi_gr/NGP/001_099/011/01.01.01_60/gr_ngp011v010101p.pdf

ETSI's E2E Network Slicing identifies a next-gen network slicing (NGNS) framework defined here is a generalized architecture that would allow different network service providers to coordinate and concurrently operate different services as active network slices, including slicing design principle (service-oriented approach, slice abstraction, slice reusability, slice autonomy), an information model, network slice functions specification and slice enablement.

The NS methods are aimed at providing custom design of networks suitable for a specific use case (vertical market). Such methods need to be able to translate a service requirement into normalized description of resources across different type of network domains based on NGMN's description of network slicing. The 3-layer approach is as follows:

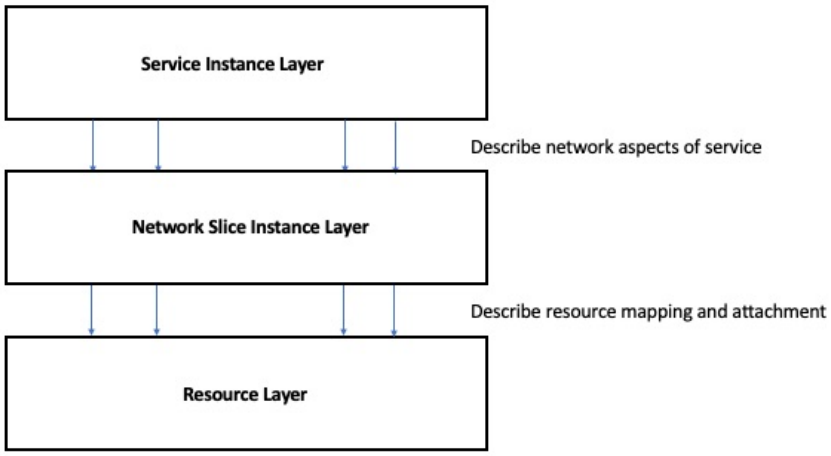


Figure 5: 3-layer network slice and service concept

According to ETSI NGP there are three key areas of consideration for the NS architecture:

1. Service description (corresponds to service instance layer): A sketch of services instantiated, independent of any technology or underlying control plane.
2. Network slice to abstract resource mapping (corresponds to network slice instance layer).
3. Resource allocation (across different networks).

2.7 ONF Slicing

Reference: TR-526 Applying_SDN_Architecture_to_5G_Slicing_TR-526.pdf

The ONF Slicing adopts the NGMN terminology of network slicing.

Applying slicing to the SDN architecture, the client context provides the complete abstract set of resources (as Resource Group) and supporting control logic that constitutes a slice, including the complete collection of related client service attributes. As such, a 5G slice is comparable to, if not the same as, an SDN client context, isolated by the controller's virtualization and client policy functions and continuously optimized by the orchestration and global policy functions. Going a step further, a 5G controller is an SDN controller, or vice versa. Both 5G and SDN controller manage-control combinations of all relevant network resources / functions / assets required to serve client- specific purposes. The client context also offers to the client functions to manage-control the slice resources, including OAM related-functions, as visible by / available to the client according to administrative policy.

Recursion in the SDN architecture allows for a high-level client context to be virtualized and orchestrated over a number of lower-level resource groups, spanning geographic, business and technology boundaries as needed. Each lower-level resource group may itself be exposed by the client context of a lower-level SDN controller. The SDN architecture thereby naturally supports a recursive composition of slices.

The SDN controller, at the centre of a feedback loop and acting autonomously according to administratively configured policies, enables dynamic allocation, modification and optimization of resource usage. The resources provided to the controller's clients are virtualized (abstracted views and services) from the underlying resources and presented to slices through resource groups.

Figure 6 shows a mapping of the SDN architecture, terminology and abstractions to 5G slicing.

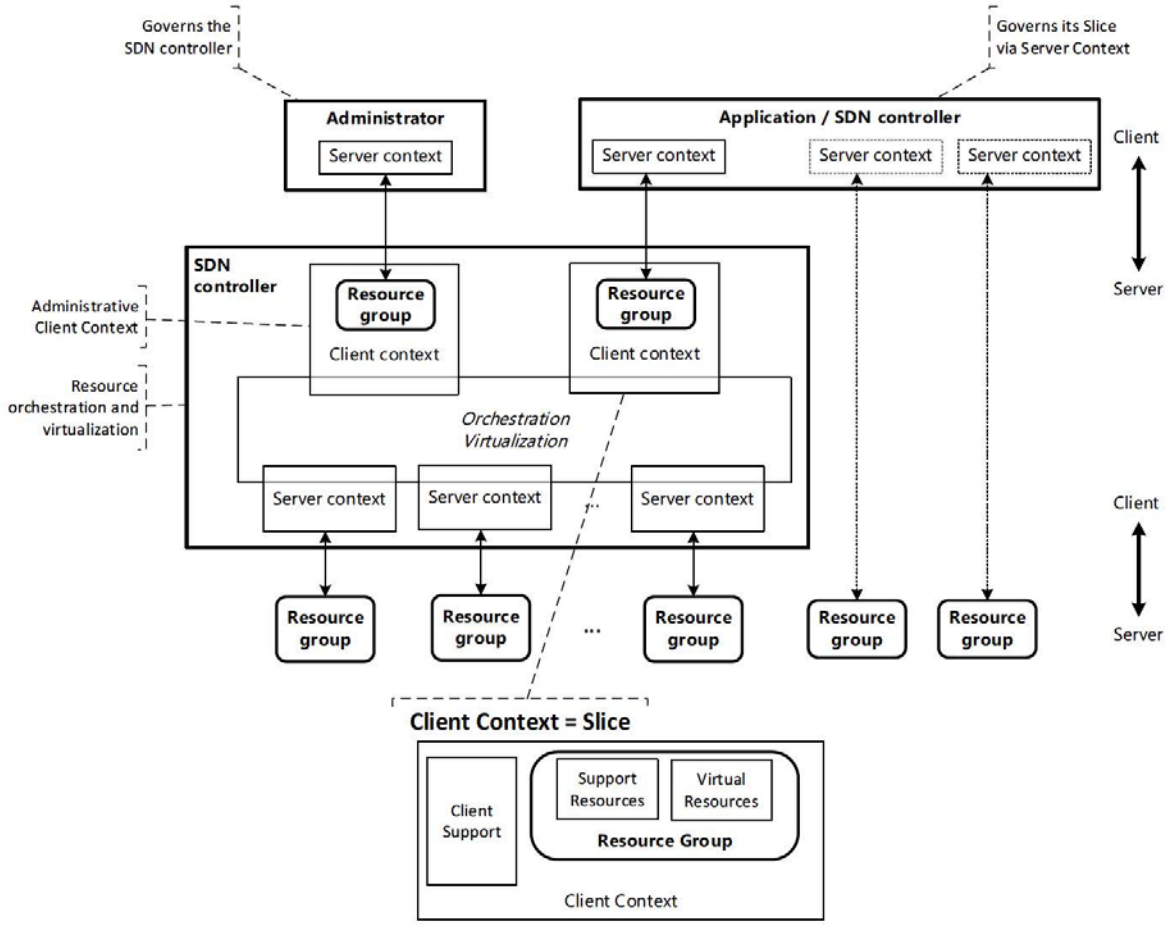


Figure 6: ONF slicing abstractions

2.8 Cloud partitioning

Reference: Rochwerger, J. Caceres, R. Montero, D. Breitgand, A. Galis, E. Levy, I. Llorente, K. Nagin, Y. Wolfsthal "The RESERVOIR Model and Architecture for Open Federated Cloud Computing", the IBM System Journal Special Edition on Internet Scale Data Centres, vol. 53, no. 4, 2009, http://dl.acm.org/citation.cfm?id=1850663; The paper is ranked first out of approx. 25,000 papers published in the last 7 years in Cloud Computing as far as architectural citation is concerned as stated in the review paper 'A Scientometric Analysis of Cloud Computing Literature'- Heilig, L., Voss, S. - IEEE Transactions on Cloud Computing, Volume: PP, Issue: 99, 30 April 2014, ISSN: 2168-7161; DOI: 10.1109/TCC.2014.2321168

Different models are being developed for federated cloud computing. One such model - Reservoir model (see above reference), provides a clear separation between the functional roles of service providers and infrastructure providers. Service providers are entities that understand the needs of a particular business and offer service applications to address those needs. Service providers do not own the computational resources needed by these service applications; instead, they lease resources from infrastructure providers that provide them with a seemingly infinite pool of computational, network, and storage resources.

Infrastructure providers operate cloud sites that own and manage the physical infrastructure on which service applications execute. The federation of collaborating sites forms a composite cloud. To optimize resource utilization, the computational resources within a site are partitioned by a virtualization layer into virtual execution environments (VEEs). VEEs are fully isolated runtime environments that abstract the physical characteristics of the resource and enable sharing. The

virtualized computational resources along with the virtualization layer and all of the management enablement components are referred to collectively as the VEE host.

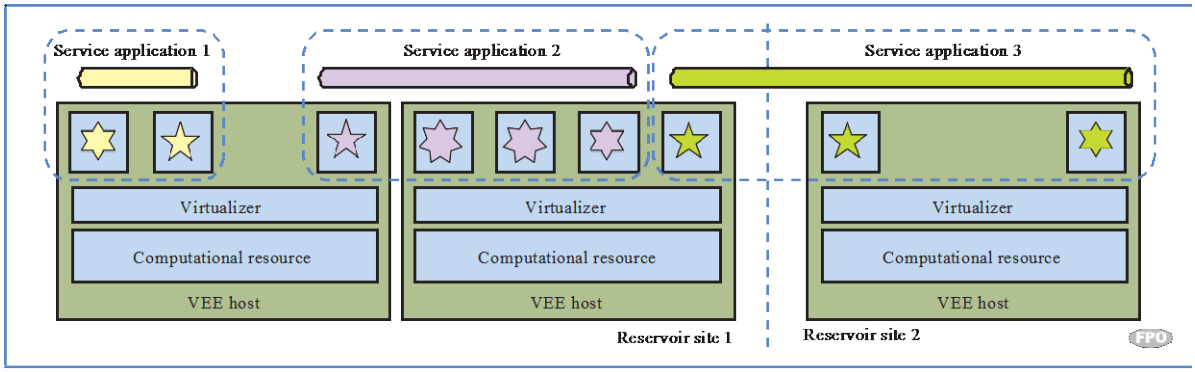


Figure 7: Partitioning Clouds by a virtualization layer into virtual execution environments

Service applications are executed by a set of VEEs (represented by squares) distributed across the VEE hosts in a Reservoir cloud. VEEs for a particular service application may all be collocated in the same VEEH (as in service application 1), they may be spread across VEEHs within the same site (as in service application 2), or they may be spread across sites (as in service application 3).

A service application is a set of software components that works collectively to achieve a common goal. Each component of such service applications executes in a dedicated VEE. The VEEs are placed on the same or different VEE hosts within the site or on different sites

(Figure 7). A service application is deployed on the composite cloud using the contract and SLA between the service provider and the infrastructure provider – the service manifest.

2.9 EU 5GPPP Projects

In this section, we present several 5GPPP projects which were selected as they have had design and implementation facets that are focussed on slicing. These include: 5GEx; 5G SONATA; 5G NORMA; 5G-TRANSFORMER; 5G-PAGODA; 5G-SLICENET and 5G-MoArch project.

2.9.1 5GEX EU Project

Reference: http://www.5gex.eu/

5GEx focused on the transition from dedicated physical networks with dedicated control and dedicated services and resources for different applications to a “network factory” where resources and network functions are traded and provisioned – a new infrastructure and services “manufactured by SW” as depicted in Figure 8.

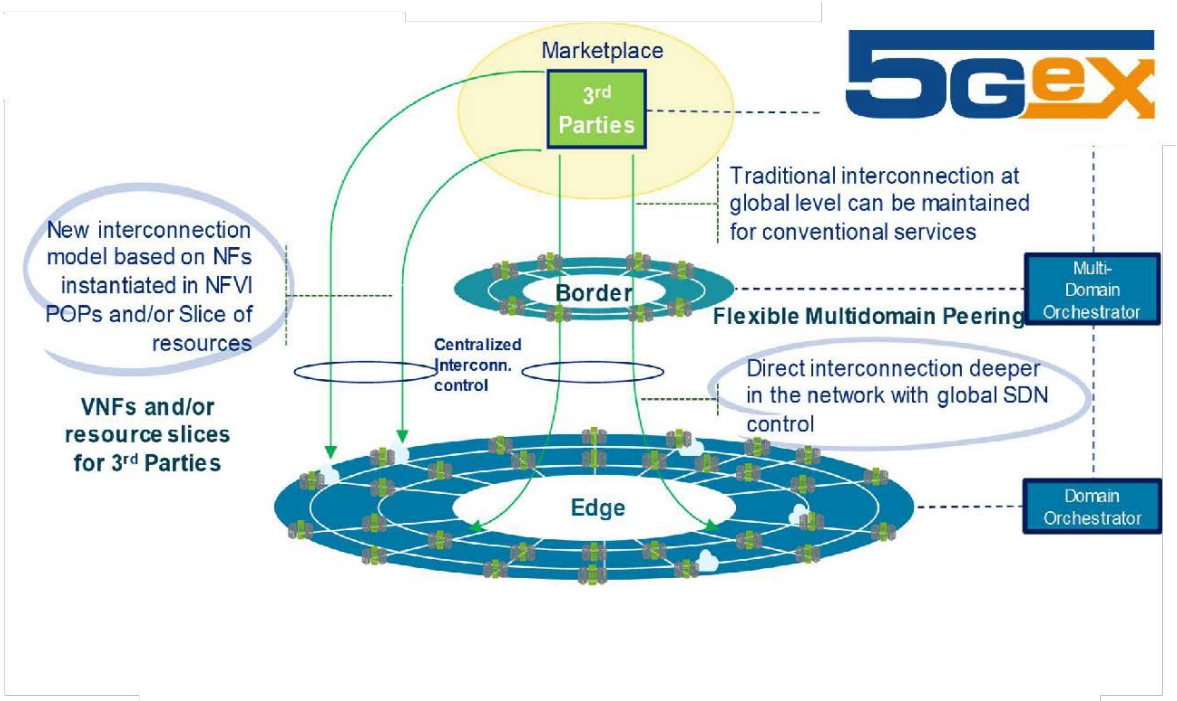


Figure 8: 5GEX network softwarization process - Infrastructure and services manufactured by SW

In 5GEx a Slicer management function is design with the view to enrich the Resource Orchestrator functionality with multi-tenancy resource slicing and generic lifecycle VNF management functionality. Additionally, the Slicer function coordinates and integrates multi-vendor / 3rd party specific lifecycle VNF management functions.

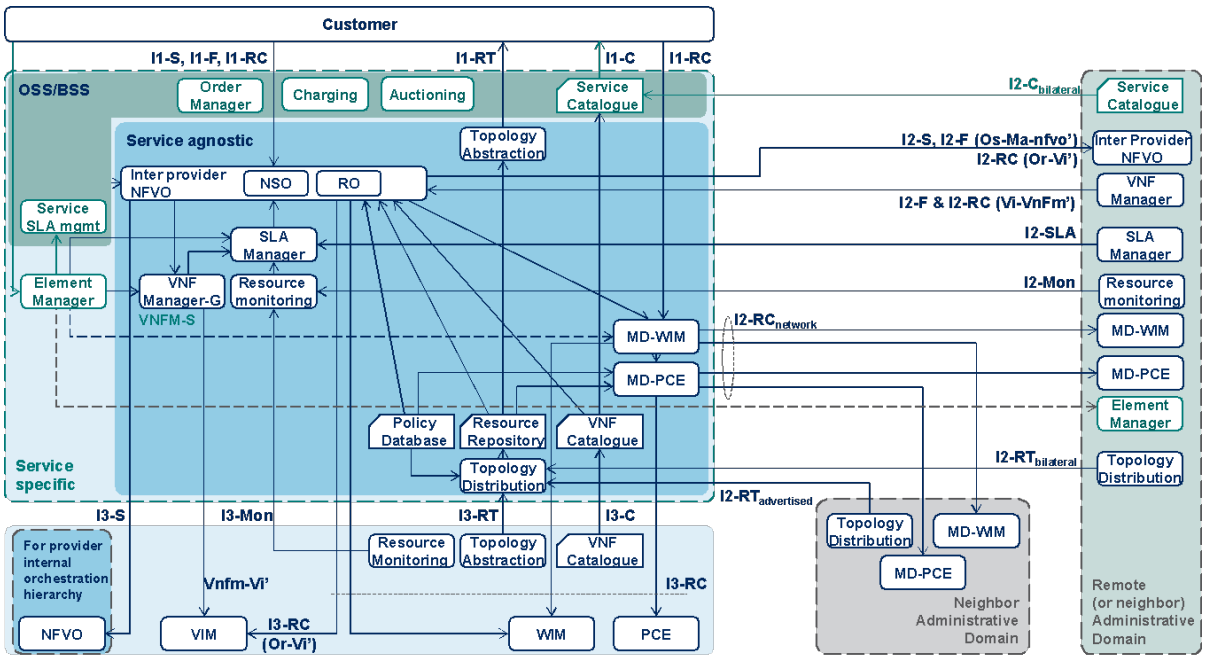


Figure 9: Functional model of 5GEX multi-domain orchestration

2.9.2 5G-SONATA EU Project

Reference: http://sonata-nfv.eu

5G SONATA focussed on flexible programmability of software networks and the optimization of their deployments.

As far as SONATA is concern a slice is an aggregated set of resources that can be used in the context of an end-to-end networked service comprised of virtual network functions. Slices are composed of multiple resources which are isolated from other slices and allows logically isolated network partitions, with a slice being considered as the basic unit of programmability using network, computation and storage.

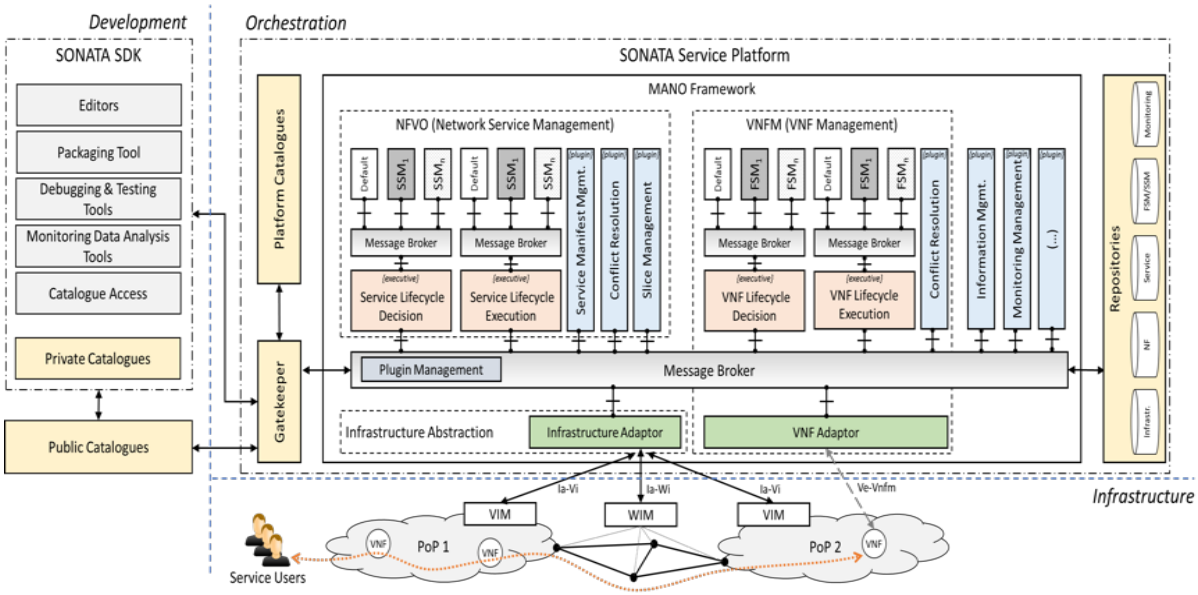


Figure 10: SONATA high level architecture

2.9.3 5G-NORMA EU Project

Reference: http://www.it.uc3m.es/wnl/5gnorma/

The 5G-NORMA focus is on: i) Adaptive (de)composition and allocation of NFs Joint; ii) Optimization of RAN and Core Network and iii) SW-defined Mobile Control.

This involved the design for Service management; Mapping of customer-facing services and procedures to resource-facing services and procedures; Network slicing (Service and Resource); Orchestration; Inter-slice and intra-slice and Network programmability as depicted in Figure 11.

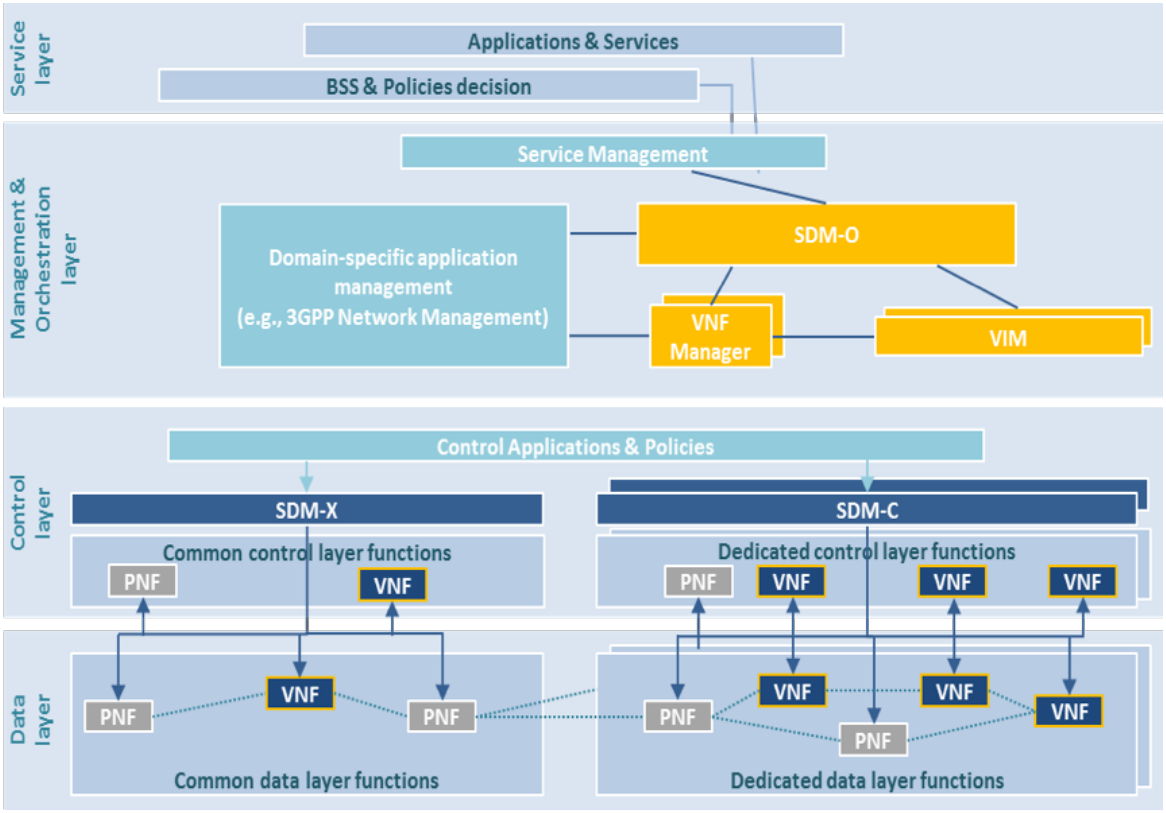


Figure 11: 5G NORMA functional architecture

2.9.4 5G-TRANSFORMER EU Project

Reference: http://5g-transformer.eu

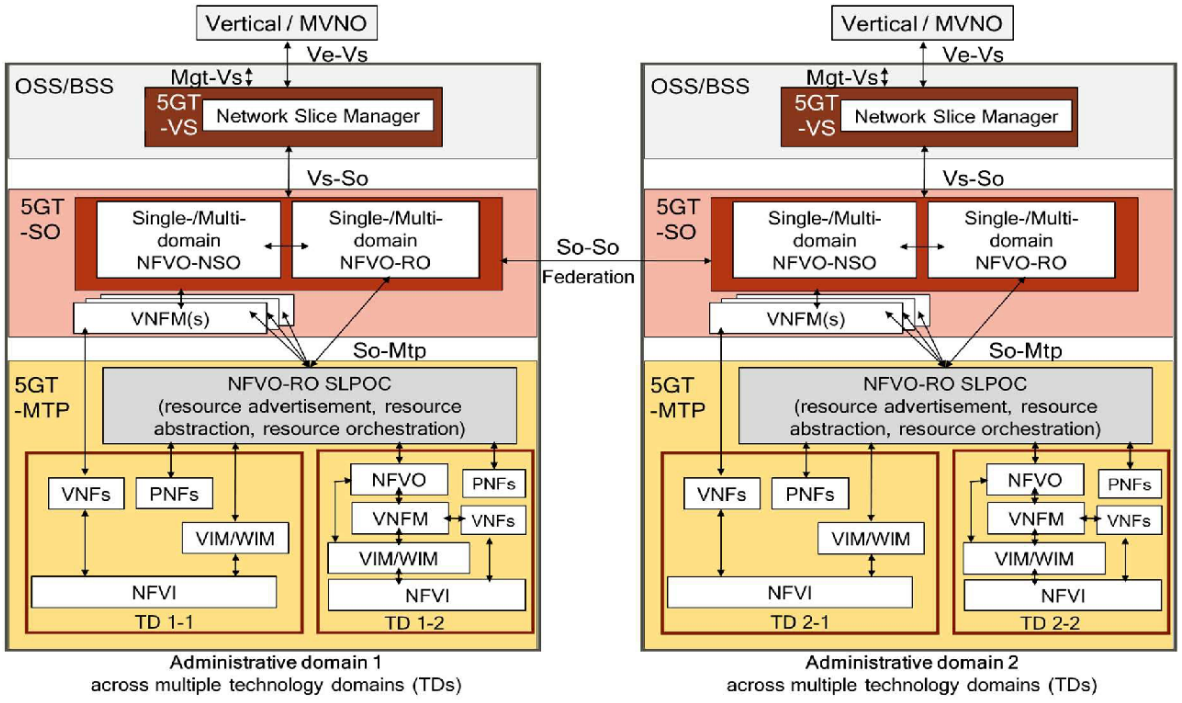


Figure 12: 5G Transformer slice architecture

5G-Transformer focuses on (i) Vertical Slicer as the logical entry point for verticals to support the creation of their respective transport slices; (ii) Service Orchestrator to orchestrate the federation of transport networking and computing resources from multiple domains and manage their allocation to slices, and (iii) Mobile Transport and Computing Platform or integrated fronthaul and backhaul networks as depicted in Figure 12.

2.9.5 5G-PAGODA EU Project

Reference: https://5g-pagoda.aalto.fi
5G PAGODA proposal is based on: (i) scalable 5G slicing architecture; (ii) extending the current NFV architecture towards support of different specialized network slices composed of multi-vendor virtualized network functions scalable 5G slicing architecture, and (iii) extending the current NFV architecture towards supporting different specialized network slices composed of multi-vendor virtualized network functions as depicted in Figure 13.

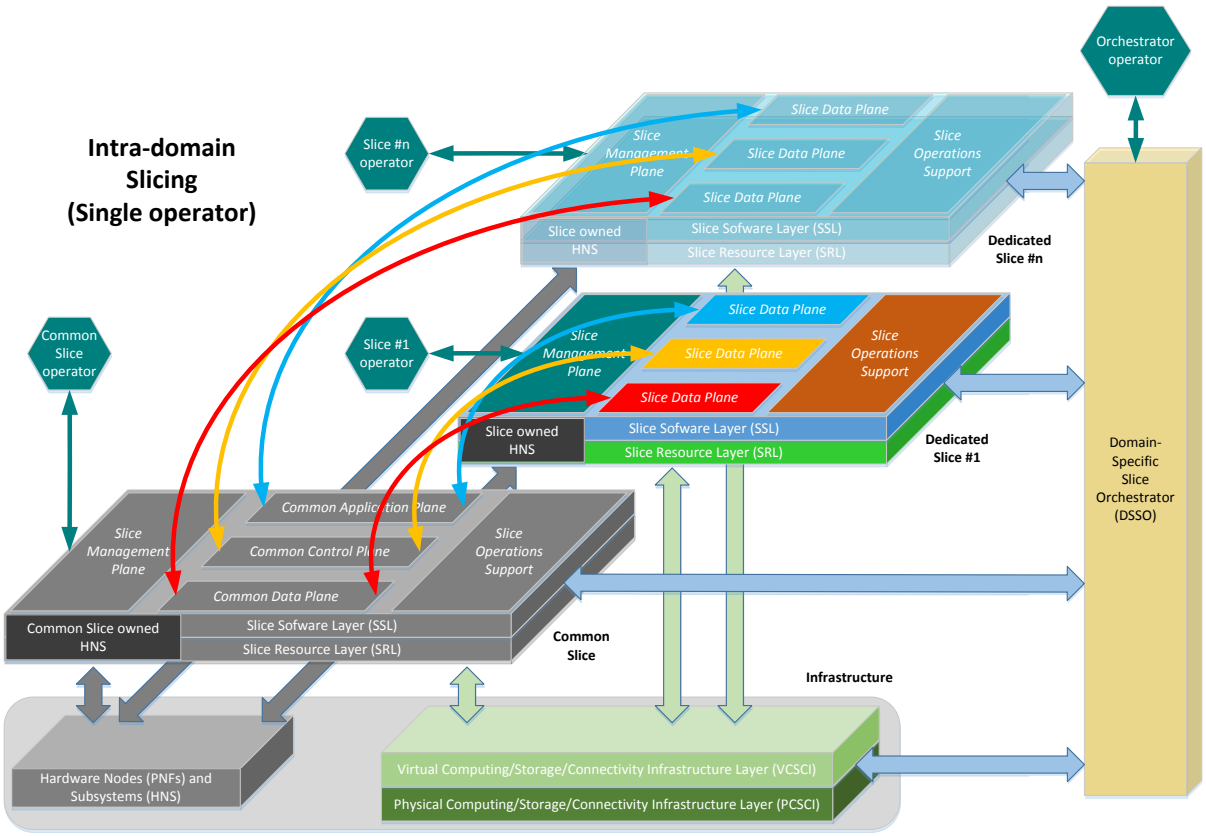


Figure 13: 5G PAGODA intra-domain slicing architecture

2.9.6 5G-SLICENET EU Project

Reference: https://slicenet.eu
5G-SLICENET is focusing on: (i) End-to-End Cognitive Network Slicing and (ii) Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks as depicted in Figure 14.

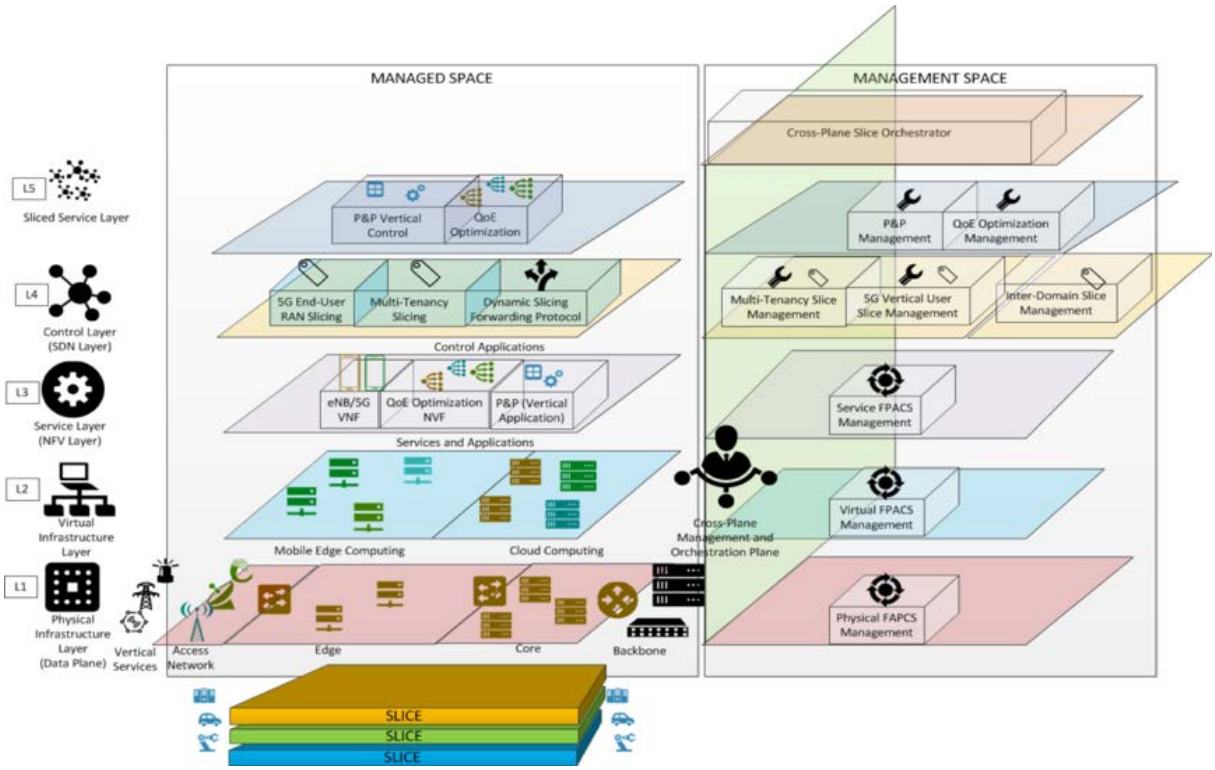


Figure 14: 5G SLICENET high level slicing architecture

2.9.7 5G-MoArch EU Project

Reference: 5G Mobile Network Architecture - https://5g-monarch.eu

Slice-aware elasticity refers to the ability to serve multiple slices over the same physical resources while optimising the allocation of computational resources to each slice based on its requirements and demands. The algorithms/policies for slice aware elasticity reside at the E2E service M&O Layer and more precisely at the Cross-Slice M&O entity, which, triggers the above-mentioned resource optimisation via the 3GPP Network Management or the NFV MANO entities, as depicted in Figure 15.

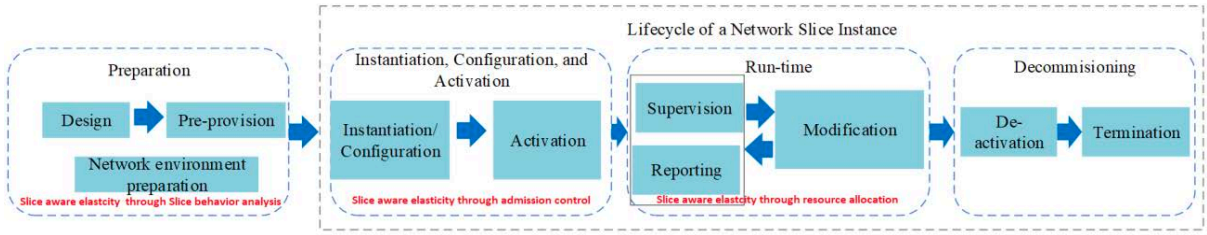


Figure 15 :Slice aware elasticity support in different phases of the lifecycle of network slice instances

In 5G-MoNArch, slice-aware elasticity is supported through three different approaches that can be mapped to three different phases of the lifecycle of a slice instance (Figure 15):

- Preparation-time approach: Slice behaviour analysis can be a critical asset for elasticity provisioning, since statistics can be exploited in the slice preparation phase to efficiently decide the basic configurations and set the network environment.

- Instantiation-time approach: Flexible slice admission control and network slice blueprint1 or template2 analysis (as the scheme proposed in Section 4.2.2.1) are applied during the slice instantiation and configuration phase, so as the activated slices have reduced probability to experience a computational resource outage.
- Run-time approach: Advanced sharing of computation resources among VNFs of multiple slices provide resource elasticity as the involved slices are in operation, by exploiting multiplexing capabilities.

2.10 Conclusions on the Emerging Slicing Concepts and Early Definitions of Slicing

After introducing the emerging slicing concepts, we observe that early definitions of slicing have a diverse scope as well as significant differences in the enabling technological underpinnings (e.g., SDN, NFV, etc.). Ongoing standardization efforts around slicing are closer to recent broadly scoped definitions of slicing and the key characteristics of cloud network slicing. However, standardization activities are too siloed off from each other and present fundamental divergences on their approach (i.e. where slicing is done and functional scope). We observe that existing approaches do not fully attend the current and foreseeable needs of cloud network slicing in its broadest capabilities.

One of the limitations we observed is related to the recurrent peer-to-peer interaction between different providers / domains sharing resources for the slicing. We expect the implementation of an alternative approach where sliced resources are directly accessible and attached to an end-to-end Slice by a provider initiating the Slice creation workflow. This will overcome the need for peer-to-peer interactions among different providers after the end-to-end cloud network slice has been put in place and becomes the driving objective of the NECOS project.

3. NECOS view on slicing: Key terms and definitions

Network technologies are making rapid progress in supporting new 5G communications (e.g., URLLC - Ultra Reliable and Low Latency Communication, mMTC - massive Machine Type Communication, eMBB - enhanced Mobile Broadband, and UDN – Ultra Dense Networking) and IoT trends. Emerging vertical industry applications, such as autonomous/cooperative driving, drone surveillance, remote sensing, and data analytic, are causing the conventional cloud computing model to evolve towards edge computing, by utilizing these advanced communications so that obtaining ultra-low latency, high broadband, and customized data transport service. In light of this observation and the native integration of network and cloud computing domains, investigating into the interaction between ***cloud*** computing and advanced ***net*****working** is a goal of the NECOS project.

One of the critical driving issues for such integration and interworking is the simple fact that is extremely inefficient and expensive to build a separate computing & networking infrastructure for each and/or new service.

The followings are key concepts utilized in the NECOS project architecture and system design.

3.1 Key Roles

Resource Provider –- It owns the physical resources and infrastructure (network/ cloud/ datacentre) and provides / leases them to operators.

Slice Provider– A slice provider, typically a telecommunication service provider, is the owner or tenant of the infrastructures from which cloud network slices can be created.

Slice Tenant – A slice tenant is the user of a specific network/cloud/datacentre slice, in which customized services are hosted. Infrastructure slice tenants can make requests for the creation of new infrastructure slice through a service model.

3.2 Key Concepts

Cloud Network Slice –- A set of infrastructures (network, cloud, data centre) components/network functions, infrastructure resources (i.e., connectivity, compute, and storage manageable resources) and service functions that have attributes specifically designed to meet the needs of an industry vertical or a service. As such a Cloud Network Slice is a managed group of subsets of resources, network functions/network virtual functions at the data, control, management/orchestration, and service planes at any given time. The behaviour of the Cloud Network Slice is realized via network slice instances (i.e., activated slices, dynamically and non-disruptively re-provisioned). The Cloud Network Slice key concepts are provided below:

- A cloud network slice supports at least one type of service.
- A cloud network slice may consist of cross-domain components from separate domains in the same or different administrations, or components applicable to the infrastructure.
- A resource-only partition is one of the components of a Cloud Network Slice, however on its own does not fully represent a Network Slice.
- A collection of cloud slice parts from separate domains is combined, connected through network slices, and finally aggregated to form an end-to-end cloud network slice.
- Underlays / overlays supporting all services equally (with ‘best effort” support) are not fully representing a Network Slice.

Infrastructure Slicing –- A management mechanism that a Resource Provider can use to allocate dedicated infrastructure resources and service functions to users. Broadly, partition strategies can be classified into three categories:

- Physical separation, such as dedicated backbones or dedicated data centres. However, this strategy is not cost efficient.
- Underlays/overlays supporting all services equally ('best effort" support), such as underlays / overlays, in the form of VPN as overlay solution. These solutions are neither flexible nor agile.
- Slicing, through cloud network resources allocation, where dedicated resources per customer/service ensure both isolation and customization on top of the same infrastructure.

3.3 NECOS Slicing, sharing and partitioning of resources

- From **a business point of view, a Cloud Network slice** includes a combination of all the relevant network and compute resources, functions, and assets required to fulfil a specific business case or service.
- From the **infrastructure point of view, Cloud Network slice instances** require the partitioning and assignment of a set of resources that can be used in an isolated, disjunctive or non- disjunctive manner for that slice.
- From the **tenant point of view, Cloud Network slice instance provides** different capabilities, specifically in terms of their management and control capabilities, and how much of them the network service provider hands over to the slice tenant. As such, there are two types of slices:
  1) **Internal slices**, understood as the partitions used for internal services of the provider, retaining full control and management of them.
  2) **External slices**, being those partitions hosting customer services, appearing to the customer as dedicated networks/clouds/data centres.
- From the **management plane point of view, Cloud Network slices** refer to the managed fully functional dynamically created partitions of physical and/or virtual network resources, network physical/virtual and service functions that can act as an independent instance of a connectivity network and/or as a network cloud. Infrastructure resources include connectivity, compute, and storage resources.
- From the **date plane point of view, Cloud Network slices** refer to dynamically created partitions of network forwarding devices with guarantees for isolation, customization and security.

3.4 NECOS Slicing operational modes

After reviewing the state of the art on slicing, including the academic literature, the very much fragmented standardization landscape, as well as the enabling technologies and early realizations of slicing concepts, one relevant observation of the NECOS project is the lack of a unique approach (even at the conceptual level) to afford Cloud Network slicing. We understand that the quest of providing sliced services by logically partitioning resources of a (multi-domain) software-defined infrastructure can be pursued through different slicing approaches depending at which level (and entry-point) of the stack slicing is performed. These different options are what we call *slicing operation modes* (*modus operandi*).

The choice of the slicing operational mode has a critical impact on the amount and type of components that need to be slice-aware, i.e., requiring software changes to support the slicing functionality, as well as the degrees of freedom on implementation choices, the isolation properties and the control and management responsibilities of both Slice Providers and Slice Tenants. Figure 16 presents a high-level layered view of software-defined infrastructure organized in the three planes:

(i) the *infrastructure* consisting of resources of different types (e.g., computing, networking),

(ii) the *control and management* realm encompassing resource management entities (e.g., VIM, WIM, VNFM), control and orchestration functions, and

(iii) the *business* plane featuring the applications and functions of an end-to-end service composed from an arbitrary number of network-related and application-specific end- or middle-point functions. Monitoring functionalities are assumed on each plane.

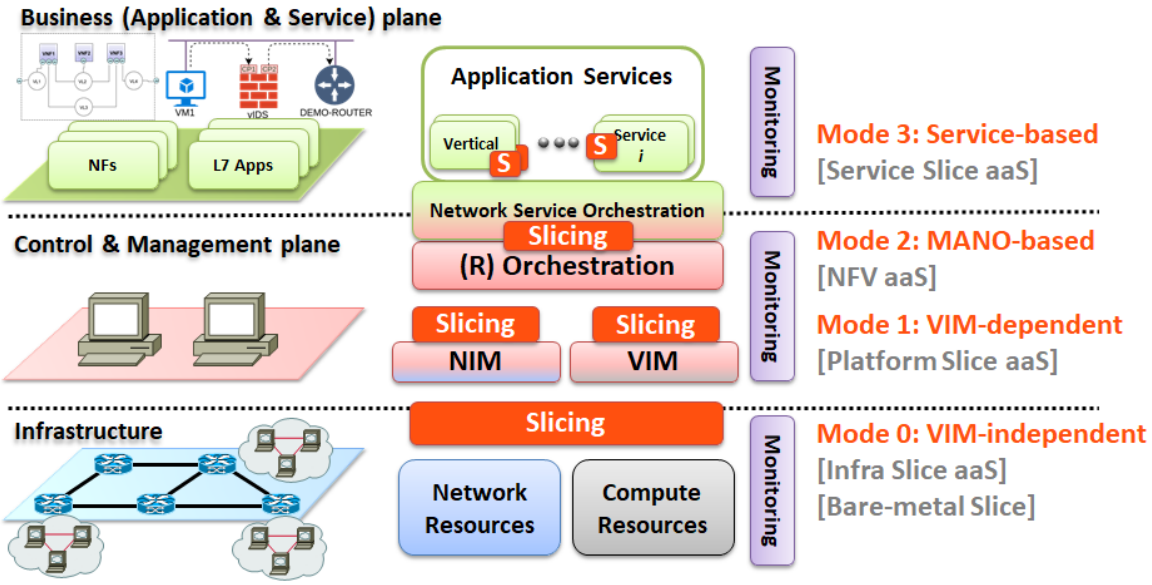


Figure 16: Candidate alternative slicing approaches

As illustrated by the dark orange boxes, labelled with Slicing in Figure 16, the logical partition of resources and their control and management functions to provide the slices can be realized at different architectural points. One fundamental question arises when considering the different plausible alternatives, which revolves around the need for changes. So, do we change all of the software to deal with slices, or do lower layer abstractions suffice to present a slice that all software just runs on?

Slicing at lower layers means that upper layers, such as VIMs and Orchestrators do not need to know about slicing. If a slice is presented to them, they can carry on working with no change or minimal change. If slicing is done otherwise, then all the main software elements need to be updated and adjusted to know about slices, i.e., all of the APIs, the modules, and internal function paths need to be adjusted and adapted to factor in slices. Hence, there are inherent trade-offs when selecting one or the other slicing operational mode. Ultimately, the actual decision on which slicing approach will depend on multiple criteria on key aspects of the use case scenario under consideration, dominated by the technical footprint of the provider, and the technical abilities and technological choices of the tenants and interest in opting for more or less control and responsibilities including customized software components. In the following, we discuss the key aspects of each slicing approach before we conclude the best fits to the goals and scope of NECOS.

These points show how candidate approaches (slicing modes) and the architecture meet, discussing the main characteristics, the SW dependencies and impact of slice-ready APIs:

- **Mode 0 - VIM-independent slicing:** the VIM on-demand model with DC slicing that allows direct inter-domain orchestrator to VIM interaction using a specific allocated VIM. Multi-tenancy happens at the lower resource layers. Being a VIM-independent approach, tenants can choose and have direct control of the VIMs, which can be as lightweight as desired. However, the freedom of choice and level of control come along potentially heavyweight operational responsibilities that the tenant must take by himself. Slicing support at resource providers can be considered lightweight since they do not need to run large slicing-capable VIM instances;

- **Mode 1 - VIM-dependent slicing:** the slicing in the VIM model, in contrast, is based on multi-tenancy at the VIM level and allows direct inter-domain orchestrator to VIM interaction using a specific allocated shim object. The tenant (Service/Resource) Orchestrator may require changes to use the slice-ready APIs of the Providers' VIM, which may also require profound software changes and customization to support slicing. This VIM-dependent approach requires arguably heavyweight VIMs – becoming a lock-in choice of the Provider – but frees the tenant from the VIM responsibility;
- **Mode 2 - MANO-based slicing:** the slicing in the Orchestrator model, which uses the inter-domain orchestrator API interaction, a peer to peer approach, where a slice is a set of data structures in the Orchestrator. From the tenant's perspective, depending on the actual split between a Service Orchestrator and a Resource Orchestrator, different software changes and standardized interfaces may be required. Eventually, the tenant BSS/OSS interfaces to the slice-ready APIs of the Provider orchestrator may also need some adaptation;
- **Mode 3 - Service-based slicing:** slicing at the application layer will not provide the relevant slice as a service, and is therefore not considered. This mode of slicing is constrained by the nature of a service-specific vertical solution, requiring per-service Tenant-Provider APIs plus eventual slicing support in the underlying layers, including multi-domain interactions.

4. NECOS Architectural Overview

4.1 Introduction and Context

This section presents an overview of the design of the overall NECOS system architecture required to support Slice as a Service. We have identified all the main architectural functions required for the implementation of NECOS. The architectural design will allow for performing the required service and resource orchestration in a federated and multi-cloud environment way, using the Slice as a Service concept.

Slicing is a move towards segmentation of resources and deployment of virtual elements for the purpose of enhanced services and applications on a shared infrastructure. The model we have for NECOS has the Service Orchestrator interacting with a Slice over the currently existing resources (Data Center and Network resources), as shown in Figure 17

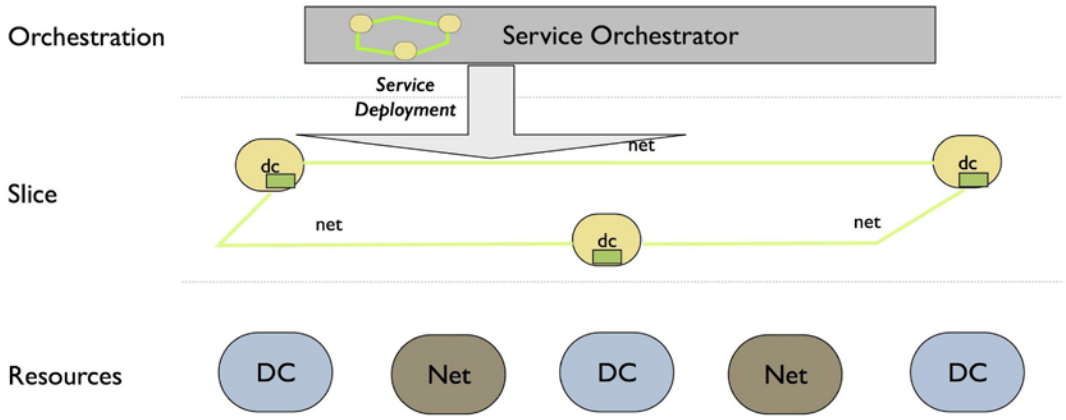
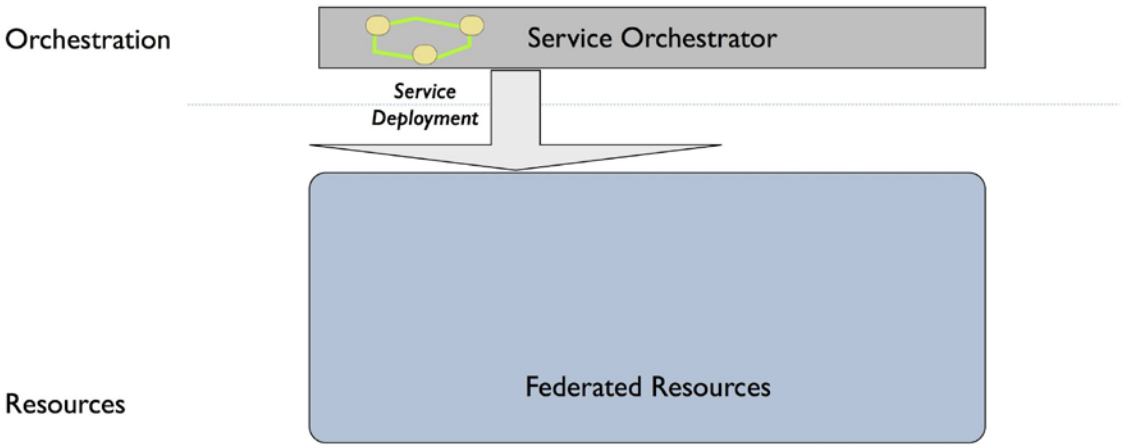
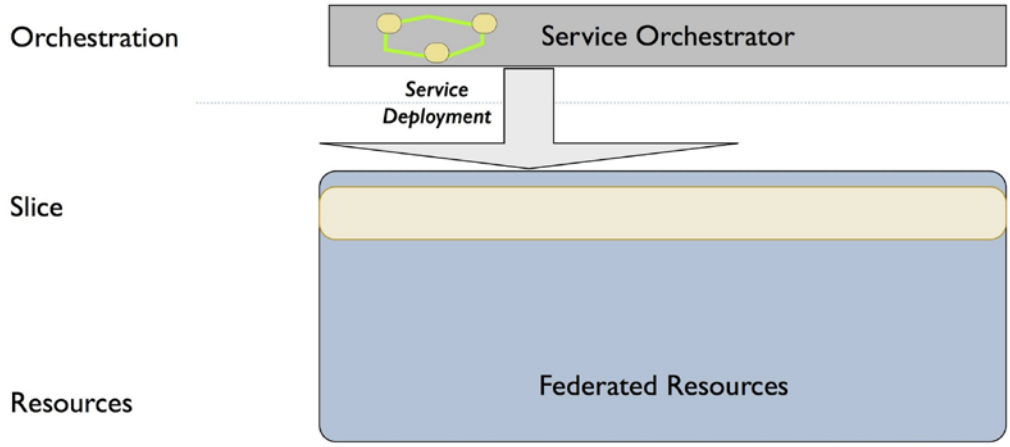


Figure 17: Model of Service Orchestrator interaction with a slice

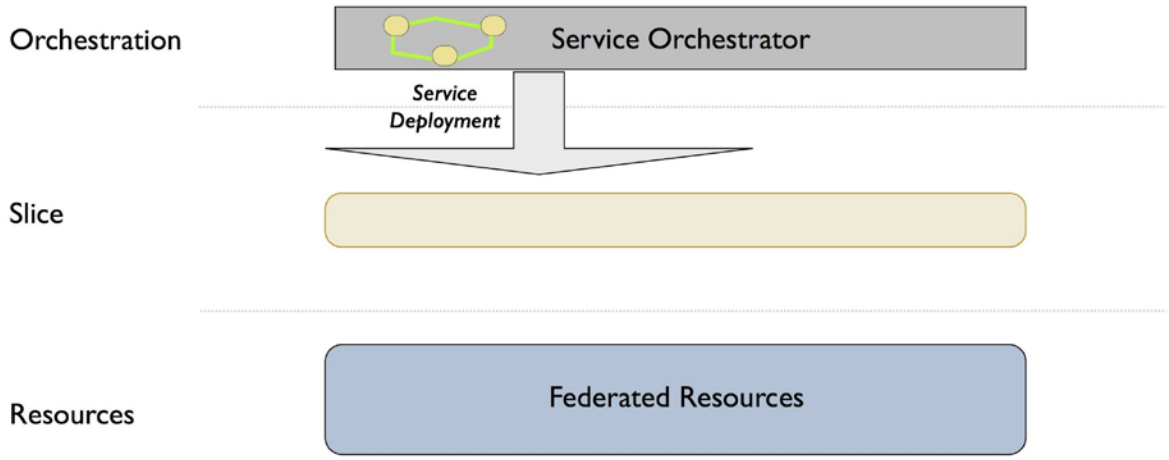
Currently, the approach is that an Orchestrator does service deployment across some federated resources, where there are no slices.



Differently, for NECOS, we plan to have a Slice as a Service where an Orchestrator performs service deployment across some federated resources, with slices, based on our designed model depicted in the following figure.



The idea is that the Service Orchestrator does service deployment directly to the slice with as little change as possible to the Service Orchestrator. To achieve this goal the slice should look like a small instance of the resources.



4.2 Functional Architecture Description

As mentioned, there are various schemes for creating slices across infrastructure domains that are currently being devised. They have different ideas regarding what a slice is, how to envision a slice within the existing systems, and how to manage the lifecycle of a slice, and how it is presented to the requester.

In NECOS, we have designed a specific slicing model that has as its foundation a mechanism to partition the underlying infrastructure into constituent *slice parts* and then combine these *slice parts* into a full end-to-end network cloud slice instance.

We are particularly focussed on a set of layered abstractions using slicing elements, ensuring that they all fit together for the purpose of service provisioning. The combination of the right abstractions, the right layering, and a separation of concerns in the architectural design will allow NECOS to create a powerful and flexible Slice as a Service mechanism.

In particular, the Slice as a Service approach provides an adaptable control plane, supporting features for: creating, growing, shrinking, and closing slices, as well as adapting slices at run-time, while considering service requirements and current cloud resource conditions.

In this section, we present the functional architecture that has been devised to support the Slice as a Service approach. The architecture contains three main high-level sub-systems, namely:

1) the NECOS (LSDC) Slice Provider (coloured in blue),
2) the Resource Marketplace (coloured in yellow) and
3) the Resource Providers (coloured in green). These sub-systems are provided in order to support the tenants of NECOS who wish to use Slice as a Service.

The tenant of NECOS is expected to be an organisation that requires slices for running their own services. Figure 18 presents these main elements, how they are grouped and the way they interact with the tenants (coloured in red) within the scope of NECOS.

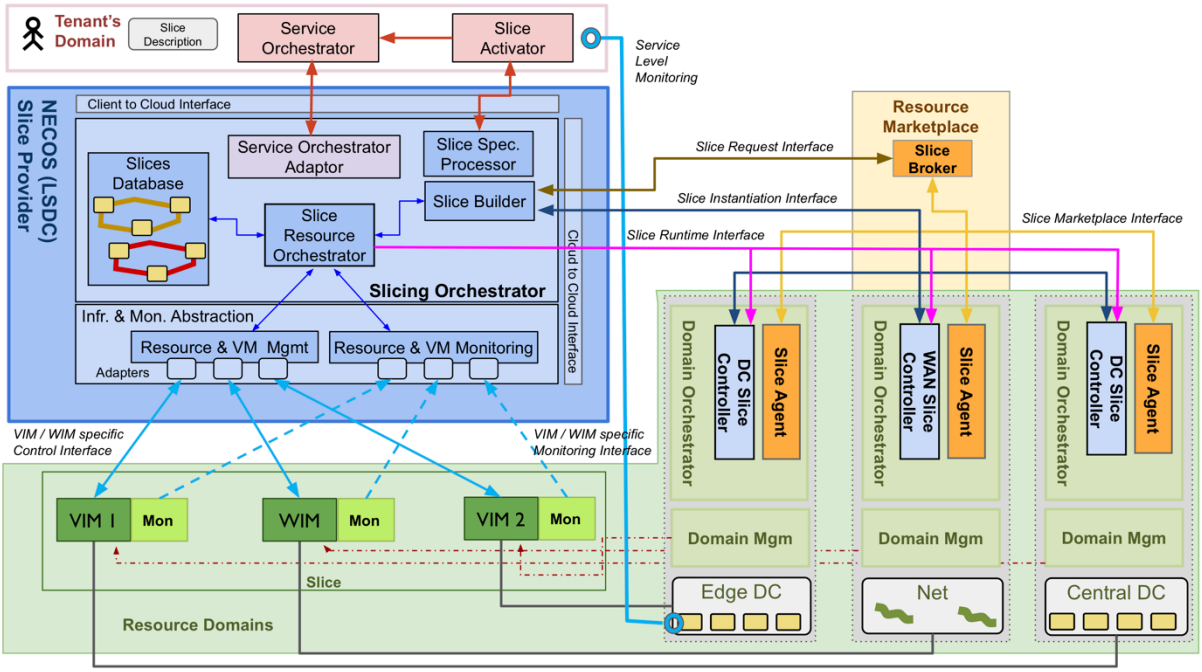


Figure 18: NECOS functional architecture

4.2.1 NECOS Tenants

The tenant of NECOS is expected to be an organisation that requires slices for running their own services. An example of such a tenant, could be a CDN company. The tenant is not expected to be a small organisation that needs a handful of Virtual Machines nor an end-user who wishes to use an online service. Both of these kinds of user already have multiple options available to them.

The NECOS tenant is depicted in the top-left part of Figure 18 – it can be an organisation that is expected to request to a NECOS Slice Provider the instantiation of heterogeneous resources including computation, storage and network in the form of end-to-end slice, and to run their own Service Orchestrator for deploying and managing their own services on that allocated slice. The tenant is likely to have a range of options for choosing a Slice Provider, and such could be based on factors such as commerce, geography, contractual obligations.

To participate in Slice as a Service environments, and for initiating and interacting with slices, the tenant also needs to run their own Slice Activator component. This is a new component that is responsible for two main tasks: (i) is allowing a Slice Description to be sent to the NECOS platform and (ii) is for handling the response back from the NECOS Slice Provider, which will inform a tenant that the slice is live and ready, and therefore once this response is received, the tenant can progress and deploy services onto the slice.

The Tenant can also deploy their own monitoring system to check the performance of the services (e.g., service specific KPIs) running on its slice (via the Service Level Monitoring Interface) or can delegate that to the NECOS provider according to the desired level of abstraction. The monitoring information related to the service instances will be provided as feedback to the Service Orchestrator to perform proper services' life-cycle management and to propagate life-cycle events to the NECOS slicing orchestrator if needed (via the Service Orchestrator Adaptor).

4.2.2 NECOS (LSDC) Slice Provider

The NECOS (LSDC) Slice Provider is the sub-system that allows for the creation of full end-to-end Slices from a set of constituent Slice Parts. In NECOS, a Slice looks the same as the full set of federated resources, with the main attribute being that the domains look a lot smaller, as they have been sliced. Therefore, we call the Slice, the LSDC - the Lightweight Software Defined Cloud.

The NECOS (LSDC) Slice Provider presents a northbound API compatible with a tenant's Service Orchestrator, thus enabling Tenants to operate on the full infrastructure, or to choose to interact with Slice as a Service providers, using NECOS.

When requesting a Slice from a NECOS provider, there is a Slice Builder component that goes out to a specially designed and configured Resource Marketplace that can find Slice parts across various participating Resource Domains, based on a Slice Specification.

Within the NECOS (LSDC) Slice Provider, the Slice Resource Orchestrator is the component that combines the Slice Parts that make up a slice into a single aggregated Slice. It is responsible for the orchestration of the running end-to-end Slices, including their run-time management of their life-cycle. It is also responsible for orchestrating the service elements across the Slice parts that make up the full end-to-end slice. Finally, it is the component that is responsible for the actual placement and embedding of VMs and virtual links for the services into the resource domains.

In order to interact with the actual remote cloud elements, the NECOS (LSDC) Slice Provider has an Infrastructure and Monitoring Abstraction (IMA) mechanism. This allows the Slice Provider to interact with various remote VIMs, WIMs, and monitoring sub-systems in a generic way, using plug-in adaptors with the relevant API interactions. The IMA allows the Slice Resource Orchestrator to interact with the remote clouds in order to provision the actual tenant services, and to monitor the remote resources running those services (via additional monitoring data that is not available via the Service Level Monitoring Interface).

4.2.3 Resource Marketplace

The Resource Marketplace in NECOS is the way that the NECOS (LSDC) Slice Provider is able to find the slice parts to build up a slice. Rather than having a pre-determined set of providers that have been configured in a federation, we chose to use a more flexible model of a marketplace from which we could provision slice parts. This was done for two main reasons: (i) in order to build end-to-end slices, we need a mechanism to reach and interact with providers in multiple geographic places - including mobile edge and sensor networks - which are often not covered with existing federation approaches; and (ii) slice creation is generally more dynamic than federation agreements, so we need a highly dynamic run-time mechanism for finding providers. We observed that there are online marketplaces for flights and for hotel rooms that provide a good operational model to follow. There is a broker which can go to multiple agents, dynamically, and get a collection of offers for the required resource. The end-user can then make a choice as to which offer is best suited. The agents can be a division of the resource provider, or they may be an external aggregator that provides compound offers. From the perspective of the broker and the end-user, the agent just provides offers at a price point with certain conditions. Such an approach fitted with the NECOS Slice as a Service system very well.

Within the NECOS (LSDC) Slice Provider, the *Slice Builder* is responsible for building a full end-to-end multi-domain slice from the relevant constituent slice parts. In order to do so, the Slice Builder has to find available resources from the marketplace. This search involves communication with a *Slice Broker*, the entity responsible for contacting provider *Slice Agents*. In the NECOS approach, there is

expected to be multiple Slice Brokers in various locations - maybe multiple per country. Each *Slice Broker* will have access to many *Slice Agents* across many geographical domains, which are able to provide offers for the required slice parts that match a set of request constraints. For example, the slice may need DC slice parts in Spain, Greece, and the UK, with network slice parts that connect the DC slice parts.

On the basis that the various offers for the parts have been collected by the *Slice Broker*, it responds to the *Slice Builder* request with alternative options (i.e., the offers) for each slice part. Then the *Slice Builder* can select the relevant resources for each slice part, based on the original slice request.

To aid in flexibility, we consider that different Marketplaces can be setup and configured, based on the use-cases of the various participants. Examples of Marketplaces include:

- *the telecoms Marketplace*, which is a limited setup between close co-operating telecoms providers, i.e., allowing the creation of slices across each other's infrastructure,
- *the contract Marketplace*, which is a setup between partners with a level of trust based on signed contracts, i.e., allowing their resources to be used by others for slices,
- *the open Marketplace*, which allows any provider to offer resources and register their offerings for users.

4.2.4 Resource Providers

The *Resource Providers* are those organisations that can provide the resources required for the slice parts - namely, Data Center resources in the form of servers and storage, and Network resources. Further resources can be provided by organisations that have Mobile Edge, Sensor Networks, Wireless, etc. Each resource provider will be capable of providing slice parts, which will be part of a full end-to-end slice. The *Resource Provider* also needs to provision the relevant manager for each slice part, so a VIM for a DC slice part, or a WIM for a network slice part. These managers allow those resources to be managed and utilised by the *Slice Resource Orchestrator*.

In order to support service provisioning over the slices, it is necessary to have a mechanism to support the slicing of the DC compute and storage resources, as well as the network resources. To manifest this slice approach, in NECOS, two components at the infrastructure level act as a point of control and management of DC and Network slices.

For each data centre, a *DC Slice Controller* is in charge of creating DC slices within the data centre, allocating the required compute and storage resources for a given Slice part, and returning a handle to a VIM running on it. The VIM can either be a separate instance deployed on demand based on the tenant specification (e.g., if an explicit reference was part of the Slice request) or an existing VIM running in that Resource Domain on which an isolated interaction point (such as a shim) was created for the tenant.

Similarly, for each network domain, a *WAN Slice Controller* is in charge of instantiating a network slice between two DC slices and either deploying an on-demand WIM or adding an isolated interaction point for the tenant to an existing WIM instance.

The mechanism to find these slice parts was described above, in the *Resource Marketplace* section. To participate in a Marketplace, a resource provider needs to run a *Slice Agent*, which can find available resources within the local resource domain, and then provide an *offer* for those resources to the *Resource Broker* (in the Marketplace), with a specified timespan and pricing model associated with the slice part.

5. Architectural Functional Blocks

This section contains a review of the main architectural blocks which are in the three-main high-level sub-systems described in the previous section. The NECOS (LSDC) Slice Provider, which is the sub-system that allows for the creation of full end-to-end Slices from a set of constituent Slice Parts, has two main functional blocks:

(i) the Slicing Orchestrator, and

(ii) the Infrastructure & Monitoring Abstraction (IMA) based upon the following components:

Slicing Orchestrator

- Slice Resource Orchestrator
- Service Orchestrator Adaptor
- Slice Specification Processor
- Slice Builder
- Slice Database

Infrastructure & Monitoring Abstraction (IMA)

- Resource and VM Monitoring
- Resource and VM Management
- Adaptors for VIM/WIM Control

There are also the *Resource Domains* which contain these components:

- DC Slice Controller
- WAN Slice Controller

And the *Marketplace* that has these components:

- Slice Broker
- Slice Agent

All of these functional blocks and their associated components are described in more detail in the following sections.

5.1 Slicing Orchestrator

The Slicing Orchestrator performs operations that deal with slices via its two main components: (i) the Slice Builder, responsible for requesting to the Resource Marketplace the different Slice parts that will be included in an end-to-end Slice via performing an initial orchestration phase to work out a subset of domain that could be used as candidates for the end-to-end slice creation; (ii) the Slice Resource Orchestrator (SRO), which combines the allocated Slice Parts into a single aggregated Slice on which it holds the overall end-to-end view and manages the lifecycle of each individual slice part. Furthermore, the SRO is also responsible for orchestrating the service elements across the Slice parts that make up the full end-to-end slice as it performs the actual placement and embedding of VMs into the resource domains. It also takes care of the life-cycle management of the services running on the slices.

To do all the above-mentioned operations the Slicing Orchestrator uses the named interfaces and APIs being called:

- *The Slice Request Interface:* provides the mechanisms to the Slice Builder to initiate the instantiation of an end-to-end Slice via interacting with the Slice Broker in the Resource Marketplace. It is used by the Slice Broker to provide the description of the Slice to be allocated, by using *the Slice Marketplace Interface* for interaction between the Slice Broker

and the Slice Agents to implement mechanisms for the propagation of resource offerings between (external) resource domains. Different Slice Agents will register the availability of their resources to the Slice Broker.

- *The Slice Instantiation Interface:* is used by the Slice Builder after available resources offerings have been identified via the Slice Broker; individual resource allocation requests are sent to the relevant Slice Resource Controllers which allocate resources for every single Part of the end to end Slice.
- *The Slice Runtime Interface:* implements the interface that provides functionalities to dynamically modify the resource allocation for a Slice Part. This is required by the Slice Resource Orchestrator to perform lifecycle operation on the end to end Slice according to the feedback received for each Slice Part via the collected monitoring measurements.

5.1.1 Slice Resource Orchestrator

The Slice Resource Orchestrator (SRO) component is responsible for combining the Slice Parts that make up a slice into a single aggregated Slice. It is also responsible for orchestrating the service elements across the Slice parts that make up the full end-to-end slice, and as such it is the component that handles the actual placement and embedding of VMs into the resource domains.

5.1.1.1 End-to-End Slice Activation and Registration

The Slice Resource Orchestrator gathers data on the overall view of the slice in terms of allocated computing resources and network topologies, and stores this in the Slice Database, which is the component holding such knowledge. The SRO will be the component interacting with the DC/WAN Slice Controllers (returned by the Slice Builder) to complete the creation of the end-to-end slice. This will be performed by interconnecting each VIM/WIM instance to its closest domain network edge points and taking care of setting up the inter-domain network connectivity between the different Slice Parts.

5.1.1.2 End-to-End Slice Elasticity

The Slice Resource Orchestrator will take care of performing the slice life-cycle management, i.e., continuously checking whether the allocated slice is capable of fulfilling the requirements that were initially requested by the Tenant. With this goal in mind, the SRO must have access to a set of monitoring measurements (coming from each Slice Part) that provide information about respective slice resource utilisation patterns (e.g., the number of available cores, memory, network delay/loss, etc.). This information sets will be at the granularity of the slice and will not directly be linked to any of the service instances running on that slice. The monitoring information is expected to be propagated to the SRO via the underlying monitoring abstraction implemented by the IMA. We foresee multiple elasticity scenario where the SRO might be involved in. Later in this document, there is a description of some of those scenarios that will be addressed by NECOS.

5.1.1.3 Decommissioning of the End-to-End Slice

In this case, the Slice Resource Orchestrator contacts all DC/WAN Slice controllers, again via the Slice Runtime Interface, informing them appropriately to release the resources.

5.1.1.4 Instantiation of Virtual Resources for the Services

The Slice Resource Orchestrator will perform the allocation of the resource elements that are required for the instantiation of the service instances (possibly described by Forwarding Graphs) on the global resource view available in each end-to-end slice. This will be performed by embedding service elements (e.g., a Container, a VM, a VNF or a virtual link), on the slice resource view. The orchestrated instantiation of the different service virtual elements will happen via the northbound interface of the Infrastructure & Management Abstraction. The IMA will take care of adapting the above instantiation requests to the specific underlying VIM/WIM technology. The SRO will have to request to the IMA the allocation of different adapters required to interact with the different Infrastructure Managers running in the Slice Parts. This will happen via the IMA API providing the entry point to a VIM / WIM the adapter should interact with.

5.1.2 Service Orchestrator Adaptor

The Service Orchestrator Adaptor is the component that interacts with the tenant's Service Orchestrator. Its northbound interface talks with the southbound interface of the tenant's Service Orchestrator. It provides the features to adapt the calls from the tenant's Service Orchestrator into calls internal to the Slicing Orchestrator. The Service Orchestrator Adaptor will interact with the other components in the Slicing Orchestrator, for various tasks. This Adaptor provides the linkage that allows a created Slice to look like a small version of the whole infrastructure.

Due to the modular nature of the architecture, we expect that there can be different implementations of the Service Orchestrator Adaptor which can be devised to interact with different Service Orchestrators. A new Service Orchestrator Adaptor can be allocated to each tenant, instantiating the relevant one, based on the tenant's request.

5.1.3 Slice Specification Processor

The Slice Specification Processor plays a fundamental role in the interaction between a Tenant and the NECOS system via the Client-to-Cloud API. It takes a Slice Specification as input from the Slice Activator in the Tenant's Domain, and performs the processing tasks required to build a Slice creation request for the Slice Builder. According to the particular API call invoked by the Tenant (i.e., the create slice call discussed in D4.1, Section 5.1), the Slice Spec Processor will be able to handle a variety of tenant's slice requests.

5.1.4 Slice Builder

The *Slice Builder* is responsible for building a full end-to-end multi-domain slice from the relevant constituent slice parts. In order to do so, the *Slice Builder* has to search for available resources in the marketplace. This search involves communication with the *Slice Broker*, the entity responsible for contacting provider *Slice Agents*, as described in The Resource Marketplace section of this document. Slice part requests must reflect the structure of the slice that the tenant wishes to create, as they are described in deliverable D4.1. *Slice Agents* in the remote domains will respond to the *Slice Broker*.

5.1.5 Slice Database

The Slice Database stores all the information related to the topology of each slice. It keeps information on the slice parts, which can be a DC slice part, a Network slice part, or an edge part, as well as the services running in each slice. Figure 19 presents the relationship between the model entities.

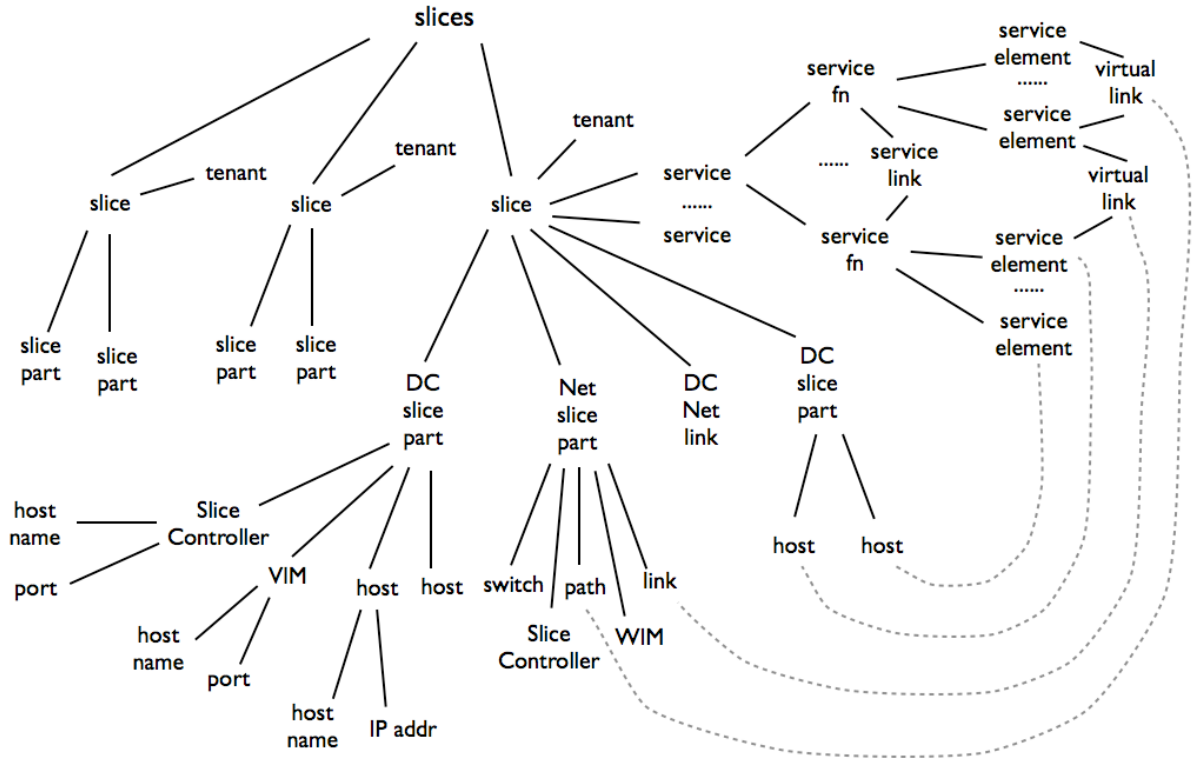


Figure 19: The Slice Database contains elements from the entity model, and is used by the Orchestrator for making its decisions (this model is described in more detail in deliverable D4.1)

5.2 Infrastructure & Monitoring Abstraction

The Infrastructure & Monitoring Abstraction (IMA) component provides a uniform abstraction layer above the heterogeneous VIM / WIM and monitoring subsystems that are part of an end-to-end Slice. Different Slice Parts will constitute the end-to-end Slice, and each part can potentially rely on a different technology (e.g., specific VIM / WIM and monitoring subsystem implementations).

IMA will provide the Slice Resource Orchestrator with an abstract way to perform the following functions while interacting with the underlying distributed resource domains that are part of the end-to-end Slice:

- Collecting information about the resource topology of each Slice Part.
- Collecting resource monitoring information from each Slice Part (to be used for the Slice life-cycle management).
- Requesting the allocation on some specific Slice Parts of the virtual elements implementing the services requested by the Tenants.
- Checking and verifying the status of the virtual elements allocated in each Slice Part (e.g., detecting failures associated to any of virtual service elements).
- Collecting resource facing KPIs (such as CPU, memory, storage consumption) for the virtual service elements running in the Slice Parts.

IMA consists of three main components, each one dealing with the implementation of the abstraction mechanisms required to perform the functions from the above list:

(i) Resource and VM Management,

(ii) Resource and VM Monitoring and

(iii) Adaptors for VIM/WIM Control & Monitoring Interface.

5.2.1 Resource and VM Management

This is the IMA sub-component responsible for providing the abstract API to be used by the Slice Resource Orchestrator to interact with the VIM / WIM inside the different Slice parts to perform both resource operations and VM management operations. The SRO will use some generic methods to collect the infrastructure view of each Slice part in order to build the global view of the end-to-end Slice on which the service elements requested by the Tenants will be embedded.

5.2.2 Resource and VM Monitoring

This is the IMA sub-component that implements the set of abstractions required to provide the Slice Resource Orchestration with a homogeneous view on the resource utilisation using monitoring information gathered from the different Slice parts. The real-time data gathered from the monitoring is directly associated with the structured global view gathered by the Resource and VM Management component. This type of monitoring information complements the resource state information collected by the Resource and VM Management components – while the latter provides a feedback about the status of the virtual resources deployed on the slice (mainly to detect faults and check the consistency between the performed embedding operations and the actual run-time resource status), the former provides an actual monitoring real-time information flow that will be used to check the performance of the physical and virtual resources elements of the slice. By combining both the structure and the real-time flows, we get a clear view of the current status of all slice elements.

5.2.3 Adaptors for VIM/WIM Control & Monitoring Interfaces

Specific adaptors for particular VIMs and WIMs are required to support multiple technologies usually providing different levels of abstraction and a variety of supported features. The VIM and WIM Adaptors are specific wrappers for different VIM/WIM implementations, providing basic service platform agnostic wrapping to common NECOS functions. In practice, the adaptors should translate generic calls into specific commands or methods in the context of specific VIMs and WIMs. Typical VIM and WIM adaptors include:

- Cloud adaptors: OpenStack, Heat, Kubernetes, VLSP;
- VNF adaptors: OpenMano, Open Baton, OPNFV;
- Transport adaptors: SDN controllers such as Opendaylight, Floodlight;
- RAN adaptors;
- Edge adaptors.

5.3 Resource Domains

In order to support service provisioning over the slices, it is necessary to have a mechanism to support the slicing of the DC compute and storage resources, as well as the network resources. To manifest this slicing approach, in NECOS, two components at the infrastructure level act as a point of control and management of DC and Network slices. For each data centre, a DC Slice Controller is in charge of creating DC slices within the data centre and allocating the required compute and storage resources. This process can be different according to the level of abstraction used by a Tenant for a Slice request and the type of mode the NECOS system is asked to operate. The resource allocation can end-up with the deployment of an on-demand VIM based on a particular tenant specification or the creation of separate handle for a Slice in a shared VIM to be passed back to the requesting Tenant. Similarly, for each network domain, a WAN Slice Controller is in charge of instantiating a network slice between

two DC slices and deploying the on-demand WIM or a reference to a Slice created in an existing WIM instance. In the following, we provide a brief description of both the DC and WAN Slice Controller components.

5.3.1 DC Slice Controller

The DC Slice Controller is the component that resides in each Data Center Provider and dynamically creates a data centre slice. To accomplish this task, the DC Slice Controller performs the following operations:

- *Resource Management*: the DC Slice Controller manages a pool of all of the DC (compute and storage) resources in the data centre that are allocated to participate in slicing and keeps track of which resources have been allocated to which slice, together with meta-data such as the VIM entry point, and the keys used for access to all the data centre resources.
- *Slice Creation*: the DC Slice Controller handles requests for DC slices coming from the Slice Agent and determines if it is possible to create a new DC slice in the data centre based on local resource availability.
- *Resource Allocation*: if the DC slice creation is possible, the DC Slice Controller is responsible for selecting the DC resources, from the pool, that should be allocated to the slice.
- *VIM Deployment/Shim Deployment*: In Mode 0, for each DC slice there is an on-demand VIM. In this case, the DC Slice Controller is responsible for allocating and deploying a VIM of a particular type to the DC slice, and configure it to use the DC resources which have been picked for the slice. This is achieved by creating VIM template images that are stored in an image repository and adapted according to the slice specification. In Mode 1, for each data centre, there is an existing VIM running in that Resource Domain. In this case, the DC Slice controller is responsible for creating an isolated interaction point (such as a shim) for the tenant.
- *Slice Elasticity*: under the control of the Slice Resource Orchestrator, the DC Slice Controller updates (adds or removes) the DC resources assigned to a slice on-the-fly as a slice can grow or shrink at runtime.
- *Slice Deletion*: the DC Slice Controller handles requests for the deletion / shutdown of DC slices. The resources used by the slice will be returned to the resource pool. In the Mode 0 case, the VIM allocated for the slice will also be shutdown.
- *Slice to Network Connectivity*: the DC Slice Controller is responsible for connecting an allocated DC slice part to a specified external network end-point. This will create a special tunnel connecting the VLAN of the DC slice part to the specified external network end-point.

5.3.2 WAN Slice Controller

The WAN Slice Controller is the component that resides in each Network Provider and that dynamically creates a Network slice. A Network slice is a set of virtual links that connects two DC slices. In order to create a Network slice, the WAN Slice Controller performs the following operations:

- *Network Management*: the WAN Slice Controller manages all of the network resources in the network provider domain that are allocated to participate in slicing and keeps track of which network resources have been allocated to which slice.
- *Slice Creation:* the WAN Slice Controller handles requests for Network slices coming from the Slice Agent and determines if it is possible to create a new network slice in the network domain based on local resource availability (e.g. bandwidth).
- *Resource Provisioning*: if the Network slice creation is possible, the WAN Slice Controller is responsible for providing the set of virtual links required to connect the given DC slices.

- *WIM Deployment/Shim Deployment*: In mode 0, for each Network slice there is an on-demand WIM. In this case, the WAN Slice Controller deploys a WIM to the Network slice, and configure it to use the network resources which have been assigned for the slice. In Mode 1, for each network domain, there is an existing WIM running in that domain. In this case, the WAN Slice controller is responsible for creating an isolated interaction point (such as a shim) for the tenant.
- *Network Slice Elasticity*: under the control of the Slice Resource Orchestrator, the WAN Slice Controller updates the network resources assigned to the slice on-the-fly as a slice can grow or shrink at runtime.
- *Slice Deletion:* the WAN Slice Controller allows handles requests for the deletion/shutdown of WAN slices. All resources used will be returned to the resource pool, and the WIM allocated for the slice will also be shutdown.

5.4 Resource Marketplace

The *Marketplace* is a supporting fully-distributed system that locates suitable slice parts from a set of participating resource domains. As mentioned, the *Slice Builder* will interact with a *Slice Broker*, pass in a request for a slice, and receive a list of suitable slice parts. The *Slice Broker* discovers these slice parts by interacting with a set of *Slice Agents* that are hosted by the involved resource domains.

Due to the distributed nature of the Marketplace, it is possible to have more than one Marketplace installations. We view the Marketplace in NECOS in a similar way to existing online marketplaces such as those for hotels (e.g., booking.com) or for flights (e.g., skyscanner.net), yielding hopefully the same business opportunities for the participating resource providers.

Within NECOS, we consider the following kinds of Marketplace:

- *the telecom Marketplace*, which is a limited setup between close co-operating telecoms providers, i.e., allowing the creation of slices across each other's infrastructure,
- *the contract Marketplace*, which is a setup between partners with a level of trust based on signed contracts, i.e., allowing their resources to be used by others for slices,
- *the open Marketplace*, which allows any provider to offer resources and register their offerings for users.

The internal operation between the *Slice Broker* and the *Slice Agents* is not exposed externally, but we propose the following two models of operation:

- a *pull model* – where the *Slice Broker* interacts with the *Slice Agent*, and the *Slice Agent* dynamically determines the availability with the resource domain, at run-time,
- a *push model* – where the resource provider pre-determines which resources and groups of resources to be allocated for slices, and these are pushed out to a catalogue in the *Slice Broker*.

The mechanisms to realize the above operation models are detailed in deliverable D5.1, while a resource discovery workflow with an information model representation of the requested and offered resources in deliverable D4.1. We give a functional description of the *Slice Broker* and the *Slice Agent* components below.

5.4.1 Slice Broker

The *Slice Broker* is the central component of a Marketplace. It accepts requests for slices and is responsible for locating resource providers that are able to satisfy the requests, based on a set of constraints and requirements, such as the amount of resources and location, as specified in deliverable D4.1.

In order to realize the resource discovery process, the *Slice Broker* decomposes the original request to a set of slice part requests and addresses the requests to a number of *Slice Agents*, each belonging to a particular resource domain. For each slice part, for instance a requirement for N hosts in a particular country, the *Slice Broker* may enquiry multiple *Slice Agents* whether they can satisfy that request. The answers collected for each slice part present alternative options for the slice resources to be instantiated to create the slice. Alternatives for each slice part will be returned as a reply to the request originating from the *Slice Builder*. The *Slice Builder* will select the resources for each slice part based on a relevant decision mechanism (i.e., considering the cost model and the suitability of the resources), as described in Section 5.1.4 Slice Builder.

5.4.2 Slice Agent

The *Slice Agent* component resides at each infrastructure provider. It is responsible for determining if a slice part request can be satisfied, and responds with resource volumes, a cost model and possible SLA/SLO attributes. In order to reply to the *Slice Broker's* requests, the *Slice Agent* communicates with its local DC/WAN domain controller, i.e., offering updated information on the available resources. Thus, the *Slice Agent* is a "translation" component that allows local domain controllers to expose resource information to the marketplace. This allows NECOS to interoperate with a diverse range of controllers, given that technology specific *Slice Agents* will be realised. In other words, a typical *Slice Agent* implementation abstracts and translates the offered resources from heterogeneous domain controllers in the NECOS information model representation (i.e., defined in deliverable D4.1), using a technology-agnostic north interface, the *Slice Marketplace Interface*, and a technology-dependent south interface.

6. Principal Architectural Artefacts

6.1 Service Slice Control Loops

In [ETSI2018], ETSI defines an architecture for network slice management systems that allow different network service providers to coordinate and concurrently operate different services as active network slices.

The NS methods are aimed at providing custom design of networks suitable for a specific use case (vertical market). Such methods need to be able to translate a service requirement into a normalized description of resources across different type of network domains based on NGMN's description of network slicing. The 3-layer approach with a Service Layer, a Network Slice Instance Layer, and a Resource Layer as in Figure 20.

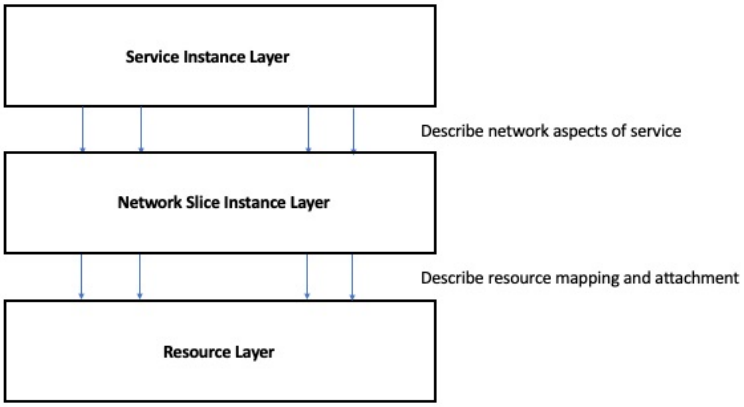


Figure 20: 3-Layer Network Slice and Service Concept (source [ETSI 2018])

As mentioned above, NECOS follows a similar approach in the context of network cloud slicing. Resources are partitioned, connected through different domains and organized with the purpose of serving a particular high-level service; this is, a particular set of functional and non-functional requirements. As in traditional virtualization services, it is expected that NECOS keeps honouring those requirements in front of workload changes or failures with the additional complexity of dealing with the same dynamics in each slice that is providing. In the case of NECOS, the three layers are depicted in the already presented Figure 17.

If we think about the assurance processes involved in such an organization, it is possible to identify several overlapping control loops. First, it is easy to think that different loads in the service, for example, more users requesting more video plays, may trigger an elasticity process that will boot more virtual machines running streaming servers. That would include monitoring metrics from the top service and actions to be performed on top of the slice. However, more virtual machines running on top of the slice will consume more CPU, memory, etc. adding pressure on the physical resources reserved for the slice. Thus, an increasing physical CPU usage may need the triggering of a lower level elasticity process in which more resources have to be added to the slice, locally or remotely in some part of a NECOS federation. Such a control loop will need monitoring data acquired from the physical devices and take actions on the physical layer and on the slicing organization of those physical resources.

Those two loops may be seen, in a simplified way, in Figure 21:

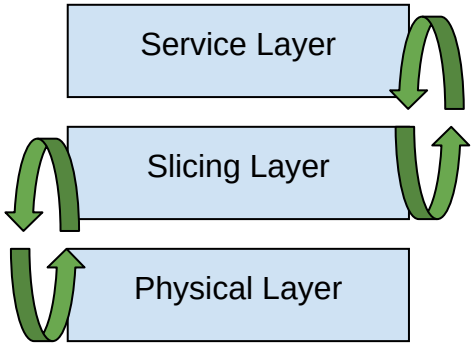


Figure 21: Simplified view of the assurance loops in a network cloud slicing framework such as NECOS

However, when we consider that a slice is a partition of physical resources that are shared among other slices, we have to see more complex interactions such as those in Figure 22.

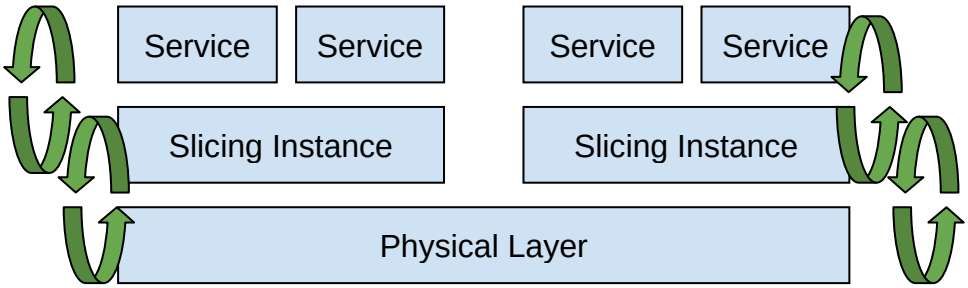


Figure 22: Interconnected assurance loops

It is the task of a manager such as NECOS to coordinate all those control loops in a stable and fair manner.

The Internet-draft in [Homa et al. 2019] presents three types of provision models: SaaS (Software as a Service), PaaS (Platform as a Service), IaaS (Infrastructure as a Service).

In the **SaaS-like model**, a network slice provider designs network slice templates in advance, and a tenant selects and uses one which fulfils most its requirement among the templates. The specifications of network slices are abstracted to KPIs as networks and servers and shown to tenants. In short, detailed parameters of infrastructure network are hidden from tenants.

In the **PaaS-like model**, a tenant makes its request, including connected area, path routes, the KPIs, and included service functions, and a Network Slice provider designs a template and instantiate a network slice based on the request dynamically. The configurable values would vary depending on the policy of each NS provider. Please notice that this model is not exactly as the Mode 1 presented in deliverable D3.1. While it is a similar concept, it differs in the inclusion of the service functions as part of the slice itself.

In the **IaaS-like model**, a tenant designs its own network slice templates and instantiates NSs by indicating concrete resources to infrastructure operators. In other words, infrastructure operators provide just their resources, and NSs are coordinated by the tenant.

NECOS' model is similar to the **PaaS-like model** and therefore the management of the elasticity and assurance control loops needs to address a hybrid model in which the Slice Orchestrator is responsible

for monitoring and controlling platform related variables, and the tenant is responsible for monitoring and controlling the top services variables and to request to NECOS the needed scaling actions.

The picture in Figure 23 provides a view of the NECOS scenario focused on the main monitoring components, the Slice Resource Orchestration and the Tenant's Domain. On the same figure, two separate assurance / control loops are represented: an external one (dashed pink line) involving the Tenant's Domain and service-specific KPIs (Service Level Control Loop), whose values are collected by specific Service Level Monitoring probes instantiated by the Tenant together with their service; an internal one (dashed dark-blue line) involving the Slicing Orchestrator and slice specific KPIs, which are related to the measurements collected from the slice resources via the IMA (Slice Control Loop). The latter will also include resource facing KPIs related to the service elements running in the slice itself as this information is usually made available by the monitoring subsystem running in each Slice Part.

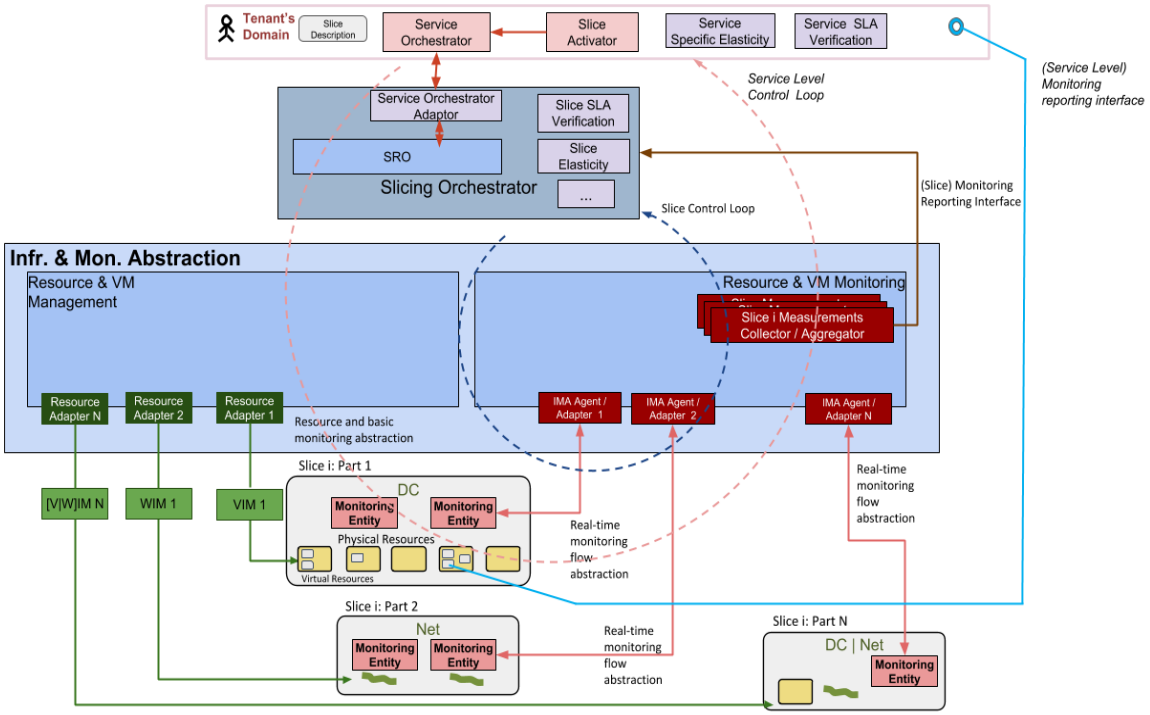


Figure 23: NECOS assurance loops

It is straightforward realising that the two loops are inherently related, and events happening on the external one are likely to affect the behaviour of the components of the internal one. For example, when the measured service related KPIs received by the components of the Tenant's domain indicate a possible violation of the expected QoS or SLA for a given service instance, the Service Orchestrator might interact to the SRO via the Service Orchestrator Adaptor in order to request a change of topology (service elasticity) for that service via the IMA (e.g., adding an additional service component).

This request can result in the execution of two different workflows, according to the status of the resources of the end-to-end Slice on which that service is running.

1. A new service element can be added to one of the existing Slice Parts: the SRO embeds the service element in the right Slice Part and interacts with the specific VIM via the Resource Adaptation.
2. A new service element cannot be instantiated in any of the available Slice Parts due to the lack of resources. In this case, the SRO will trigger the proper slice elasticity workflow,

namely either increasing the resources of an existing Slice Part or adding additional Slice Parts to the end-to-end Slice.

6.2 End to End Slicing

Frameworks as the one described in [ETSI2018] dedicated to network slices (NS) allows infrastructure providers to create and operate slice subnets in order to create/activate an end-to-end network slice. In [ETSI2018] three entities interact for the use of a network slice: the tenant, the Network Slice Provider (NSP), and the Network Slice Agent (NSA). The former uses the network slice; the NSP provides network slice as a service to tenants, while the NSA is a component in the infrastructure provider's domain that maps NSP's information and operations within its domain. In [ETSI2018], network slices are defined in terms of two types of resources: links, that relate to traffic or path related constraints, and nodes that represent either compute or store resources. End-to-end network slices are then created and setup by a set of managed objects responsible for resource discovery, slice provisioning based on NS service profiles, and NS instantiation from NS service profiles. Operations related to the network slice are grouped into two types: NS service operations that create, remove, modify and augment an end-to-end network slice, and NS subnet operations that take place in the context of resources (e.g. monitoring).

As shown in Figure 24, NECOS follows a similar approach in the context of network cloud slicing. Tenants are the users of the slice. The NECOS Slice Provider sub-system maps to the NSP entity and it is responsible for slice provisioning based on the tenant's service description. The Slice Controllers (DC and WAN) execute similar functions performed by the NSA, understanding the NECOS Slice Provider requests and translating them into concrete actions (e.g. resource allocation, VIM instantiation, etc.) inside the domain, so that an end-to-end slice can be created, removed, modified and augmented accordingly. The resource discovery function in NECOS is performed by the Marketplace sub-system, the only sub-system that does not have a direct mapping to the entities described in ETSI reference framework. In the following, we discuss in more detail how NECOS provides end-to-end slices.

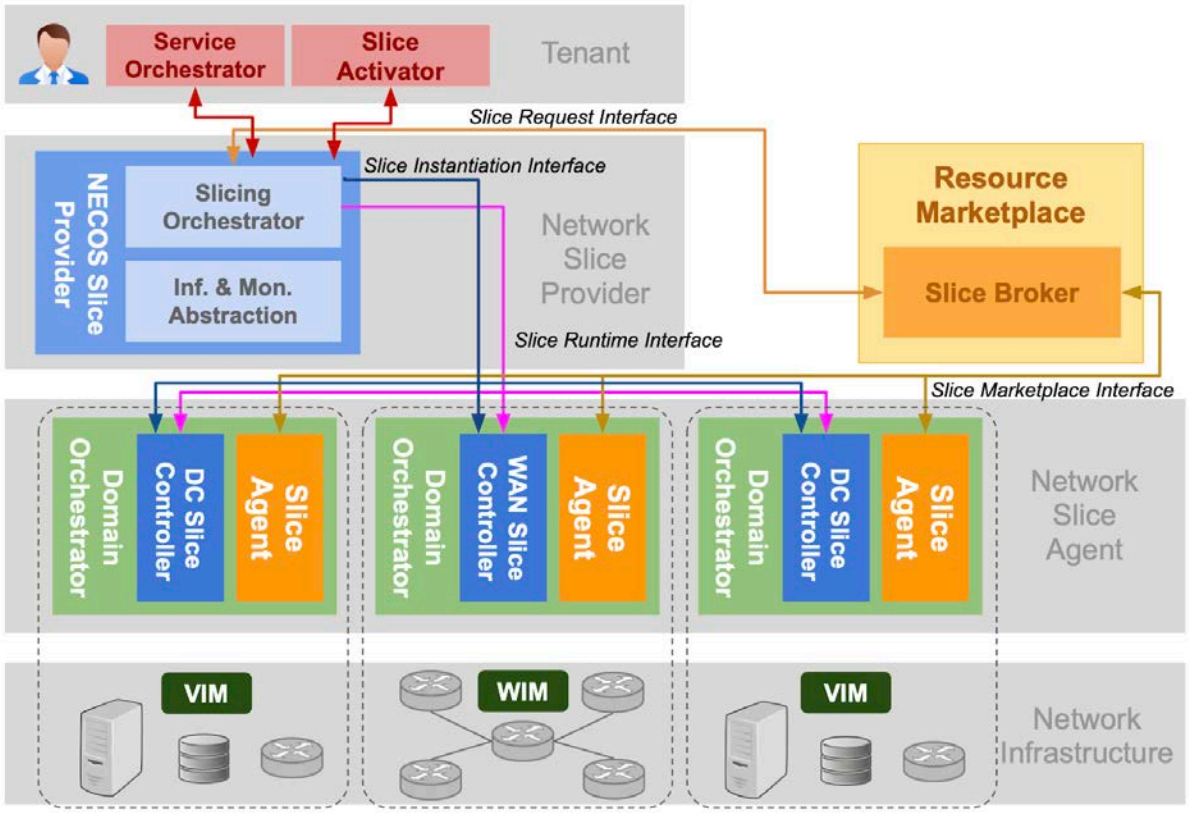


Figure 24: Mapping of NECOS components in Slice Reference Framework [SC10]

The creation of an end-to-end slice is one of the most important operation in frameworks dedicated to cloud network slices. In NECOS, the creation of an end-to-end slice involves all the NECOS sub-systems, as illustrated in Figure 24.

The creation of an end-to-end slice starts in the tenant domain, by the tenant running their own Slice Activator component. This component allows a Slice Description to be sent to the NECOS platform.

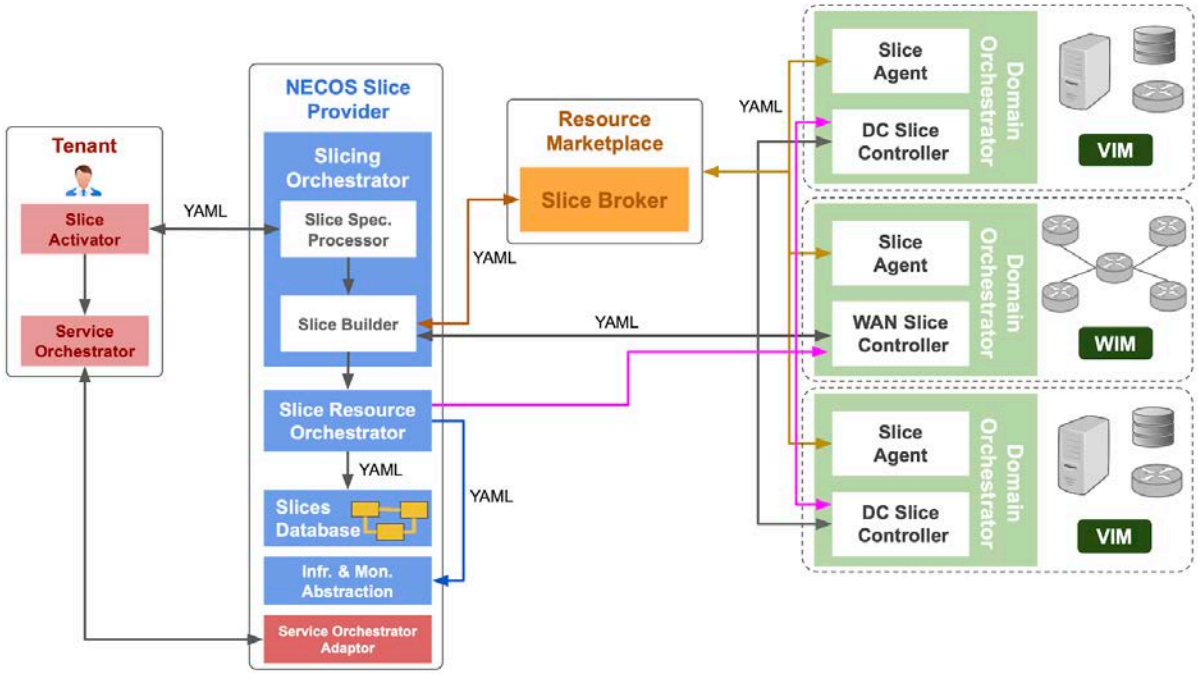


Figure 25: Interactions among NECOS components for creating the slice

The slice descriptor contains constraints to be applied to the end-to-end slice (such as locations, number of slice parts) and the specification of the VNFs to be hosted and their resource requirements in the form of *Virtual Deployment Units* (VDUs). This descriptor is passed to the Slice SPEC Processor which decides on the slice graph, the number of slice parts and VDUs residing in each part.

The descriptor generated by the Slice SPEC Processor is passed to the Slice Builder component, which forwards it to the Marketplace via the Slice Broker assistance. As will be described in Section 6.5, the Slice Broker decomposes the descriptor it receives from the Slice Builder and creates a different query for each specified slice part. The Slice Broker than proceeds with the collection of all alternative responses and responds back to the Builder. Responses for each alternative slice part (dc-slice and network-slice parts) are lists of alternative resources along with cost and other information so that the Slice Builder becomes capable to select which configuration of the slice best matches the tenant's needs.

For each selected slice part, the Slice Builder forwards a slice part descriptor to the corresponding DC and WAN Slice Controllers, so that they can afford the required resources reservations. In each domain, the DCs/WAN Slice Controller allocates the required resources, deploys, and configures an on-demand VIM/WIM. The DCs/WAN Slice Controllers return to the Slice Builder the details of each slice part, the VIM/WIM entry-point, and the monitoring end-point associated with them. This knowledge is forwarded to the Slice Resource Orchestrator that updates the Slice Database with the allocated computing resources and network topologies accordingly. The SRO in turn, interacts with the DC/WAN Slice Controllers to complete the creation of the end-to-end slice by interconnecting each VIM/WIM instance to its closest domain network edge. Finally, the SRO informs the IMA about the allocated VIMs, WIMs, and monitoring endpoints.

The IMA configures the adapters required for VIMs, WIMs, and monitoring endpoints in each resource domain. Lastly, the end-to-end slice is setup and the SRO informs the Slice Activator. The latter triggers the Service Orchestrator to start the service deployment over the created slice.

6.3 VIM on-demand

The NECOS architecture will provide automated provisioning and slicing of cloud resources which comply with customer requirements, as well as adapting them at run-time, in response to the system dynamics. Having Slice as a Service will be an advance in the state-of-the-art of current cloud computing architectures. This requires advances in heterogeneous resource management, federation and orchestration of compute, storage, and mobile entities in the form of slices that will enable an isolated, secure and scalable delivery of multi-cloud distributed services.

As a fundamental concept of Data Center (DC) slicing within the full NFVI foundation, we have to ensure that the attributes prescribed to network slices are propagated into the DC. We make a case for creating a VIM on-demand and dynamically allocating a new VIM (Virtual Infrastructure Manager) for each slice, rather than having one for the whole DC, which can be beneficial for those scenarios. There are some scenarios in which it is important to have a separate Data Center slice within a full Network Slice.

6.3.1 DC Slicing

In this section, we present an overview of some of the mechanisms, components, and abstractions that can be utilized in order to encompass network slicing into a bigger picture for NFV delivery.

The DCs and the networks are physically connected. Slices can be requested from networks. This is on-going work in many arenas such as IETF and IEEE. Slices should also be a feature that can be requested from DCs.

6.3.1.1 DC Slice

A DC slice is an abstraction over the resources of a DC

- It presents a collection of resources that look like a DC
- It can be controlled and managed independently from any other slices

A DC slice can be allocated at any Data Center: large central DCs, medium DCs, and edge DCs.

A slice is a basis for control in virtualized environments:

- a DC slice needs to be as elastic as other elements
- it can grow or shrink dynamically under the control of a Slice Controller

For each slice, there will be an on-demand VIM allocated for any kind of lower level virtualization, including Xen and KVM, or for containers such as Docker / Kubernetes. As a consequence, this choice is not a feature pre-determined once by the DC or the provider, but can now be an option for the customer. The customer will have a lot more configuration options for their VIM. It also means that the customer could also be billed for their VIM as opposed to it being part of the shared infrastructure.

6.3.1.2 VIM on-demand Deployment

A VIM needs to be allocated for each slice. We need a management component in each domain which can allocate a slice. A slice owner can manage, configure, and control their own VIM. The actual VIM that is deployed could be one of many: e.g. OpenStack, OpenNebula, OpenVIM, Kuberbetes. The VIM will be chosen from a pre-determined catalogue of VIMs that the DC owner has. As a VIM can be its own distributed system we need to decide how and where to deploy the VIM components.

6.3.1.3 Structural View

Figure 26 to Figure 29 show the structural view of how a DC can be sliced. We see it from the viewpoint of a single Data Center. These DC slices are then composed into a single multi-domain topology to form a NECOS slice.

6.3.1.4 One VIM Per DC

This approach has one VIM for all of the resources of the DC. This is the current situation in DCs. The VIM gets more and more functionality in order to do more or different tasks.

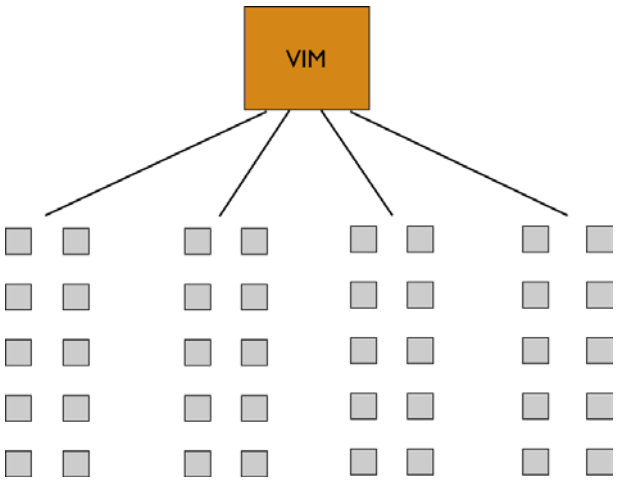


Figure 26: One VIM per data centre

This approach has one VIM for all of the resources of the DC. This is the current situation in DCs. The VIM gets more and more functionality in order to do more or different tasks. The VMs for the NFVs are scattered across the infrastructure, and each slice is inter-mingled with the others. The attributes of the network part of the slice do not apply in the DC.

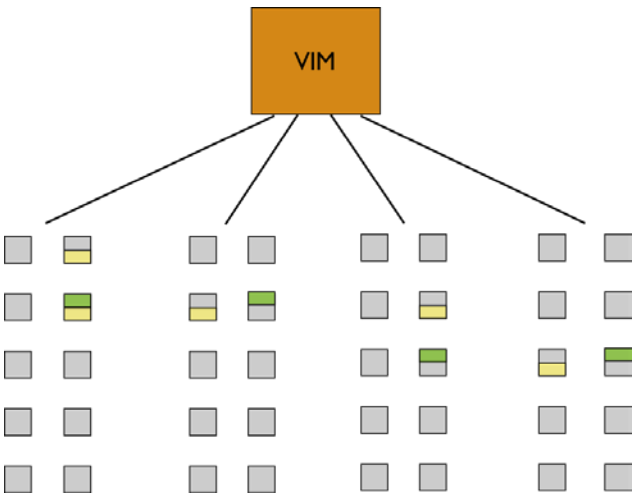


Figure 27: One VIM per DC with scattered NFVs

6.3.1.5 One VIM for all slices

The approach of one VIM for all of the resources of the DC has many issues when we introduce slices. It forces all of the slices to have the same strategies and policies, as there is only one VIM. The owner of each slice does not have the flexibility to control the slice how they wish, with different placement strategies of different approaches to energy management. The VIM needs to be even more complex to deal with this.

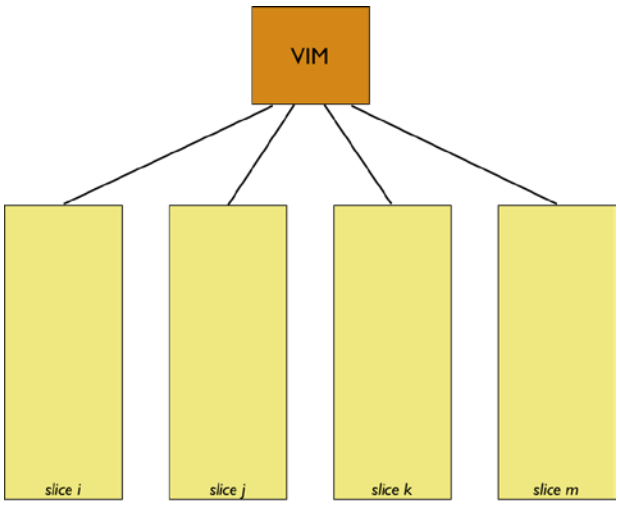


Figure 28: One VIM for all slices

We have found an approach that overcomes these complexities, and allows for building modular and scalable slices.

6.3.1.6 One VIM per Slice

This approach has many VIMs - one per slice. Each VIM can have its own independent strategies and can be managed differently from the other slices. The slice owner can configure the own VIM as needed for their use. We need to ensure that each slice is isolated from the others, and that a VIM cannot manage the wrong resources. The one VIM per slice is a basic premise of our approach.

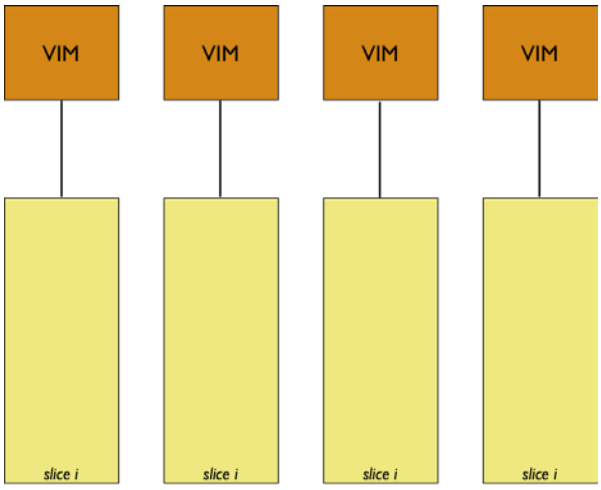


Figure 29: One VIM for each slice

A VIM is not special, and there is no requirement to have only 1, it is just another piece of software and can be started and stopped at any time.

6.4.WIM on-demand

Similarly, to the option considered for the tenants of being able of instantiating on demand a VIM for managing and control the cloud infrastructure allocated when providing a slice, it could also be considered the need of controlling and managing the connectivity paths interconnecting such slices when spread across several locations. Then the concept of WIM-on-demand arises naturally as a complementary element to be offered to the tenants, of required for running their services.

This section explores the concept, the expected functionality to be offered, as well as some of the possible scenarios and their requirements.

6.4.1 Functionality expected from a WIM

The functionality of a VIM is defined in [ETSI NFV]. The following list highlights relevant expected capabilities:

- Orchestration of the **allocation / upgrade / release / reclamation of NFVI resources**, and also their optimization
- Management of the **association of the virtualised resources to** the **physical** compute, storage, networking resources.
- Management of VNF Forwarding Graphs (create, query, update, delete), e.g. by **creating and maintaining Virtual Links, virtual networks, subnets, and ports**
- Management of security group **policies** to ensure network/traffic access control.
- Maintenance of a **repository inventory** with related information of NFVI hardware resources (compute, storage, networking) and software resources (e.g. hypervisors)
- **Management of the virtualised resource capacity** (e.g. density of virtualised resources to physical resources), including **usage reporting.**
- **Collection of performance and fault information**.
- Management of **catalogues of virtualised** resources.

By analogy, we could expect from a WIM similar behaviour with respect the connectivity among sites in order to facilitate the interconnection of the slices. The VIM will manage the internal connectivity to the slice, while the WIM will be in charge of the inter-slice connection. In order to stitch all the elements end-to-end, VIM and WIM should interact.

Then, as expected functionality to be provided by the WIM we could consider:

- The allocation / upgrade /release / reclamation of WAN resources (i.e., paths / connectivity) necessary for the interconnection of slices from different data centres, including their optimization
- Association of the virtualised resources, in this case interconnection paths, to the physical networking resources, which can refer to MPLS LSPs, GRE tunnels, VxLAN connections, optical lambdas, etc.
- Management of an inventory keeping track of the connectivity resources allocated in a per tenant way, in the form of a catalogue of resources and connectivity.
- Capacity management of those interconnection paths and the reporting of their usage, in a per tenant way.
- Collection of performance and fault information, facilitating the monitoring and alarm management of such interconnections.

The final objective by the tenant would be to dictate actions such as re-routing traffic among slice sites, dynamic establishment of paths, dynamic management of bandwidth, etc.

6.4.2 Is there a unique kind of deployable WIM?

Depending on the kind of slice requested by the tenant, it is possible that all of these capabilities could not be necessary for all kind of tenants. This can be related to the type of slice requested (in line with the type of potential slices reported in Deliverable 2.1 [NECOS D2.1].

In consequence, different WIM on demand flavours could be foreseen:

- *Full WIM* flavour, where the tenant can have full control of the allocated connectivity resources. This is only feasible if the resources are dedicated to the tenant, in a manner of "network partition" where allocated resources are entirely under control, in order to avoid any kind of conflict and provide full isolation.
- *WIM agent* flavour, where the actual control remains on the central WIM of the network provider, offering a customized (even limited) view to the tenant's WIM agent. This is indicated when the connectivity resources are not dedicated solely to a certain tenant, but shared among a number of them. The WIM agent will interact with the central WIM, which actually has the capability of controlling, managing and operating the connectivity resources.

In scenarios where the necessary connectivity traverses multiple transport network providers, a full WIM should interact with infrastructures of multiple domains, whereas the WIM agent should interact with a number of WIMs (one per domain). However, slices that require the use of IoT could also be feasible a situation where the tenant's WIM could act differently with each of them. That is, mixed scenarios could be feasible where the tenant's WIM could act in some domains as full WIM for directly handling connectivity resources in some domains, while behaving as WIM agent in others, where such direct control is not possible. In any case, the expectation would be the instantiation of a unique WIM on demand, offering the same appearance to the tenant but acting in a different manner towards the supporting domains.

6.4.3 Problems of having multiple WIMs with full control on the same network infrastructure

Multiple tenants interacting on the same transport infrastructure resource with full control can produce inconsistencies on the configuration and behaviour of the transport network. This can be due to the population of contradictory actions from different tenants, reactive measures to actions from other tenants, etc. In summary, lack of guarantee on the necessary isolation expected in a slicing environment. Such isolation can only be totally guaranteed except if dedicated resources are allocated to each tenant that needs complete isolation. However, the deeper we go on the approach of dedicating resources, the far we stay from the slicing concept

Because of that, it is necessary to clearly understand the control needs of each tenant in order to provide the more convenient type of slice and the corresponding level of control on it and the associated transport network. The two extreme situations could range from a dedicated allocation of resources (network partition) in one hand, to a simple, logical assignment of static inter-site capacity (network overlay) on the other hand.

6.4.4 Scenarios of deployment for WIM-on-demand

The two envisaged scenarios for WIM on demand are depicted in Figure 30 and 31. Figure 30 presents the case where the WIM on demand behaves as a full WIM, with total control on the allocated resources in each domain. In contrast, Figure 31 depicts the scenario where the tenant's WIM behaves as a WIM agent, interacting with a central or master WIM which is the actual entity having full control of the underlying transport architecture. The former can be seen as NECOS "Mode 0" for WAN control, whereas the latter could correspond to NECOS "Mode 1".

6.4.5 Requirements for full WIM and WIM agent

In order to consider WIM on demand as an operational solution, some requirements have to be taken into account. For instance, in the case of the full WIM, it is necessary to have standard South-bound Interfaces (SBIs) from the WIM towards the transport network infrastructure, in order to easily

integrate (on demand) the control plane represented by the WIM with the data plane lying in the transport infrastructure. Examples of that are the device models being defined in IETF [IETF3]. Also, for each transport network provider, it is necessary to dispose of some mechanism or artefact for allocating dedicated infrastructure to the tenant.

In the case of the WIM agent, it is necessary to offer a standard North-Bound Interface (NBI) from the centralized WIM in each transport network domain for allowing multiple agents running on top, without ad-hoc integration efforts. As an example of standard NBI, the ONF Transport API (TAPI) could be considered [ONF2017]. Complementary to that, and depending on the tenant's needs for network control, isolation mechanisms must be in place to avoid affection from one tenant to another.

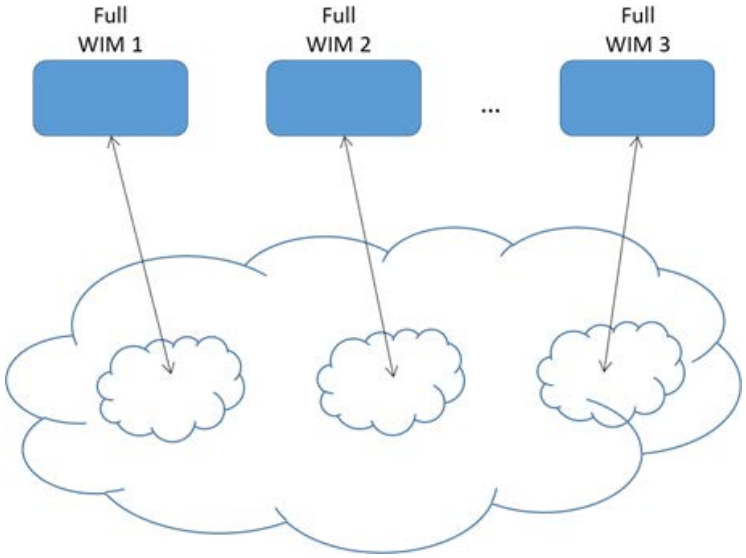


Figure 30: Full WIM scenario

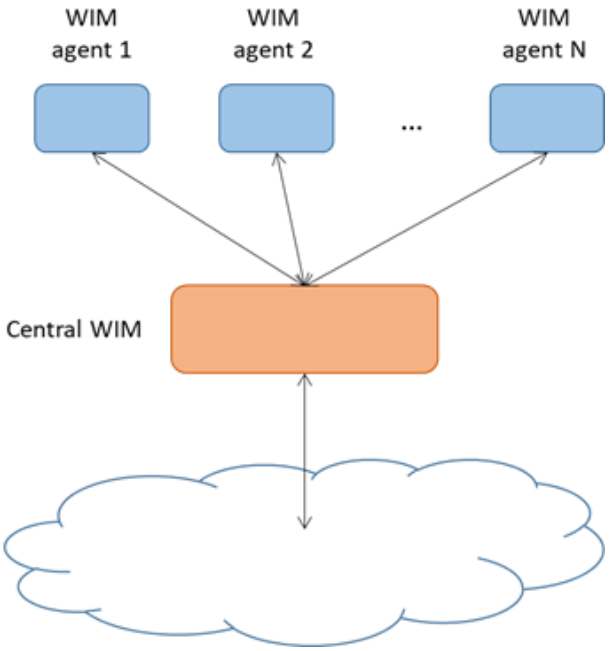


Figure 31: WIM agent scenario

6.5 Resources Marketplace

The Marketplace is an important aspect in NECOS, and creates a mechanism for dynamic discovering of resources, as well as creating a platform for deployment of different economic models in order to leverage business participation. In this subsection, we present a better description of the Marketplace components and interactions. Moreover, we explain how business can exploit the Marketplace via slice provision models.

6.5.1 Main Components - Marketplace

The Marketplace-based slice resource discovery approach is an essential feature of the NECOS platform. As already mentioned, several infrastructure providers can participate in the slice resource Marketplace, by offering resources dynamically to slice owners or tenants with specific requirements. So, the Resource Marketplace component is a fully-distributed system responsible for locating suitable slice parts on behalf of the tenants, from a set of participating resource domains.

To summarize the slice creation process in NECOS briefly, the Slice Orchestrator requests from the Resource Marketplace the different slice parts to be included in an E2E slice; and stitches the allocated slice parts into a single aggregated slice. To discover the needed resources, the Resource Marketplace component Slice Broker interacts eastbound with the Slice Builder, the latter being the NECOS sub-component between the Slice Orchestrator and the Resource Marketplace. The Slice Broker discovers these resources by interacting with a set of Slice Agents hosted by the resource domains involved. Figure 32 depicts the resource discovery workflow for further discussion.

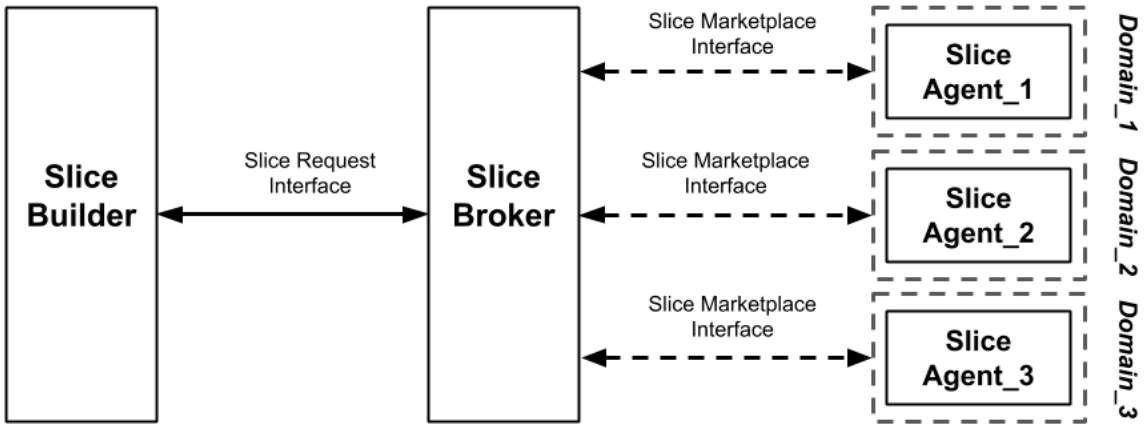


Figure 32: An overview of the resource discovery workflow

The resource discovery framework is responsible for locating the appropriate resources that compose a slice. These resources are slice components that correspond to service functions and service links, according to a specific information model. The Marketplace utilizes the NECOS information model, which enables interaction between the different stakeholders and, therefore, is also an important resource discovery feature that must be taken into account. Briefly, a Partially Defined Template (PDT) message acts as the input to the resource discovery framework and defines the general slice requirements. The Slice Specification Processor originates this message and passes to the Slice Builder which is closely coupled with the former. The Slice Broker searches for resources from resource domains to fulfil the slice requirements and prepare a corresponding response. This response is named Slice Resource Alternatives (SRA) message which, in practice, annotates the PDT message with alternative slice component options. More precisely, the aforementioned process employs the following three architectural functional components of NECOS:

- The *Slice Builder* is responsible for initiating the slice resource discovery process, by forwarding the PDT message to the Broker, and selecting the most appropriate slice

components, among alternatives returned by the latter in the form of an SRA message. The PDT consists of DC slice parts, annotated with computing resource constraints, and WAN slice parts that describe the desirable connections among providers.

- The *Slice Broker* is responsible for decomposing the template it receives from the Slice Builder and creating a different query for each slice part. Given the structure of the PDT message, the message decomposition can be performed easily, since each slice part corresponds to a different resource provider. The Slice Broker has all the necessary information to form query messages that contain all the constraints, preferences, and resources needed for the component request message. Once the PDT has been decomposed and submitted, it proceeds with the collection of all alternative responses and responds back to the Builder. Responses for each alternative slice part (dc-slice and network-slice parts) are lists of alternative resources originating from the providers Slice Agents. Since each agent supplies references to the offered slice parts, along with cost and other information, the Builder is in a position to select which configuration of the slice best matches the tenant's needs.
- A set of *Slice Agents* that reside on the providers' domains. The role of an agent is to answer requests for resources originated from the Slice Broker and regard specific dc- or wan-slice part. This message is translated into a form that the corresponding DC or WAN provider can process, and the answer received from the controller is passed to the Slice Broker.

6.5.2 Marketplace Interactions

Figure 33 expands the Marketplace architectural view and details the interactions among components, realized through messages exchanged during the slice resource discovery process.

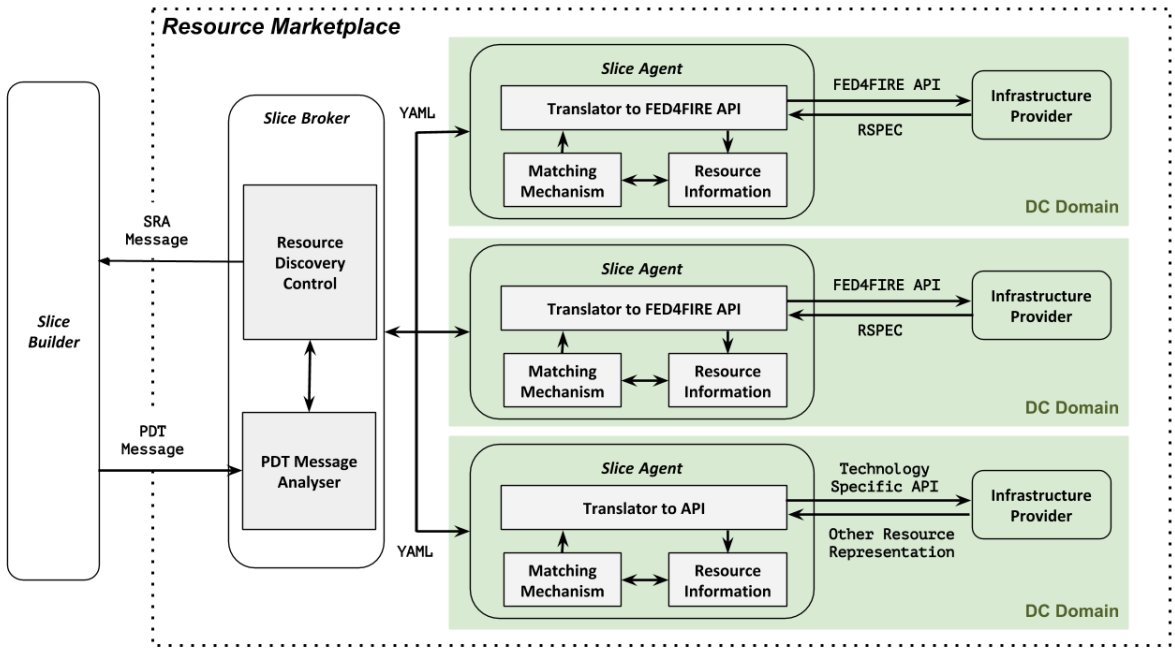


Figure 33: Interactions among the Marketplace components

Upon the reception of the PDT message, the Slice Broker decomposes it to a number of requests, one for each slice part. Such a request must contain all the necessary information about resource characteristics' demands for each part, which are not all defined in the same section of the PDT message. This happens because different components refine the original tenant's request for a slice, e.g. *Slice Activator* is responsible for specifying characteristics such as EPA attributes, where the *Slice*

Specification Processor decides on the slice graph, with the number of slice parts and *Virtual Deployment Units* (VDUs) residing in each part.

The role of the *PDT Message Analyser* component is to aggregate all that information, in order to format a complete request message in terms of necessary demand information. Follows an example slice part request for a particular DC-VDU (i.e., a web_server_VM):

The *PDT Message Analyser* collects all the necessary information in the request message for the resource discovery by filtering out the unneeded data. The output is directed to the *Resource Discovery Control* component to initiate the discovery process.

The *Resource Discovery Control* component inquiries all agents in the marketplace for each slice part and collects all the responses. This component is also responsible for forming all alternative slices (i.e. the slice part allocations to the different providers).

To retrieve the real-time status from the resource domains, the Slice Agent is equipped with a *Translator* component which communicates directly with each domain (e.g. DC domain), using the specific API for that domain such as the FED4FIRE API. The resource domains respond with the status of their available resources in an RSpec format [GENI]. The *Translator* is also responsible for translating the response message into a more consistent format, in compliance with the NECOS information model. After that, it forwards the message to the resource *Matching Mechanism* component. With the task of discovering which cluster can accommodate each slice part, the *Matching Mechanism* component translates the request originated by the Slice Broker to a set of resource availability constraints the part should satisfy. As an example, the previous message is translated to the following:

This whole interaction process allows a simple matching process that discovers all DC-VDUs a provider can host in a node cluster. The output forms a (simple) resource allocation problem (i.e. DC-VDUs to node cluster's) that the Slice Agent solves to answer positively to the request. Although a simple backtracking process is employed to avoid overwhelm a cluster with more DC-VDUs it can manage due to node availability constraints, a constraint logic programming approach could solve the problem much more efficiently, by minimizing certain optimization criteria stated by the infrastructure provider. Upon a query success, the Slice Agent fills the PDT request message with its offering and responds it back to the Slice Broker.

6.5.3 Slice Provision Models

Cloud network slicing is getting attention due to the possibility of diverse offerings in terms of services and requirements, standing as one of the main concepts of the upcoming 5G networks, in which several business models are expected in order to provide cloud network slices as a service for on-demand vertical clients. These business models will allow for the deployment of different types of marketplaces.

The NECOS architecture is expected to support the implementation of different types of marketplaces, as for example:

- Telecoms Marketplace, which is a limited setup between close co-operating telecoms providers;
- Contract Marketplace, which is a setup between partners with a level of trust based on signed contracts;
- Open Marketplace, which allows any provider to offer resources and register their offering for users.

Moreover, the deployment of the Marketplace is expected to be dynamic enough to support diverse types of federations and/or resources being traded, such as network or computing. Provision models play an important role in this kind of environment and a more in-depth analysis of how those models can be applied in the NECOS Project is highly desired.

According to the Internet-Draft "Network Slice Provision Models" [I-D.homma-slice-provision-model], the provision models for network slicing are similar to the basic ones adopted in the cloud computing market. More specifically, they are categorized into the three basic *-as-a-Service (*aaS) provision models: Software-aaS, Platform-aaS and Infrastructure-aaS. Certainly, it affords hybrid models for the provisioning of resources and share the same abstraction levels established in the cloud computing market, as explained below.

In a SaaS-like provision model, a network slice provider designs models in advance. The tenant is responsible for selecting and using the one that fits mostly its requirements.

The network slice specifications are abstracted in the form of KPIs and detailed parameters of the infrastructure are not exposed to the tenants.

In a PaaS-like provision model, the tenant exposes its request and the network slice provider creates a model to be instantiated dynamically based on this request. The tenant's request may include technical parameters such as connected area, paths, KPIs and service functions.

In an IaaS-like provision model, the tenant is responsible for designing his own models and instantiating the network slices. The infrastructure operators only provide the resources indicated by the tenant, who is also responsible for coordinating these resources.

As mentioned before, a hybrid scenario can be provided as, for example, by a broker coordinating a PaaS type model in a SaaS-like manner. This way, the broker acts recursively and becomes a network slice provider as well.

The several interaction/federation models need to be considered for the Marketplace components' workflows, which will provide a better understanding on how to develop the internal components and what type of situation a specific type of Marketplace should be deployed.

6.6 Elasticity and Scalability

6.6.1 Elasticity

Elasticity is one of NECOS's Critical Success Factors presented in Deliverable D2.1. In general terms, the concept of elasticity can be found in several standard organizations, projects, and papers with similar definitions, as detailed below. A common synonymous to elasticity is auto-scaling and is thoroughly used by the cloud computing platforms.

Several standard organizations have somehow defined the concept of elasticity. The NGMN mentions that actual physical resources and their configuration may vary with time, including on-demand allocation or scaling [NGMN 2016]. The ITU-T defines the slice as an entity that needs to be dynamic. The slice lifecycle management of ITU-T includes the elasticity function which should be capable of giving or removing physical and virtual resources to a slice [ITU-T 2016]. ONF clearly defines that the resources may be sliced dynamically to attend the requirements of the clients. There is a controller responsible for continually adapting the resources based on the load and on the policy constraints [ONF, 2016].

As already defined in D5.1, for the NECOS Project, elasticity is defined as the degree to which a system can adapt to workload changes by provisioning and de-provisioning resources (computing, networking and storage) in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible. More than that, in the NECOS Project, this elasticity considers that the slice is provisioned in a multi-domain and multi-technological environment and the run-time change of resources as such requires a sophisticated mechanism for orchestration. As such mechanisms and triggers of elasticity include: computational-driven elasticity, connectivity-driven elasticity, orchestration-driven elasticity, service-driven elasticity and slice-driven elasticity.

We have also defined vertical and horizontal elasticity in line with the literature for resource virtualization [ETSI, 2019] [ONF, 2016]; in NECOS, we define vertical elasticity as the ability to resize slice parts dynamically as needed to adapt to workload changes. Horizontal elasticity, in

change, is the ability to create or remove slice parts dynamically, using resources of other(s) provider(s), following the need to adapt to the workload evolution. For example, as the service workload increases and the resources available for its supporting slice at a particular data centre are not enough to cope with the needed computing power, it may be possible to scale out resources creating new slice parts (e.g., a new on-demand VIM) in another data centre and connecting them appropriately (e.g., creating a new networking slice part).

As described in [Al-Dhuraibi] for the cloud computing realm, both scalability and efficiency are associated (although not equivalent to) with elasticity. Scalability is the ability of the system tosustain increasing workloads by making use of additional resources [Herbst] it is time independent and it is similar to the provisioning state in elasticity but the time has no effect on the system (static property). Those authors explain this interrelation with the following equation that summarizes the elasticity concept in cloud computing.

$$Elasticity = \underbrace{scalability + automation}_{\text{auto-scaling}} + optimization$$

Figure 34: Elasticity and Scalability, (source [Al-Dhuraibi])

It means that the elasticity is built on top of scalability. It canbe considered as an automation of the concept of scalability (defined in more detail bellow), however, it aims to optimize at best and as quickly as possible the resources at a given time.

Elasticity stands as an analogous concept in a network cloud slicing environment, such as NECOS's, it is an analogous concept. It includes not only to dynamically adapt computing plus network resources, but also to adapt the target network services accordingly. That means, for example, to migrate VNFs and all the related connectivity in the case of a horizontal scale-in action, or to migrate critical service functions such as a load balancer.

With regards to adapting network resources, there are some decades of research and deployed protocols and technologies on how to adapt network resources to variable traffic patterns. From QoS routing to Adaptive Traffic Engineering, there are enough related work to adapt for the particular case of NECOS slices. The challenge in this case, resides in orchestrating those autonomous procedures and protocols with more pure cloud computing elasticity and service assurance techniques.

The elasticity-model described above affects the view that the tenant has of the sliced system. Since NECOS adapts the volume and location of resources following platform-related variables, the main idea of this process is to present the slice as a system that is stable and isolated in terms of those variables.

6.6.2 Scalability

As stated in Deliverable D2.1, although there are several definitions of scalability, for the scope of this project we define scalability as the ability to increase workload size within existing infrastructure (hardware, software, etc.) without seeking to impact performance. We called the attention on two different dimensions of scalability in the context of NECOS: scalability of a particular provisioned slice and scalability of the number and size of the slices provided.

The main tool that NECOS has by design to cope with scalability issues is its distribution. NECOS has been designed as a distributed system, each component in its architecture is designed as an independent service that may actually be implemented as a set of different servers dividing the management load geographically, by client, or by service. Even those components that seem to be unique, such as the SRO, may have several instances (one per entry slice provider, for example) or, in the extreme case, one per slice. This is particularly important in the monitoring and auto scaling functionalities where the volume of data may be enormous and the reaction time must be near real-time.

6.6.3 Architectural impact of elasticity and scalability on NECOS

As foreseen in Deliverable D3.1, the main functionalities related with elasticity and scalability are commanded from the Slice Resource Orchestrator and the Service Orchestrator, and enabled and executed by most of the NECOS components. In Figure 35 we depict the two control loops introduced in Section 6.1as are implemented on the NECOS architecture. In green, it is possible to see the platform elasticity loop: platform-related monitoring data such as CPU usage or performance/temporal isolation are monitored by tools at the resource provider. That information is processed by the IMA and left in a time-series database at the disposition of the SRO. The SRO decides if the slice needs a different configuration of resources, asks the Slice Builder to choose them, and deploys them using the domain-specific Slice Controllers and the IMA. In *red* we depict the loop related with the service elasticity. The service specific monitoring data is collected by the tenant, its service orchestrator decides if the slice needs more resources, and it asks NECOS' Slice Resource Orchestrator to add or remove or relocate resources.

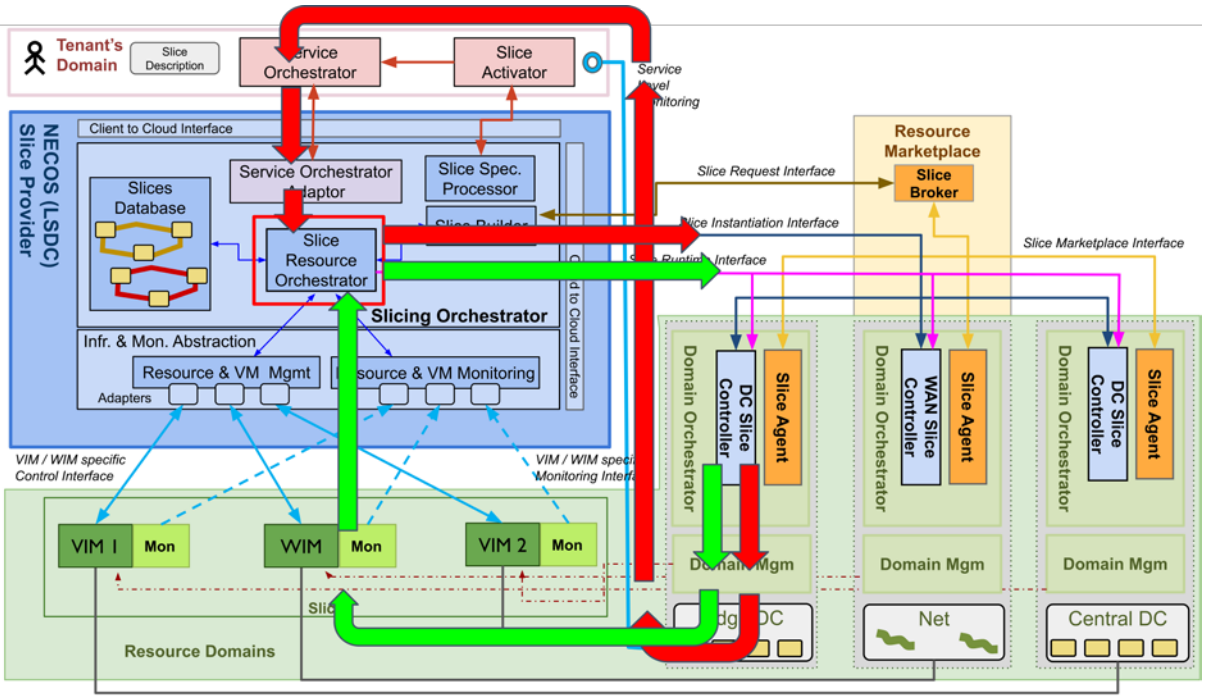


Figure 35: Two control loops for elasticity in the PaaS-like provision model

In Figure 36 we depict the internal architecture of the SRO. It follows the same distribution idea and each internal component is an independent service; they are organized following the workflows provided by Deliverable D5.1 and are organized in three main functionalities: the *SRO Core*, in charge of common tasks and external interactions, the *Slice Instance Management*, in charge of the slice lifecycle tasks, and the *Slice Intelligent Run-time Management* which is the internal component in charge of implementing the elasticity functionalities described above.

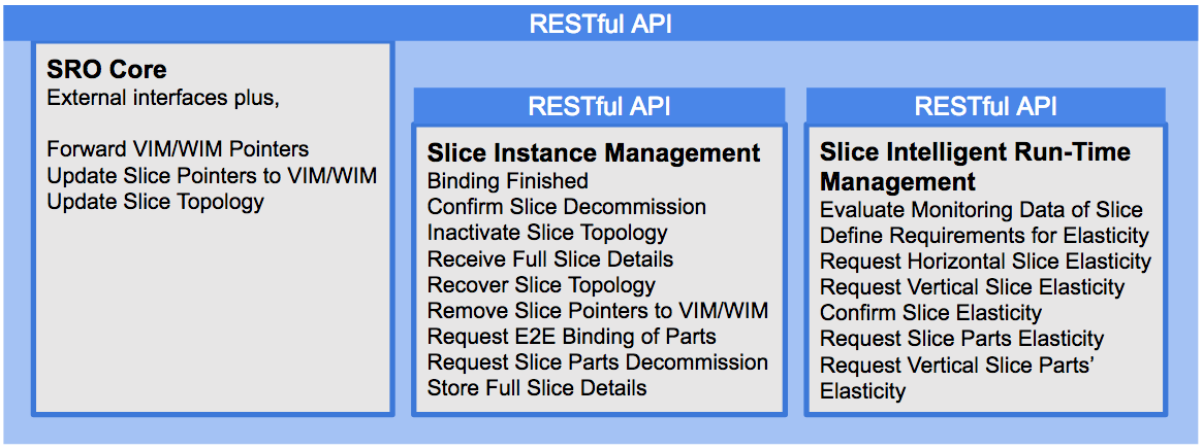


Figure 36: The internal SRO architecture. It is organized in such a way that all elasticity-related functionalities are together and detachable as a plug-in

6.7 Infrastructure Monitoring

The purpose of the Infrastructure & Monitoring Abstraction (IMA) component is to provide a uniform abstraction layer, above the heterogeneous VIM / WIM and monitoring subsystems that are part of an end-to-end network slice. Different Slice Parts will constitute the end-to-end slice, and each part can potentially rely on a different underlying technology (e.g. specific VIM / WIM and monitoring subsystem implementations), and therefore have different adaptors.

The aim of IMA is to provide an abstract interface at its northbound to allow the SRO performing its functions without taking care of the implementation details of each Slice Part.

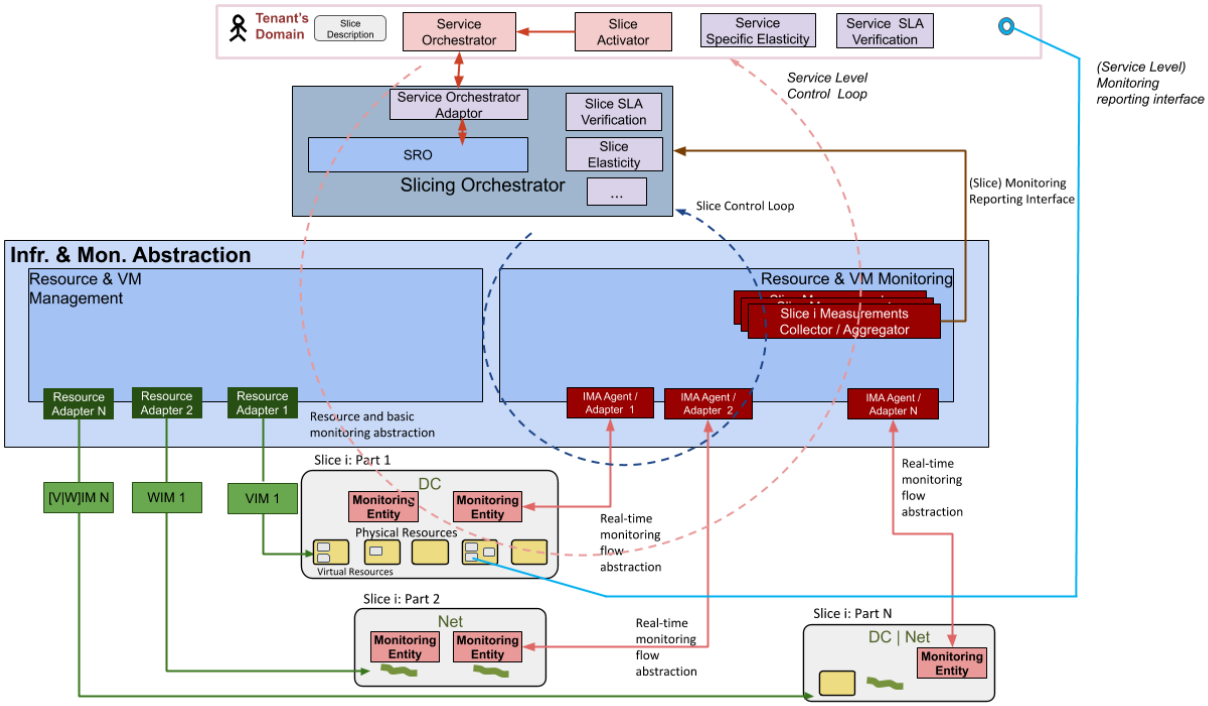


Figure 37: Infrastructure and Monitoring Abstraction (IMA) architectural blocks

Figure 37 shows the two main IMA subcomponents, each one dealing with the implementation of the abstraction mechanisms required to perform the NECOS functions, namely:

(i) Resource and VM Management

(ii) Resource and VM Monitoring.

A detailed description of the internal blocks of each IMA subcomponent is provided in the following.

6.7.1 Resource and VM Management

The Resource and VM Management stands as the IMA sub-component responsible for providing the abstract API to be used by the SRO to interact with the VIM / WIM inside the different Slice parts to perform both resource operations and VM management operations.

This subcomponent includes two main functional blocks, i.e., the Resource Engine / Controller and the Resource Adaptation Layer (i.e., Resource Adapters) and uses three interfaces in order to interact with the other NECOS' entities, i.e., the Resource Control Interface, the Abstracted WIM / VIM Interface and the Resource and Basic Monitoring Abstraction Interface.

6.7.1.1 Resource Engine / Controller

This is the functional component of the Resource and VM Management responsible for the instantiation, configuration and control of the different Resource Adapters that are requested at run-time (by the Slice Resource Orchestrator) to build the required abstraction on top of different Slice Parts. In order to allocate such an abstraction on-demand, the Resource Engine / Controller needs as input from the SRO the information on the types of VIM / WIMs of the end-to-end Slice, as well as their associated endpoints. This interaction (represented in Figure 37 via a dashed green arrow) will happen via the northbound interface API of the Resource Engine / Controller, namely the Resource Control Interface.

This component will also be responsible for decommissioning the on-demand instantiated abstraction layer at the end of the life-cycle of a given Slice Part, e.g., when a Slice Part is no longer needed or the whole end-to-end Slice has been deallocated. In a similar way, it will deal with the on-demand re-configuration of the Resource Adaptation Layer when an elasticity workflow is executed and Slice Parts are added (removed) to (from) an existing end-to-end Slice. All the above-mentioned inputs to the Resource Engine / Controller will be provided by the SRO again using the Resource Control Interface.

6.7.1.2 Resource Adaptation Layer

This is a distributed adaptation layer based on the concept of on-demand allocated Resource Adapters. Each resource adapter is created in order to hide the specific technological implementation of a Slice Part, such as a running, on-demand created VIM or WIM. More specifically, at the southbound of a Resource Adapter, the specific VIM / WIM interface / API will be used to provide resource and basic monitoring abstractions to the SRO (as presented in Figure 37). This includes the status of the physical resources and their relationship, together with a map of how virtual elements are placed on the resources. This relationship does not give any real-time information, as such flows are handled instead by the Resource and VM Monitoring subcomponent of the IMA. The SRO will use the northbound interface exposed by each Resource Adapter (abstracted VIM / WIM interface in Figure 37) in order to interact with the different heterogeneous Slice Parts of the end-to-end Slice in a uniform way.

6.7.1.3 Resource Control Interface

This is the interface exposed at the north side of the Resource Engine / Controller toward the SRO to perform the dynamic instantiation, reconfiguration and control of the different adapters in the Resource Adaptation Layer. This interface / API provides the following main calls:

- *Start Adapter (type, endpoint): Adapter ID*
- *Stop Adapter (Adapter ID)*

- *Configure Adapter (Adapter ID, settings)*

6.7.1.4 Abstracted WIM / VIM Interface

As specific adaptors for particular VIMs and WIMs are required to support multiple technologies within an end-to-end Slice, the southbound interface of each adapter will be used to translate generic calls received by the SRO via the Resource and Basic Monitoring Abstraction Interface into particular commands or methods in the context of specific VIMs and WIMs. Typical types of VIMs and WIMs include:

- *Cloud adaptors: OpenStack, Heat, Kubernetes, VLSP;*
- *VNF adaptors: OpenMANO, Open Baton, OPNFV;*
- *Transport adaptors: SDN controllers, such as OpenDaylight, Floodlight;*
- *RAN adaptors;*
- *Edge adaptors.*

6.7.1.5 Resource and Basic Monitoring Abstraction Interface

This interface is exposed at the northbound of the Resource Adaptation Layer and can be utilised by the SRO as a uniform way to interact with the element of the different Slice Parts that have been allocated for an end-to-end Slice. In particular, this interface will support mechanisms that allow the SRO to retrieve an up-to-date topology view from each Slice part that can be used to build the global resource view of the end-to-end Slice. Moreover, it can be used by the SRO in order to check the status of the virtual elements allocated on a Slice Part for each embedded service instance (e.g., to detect any failures on the VIM).

The Resource and Basic Monitoring Abstraction Interface provides the following main methods to the SRO:

- *Get Slice Part topology (Slice Part ID): topology view*
- *Get Slice Part status (Slice Part ID): embedding view*
- *Instantiate Service Element (Type Element, Slice Part ID): service element ID*
- *Delete Service Element (Element ID)*

6.7.2 Resource and VM Monitoring

The Resource and VM Monitoring subcomponent of the IMA implements the set of abstractions required to provide the SRO with a homogeneous view on the resource utilisation of an end-to-end Slice via gathering monitoring information from the different Slice parts (i.e., from the specific monitoring systems deployed therein). The real-time data gathered from the Resource and VM Monitoring is then directly associated (by the SRO) with the structured global view gathered by the Resource and VM Management IMA subcomponent that was described earlier in this section.

6.7.2.1 Monitoring Engine / Controller

The Monitoring Engine / Controller is the functional component of the Resource and VM Monitoring that takes care of starting, configuring and controlling relevant IMA Agents / Adaptors in the Monitoring Adaptation Layer in order to hide the implementation details of the specific monitoring subsystems deployed in each Slice Part. An IMA Agent / Adapter is allocated / instantiated / configured for each Slice Part in order to gather measurements from the monitoring entities (e.g., agents, exporters, probes) and sending them to the relevant Slice Measurements Collector via a *measurements distribution mechanism*. The SRO will trigger the instantiation of IMA Agents / Adapters using the Monitoring Control Interface / API of the Monitoring Engine / Controller, and specifying both the type of monitoring technology to be abstracted and the monitoring endpoint as input parameters. As part of the end-to-end Slice monitoring layer, one (or more) Slice Measurements Collector / Aggregator will be instantiated on demand as soon as the allocation of the resources for an end-to-end Slice is completed.

6.7.2.2 Slice Measurements Collector / Aggregator

This is the component instantiated in order to aggregate / multiplex the monitoring information coming from different Parts of an end-to-end Slice. Although we expect one Collector / Aggregator to be associated with an end-to-end Slice, the number of Collectors / Aggregators can (dynamically) be scaled out to prevent the creation of bottlenecks when the aggregated rate of received measurements exceeds a given threshold (e.g., due to a large number of Slice Parts being active at the same time).

A Slice Measurements Collector will gather measurements from the IMA Agents / Adapters that have been allocated for each Slice Parts, will (re-)structure them according to a given data model (known by the SRO) and will finally send them out to the SRO on the Monitoring Reporting Interface. The interaction between the IMA agents and the Slice Measurements Collectors / Aggregators is not limited to a specific technological implementation, as the Measurements Distribution mechanisms depicted in Figure X-IMA can be implemented using different approaches, such as push/pull, pub/sub, intermediate Database, etc.

6.7.2.3 Monitoring Adaptation Layer

Similarly, to the Resource Adaptation Layer described earlier, the Monitoring Adaptation Layer also relies on the on-demand instantiation of adapters. In this case, the required adaptation will be implemented (per end-to-end Slice) by a distributed layer formed of IMA Agents / Adapters, each one interacting with the VIM / WIM and / or the monitoring entry points that were allocated and returned for each Slice part. The interaction with the above entry points will be strictly related to the particular technology being used for a Slice Part (see Real-time Monitoring Flow Abstraction Interface in Figure 37). Each IMA Agent will collect resource facing KPIs (e.g., CPU, Memory, Storage, delay, bandwidth, network I/O) of both the physical resource and the virtual service elements running on that VIM / WIM. Service-specific KPIs are out of the scope of this component and are left to specific monitoring solutions provided by the Tenant.

6.7.2.4 Monitoring Control Interface

This is the interface exposed by the Monitoring Engine / Controller to the SRO, in order to manage the on-demand instantiation of the monitoring abstraction layer for an end-to-end slice. As the specific technological implementation of each Slice Part will only be known at run-time, it is not possible to allocate IMA Agents / Adapters in advance. As such, after the successful allocation of an end-to-end slice, the SRO will interact with the northbound interface of the IMA Resource and VM Monitoring, namely the Monitoring Control Interface, to trigger the instantiation of the necessary components with the relevant adaptation logic.

The Monitoring Control Interface exposes the following functionalities:

- *Instantiate Agent / Adapter (Monitoring Type, Endpoint, [Slice ID], Slice Part ID, Report to Aggregator, [KPI list], data rate): Adapter ID*
- *Instantiate Measurements Collector / Aggregator (Slice ID, Reporting Type, Report to endpoint): Collector ID*
- *Stop Agent / Adapter (Adapter ID)*
- *Stop Collector / Aggregator (Collector ID)*

6.7.2.5 Monitoring Reporting Interface

This is the interface between the Measurements Collectors / Aggregators and the SRO. Although the interface can be implemented using different technologies / approaches (e.g., push/pull, pub/sub, intermediate database), the data model exchanged by the Slice Measurements Collector(s) and the SRO on the Monitoring Reporting Interface must be commonly agreed. The data model will include the following information:

- *KPI name*
- *Resource Type (e.g., Host, VM, link, etc.)*
- *Resource ID*

- *Slice Part ID*
- *Slice ID*
- *Value*
- *Timestamp*

6.7.2.6 Real-time Monitoring Flow Abstraction Interface

This is the interface at the southbound of the IMA Agents / Adapters which is specific for the particular type of VIM and / or monitoring subsystem running in a Slice Part. The type of monitoring information collected via this interface complements the resource state information collected by the Resource and VM Management components. While the latter provides feedback about the status of the virtual resources deployed on the slice (mainly to detect faults and check the consistency between the performed embedding operations and the actual run-time resource status), the former provides the actual real-time monitoring flows that will be used to check the performance of the physical and virtual resources elements of the slice. By combining both the structural data with the real-time flows, we get a clear view of the current status of all slice elements.

7. Interactions between the Functional Blocks

This section presents the interactions between the functional blocks within the NECOS architecture.

7.1 Slice Lifecycle

Slice is a dynamic entity that has its own lifecycle management and operation.

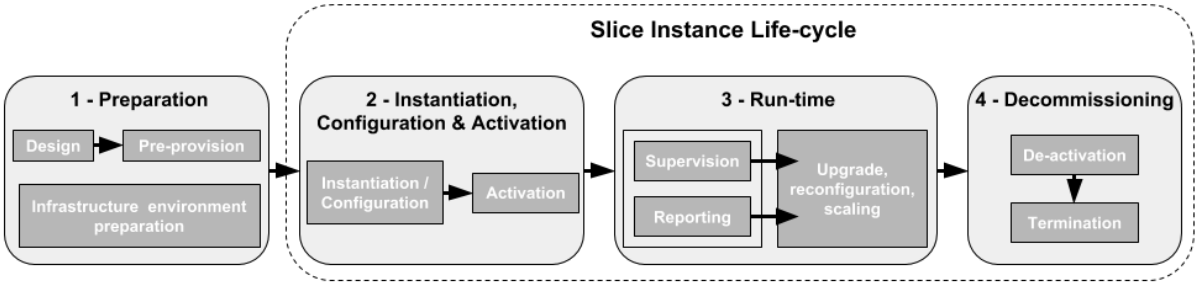


Figure 38: Lifecycle phases of a slice instance

The main tasks performed by the NECOS platform in each of the four phases are as follows:

1. Preparation Phase: this starts with the specification of slice templates performed by the Tenant; it is followed by the discovery and pre-provisioning of resources involving the Slice Builder, Slice Broker and Slice Agents via the Slice Request Interface and Slice Marketplace Interface; continues with the preparation of the allocated infrastructure for the instantiation and support of the slice performed by the Slice Builder interacting with the Slice Controllers via the Slice Instantiation Interface.
2. Instantiation, Configuration and Activation Phase: as soon as all the resources shared/dedicated to the slice instance have been created and are configured, the Slice Resource Orchestrator will complete and activate the Slice using the handles to the different Slice Parts received by the Slice Builder.
3. Run-time Phase: normal operation of the slice and monitoring of its performance (via the IMA), allowing the Slice Resource Orchestrator to perform the slice's reconfiguration or scaling as necessary using the Slice Runtime Interface.
4. Decommissioning Phase: deactivation of the slice, releasing its allocated resources is carried by the Slice Resource Orchestrator at the end of the Slice lifetime via the Slice Runtime Interface.

In the NECOS platform, each phase of the slice lifecycle will trigger actions on different components. These tasks should be coordinated by the Slicing Orchestrator in order to provide ways of handling workflows in scenarios of success or failure.

Each of these lifecycle phases is further broken down into sub-phases, in which a specific workflow will be performed by different components, as explained in the remainder of this Section. The operations related to each sub-phase will also be linked to the numbered steps (reported in square brackets) of the overall workflow depicted in Figure 38 (e.g., Design sub-phase in the Preparation is related to step 1 of the overall workflow).

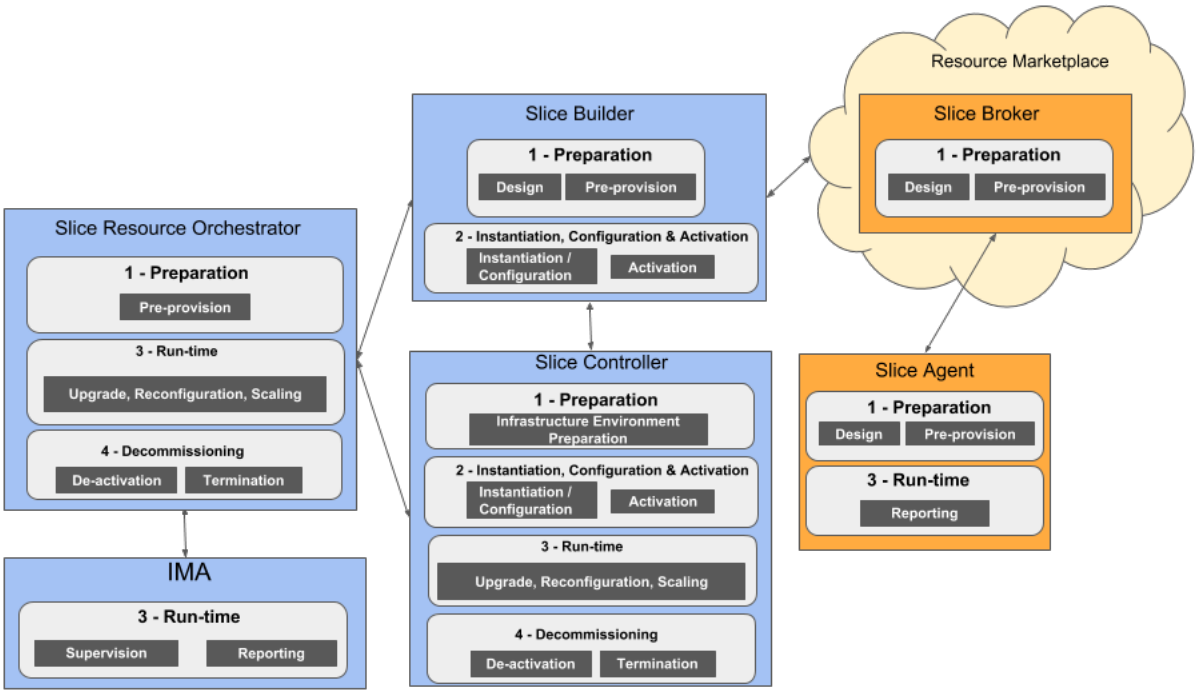


Figure 39: Mapping of slice lifecycle operations and NECOS components

7.1.1 Preparation phase

This phase starts with the specification of slice templates, followed by the discovery and pre-provisioning of resources, then the preparation of the allocated infrastructure for the instantiation and support of the slice, and the exchanging of agreement between tenants and providers. It is divided into three sub-phases:

- ***Design***

  The Slice Builder uses the slice blueprints/descriptors to design a new slice instance. Here the different slice parts are described as well as their features, configurations, resources needed and associated workflows. [Step 1]

- ***Pre-provision***

  The Slice Builder together with the Slice Broker and the Slice Agents makes an on-demand request of resources via the Slice Request and Marketplace Interfaces using admission control, resource negotiation and charging. Network resource availability, latency and resiliency, as well as polices, are taken in consideration before doing the allocation. [Steps 2,3,4,5,6,7]

- ***Infrastructure environment preparation***

  The Slice Builder uses the Slice Instantiation Interface to prepare all the infrastructure resources needed for the instantiation and support of the new slice to be created, via contacting the relevant DC or WAN Slice Controllers, with the *offer* from the related Slice agent. [Step 8]

7.1.2 Instantiation / configuration and activation phase

In this phase, all resources of the slice instance are created and configured and become ready for operation. These tasks occur in the Resource domains.

- ***Instantiation/Configuration***

The Slice Controller finalises the allocation of the resources for the Slice: the required infrastructure resources (shared and/or dedicated) are configured and instantiated but not yet activated. Allocating the VIMs. [Steps 9,10]

- ***Activation***

  The Slice Resource Orchestrator finalises the activation of the slice performing the steps required to complete its topology; any pending service deployment is processed; service instances are activated and becomes ready to start their operation. [Steps 11,12,13,14,15,16]

7.1.3 Run-time phase

In this phase, the slice instance is normally operating and has a feedback loop provided to the SRO via the IMA that is used to monitor the slice performance as well as the overall infrastructure needs.

- ***Supervision & Reporting***

  The performance and status of the Slices are monitored and reported to the SRO through the IMA component. According to the requirements of each slice and also to the overall status of the system, the SRO can take the decision of performing reconfiguration and / or scaling on (some of) the running slices.

- ***Upgrade, Reconfiguration, Scaling***

  If required, a slice instance is re-configured and / or scaled according to its specific requirements and to the overall status of the system. This process allows performing resource optimisation via slicing elasticity and is further detailed in the remainder of this Section.

7.1.4 Scaling and elasticity

Supported by the monitoring data, the management of the slice instance involves strategies to provide elasticity, adding or removing resources from the slice according to its workload, ensuring optimized service performance with efficient resource usage. These methods may be applied both by the NECOS Slicing Orchestrator, to add/remove or replace resources allocated to a given slice.

Next, we elaborate on the slice scaling options and present how slice elasticity can be delivered, which can categorized as presented in Figure 40.

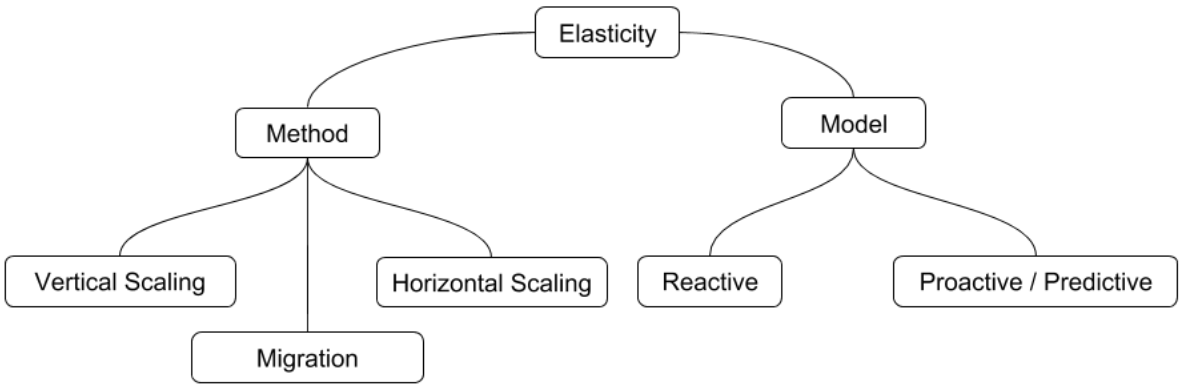


Figure 40: Categorization of methods and models of elasticity solutions

The 3GPP approach considers two scaling methods which, when adapted, can be performed by NECOS as follows:

- ***Horizontal Scaling***

  This scaling is appropriate when new Slice parts are needed or when an existing Slice part can be shutdown. To add a new Slice part, the Slice Builder / Slice Broker mechanism needs to be triggered to find the relevant part. When doing a shutdown of an existing part, the Slice

Resource Orchestrator can use the Slice Runtime Interface to tell the relevant DC Slice Controller to close the Slice part.

- ***Vertical Scaling***

  This scaling is appropriate for resources in existing Slice parts. It involves adding/removing resources from a slice part (e.g. computing, memory or storage) or network part. The NECOS Slicing Orchestrator might add/remove physical devices or resources from a slice part to better support its current demand. This is achieved by the Slice Resource Orchestrator using the Slice Runtime Interface to make a request to the relevant DC Slice Controller.

- ***Migration***

  For a slice, migration needs to be analysed and defined.

Two elasticity models regarding the triggering event can be defined:

- ***Reactive***

  The elasticity procedures are triggered by thresholds in resource usage or SLA violations, reacting to the current workload.

- ***Proactive / Predictive***

  Uses predictive mechanisms, possibly based on workload history, to trigger the elasticity procedures according to anticipated load.

7.1.5 Decommissioning phase

After a slice instance is no longer needed, the Slice Resource Orchestrator proceeds to finish the lifecycle of the slice.

- ***De-activation***

  The slice instance is deactivated, stopping all its operations and services.

- ***Termination***

  The slice instance is terminated and all the allocated resources are released and reclaimed.

7.2 Workflows and Interactions

This section presents the interactions between the functional blocks with the following named APIs and interfaces:

- *The Slice Request Interface:* provides the mechanisms to the Slice Builder to initiate the instantiation of an end-to-end Slice via interacting with the Slice Broker in the Resource Marketplace. It is used by the Slice Broker to provide the description of the Slice to be allocated.
- *The Slice Marketplace Interface:* is related to the interaction between the Slice Broker and the Slice Agents to implement mechanisms for the propagation of resource offerings between (external) resource domains. Different Slice Agents will register their resources availability to the Slice Broker.
- *The Slice Instantiation Interface:* is used by the Slice Builder after available resources offerings have been identified via the Slice Broker; individual resource allocation requests are sent to the relevant Slice Resource Controllers which allocate resources for each single Part of the end to end Slice.
- *The Slice Runtime Interface:* implements the interface that provides functionalities to dynamically modify the resource allocation for a Slice Part. This is required by the Slice Resource Orchestrator to perform lifecycle operation on the end to end Slice according to the feedback received for each Slice Part via the collected monitoring measurements.

7.2.1 Preparation and Instantiation Workflow

The lifecycle flow for the preparation and instantiation is shown below. More details of all the lifecycle functions are presented in the Slice Lifecycle section.

The steps for preparation of a slice plus its instantiation are presented in the following list:

1. The end-user makes request to a Slice Activator for a Slice using a Slice Description, with a specification for the slice, either as a resource focussed spec or as a service focussed spec with some service level details
2. The Slice Activator makes a request to the NECOS Slice Provider handled by the Slice Specification Processor.
3. The NECOS Slice Provider asks Slice Builder to create Slice elements based on the specification
4. The Slice Builder goes to the Marketplace of choice, and interacts with a Slice Broker, to find some slice parts.
5. The Slice Broker interacts with a number of Slice Agents, to try to find suitable and available Slice Parts
6. The Slice Broker collects the details of the infrastructure provider's offers from each of the Slice Agents
7. The Slice Builder receives these details from the Slice Broker in the Marketplace, and makes a decision as to which offers to choose for the Slice
8. The Slice Builder contacts all the relevant infrastructure Slice Controllers with a request for a slice, plus some spec
9. The Slice Controllers allocate the relevant resources.
10. The Slice Controllers allocate / instantiate / configure the on-demand VIM and WIM to manage the resources of the allocated Slice part, as well as the relevant monitoring end-points.
11. The Slice Controllers return the details of the Slice part, together with the access details for the VIM or WIM, plus the associated monitoring end-point.
12. The Slice Builder returns all the Slice Controller details plus Slice component details (e.g. VIMs) to the Slice Resource Orchestrator
13. The Slice Resource Orchestrator interacts with the DC/WAN Slice Controllers to bind/glue the separate slice parts together. This completes the topology as it connects DC part to Net part.
14. The Slice Resource Orchestrator keeps a representation of the slice topology and its constituent parts.
15. The Slice Resource Orchestrator informs the IMA of the allocated VIMs, WIMs, and monitoring end-points, and then the IMA sets up the relevant Adaptors for the VIMs, WIMs, and monitoring end-point at the resource domains.
16. The Slicing Orchestrator informs the Slice Activator of the details of the slice.
17. Slice Activator triggers the Service Orchestrator and the deployment of service is started into the Slice
18. The VMs are instantiated

7.2.2 Run-time and Decommissioning Workflow

The lifecycle flow for the run-time phase is presented below. More details of all the lifecycle functions can be found in the Slice Lifecycle section.

1. The Slice Resource Orchestrator (SRO) functional block is meant to continuously receive feedback sets from the Infrastructure Monitoring Abstraction (IMA). These feedbacks include relevant slice-specific KPIs (e.g., network load, utilization of allocated resources).
2. Lifecycle actions are triggered according with the IMA feedback, including automatic reconfiguration of slices or decommission process, which is carried out by SRO via the Slice Runtime Interface.
3. If a slice needs to add a new slice part in a new location or domain, the SRO contacts the Slice Builder to enforce the process of discovering new slice parts and attached them to the existent slice in a similar manner as described for the Slice Instantiation Phase.
4. If a slice needs to expand, the SRO contacts the relevant Slice Controllers in order to extend the slices parts, increasing the allocated resources associated to the slice parts.
5. If a slice needs to shrink, the SRO contacts the corresponding Slice Controllers in order to remove slices parts, releasing the allocated resources associated to slice parts
6. After adding or removing slice parts, the SRO updates the associated slice topology in the Slice Database.

7.3 Cross-Domain Spanning

In this section, we consider how the modes manifest with respect to the architecture, and present the outcome of a slice spanning across multiple domains.

Each mode is shown together with it's cross-domain aspect:

- **Mode 0** - the VIM on-demand model with DC slicing that allows direct inter-domain orchestrator to VIM interaction using a specific allocated VIM
- **Mode 1** - the slicing in the VIM model, that allows direct inter-domain orchestrator to VIM interaction using a specific allocated shim object

Figure 41 shows how a slice overlays the architectural elements in a Mode 0 manifestation. In Mode 0, there is no Inter-domain Orchestrator interaction, but there is a VIM on-demand allocated for each slice part. This allows a direct Inter-domain Orchestrator to VIM interaction.

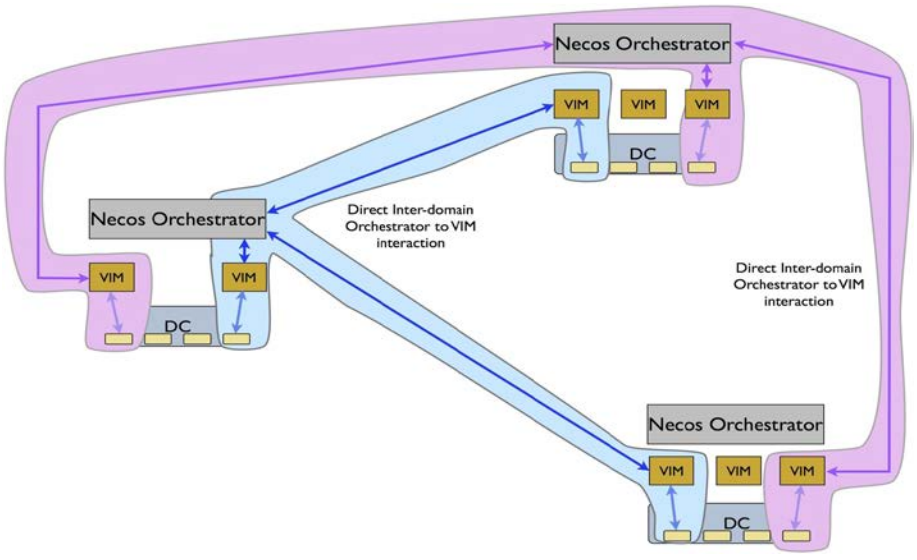


Figure 41: Slices overlaying multiple domains using mode 0

8. Conclusions and Positioning NECOS in the Slicing Ecosystem

This section provides the conclusions of the architectural work in NECOS. First there is a description of the slicing characteristics which have been observed during this project, together with a table which summarises how the key slice characteristics are being addressed. Second, is a presentation of the deployment & economic considerations when slicing is utilised. Finally, we discuss the further work and further challenges in the arena of slicing.

8.1. Slicing Characteristics

After considering the set of relevant network slicing activities in standardization and research projects, it became clear that there was a lack of common terminology and consistent concepts to clearly characterize the scope and approach of each work embracing network slicing principles. As an effort to provide some qualitative comparison on how each work is approaching network slicing, and also to better position these work activities within the NECOS project scope, we identified a list of 14 key characteristics at a high and common level, sufficient to provide a comprehensive snapshot on the state of affairs towards network slicing. The distinguishing characteristics are as follows:

- Connectivity Resource only Slicing
- Connectivity Service Slicing
- Network Cloud Slicing
- E2E Slicing
- Multi-domain Slicing
- Uniform Lifecycle Management
- Tenant Slice Management
- Slices as a Service
- VIM on Demand
- WIM on Demand
- Resource Marketplace for Slices
- Network Elasticity
- Network Cloud Elasticity
- Infrastructure Monitoring

While the set of selected initiatives and projects investigated, early in the project, was representative enough for our main goal of positioning NECOS with regard to relevant related work, we recognize that as more projects address slicing, we could bring these into the picture, especially as time passes, since the support of some type of slicing features is increasingly becoming the norm of any network and cloud-oriented initiative. Certainly, further characteristics could be identified in the future for a more fine-grain and scoped discussion.

The table below summarises how the key slice characteristics in the first *blue* column (which are being addressed) are mapped to the different initiatives in *green* per column. While we opted for a binary check on each characteristic, we recognise that finer-grain quantification and qualification could be done especially for those characteristics ticked as positive by different work groups and projects.

Table 1: Slicing Characteristics

	Initiative						EU 5GPPP Project							EU Project	
	ITU-T Slicing	*NGMN Slicing*	*IETF Slicing*	*3GPP Slicing*	*ETSI Slicing*	*ONF Slicing*	*5GEX Slicing*	*5G-Sonata Slicing*	*5G_NORMA Slicing*	*5G-Transformer Slicing*	*5G-Pagoda Slicing*	*5G-Slicenet Slicing*	*5G MoArch Slicing*	*RESERVOIR Cloud*	***NECOS Slicing***
Slicing Characteristics															
Connectivity Resource only Slicing	✓	✗	✗	✓	✗	✓	✓	✗	✗	✓	✓	✓	✓	✗	✓
Connectivity Service Slicing	✗	✓	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓
Network Cloud Slicing	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
End-to-end Slicing	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Multi-domain Slicing	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓
Uniform Lifecycle Management	✗	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Tenant Slice Management	✗	✗	✓	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	✓
Slices as a Service	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
VIM on Demand	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
WIM on Demand	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Marketplace for Slices	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Network Elasticity	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
Cloud Elasticity	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Infrastructure Monitoring	✓	✗	✗	✗	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓

From the table, we can observe that some characteristics, namely:

(i) Connectivity Resource only Slicing,

(ii) Connectivity Service Slicing,

(iii) Uniform Lifecycle Management, and

(iv) Tenant Slice Management,

have been worked out by different initiatives. This is sensible considering the initial standpoint of *network slicing* was from network connectivity-oriented initiatives and the common management requirements of resources and tenant slices.

In contrast, we note that few works addressed cloud slicing, or jointly addressed cloud and network slicing, or considered end-to-end and multi-administrative domain scenarios, or embraced the Slice-

as-a-Service paradigm. NECOS not only addresses those key characteristics, but in addition goes beyond the state of the art by considering a number of unique features, namely the VIM on-demand and WIM on-demand slicing models, and the Marketplace approach, which have not been considered by other slicing approaches. These will help deliver a more flexible concept of slicing which goes beyond the peer-to-peer orchestration and management interactions on which most of the above initiatives are based. Our architecture started from these concepts to introduce and design the blocks that will allow the realisation of a slicing approach that covers the highlighted characteristics.

8.1.1 Deployment & Economic Considerations

Here is a consideration of the deployment & economic aspects when slicing is utilised.

- **Deployment Options:** There are architectural, engineering, performance, flexibility and service agility without disruption challenges in terms support of many next-generation services in a network cloud enabled infrastructure. In terms of deployment options an operator could deploy a single multi-service network, with a shared physical layer supporting a shared functional layer. Alternatively, the operator could deploy separate physical sub-networks, each with their own physical resource layer and functional layer on top of that; Or the operator could deploy discrete virtual networks, built on one shared physical resource layer, with multiple functional layers dedicated to each application or service type.
- **Economy of Scale in Slicing**: The benefits of slicing grow as the number of service types that you are trying to launch grows. In addition, significant investment is needed in automation to be able to do this at scale, otherwise the complexity and operational challenges are likely to mount up. It's key that the rest of the organisation gears up to support this ambition – development, delivery, marketing, operations and so on - otherwise the operator won't be able to exploit the technology commercially.
- **Service Diversity:** the key challenge is how to support and operate different kind of services with very distinct needs, KPIs, QoS characteristics onto the same infrastructure. One practical approach is to position segregated services on specialized partitions, designed and optimized for the type of service to be provided.
- **Interaction with the vertical customers**: Proper abstractions and templates have to be defined for ensuring the provision of a consistent service portfolio and their integration with the internal network management and orchestration.

8.2 Further Challenges

Many aspects have been considered during the architectural work in NECOS. Many characteristics have been discovered and designed, but there is still a significant amount of work to be undertaken before slices can be utilised in an easy and effective manner.

8.2.1 Challenges in Realising Slice Capabilities

In pursuit of effective solutions for slicing, new types of networking, computing and servicing there is a need to address additional challenges and outcomes, which we have published in [CC1]. The following list presents the set of challenges for realising slice capabilities:

- A Uniform Reference Model for Large Scale Network Cloud Slicing that describes all of the functional and non-functional elements and instances of a slice. It also describes shared non-sliced parts.
- Uniform Slice Templates: Providing the design of slices to different scenarios. This outlines an appropriate slice template definition that may include capability exposure of managed partitions of network cloud resources (i.e. connectivity compute and storage resources), physical and/or virtual network and service functions that can act as an independent network cloud.

- Full Networks Isolation – Highly Efficient slice creation with guarantees for isolation in each of the Data / Control / Management / Service planes. Having enablers for safe, secure and efficient multi-tenancy in slices. Methods to enable diverse requirements for NS, including guarantees for the end-to-end QoS of a service within a slice.
- Slicing Service Mapping – creating an efficient service mapping model binding across network cloud slicing; specifying policies and methods to realize diverse service requirements without re-engineering the infrastructure.
- Slicing Resource Mapping - creating an efficient resource to slice mapping model binding across network cloud slicing
- Recursion, namely methods for slicing segmentation allowing a slicing hierarchy with parent–child relationships.
- Customized security mechanisms per slice - In any shared infrastructure, security is a key element to guarantee proper operation, and especially a fair share of resources to each user including Resource isolation and allocation policy at different levels and Isolation of network service management for multiple tenants.
- Network Slices Reliability - Maintaining the reliability of a network cloud slice instance, which is being terminated, or after resource changes.
- Optimisation, namely methods for automatic selection of network resources for slicing; global resource views; global energy views; Network cloud Slice deployment based on global resource and energy efficiency.
- Capability exposure for network cloud slicing (allowing openness); with APIs for slice specification and interaction.
- Programmability and monitoring control of Network Cloud Slices.
- Per-tenant Policy Management - In a multi-tenant, multi-slice end-to-end hosting and networking scenario, closed-loop automation requires both per-tenant policies, as well as the network operator's own. Per-tenant policies would be set to limit compute, storage and network resource usage, block the execution of unauthorized operations, trigger actions including scaling, healing, and topology reconfiguration to meet the service-level agreement (SLA) with a tenant.
- Efficient and Light Weight Slice lifecycle management including creation, activation / deactivation, protection, elasticity, extensibility, safety, and sizing of the slicing model per network and per network cloud for slices in access, core and transport networks; for slices in data centres, and for slices in edge clouds.
- Dedicated network - Each slice must behave as a dedicated network while sharing underlying resources, physical and virtual. Monitoring the status and behaviour of network cloud slice in a single and/or multi-domain environment and maintenance mechanisms have to be defined in order to show and abstract the proper information for each slice customer.
- Scalability: In order to partition network resources in a scalable manner, it is required to clearly define to what extent slice customers can be accommodated or not on a given slice. The application of different SLAs on the offered capabilities of management, control and customization of slices will directly impact the scalability issue.
- Slice dimensioning: Over-dimensioning has been the normal way in the past for avoiding any kind of congestion. With slicing the traffic sources and destinations become much less predictable, if at all. Appropriate planning, dimensioning and enforcement are needed to make sustainable the transition to this new form of service.

- Autonomic slice management and operation, namely self-configuration, self-composition, self-monitoring, self-optimisation, self-elasticity for slices that will be supported as part of the slice protocols.
- Automated instantiation, scaling and topology reconfiguration of slices.
- Very large scale slicing with efficient elasticity.
- Efficient and large scale slice stitching / composition by having enablers and methods for efficient stitching / composition / decomposition of slices: vertically (through service + management + control planes); horizontally (between different domains as part of access, core, edge segments); or a combination of vertically + horizontally.
- Efficient End-to-end network clouds orchestration of slices.

Version History

Version	Date	Author	Change record
4.7	21/04/2019	Alex Galis	Full draft ready for final review
4.8	22/04/2019	Sand Correa	Review of Section 6.2
4.9	23/04/2019	Augusto Neto	Full reviewed draft
4.10	25/04/2019	Stuart Clayman	Draft with merged reviews
4.8_JS	26/04/2019	Joan Serrat	Full reviewed draft
4.11_SC	30/04/2019	Stuart Clayman	All reviews merged draft
4.11	30/04/2019	Joan Serrat	Final version

Appendix A – Revised Prioritization of Requirements

Based on deliverable 3.1 [D3.1], Appendix A presents the prioritization of scenarios requirements identified in the deliverable 2.1 [D2.1] and enabled by the Quality Function Deployment (QFD) method.

Based on the requirements prioritization it was possible to identify a set of features as presented in table 1:

Table 2: Prioritization of requirements from D2.1

Feature	D2.1 requirements
Slice Provisioning	RF.vRAN3, RF.5G.3, and RF.vCPE.1
Isolation	RN.vRAN.1, RN.5G.1, and RN.vCPE.1
Management	RF.5G.4, RF.vCPE.2, RF. Touristic(CD).1, RF.Emergency.1, RF.Emergency.3
Elasticity	RF.vCPE.6, RN.Touristic(CD).3, RN.Touristic(APP).4
Scalability	RN.Touristic(CD).5, RN.Touristic(APP).2
Monitoring	RF.Touristic(CD).4
VIM-independence	RF.vCPE.3
Bare-metal slice	RF.vCPE.4

Next, the deliverable 2.2 [D.2.2] presented a new set of requirements focusing on the NECOS platform but using the scenario's requirements from D2.1 as the base. Thus, in order to reuse the QFD analysis previously mentioned, it is necessary to provide a link between the requirements from the D2.1 into D2.2. These requirements are presented at the end of this section.

In this context, table 2 presents the mapping between the scenario's requirements [D2.1] into the NECOS Platform requirements [D2.2].

Table 3: Mapping of D2.1 requirements into D2.2 requirements

D2.1 - NFR and FR	D2.2 - NFR	D2.2 - FR
RF.vRAN.1-Service Level Agreement	NFR-2.X	FR-2.3, FR-2.6, FR-5.2
RF.vRAN.2-Accountability		FR-2.2, FR-6.2, FR-6.1
RF.vRAN.3-On-demand slice provisioning	NFR-4.1; NFR-4.3;	FR-1.1; FR-1.2; FR-1.3; FR-2.8; FR-2.10; FR-9.1
RF.vRAN.4-Isolation of slice provisioning	NFR-6.X	

RN.vRAN.5 -Fairness	NFR-6.7	
RN.vRAN.6 -Fault detection	NFR-2.4; NFR-2.1; NFR-9.3	FR-2.3; FR-2.6; FR-5.2
RF.5G.1-Service Level Agreement	NFR-2.X	FR-2.3; FR-2.6; FR-5.2
RF.5G.2-Accountability		FR-2.2, FR-6.2, FR-6.1
RF.5G.3-On-demand slice provisioning	NFR-4.1; NFR-4.3;	FR-1.1; FR-1.2; FR-1.3; FR-2.8; FR-2.10; FR-9.1
RF.5G.4-External control and management of the offered slices	NFR-4.1, NFR-4.3, NFR-7.3	FR-1.1; FR-1.2; FR-1.3; FR-9.1
RN.5G.5-Isolation of slice resources	NFR-6.X	
RN.5G.6-Fairness	NFR-6.7	
RN.5G.7-Fault detection	NFR-2.4; NFR-2.1; NFR-9.3	FR-2.3; FR-2.6; FR-5.2
RF.vCPE.1-On-demand slice provisioning	NFR-4.1; NFR-4.3;	FR-1.1; FR-1.2; FR-1.3; FR-2.8; FR-2.10; FR-9.1
RF.vCPE.2-Manageable slice	NFR-7.X	FR-1.3;FR-2.1;FR-2.4;FR-5.3;FR-6.1
RF.vCPE.3-VIM-independence	NFR-5.3	FR-4.3
RF.vCPE.4-Bare-metal slice	NFR-6.X	
RF.vCPE.5-Lightweight virtualization	NFR-6.1, NFR-7.2, NFR-14.1	FR-5.3
RF.vCPE.6-Elasticity	NFR-4.X	FR-2.11; FR-2.12
RF.vCPE.7-Zero touch service provisioning	NFR-7.2, NFR-7.4, NFR-12.1, NFR-12.3	
RF.vCPE.8-Fault detection	NFR-2.4; NFR-2.1; NFR-9.3	FR-2.3; FR-2.6; FR-5.2
RN.vCPE.9-Isolation of slice resources	NFR-6.X	
RN.vCPE.10-SLA monitoring (QoS)	NFR-7.6	
RN.vCPE.11-Low latency	NFR-2.2	

RN.vCPE.12-High throughput	NFR-2.3	
RN.vCPE.13-High availability	NFR-2.4	
RF.Touristic(CD).1-Slice and slice-resource management	NFR-4.1, NFR-4.2, NFR-7.1	FR-1.2, FR-1.3, FR-2.1
RF.Touristic(CD).2-Automated Virtual Machine deployment	NFR-7.2	FR-5.3
RF.Touristic(CD).3-Traffic load-balancing for content delivery	NFR-5.3, NFR-12.3	
RF.Touristic(CD).4-Slice resource and service monitoring	NFR-12.4	FR-1.1, FR-2.7, FR-2.8, FR-2.9, FR-2.10, FR-3.4, FR-7.2, FR-7.3, FR-8.1, FR-8.2, FR-9.5, FR-2.3, FR-2.6, FR-5.1, FR-5.2, FR-10.2
RF.Touristic(CD).5-Service planning		FR-1.2, FR-2.7, FR-2.8, FR-2.9
RN.Touristic(CD).6-Transparent end-user performance	NFR-12.3, NFR-5.3	FR-3.6, FR-6.2, FR-10.2
RN.Touristic(CD).7-Heterogeneity handling	NFR-5.3	
RN.Touristic(CD).8-Elasticity	NFR-4.X	FR-2.11; FR-2.12
RN.Touristic(CD).9-Resource-efficiency		FR-7.2, FR-7.3, FR-8.1, FR-8.2, FR-9.5
RN.Touristic(CD).10-Scalability	NFR-4.4	
RF.Touristic(APP).1 Service function chain orchestration	NFR-5.4.	FR-2.7, FR-2.8, FR-2.11, FR-2.12
RF.Touristic(APP).2 Resource and user-demand prediction capabilities	NFR-4.X	FR-2.3, FR-2.6, FR-2.7, FR-2.8
RF.Touristic(APP).3 Resource offloading between edge, core clouds and cloud providers	NFR-4.X	FR-2.3, FR-2.6, FR-2.7, FR-2.8
RF.Touristic(APP).4 Resource federation and intelligent multi-domain orchestration	NFR-5.4	FR-4.1, FR-4.4
RF.Touristic(APP).5 Scalability	NFR-4.4	
RF.Touristic(APP).6 Efficient next-generation touristic application performance	NFR-2.X	FR-2.7, FR-2.8, FR-2.11, FR-2.12

RF.Touristic(APP).7 Elasticity	NFR-4.X	FR-2.11; FR-2.12
RF.emergency.1 Dynamic slice management	NFR-7.X	FR-1.3;FR-2.1;FR-2.4;FR-5.3;FR-6.1
RF.emergency.2 Dynamic service definition		FR-7.2, FR-7.3, FR-8.1, FR-8.2, FR-9.5
RF.emergency.3 Timely slice management	NFR-7.X	FR-1.3;FR-2.1;FR-2.4;FR-5.3;FR-6.1
RF.emergency.4 Orchestration	NFR-5.4	FR-4.1, FR-4.4
RF.emergency.5 High Reliability	NFR-9.X NFR-1.2	
RF.emergency.6 High Availability	NFR-2.4	
RF.emergency.7 High Survivability	NFR-2.4	

It's possible to identify in table 4 that functional requirements from the D2.1 are mapped to D2.2 functional and non-functional requirements, the same behaviour is found on the mapping of D2.1 non-functional requirements. We understand that the scenarios' requirements from D2.1 were more abstracts and they were refined in D2.2, thus, a single functional requirement in D2.1 could be mapped in multiples functional and non-functional requirements from D2.2.

Table 4: Prioritization of requirements from D2.2

Feature	D2.2 requirements
Slice Provisioning	NFR-4.1, NFR-4.3, FR-1.1, FR-1.2, FR-1.3, FR-2.8, FR-2.10, and FR-9.1
Isolation	NFR-2.X, FR-2.3; FR-2.6; FR-5.2, NFR-4.1, NFR-4.3, FR-1.1, FR-1.2, FR-1.3, FR-2.8, FR-2.10, and FR-9.1
Management	NFR-6.X, NFR-7.X, FR-1.3;FR-2.1;FR-2.4;FR-5.3;FR-6.1, NFR-4.1; NFR-4.2;NFR-7.1, FR-1.2; FR-1.3; FR-2.1, NFR-7.X, FR-1.3, FR-2.1, FR-2.4, FR-5.3, FR-6.1
Elasticity	NFR-4.X, FR-2.11; FR-2.12, NFR-5.3, NFR-12.3, NFR-5.4, FR-4.1, FR-4.4
Scalability	FR-1.2, FR-2.7, FR-2.8, FR-2.9, NFR-4.X, FR-2.3, FR-2.6, FR-2.7, FR-2.8
Monitoring	NFR-12.4, FR-1.1, FR-2.7, FR-2.8, FR-2.9, FR-2.10, FR-3.4, FR-7.2, FR-7.3, FR-8.1, FR-8.2, FR-9.5, FR-2.3, FR-2.6, FR-5.1, FR-5.2, FR-10.2
VIM-independence	NFR-5.3 and FR-4.3
Bare-metal slice	NFR-6.X

Using the mapping of D2.1 into D2.2 presented in Table 2 and the prioritization of features presented in Table 1. It is possible to create a new prioritization table with the requirements from D2.2. This new prioritization is presented in Table 3.

Functional Requirements

These are the Functional Requirements from Deliverable D2.2.

Slice Specification Processor

FR-1.1 - Creation of PDT messages from slice descriptions

FR-1.2 - Provide slice creation interface

FR-1.3 - Provide slice management interface

Slice Resource Orchestrator

FR-2.1 - Management of connections between slice parts to have a end-to-end slice

FR-2.2 - Interact with Slice Database for query and to provide slice updates

FR-2.3 - Continuously update IMA regarding VIM, WIMs and Monitoring pointers

FR-2.4 - Provide operational functions for slice management

FR-2.5 - Request slice parts removal

FR-2.6 - Process slice monitoring information provided by the IMA

FR-2.7 - Evaluate the need of upgrade or downgrade of resources within a slice

FR-2.8 - Evaluate requests for upgrade or downgrade of resources within a slice

FR-2.9 - Selection of resources that will be allocated or freed in the elasticity process

FR-2.10 - Request the addition or removal of resources within slice parts

FR-2.11 - Provide an elasticity process that prioritizes vertical elasticity over a horizontal approach

FR-2.12 - Implement horizontal elasticity by requesting the creation or removal of slice parts

Slice Builder

FR-3.1 - Analysis of specific policies and rules for slice requests

FR-3.2 - Request resources for slice parts

FR-3.3 - Provide different mechanisms for defining the final slice specification

FR-3.4 - Creation of contracts for resources reservation among resource providers, NECOS platform and tenants

FR-3.5 - Perform reservation and activation of slice parts

FR-3.6 - Provide slice and slice parts information

Slice Controller

FR-4.1 - Support allocation, removal and modification of resources for slice parts

FR-4.2 - Accept or reject contracts for resource reservation

FR-4.3 - Instantiation and removal of VIMs and WIMs for slice parts

FR-4.4 - Support requests for connecting and disconnecting slice parts together

FR-4.5 - Process requests for upgrade and downgrade of resources within slice parts

Infrastructure & Monitoring Abstraction

FR-5.1 - Process requests for setup, removal and update of relevant adaptors for VIMs, WIMs and Monitoring endpoints

FR-5.2 - Provide slice monitoring metrics

FR-5.3 - Provide management operations for deployment of virtual functions

Slice Database

FR-6.1 - Provide an interface for slice management

FR-6.2 - Provide information about slices, slice parts and services

Slice Broker

FR-7.1 - Support a search mechanism for requesting resources from infrastructure providers

FR-7.2 - Capability of identify potential network resource providers based on DC resource offers

FR-7.3 - Provide resource offers in the form of alternative resources

FR-7.4 - Accept registration of Slice Agents

Slice Agent

FR-8.1 - Process resource requests and check local availability

FR-8.2 - Provide resource options as answers for resource requests

FR-8.3 - Constantly check the availability of resources in the local domain

FR-8.4 - Registration with the Slice Broker

Slice Activator

FR-9.1 - Creation of slice description from tenant's slice specification, slice requirements and service descriptors

FR-9.2 - Provide a catalogue system to support the description of slice requests

FR-9.3 - Start service deployment

FR-9.4 - Support management functions for running slices

FR-9.5 - Handle contracts for resource reservation

FR-9.6 - Support a mechanism for selection of the final slice specification

Service Orchestrator Adaptor

FR-10.1 - Process service deployment requests

FR-10.2 - Provide service access and monitoring interfaces to the tenant

Service Orchestrator

FR-11.1 - Provide service deployment within a slice

Non-Functional Requirements

These are the Non-Functional Requirements from Deliverable D2.2.

Service diversity

NFR-1.1 - Network should support diversified services accommodating a wide variety of traffic characteristics and behaviours.

NFR-1.2 - Network should support services that have different end-to-end QoS requirements.

Service Level Agreement

NFR-2.1 - The system must assure all the performance metrics negotiated for each slice.

NFR-2.2 - Low latency.

NFR-2.3 - High throughput.

NFR-2.4 - High availability.

Mobility

NFR-3.1 - The network architecture is required to support seamless and consistent user experience while moving across different access networks.

NFR-3.2 - The network architecture is required to support enhanced mobility management such as "context-aware mobility".

NFR-3.3 - The network architecture is required to support the mobility management to cope with traffic explosion and latency requirements of applications.

NFR-3.4 - The network should provide mobility that facilitates high-speed and large-scale where a huge number of UE can dynamically move across heterogeneous networks.

Functional Flexibility and programmability

NFR-4.1 - The network architecture is required to support the capability of creating the dedicated network into a dedicated slice, according to the requirement of the 3th party.

NFR-4.2 - The network architecture should be designed to have a common core that supports on-demand composition of variety of multiple network slices.

NFR-4.3 - Programmability allows third parties to control the allocated slice resourcesvia open APIs.

NFR-4.4 - High scalability with a short-time response for the deployment in the field.

Network-agnostic architecture and technology-agnostic

NFR-5.1 – The network is envisioned to be an access network-agnostic architecture.

NFR-5.2 - The access technology-agnostic unified core network should be accompanied by common control mechanisms.

NFR-5.3 - Heterogeneity handling requires service provisioning should be transparent with respect to details of the physical infrastructure.

NFR-5.4 - Resource federation and intelligent multi-domain orchestration.

Isolation of network Slice and virtual resource and security

NFR-6.1 – The network architecture should support virtualization of resources associated with network functions, and should support isolation of any network slice and virtual resource from others.

NFR-6.2 - The network architecture is required to provide softwarization capabilities with enhanced performance for wired and wireless network.

NFR-6.3 - The network architecture is recommended to provide softwarization capabilities with enhanced performance for mobile access networks.

NFR-6.4 - The network architecture is required to support programmability of network functions in data plane for easier provisioning of new emerging service.

NFR-6.5 - The network architecture is required to support the separation of control and data plane functions in the network.

NFR-6.6 - The network architecture is required to meet the service-specific security assurance requirement in a single network slice, rather than the whole network.

NFR-6.7 - Fairness. Operator must be able to optimize the resource usage without negative impact on any tenant.

Unified intelligent network management

NFR-7.1 - The network architecture is required to create, operate and manage network slice.

NFR-7.2 - The network architecture is recommended that network functions are virtualized, and it should support dynamic scale-in/scale-out per operator's policies.

NFR-7.3 - The network should be designed to simplify operations and management of the network with increased complexity due to flexible and extensible network softwarization.

NFR-7.4 - Procedures should be automated as far as possible, with well-defined open interfaces to mitigate multivendor interworking problems, as well as interoperability issues.

NFR-7.5 - Standardized management protocol is desirable.

NFR-7.6 - Enhanced end-to-end QoS management and security/privacy models should be designed.

NFR-7.7 - Customization assures that the resources allocated to a particular tenant are efficiently utilized in order to meet best the respective service requirements.

Distributed architecture

NFR-8.1 - Separation of control and data planes and the enabling technologies are the basis to make the network flexible and extensible

NFR-8.2 - Network should support a highly scalable distributed architecture to avoid signalling congestion and to minimize the signalling overhead for diverse UE/RAT/service requirements.

NFR-8.3 - Require the gateways to the core networks expected to be located closer to the cell sites resulting in distributed a network architecture.

Reliability and resilience

NFR-9.1 - The network should be designed, operated, and evolved with reliability and resilience, considering congestion and disaster conditions, and be designed for safety and privacy of their users.

NFR-9.2 - The network architecture is required to protect negative impact in one network slice offered by another network slice.

NFR-9.3 - The network architecture is recommended to maintain the established QoS of the network slice after creation regardless of the status of other network slices.

Optimization

NFR-10.1 - The network should provide sufficient performance by optimizing network equipment capacity based on service requirement and user demand.

NFR-10.2 - The network should provide dynamic data routing mechanisms that respond to changing conditions of network segments.

NFR-10.3 - Network should be designed and implemented for optimal and efficient handling of huge amounts of data.

NFR-10.4 - The slice resources should be efficiently utilized.

NFR-10.5 - Fast service deployment.

Improve energy efficiency

NFR-11.1 - The network should be designed to reduce UE power consumption and to improve energy efficiency in overall network operation.

Automation

NFR-12.1 - Automation enables an on-demand configuration of network slicing without the need of fixed contractual agreements and manual intervention.

NFR-12.2 - Allow third parties to place a slice creation request indicating besides the conventional SLA, which would reflect the desired capacity, latency, jitter, timing.

NFR-12.3 - End-user performance transparency, with respect to service usage and experience minimal service delays and interruptions.

NFR-12.4 - Provision of real-time monitored data to the operator and the tenant.

End-to-end

NFR-13.1 - End-to-end is an inherent property of network slicing for facilitating a service delivery all the way from the service providers to the end-user/customers.

NFR-13.2 - End-to-end stretches across different administrative domains, i.e. a slice that combines resources that belong to distinct infrastructure providers, and unifies various network layers and heterogeneous technologies.

Hierarchical abstraction

NFR-14.1 - Hierarchical abstraction provides a procedure which is repeated on a hierarchical pattern with each successively higher level, offering a greater abstraction with a broader scope.

NFR-14.2 - The network architecture is required to support that the 3rd parties can create and manage a network slice configuration within the limits set by the network operator.

References

[IN1] https://cloudify.co/

[IN2] https://www.terraform.io/

[IN3] https://www.scalr.com/

[IN4] http://www.5gex.eu/

[IN5] http://www.fp7-unify.eu/

[SS1] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," in IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2429-2453, third quarter 2018.

[SC1] Programmable Networks for IP Service Deployment" Galis, A., Denazis, S., Brou, C., Klein, C.–"ISBN 1-58053-745-6, pp 450, June 2004, Artech House Books, http://www.artechhouse.com/International/Books/Programmable-Networks-for-IP-Service-Deployment-1017.aspx

[SC2] "Management and Service-aware Networking Architectures (MANA) for Future Internet" – A. Galis et all - Invited paper IEEE 2009 Fourth International Conference on Communications and Networking in China (ChinaCom09) 26-28 August 2009, Xi'an, China, http://www.chinacom.org/2009/index.html

[SC3] "The RESERVOIR Model and Architecture for Open Federated Cloud Computing", Rochwerger, J. Caceres, R. Montero, D. Breitgand, A. Galis, E. Levy, I. Llorente, K. Nagin, Y. Wolfsthal; the IBM System Journal Special Edition on Internet Scale Data Centres, vol. 53, no. 4, 2009, http://www.haifa.ibm.com/dept/stt/sas_public.html

[SC4] "Infrastructure Slicing Landscape: –Galis. A, Makhijani, K - Tutorial at IEEE NetSoft 2018, Montreal 19 July 2018; http://discovery.ucl.ac.uk/10051374/

[SC5] http://groups.geni.net/geni/wiki/GENIConcepts

[SC6] ITU-T Y.3011- http://www.itu.int/rec/T-REC-Y.3001-201105-I

[SC7] https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-July-2016.pdf

[SC8] https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf

[SC9] 'A Scientometric Analysis of Cloud Computing Literature'- Heilig, L., Voss, S. - IEEE Transactions on Cloud Computing, Volume: PP, Issue: 99, 30 April 2014, ISSN: 2168-7161; DOI: 10.1109/TCC.2014.2321168

[SC10]https://www.etsi.org/deliver/etsi_gr/NGP/001_099/011/01.01.01_60/gr_ngp011v010101p.pdf

[ETSI2018] E2E Network Slicing Reference Framework and Information Model" - Kiran, Makhijani-Huawei Technologies, Kevin Smith-Vodafone John Grant - Ninetiles Alex Galis- UCL, Xavier Defoy- Interdigital - published in October 2018 - https://www.etsi.org/deliver/etsi_gr/NGP/001_099/011/01.01.01_60/gr_ngp011v010101p.pdf

[HomaEtAl2019] Homa, S. et al., "Network Slice Provision Models", Work in Progress, draft-homma-slice-provision-models-00, February 1, 2019.

[ETSI NFV] https://www.etsi.org/deliver/etsi_gs/nfv-man/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[NECOS D2.1] http://www.maps.upc.edu/public/Deliverable%20D2.1%20final.pdf

[IETF3] L. Berger, C. Hopps, A. Lindem, D. Bogdanovic, X. Liu, "YANG Model for Logical Network Elements", RFC 8530, March 2019.

[ONF2017] ONF Transport API (TAPI) 2.0, August 2017.

[I-D.homma-slice-provision-model] Homma, S., Nishihara, H., Miyasaka, T., Galis, A., Ram OV, V., Lopez, D., Contreras-Murillo, L., Ordonez-Lucena, J., Martinez-Julia, P., Qiang, L., Rokui, R., Ciavaglia, L., de Foy, X., "Network Slice Provision Models", https://tools.ietf.org/html/draft-homma-slice-provision-models-00 (work in progress), February 2019.

[ITU-T 2016] Draft Recommendation: Network Management Framework for IMT-2020. IMT-O-047.

[NGMN 2016] NGMN 5G P1 Requirements & Architecture Work Stream End-to-End Architecture. Description of Network Slicing Concept.

[Y. Al-Dhuraibi] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah and P. Merle, "Elasticity in Cloud Computing: State of the Art and Research Challenges," in IEEE Transactions on Services Computing, vol. 11, no. 2, pp. 430-447, 1 March-April 2018; doi: 10.1109/TSC.2017.2711009 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7937885&isnumber=8332642

[Herbst] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in Cloud Computing: What It Is, and What It Is Not," in Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13). San Jose, CA: USENIX, 2013, pp. 23–27.

[AG1] IETF draft (2017) "Network Slicing" Galis., A, Dong., J, Makhijani, K, Bryant, S., Boucadair, M, Martinez-Julia,P. https://tools.ietf.org/html/draft-gdmb-netslices-intro-and-ps-01

[AG2] Alex Galis "Network Slicing Tutorial" at IEEE CNSM, Rome, 5-9 November 2018- http://cnsm-conf.org/2018/files/CNSM18_SlicingTutorial_AlexGalis_5-10-2018.pdf

[NC1] "Slicing and Allocation of Transformable Resources for the Deployment of Multiple Virtualized Infrastructure Managers (VIMs)", Leandro A. Freitas, Vinícius G. Braga, Sand Correa, Lefteris Mamatas, Christian Esteve Rothenberg, Stuart Clayman, Kleber Vieira Cardoso , 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)

[NC2] NECOS Project: Towards Lightweight Slicing of Cloud-Federated Infrastructures, Felipe S. Dantas Silva, et al , 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)

[CC1] "Perspectives on Network Slicing – Towards the New 'Bread and Butter' of Networking and Servicing", A. Galis, https://sdn.ieee.org/newsletter/january-2018/perspectives-on-network-slicing-towards-the-new-bread-and-butter-of-networking-and-servicing