A Holistic, Innovative Framework for the Design, Development and Orchestration of 5G-ready Applications and Network Services over Sliced Programmable Infrastructure

"OSS FOR 5G-READY APPLICATIONS"

A WHITEPAPER WRITTEN BY THE MATILDA PROJECT

List of Authors

CNIT	CONSORZIO NAZIONALE INTERUNIVERSITARIO PER LE TELECOMUNICAZIONI
Roberto Bruschi, Chiara Lombardo	
ATOS	ATOS SPAIN SA
Fernando Diáz, Javier Melián, Aurora Ramos	
ERICSSON	ERICSSON TELECOMUNICAZIONI
Orazio Toscano	
COSM	COSMOTE KINITES TILEPIKOINONIES AE
Ioanna Mesogiti	
ORO	ORANGE ROMANIA SA
Horia Stefanescu	
UBITECH	GIOUMPITEK MELETI SCHEDIASMOS YLOPOIISI KAI POLISI ERGON PLIROFORIKIS ETAIREIA PERIORISMENIS EFTHYNIS
Anastasios Zafeiropoulos, Panagiotis Gouvas, Eleni Fotopoulou, Thanos Xirofotos	
INC	INCELLIGENT IDIOTIKI KEFALAIOUCHIKI ETAIREIA
Nikolaos Stasinopoulos	
NCSRD	NATIONAL CENTER FOR SCIENTIFIC RESEARCH "DEMOKRITOS"
Eleni Trouva	
AALTO	AALTO-KORKEAKOULUSAATIO
Miloud Bagaa, Ibrahim Afolabi	

Table of Contents

1 Overall system architecture to orchestrate 5G-Ready Applications

In order to meet the increased demands from the consumer and business networking perspectives, Multi-access Edge Computing (MEC), Network Functions Virtualization (NFV) and Software Defined Networking (SDN), brought together, will lift to higher levels the computing capabilities and system-wide utility and efficiency of underlying infrastructure. When the combination of the programmability, scalability, flexibility and low latency characteristics of the aforementioned enabling technologies are leveraged, vertical network applications can easily be orchestrated while ensuring certain set Quality of Service (QoS) within the lifecycle of applications running on the network.

By leveraging these technological paradigms, the main system architecture taken as reference by the MATILDA project is a multi-layered and multi-tenant framework, fully compliant with the latest specification of 3GPP and ETSI NFV, and able to reflect the various administrative domains in a 5G network. In details, this framework is composed of three main layers, corresponding to the applicative, the network platform and the infrastructural levels.

The infrastructural level is composed of network and computing facilities, which can be provided by different players (i.e., Infrastructure Providers), and which are exposed through state-of-the-art programming interfaces. On the Telecom Service providers' end, combined with the OSS/BSS, various end-to-end telecommunication services can be supported, covering several requirements and needed functions such as, network activation, network provisioning, service inventory, and service activation. In the context of MATILDA, the 5G-ready application is the highest-level entity from a top-down perspective. This is the end-point of 5G environments that the users will interact with. A 5G-ready application is reflected in an application graph and is implemented by a service mesh, which consists of several chainable components; the most granular entity.

The main peculiarity and advantage of the foundation of the MATILDA architecture consists of exposing 5G networks to application providers as a simple extension to today's Cloud Computing technologies, paradigms, and interfaces. This flexibility and extensibility will be achieved by relying entirely on the MATILDA architecture that combines the aforementioned technologies.

In this respect, this whitepaper focuses on the design of the MATILDA network platform layer to allow, on the one hand, the interaction with the applicative layer by exposing an abstract view of the available resources and, on the other hand, to realize and autonomically management of the lifecycle of 5G-ready applications.

It is worth noting that, thanks to the clear decomposition of administrative domains and the specific selection of programming interfaces, this architecture permits and somehow fosters the presence of small/medium infrastructure providers at any layers, from the infrastructural up to the applicative layer. For instance, small/medium infrastructure providers can offer as-a-Service computing or networking resources (e.g., a micro datacenter or some radio access nodes close to a facility, like a stadium or an industrial plant). At the applicative layer, a small/medium software provider can offer its applications/services deploying them onto 5G network slices. To manage the lifecycle of its applications/services, the MATILDA framework allows software providers to use the same paradigms and concepts of today's cloud computing

frameworks. This approach has been specifically conceived to encourage the (partial) shift from the "cloud" to the 5G "edge" of vertical stakeholders. Moreover, as will be highlighted later in the document, the architecture also satisfies the functionality of the main network functions recently proposed by the 3rd Generation Partnership Project (3GPP) [1].

2 Network Platform Layer

The MATILDA network platform layer is targeted to be owned by a Telecom Service Provider, and it is in charge of setting up, monitoring, and managing the lifecycle of any network services, and exposing them to vertical applications. Therefore, the network platform is a middle-layer with intrinsic multi-tenant and multi-domain support – i.e., it exposes northbound interfaces towards multiple diverse vertical applications, and southbound interfaces towards multiple (third-party) network and computing infrastructures. The proposed network platform is composed of four main building blocks, namely:

- The extended Operations and Business Support Systems (OSS/BSS);
- The Computing Slice Manager (CSM);
- The NFV Orchestrator (NFVO);
- The Wide-area Infrastructure Manager (WIM).

It is worth noting that, with the exception of the Computing Slice Manager, all of these building blocks and their reference points are fully compliant with the specifications of the ETSI NFV architectural framework [2].

The main role of these modules is to jointly interact with the MATILDA 5G Vertical Application Orchestrator (VAO) [3] upon vertical application requests or events coming from the infrastructural level to realize the autonomic management of the lifecycle of 5G network slices and edge computing resources. In detail, for each hosted vertical application, the MATILDA framework envisions to realize and to manage a deployment like the example shown in Figure 1.

The Computing Slice Manager allows verticals to perform operations on applications deployed over multiple mobile edge hosts. In fact, it provides the VAO with the list of VIMs and tenant spaces to be made accessible to the Vertical and, upon completion of a slice materialization, it allows interacting with the slice by performing HTTP proxying towards the application instances.

The extended OSS/BSS provides additional management capabilities and reference points with respect to the legacy ETSI-MANO specification [2]. In fact, while still supporting interactions with the Element Manager of already instantiated Network Functions, it also allows exposing the resources of the Telecom Service Provider to Vertical Industries in terms of 5G network slices. The communication of the available resources and their reservation to the above layers is performed by interacting with the VAO by means of the metamodels presented in the D1.2 report [4]. Southbound, the OSS/BSS interacts with the NFV Orchestrator (NFVO) to request the activation/deactivation/modification of NFV services, and the VNF instances for configuration purposes.

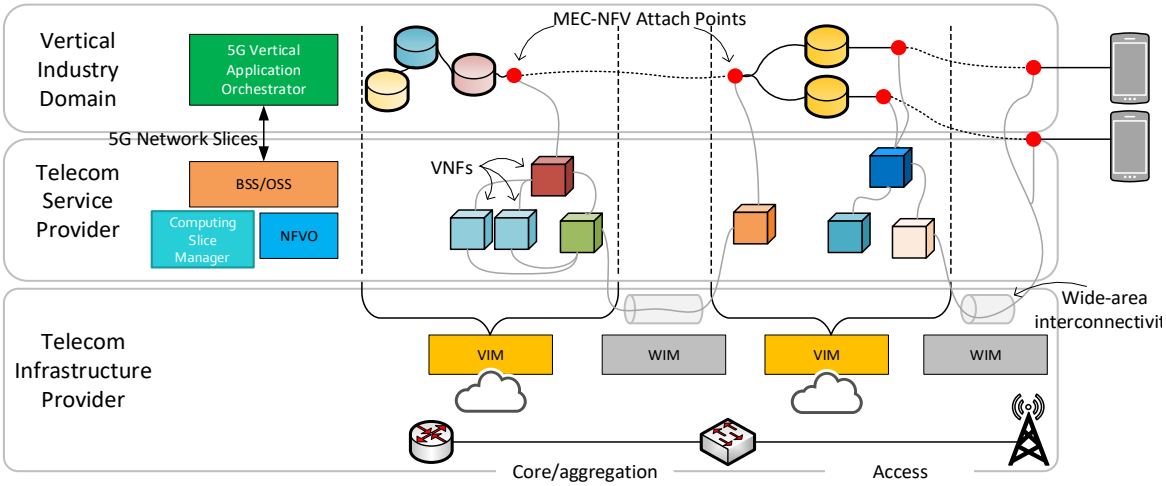

Figure 1: Key architectural building blocks and main stakeholders involved in the deployment of a vertical application onto a 5G infrastructure.

In the middle of the applicative and infrastructural layers, the MATILDA network platform is in charge of proving the required end-to-end network services to connect the application components deployed in diverse datacenters among themselves, and with the mobile terminals (i.e., 5G User Equipment – UE) consuming and/or providing data to the vertical application; such network services are meant to be provided by the NFVO, which has the role of maintaining a catalogue of available resources, driving the selection of the aforementioned resources according to the vertical application needs, and of harmonizing and abstracting complex network services to the applicative layer.

Finally, the WIM has knowledge of the Wide Area Network (WAN) topology and provides the OSS/BSS with latency-driven candidate paths among VIMs and subsequent resource mapping reservation of suitable wide-area paths (routing/tunneling resources in compliance with QoS requirements).

The implementation details of each building block are provided in MATILDA deliverable [D4.1]. One of the main goals driving the design of the architecture is to concentrate most of the extensions into the OSS/BSS while minimizing changes in the other building blocks. Although this line of work might seem to counterintuitively increase complexity in the OSS/BSS, its microservice-driven deployment guarantees a rapid provisioning and deployment of additional functionalities while exploiting already available technologies, such as OSM for the NFVO and the Ericsson Network Manager (ENM) for the WIM.

3 The extended Operations Support Systems (OSS)

The MATILDA OSS has been designed according to a highly modular architecture, composed of a mesh of microservices, in charge of managing the different components of the whole 5G infrastructure, as well as of the overlying network and applicative services. As shown in Figure 2, the OSS is composed of the following main building blocks:

- **Slicing Northbound Module**: this module is in charge of exposing the OSS northbound APIs to VAOs for the lifecycle management of the network and computing slice. For this

reason, it implements the 5G slice information models and the related transactions specified in the D1.4 report, including the "*slice intent*" and the "*materialized slice*" messages. The Slicing North Bound Module is meant to store all the data received from the VAO (e.g., slice intent requests) in the Persistency Layer, and to retrieve from this layer all the data needed to fill the candidate materialized slice.

- **Resource Selection Optimizer**: this module is charge of optimally selecting the infrastructure resources, the wide-area interconnectivity, as well as the network services to be used to meet the application-level slice requirements. It is triggered upon direct requests from the VAO (e.g., new slice, or modification of existing slices), or events from the network (e.g., faults, maintenance operations, etc.). The Resource Selection Optimizer reserves the selected resources until the confirmation from the VAO is received or a timeout expires.
- **Slicing Lifecycle Manager**: this module is responsible for applying the configurations computed by the Resource Selection Optimizer by sequentially requesting the activation of reserved WIM, VIM and NFV resources and services. The Slicing Lifecycle Manager is also in charge of suitably configuring the Computing Slice Manager to let the VAO access the selected VIMs.
- **VIM Convergence Layer**: this module is devoted to translate the requests from the Slicing Lifecycle Manager to the specific VIM APIs. In the context of the MATILDA Project, only OpenStack will be supported as VIM. The main APIs supported by this module will be the creation of tenants' projects (used by the VAO to run applicative components), to create the "*attach points*" between the tenants' projects used by the NFVO and the ones used by the VAO, as well as to reserve resources. This layer is also meant to retrieve any identifiers and data of the activated entities in the VIM.

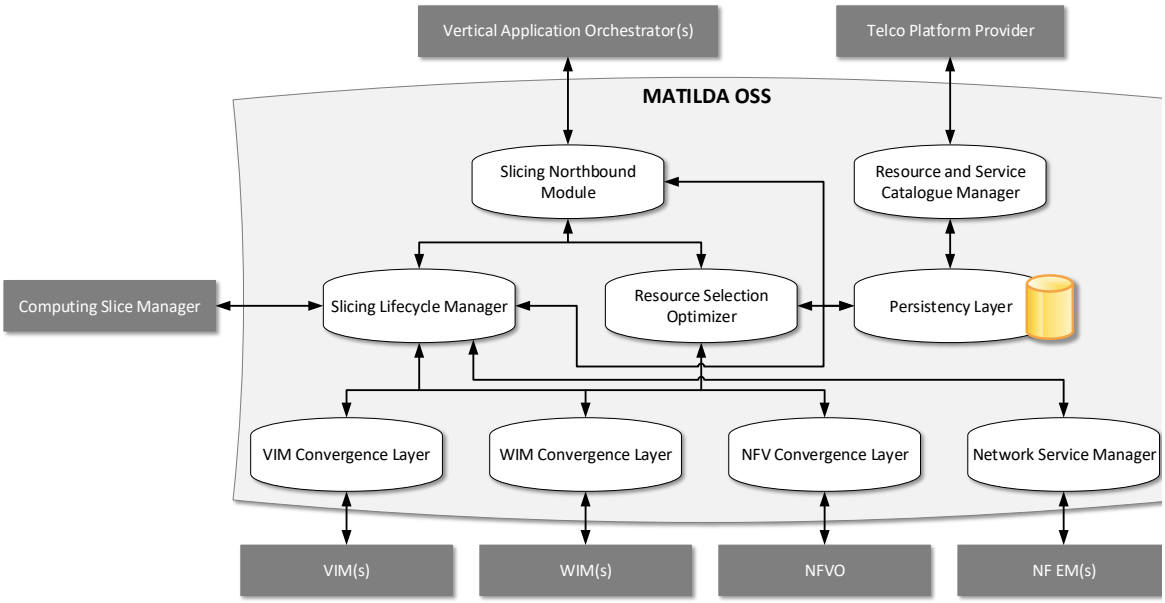

Figure 2: Overall architecture and main building blocks of the MATILDA OSS within their reference points towards external components.

- **NFV Convergence Layer**: this module is devoted to keep the Slicing Lifecycle Manager and the Resource Selection Optimizer aligned with the NFV Orchestrator and related services. The main role of this component will be to provide the Slicing Lifecycle Manager with the translation service for the specific NFVO northbound APIs under use, as well as to maintain a set of high-level information. In detail, such information and meta-data regard the available NFV service blueprints, their computing, storage, and network requirements, and the ability to transform this information into the set of descriptors required by the NFVO to set up, update, or remove a service instance. In the context of the MATILDA Project, only ETSI OSM will be supported as NFVO in the final demonstrator.
- **WIM Convergence Layer**: This module acts as translator between the Slicing Lifecycle Manager, the Resource Selection Optimizer components and the WIM. It is in charge of supporting APIs for both the identification and reservation of the resources (i.e., paths, possible end-points supporting the vertical application requirements, etc.), and the management of the lifecycle of reserved resources in the wide-area transport network. In the context of the MATILDA Project, the Ericsson Network Manager (ENM) will be supported as WIM in the final demonstrator.
- **Network Service Manager**: This module aims to manage the functionalities usually provided by "legacy" OSSs, which is interacting with the Element Manager of already instantiated VNFs for monitoring and configuration purposes to assure the correct operations of NSs.
- **Persistency Layer**: This module consists of a database where all the data regarding the slices, the resources, and the infrastructure will be stored. It has been chosen to use a non-relational database like MongoDB [5].
- **Resource and Service Catalogue Manager**: This module consists of a simple user interface, meant to be used by the Telecom Infrastructure Provider to manage the catalogues of the onboarded resources (i.e., VIMs, etc.) and services in the OSS. Also in this case, these data are stored in the Persistency Layer.

3.1 Lifecycle Management of a Network and Computing Slice

Figure 3 and Figure 4 show the main two phases performed by the MATILDA OSS for instantiating a new slice. The first phase aims at producing a candidate materialized slice upon the reception of a slice intent from the VAO. Triggered by the acceptance of the materialized slice by the VAO, the second phase is meant to reserve all the needed resources and services. Further details on the involved microservices are reported in the following subsections.

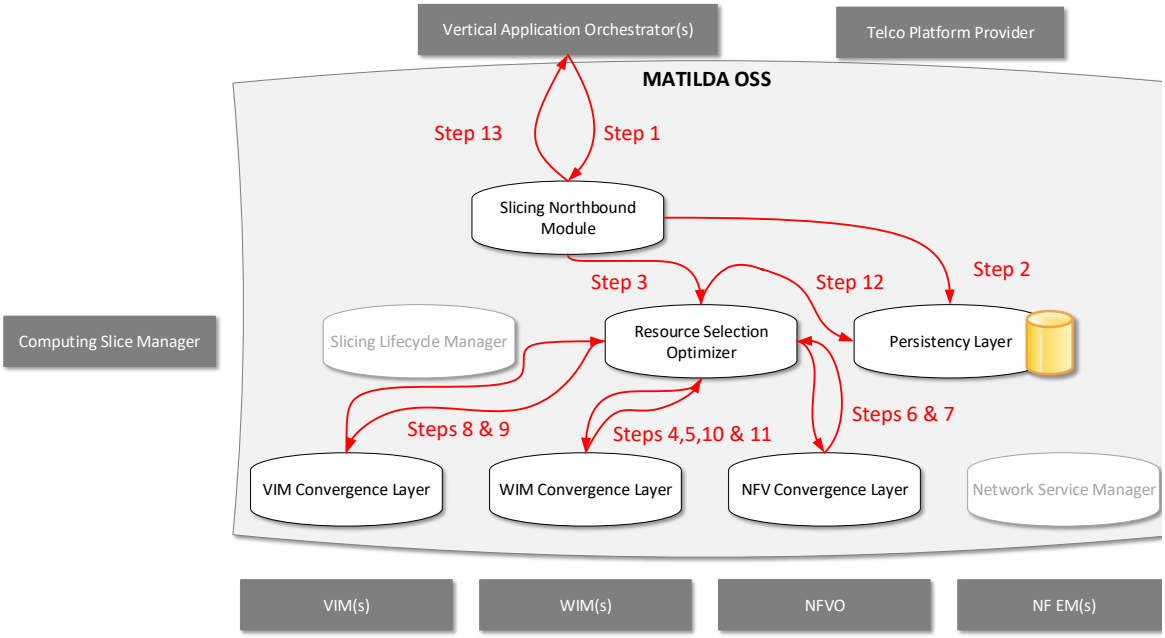

Figure 3: Phase 1 of the 5G slice creation.

Table 1: Main steps in the phase 1 of the 5G slice creation procedure.

Step #	Description	OSS module
1	A new slice intent message from a VAO is received by the OSS.	Slicing Northbound Module
2	The message is parsed and checked, and the data is stored in the Persistency Layer. In case of malformed message, a message error is raised back to the VAO, and the setup procedure is stopped.	Slicing Northbound Module
3	A trigger message is sent to the Resource Selection Optimizer. The message includes the reference to the slice intent data stored in the Persistency Layer.	Slicing Northbound Module
4	The Resource Selection Optimizer analyses the application graph included in the slice intent, and applies a graph reduction algorithm to group the components that need to be placed in the same VIM. Then, it requests a list of candidate VIMs for the deployment of all the groups to the WIM Convergence Layer. The request includes bandwidth and latency requirements between groups and towards access/UE resources, as well as identification parameters for accessing PNFs (e.g., eNBs) and UEs.	Resource Selection Optimizer

5	The WIM Convergence Layer calculates a number of possible deployment configurations that cope with the requirements at Step 4 and sends them back to the Resource Selection Optimizer. If no configuration is possible, the WIM Convergence Layer raises an alarm to the Resource Selection Optimizer, which on its turn proposes a different "reduced" application graph. If no alternative graphs are possible, an error message is issued to the VAO through the Slicing Northbound Module.	WIM Convergence Layer
6	The Resource Selection Optimizer uses the slice intent data and the reduced graph to select the network services to be deployed to materialize the slice. The Resource Selection Optimizer asks the NFV Convergence Layer for the quota of resources to be reserved for the NSs in each VIM.	Resource Selection Optimizer
7	The NFV Convergence Layer handles the request from the Resource Selection Optimizer by mapping the type of NS, along with its operating and performance requirements, against the NS blueprints available in the selected VIMs. Then, the selected blueprint is used for extracting the amount (e.g., number of vCPUs) and the type (e.g., SR-IoV hardware acceleration) of resources to be reserved for each VIM. If no suitable service blueprints are available, the procedure ends and an error message is issued to the VAO through the Slicing Northbound Module.	NFV Convergence Layer
8	The Resource Selection Optimizer aggregates the amount and the type of resources to be reserved for both the vertical application components and the VNFs, and it asks the VIM Convergence Layer for the availability of such resources at the selected VIMs.	Resource Selection Optimizer
9	Upon the request from the Resource Selection Optimizer at Step 8, the VIM Convergence Layer checks for the availability of the resources at the selected VIMs. The VIM Convergence Layer replies to the Resource Selection Optimizer with an availability report, and if all the requirements are satisfied, the VIM Convergence Layer creates the tenant spaces for the VAO and for the NFVO, reserves the required resource quotas, and includes the identifiers of the tenant spaces in the reply message.	VIM Convergence Layer
10	The Resource Selection Optimizer handles the reply form the VIM Convergence Layer. If one or more VIMs cannot provide the required types or amount of resources, then the Resource Selection Optimizer will restart from Step 6 with a different deployment option. If no alternative deployment options are available, an error message is issued to the VAO through the Slicing Northbound Module. If all the requests are satisfied, the Resource Selection Optimizer asks the WIM Convergence Layer to reserve the wide-area resources for the chosen deployment option. In this case, VIM identifiers related to traffic flows expected to enter and exit from each VIM (e.g., ports, IP addresses, VLAN tags, etc.) will be made explicit in the reservation request.	Resource Selection Optimizer
11	The WIM Convergence Layer reserves the wide-area resources (i.e., paths) for the chosen deployment option.	WIM Convergence Layer

12	Starting from the resources reserved by all the Convergence Layers, the chosen deployment option, and the identifiers of the tenant spaces/resources, the Resource Selection Optimizer stores in the Persistency Layer all the required data to fill the candidate materialized slice message, and to proceed to the slice set up in case of positive response from the VAO.	Resource Selection Optimizer
13	The Slicing Northbound Module collects the data stored by the Resource Selection Optimizer, parses them into a candidate materialized slice message, and sends it to the VAO.	Slicing Northbound Module

Table 2: Main steps in the phase 2 of the 5G slice creation procedure.

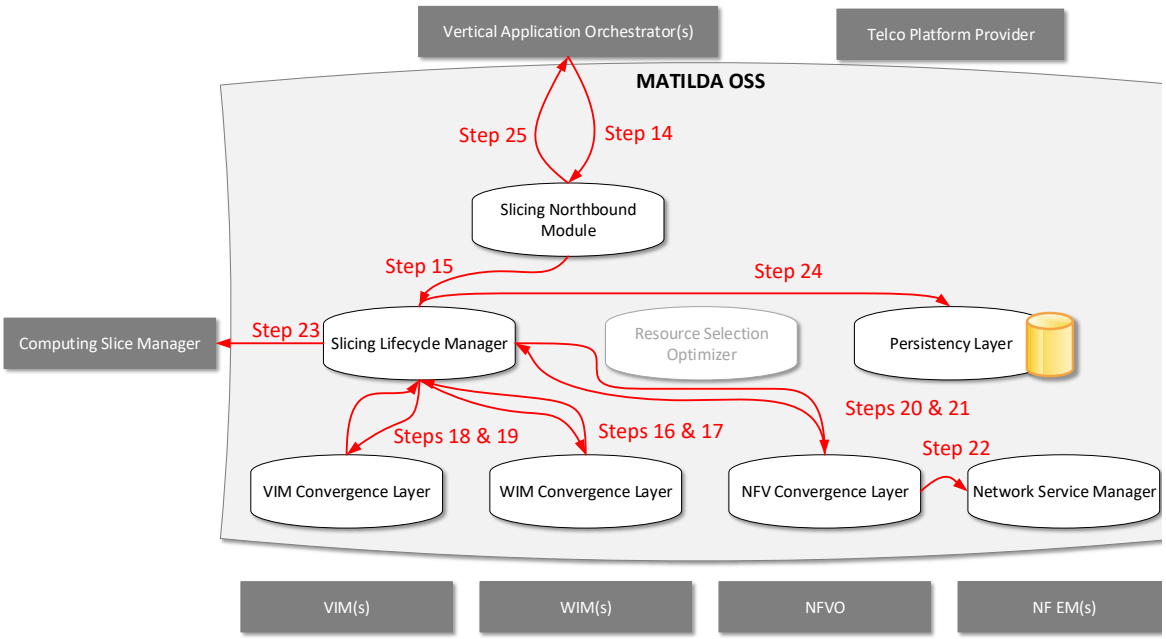

Figure 4: Phase 2 of the 5G slice creation.

Step #	Description	OSS module
14	An acceptance of a candidate materialized slice message from a VAO is received by the OSS. The content of the message is parsed and a consistency control is performed.	Slicing Northbound Module
15	When the Slicing Northbound Module receives the message from the VAO containing the candidate materialized slice acceptance, it triggers the Slicing Lifecycle Manager. If no messages are received, or if the VAO does not accept the candidate materialized slice, the reserved resources at VIMs and WIM are freed, and the status information of the procedure is removed from the Persistency Layer.	Slicing Northbound Module

16	When the trigger from Step 14 is received, the Slicing Lifecycle Manager retrieves all the data related to the slice to be materialized, and asks the WIM Convergence Layer to set up the reserved wide-area paths.	Slicing Lifecycle Manager
17	When the wide-area paths are successfully instantiated, the WIM Convergence Layer sends an acknowledgment message to the Slicing Lifecycle Manager. If setup operations are not completed successfully, the procedure ends, and an error message is issued to the VAO through the Slicing Northbound Module.	WIM Convergence Layer
18	Upon the reception of the acknowledgement from the WIM Convergence Layer, the Slicing Lifecycle Manager requests the VIM Convergence Layer to setup the attach points in all the involved VIMs to be used between vertical application and NFV tenant spaces, as well as the external network resources (e.g., floating IP addresses) to be attached with wide-area paths.	Slicing Lifecycle Manager
19	The VIM Convergence Layer creates the VIM overlay networks to be used as attach points, and configures the access rules for allowing the Vertical Application and the NFVO tenants to access them. It also creates all the needed resources/configurations for external networks to interconnect with other VIMs or with the mobile access. Upon the successful fulfilment of all the setup operations, the VIM Convergence Layer sends an acknowledgement message to the Slicing Lifecycle Manager. If setup operations do not complete successfully, the procedure ends, and an error message is issued to the VAO through the Slicing Northbound Module.	VIM Convergence Layer
20	Upon the reception of the acknowledgment at Step 18, the Slicing Lifecycle Manager asks the NFV Convergence Layer for the setup of all the needed NFV services.	Slicing Lifecycle Manager
21	The NFV Convergence Layer retrieves from the Persistency Layer all the data related to the NSs to be setup. For each service, it extracts the service blueprint that was chosen at Step 7, and fills it with additional deployment configuration data to produce a valid NFV service descriptor for each VIM involved with the service. When a service is successfully instantiated, the NFV Convergence Layer sets up a Network Service Module for the post-boot configuration, monitoring, and dynamic management of all the network elements composing the service. If setup operations do not complete successfully, the procedure ends, and an error message is issued to the VAO through the Slicing Northbound Module.	NFV Convergence Layer
22	The Network Service Module is meant to provide post-boot configuration, monitoring, and dynamic management of all the network elements composing the service. As defined by ETSI NFV, these operations are performed through the Element Manager interfaces and protocols.	Network Service Manager
23	Upon the successful fulfilment of the NFV Convergence Layer operations at Step 20, the Slicing Lifecycle Manager communicates to the Computing Slice Manager the list of VIMs and tenants spaces to be made accessible to the vertical.	Slicing Lifecycle Manager
24	The Computing Slice Manager takes care of the deployment of the application graph components in the assigned VIMs, as well as their management towards all their lifetime. Management regards a set of actions related to scaling, lifecycle actions (restart, healing) and real-time configurations.	Computing Slice Manager

25	When the Computing Slice Manager communicates the completion of its operations, the Slicing Lifecycle Manager stores all the data related to the slice in the Persistency Layer, and triggers the Slicing Northbound Module.	Slicing Lifecycle Manager
26	The Slicing Northbound Module notifies to the VAO that the materialized slice is up and ready to be used.	Slicing Northbound Module

3.2 The Slicing Northbound Module

The role of the Slicing Northbound Module is twofold. On the one hand, it is in charge of the interaction between the VAO and the OSS for the exchange of the slice intent and materialized slice messages, on the other hand, this module also stores the information contained in such messages into the Persistency Layer. The communication between the VAO and the OSS/BSS is performed through a set of APIs. At the current stage of the Project, the development has covered the messages related to the slice intent, while activities on the materialized slice are still ongoing.

3.3 The Resource Selection Optimizer

The Resource Selection Optimizer is charged with the optimal selection of resources inside the virtual infrastructure and across the wide-area interconnected infrastructure, with an additional scope to support network services that meet application-level slice requirements.

Its services are requested either by the VAO directly via the Slicing Northbound Module to prepare the accommodation of a new slice request or the modification of an existing one, or by network events that trigger a slice replacement (such is the cases of faults, maintenance operations, etc…).

Slice reservation is retained upon slice expiration timeout or a direct request by the VAO.

The Resource Selection Optimizer is critical in the preparation of a new slice since it optimizes and, in a sense, assumes a coordinating role in the process of translating a slice intent request by the Slicing Northbound Module to the various convergence layers (i.e. VIM, WIM, NFVO) before passing the control of the reserved slice to the Slicing Lifecycle Manager.

The steps in which the Resource Selection Optimizer is involved can be found in Table 1; the main Tasks that the Resource Selection Optimizer undertakes are:

- Task 1: Optimal placement for groups of application components inside the VIM infrastructure, and subsequently which WIMs can accommodate such placements given resource and performance constraints.
- Task 2: Optimal mapping of the NSs required by the application itself to the resources of the VIMs marked available in the previous task and, as a result, selection of a VIM capable of supporting the application in terms of both components deployment and NFV resources availability.
- Task 3: Collection and persistence of all the mappings made in the previous two tasks to be parsed by the Slicing Northbound Module them into a candidate materialized slice message.

3.4 The Slicing Lifecycle Manager

The Slicing Lifecycle Manager represents the counterpart of the Resource Selector Optimizer, in the sense that it takes charge upon acceptance of a candidate materialized slice from a Vertical Application Orchestrator. At this point, it is the responsibility of the Slicing Lifecycle Manager to request the activation of reserved resources and services.

To do this, the Slicing Lifecycle Manager interacts with the WIM, VIM and NFV Convergence Layers to set up, the reserved wide-area paths, the attach points between vertical application and NFV tenant spaces, and all the needed NFV services. The sequentiality of these set up operations is required because the selection of the proper network services is bound to the selection of the wide-area paths and VIMs.

The Slicing Lifecycle Manager is also in charge of providing the Computing Slice Manager with the list of VIMs and tenants spaces to be made accessible to the vertical, as well as storing the data related to the slice in the Persistency Layer for the Slicing Northbound Module.

The implementation of these operations is still under development and is currently oriented towards the exploitation of Create, Read, Update, and Delete (CRUD) functions. Suitable algorithms for the optimization of the operations are under development as well.

3.5 The VIM Convergence Layer

The intercommunication between the Vertical Industry and the Telecom Infrastructure Provider domains is still an open issue for the involved working groups [6]. It is clear that Telecom Infrastructure Providers need to expose an abstract view of the available resources, in order to provide verticals with the information needed to deploy their services. However, this goal must be achieved without plainly exposing their infrastructures.

To do this, a so called “attach point” is required to terminate the network slice, assigned to the vertical application, and abstract the underlying infrastructure. Although this concept is widely accepted, its realization, as already mentioned in the D1.1 report, is far from being formalized. The relevance of this feature becomes manifest by considering that the provisioning of NFV services should actually entail multiple heterogeneous datacenters, providing different resources, in order to guarantee satisfying scalability and performance levels.

After receiving a slice intent and while the reservation procedure is taking place, the Resource Selection Optimizer considers the service graph, containing the list of the resources and related requirements, and it groups VMs and corresponding links, according to their constraints, to be deployed together.

For example, as illustrated in the figure below, if two application components need to be close to each other in order to satisfy a certain constraint, it will be required to deploy them (along with their corresponding links) in the same VIM. While defining these aggregations of resources, the Resource Selection Optimizer also identifies the NSs required to interconnect them across VIMs. These include the NSs needed to interconnect the UE in the access network to the vertical application. Both application resources (i.e., components and links) and NSs need to be taken into account when evaluating the amount of free resources needed for the vertical application deployment.

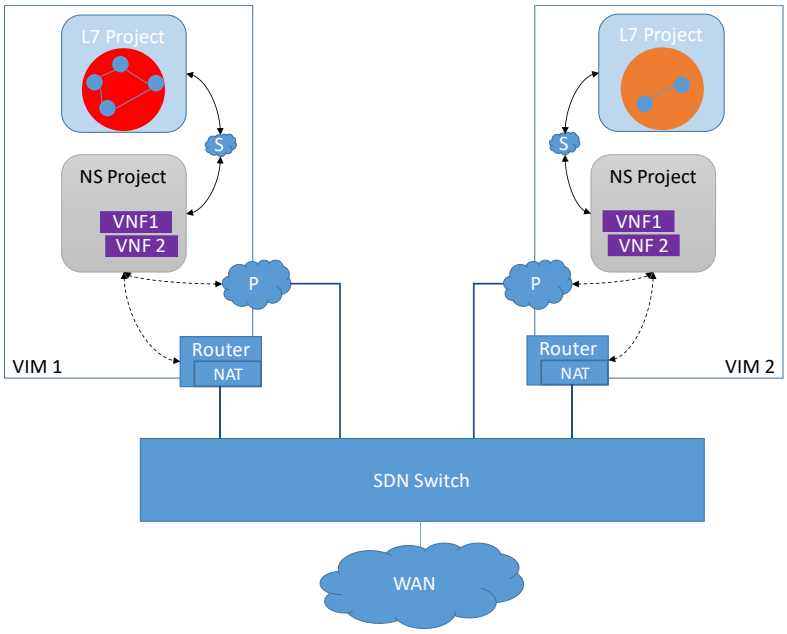

Figure 5: Example of deployment of a vertical application in two datacenters and the related attach points.

Then, once the candidate slice has been received, each selected VIM proceeds with the creation of the projects and their interconnection. A project is created for the vertical application, which contains the VMs and links composing the application graph. Moreover, another project is created for the NSs needed to provide interconnectivity among the vertical application component, deployed in different VIMs and between the UE and the vertical application. Clearly, the former is managed by the VAO owning the application, while the latter belongs to the NFVO.

In each involved VIM, a self-service network is already available at this phase. Self-service networks allow users to create their own topology (i.e., bridges between VMs, VLANs, and virtual routers). Once the corresponding projects have been created, the isolation of each vertical is achieved thanks to the Role-Based Access Control (RBAC) framework. Self-service networks are shared either between the vertical application and the access network to provide the slice with the 4/5G connectivity, or between the vertical application and NS projects for specific networking requirements of the slice.

Provider networks differ from self-service networks in the way they are directly associated with the physical network. For this reason, they can only be set up by an admin. Projects can use them and they can be shared by multiple projects via RBAC as well. But unlike self-service networks, their topology/components can't be modified. Provider networks can be used as attach points, but they can only provide layer-2 connectivity, typically associating projects with VLAN tags that correspond to the VLANs present in the physical network. Routers in OpenStack provide virtual layer-3 services, such as routing and NAT, to self-service networks belonging to a project; vertical applications can be directly connected to the WAN using floating IP addresses. The choice between these two solutions depend on the level of connectivity envisaged, and the level of dependency to the chosen NFVO technology.

3.6 The WIM Convergence Layer

The WIM provides a connectivity actuator module that acts as a convergence layer to the MATILDA framework. This module comprises all the necessary plugins to interface the network nodes (configuration, monitoring, etc.).

Among the most important plugins we can list the following:

- OpenFlow, for handling OpenFlow switches.
- XRPC, more suitable than OpenFlow for optical equipment.
- SNMP, the traditional Network and Management Protocol.
- Netconf, the emerging Network Configuration Protocol with improved network functions.

The different plugins are coordinated by the Network Service Actuator module that is in charge of paths implementations, reading the topology information, and addressing the proper network elements with the required operations (connections creation or removal, new paths signalling, etc.). These operations are performed to select the proper plugin and address the different nodes according to the MD-SAL (Model-driven Service Abstraction Layer).

After the successful conclusion of the operation, the Network Service Actuator module updates the other parts of the system with the topology and traffic variations, otherwise it is responsible for the rollback operation to achieve, again, a network stable condition and an updated controller information state.

The Network Service Actuator interfaces with: *i*) the Protection Manager that handles the defined protection mechanisms; *ii*) the Connectivity Manager that handles all the Path Creation, Cancellation, and Modification requests; and *iii*) the Connectivity Transformer that receives all the requests from the NBI and translates them to the internal formats.

3.7 The NFV Convergence Layer

The MATILDA NFV Convergence Layer provides a level of abstraction regarding the orchestration technology, aligning the system to the modular approach taking advantage of the possibility of plugging in a new orchestrator (in case of being a future requirement) without modifications in the whole OSS. The responsibilities of this layer are:

1. To keep the communication with the Resource Selection Optimizer, during the slice intent processing, producing the list of resources to be reserved on each VIM for the slice deployment.
2. To receive from the Slicing Lifecycle Manager the slice at the instantiation phase to deploy and apply the initial configuration to the NSs that compose the slice.
3. To consume the update/remove messages from the Slicing Lifecycle Manager in the operation/decommission phase and produce the actions in the NFVO to satisfy these requests.

The NFV Convergence Layer is internally composed of two modules, as shown in Figure 7: the NFV Service Mapper, to manage the tasks described above in point 1; and the NFV Service Setup, which cover tasks related to points 2 and 3.

3.7.1 NFV Service Mapper (NFVSM)

The Vertical Application Provider, when creating a vertical application, is not totally aware of the existing network infrastructure, as it does not belong to him but to the Telecom

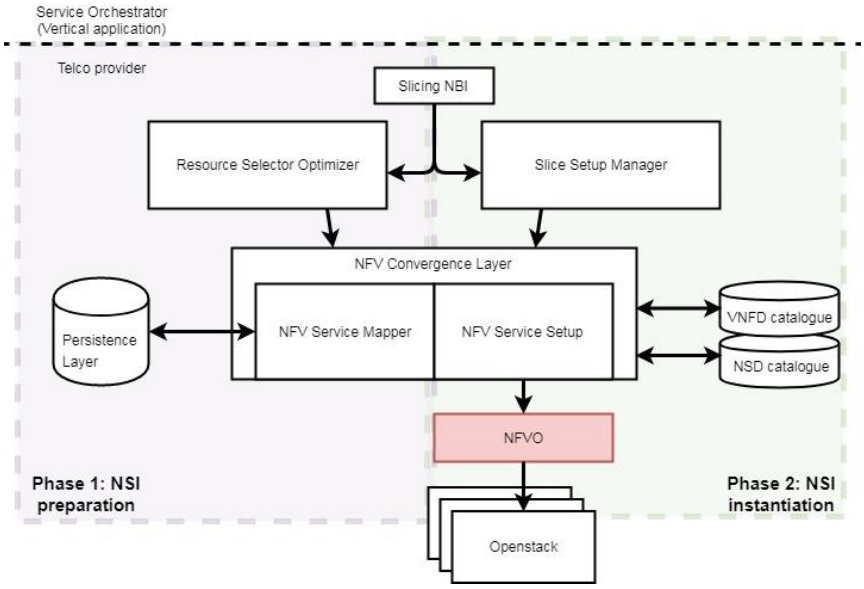

Figure 7: NFV Convergence Layer close iterations.

Infrastructure Provider. During the definition phase of the vertical application, the Vertical Application Provider describes the virtual links between the application components, specifying QoS constraints and enhanced behaviour beyond a straight link. This is abstracted through a tag offered by the Graph Composer, which indicates a logical NS; the task of the NFV Service Mapper is to resolve this logical NS into a real one, or a concatenation of NSs within the Infrastructure Provider NS catalogue, fulfilling, in turn, the QoS constraints expressed by the Vertical Application Provider in the slice intent [7]. All this is done upon request of the Resource Selector Optimizer, which needs, as an input from the NFV Service Mapper, not only the final NS that will be instantiated on the slice, but also the hardware resources to be provisioned in order to deploy the NS properly. For this, a more in-depth search is necessary, finding out which VNFs compose the selected NS and return the resources to be consumed.

3.7.2 NFV Service Setup (NFVSS)

The NFV Service Setup (hereinafter NFVSS) module is responsible for the Network Slice Instances (hereinafter NSIs) instantiation, operation and deprovision phases.

For that purpose, VNF management functions will be inherited from Open Source MANO (OSM). The NFVSS will communicate with the NFVO using the Os-Ma-Nfvo interface [8], whose RESTful protocols specification is described in ETSI GS NFV-SOL 005 [9] and implemented in the OSM northbound interface component.

In Figure 8, it is possible to see the processes in which the NFVSS is involved. During the NSI instantiation phase (with the green background in Figure 8), the NFV Convergence Layer handles the instantiation requests from the Slicing Lifecycle Manager and passes the request to the NFVSS to manage the service instantiation and initial configuration with the purpose of deploying all resources negotiated in the NSI preparation, allowing all the necessary configuration without manual actions and providing the attach points for network flows to be terminated.

Preparation

For this purpose, the NFVSS module should consume the frame from the Slicing Lifecycle Manager. This frame should be a copy of the message created to propose the slice to the Service Mesh Orchestrator.Parsing this frame, the NFVSS will have all the NS information required for the deployment and operation of the VNF forwarding graph in a multi-site way if in the request is ordered, leveraging OSM R4 capacity for multi-site deployments.

It is worth noting that in OSM, the NS onboarding process starts adding the VNF descriptor(VNFD) package to the internal catalogue. In MATILDA, this process is automated at the design phase of the VNFD, thus each time a new VNFD or a new version of one of them is developed, it will be packaged and onboarded in OSM.

For the NS descriptor (NSD), there is a fixed part that describes the logical function and will be developed in the design phase. The dynamic parts of the NSD are the deployment parameters that the Slicing Lifecycle Manager has requested to be created on each VIM, e.g., VIM tenants and networks to attach the VNFs.

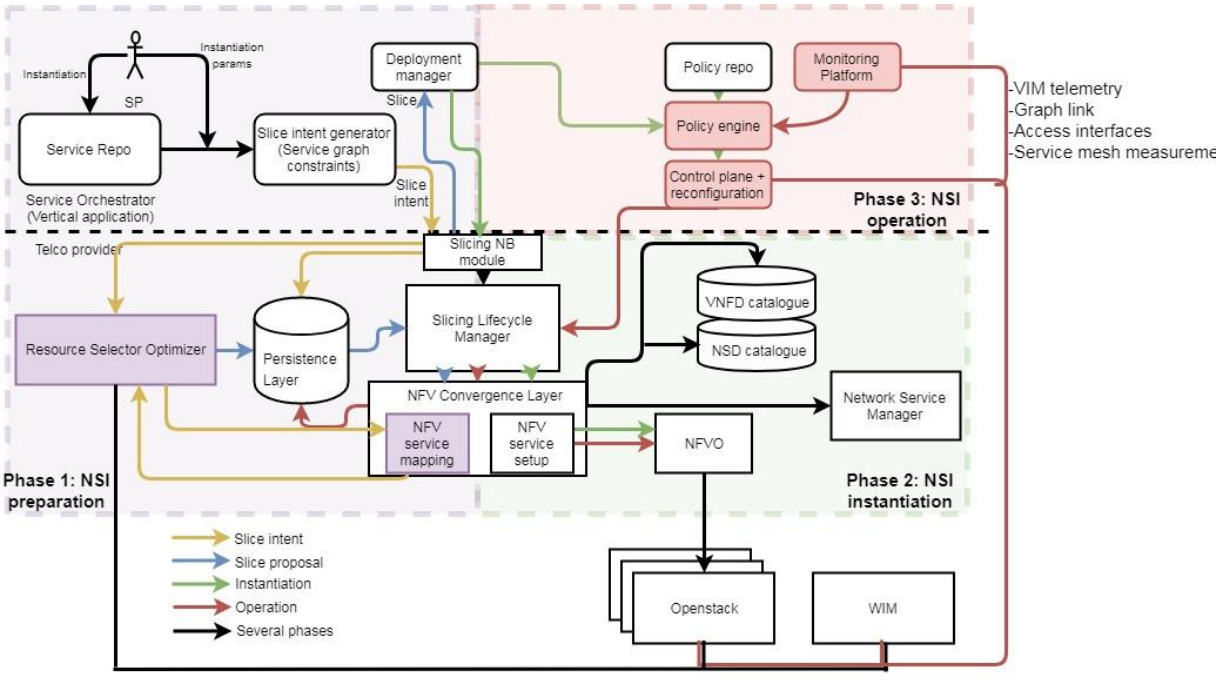

Figure 8: Overview of the NFVSS process.

Onboarding and Instantiation.

Once the placement has been decided and the frame has been parsed, the NFVSS should retrieve the NFV Service blueprint from the catalogue for each Network Service that composes the slice, and fill the information required:

1. VNF multisite deployment to specify the Point of Presence (PoP) where each component needs to be deployed. Matching each member-vnf-index with the vim_account that was registered in the preparation step.
2. Virtual links: It is needed to specify, for each PoP, the attach point where the VNF should be deployed, i.e. the vim-network-name.

With the blueprint in OSM NSD format, we can proceed to the onboarding in OSM, that is adding the object to the catalogue, and once added the instantiation can be performed. With both descriptors uploaded to OSM, the NS is prepared to be instantiated, and the Day 1 (operations) and Day 2 (optimizations) configuration can be applied to the deployed services, which is the second phase where the NFVSS should intervene is the NSI operation (red background colour in Figure 8). At this point, thanks to the monitoring platform that executes performance measurements of the status of the slice, it is possible to provide the policy engine with the input allowing it to perform deployment optimization: horizontal/vertical scaling, health, modification of the VNFFG, etc. This optimization request should be collected by the NFVSS and performed by the NFVO.

Once the slice is instantiated, the Embedding Verification process must be started to assure that all the components in the slice request are working properly, providing a reliable service for the NSI operation phase. After this verification process, as shown in Figure 8, the workflow continues to the Network Service Module, for post-boot net config blocks, where the NFVSS will ask this block for the configuration of the communication to reach the external connectivity between the slice built and the 5G-ready application.

3.8 The Persistency Layer and the Resource and Service Catalogue Manager

The database (see Figure 9) is in charge of hosting both the vertical slices and the resources available in the VIMs. In more detail, the database initially contains the available VNFs, PNFs, wide-area topologies and so on, and maintains the correspondence between these resources and the physical infrastructure hosting them. At this stage, it is not envisaged that the Project will take into consideration real-time onboarding.

During the reservation phase of an application slice, the database stores the slice intent coming from the VAO, and adds the information related to the candidate materialized slice, once it is made available by the resource selector.

The slice intent contains the list of the resources required to compose the vertical application, such as Virtual Machines (VMs) and links connecting them, and a number of requirements: number of CPUs, RAM size, link bandwidth, maximum delay, etc. Such requirements can be hard or soft, with the former group being mandatory for a successful reservation process. The candidate materialized slice has a slightly different structure, as it reports the correspondence between the VMs and the VIMs where they should be hosted, and also whether the requirements have been fulfilled or not.

4 The Wide-Area Infrastructure Manager (WIM)

The Wide-Area Infrastructure Manager (WIM) interworks with the extended OSS/BSS blocks to provide the VIM selection, according to the desired target latency. It will, therefore, suggest to the extended OSS/BSS the best VIM candidates for the specified latency range. After the OSS/BSS first tentative selection, it will perform the final path computation, adding the bandwidth constraints arising from the chosen selection.

All these tasks mainly leverage on the functional blocks provided by a refined controller, based on the Open Day Light (ODL) framework. ODL is an "open" initiative sponsored by a group of companies, comprising Ericsson, which is one of its founders and main contributors. The goal is to provide an open platform for the programmability of the network, to allow its development of SDN and NFV applications on networks of any size [10].

The solution is entirely based on a "Model-Driven SAL" or "MD-SAL" approach (where SAL stands for Service Abstraction Layer): MD-SAL is a service of the ODL platform, which provides a system "bus" that the various modules can use to communicate with each other, synchronizing their operations, transferring information, and eventually making it persistent. For this purpose, the ODL platform provides tools to describe the model of data on which it operates (the YANG language is used) and, starting from the modeling of these data using this language, it allows to automatically generate parsers and formatters for the data represented from the model, define interfaces and communication channels (RPC/Service Routing), persistence services (Data Store) and present the data (Notification subscription and publish services). Added to these are additional services, such as support for the development of user interfaces, and processes of customizable installation, made possible using OSGI technology (Apache Karaf).

At high level, the controller and its applications have the following functional decomposition:

- Multi-Layer Path Computation Engine (ML-PCE)
- South Bound Interface (SBI): the "plugins", according to the ODL terminology, implement the functions to access the network systems, i.e. the protocol interfaces described in Section 3.6. The set of all plugins implements the SBI controller.

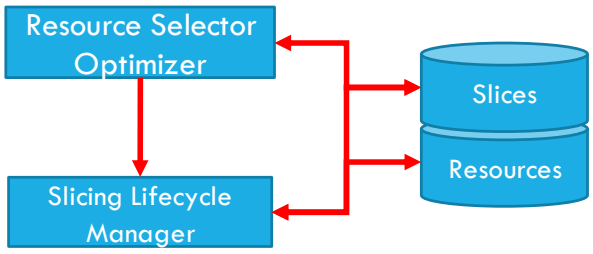

Figure 9: Structure of the Persistency Storage and its interaction with the OSS modules.

- ML-connectivity manager: it is in charge of the definition and management of routing policies. It also performs the coordination of traffic configuration activities

(implemented in the respective plug-ins) and alarms monitoring. Traffic allocation strategies are maintained by a specific module (also part of the connectivity manager).

- ML-MT Topology: it manages the topology database that it acquires from the plug-ins.
- Database Circuits and Topology: The ODL data-store implements the persistence of the whole model and therefore implements all the Circuits and Topology databases.
- North Bound Interface (NBI): NBI is a REST CONF interface derived from the YANG model of controller data, through which the external applications can access data and controller functionality.

5 The Computing Slice Manager (CSM)

The Computing Slice Manager is the building block responsible for the lifecycle management of the application graph, taking care of deployment, operational and deprovisioning aspects. Deployment regards the instantiation of the application components in the provided VIMs through the slice instance, operation regards the enforcement of application component and graph management actions (e.g. horizontal scaling, deprovision of resources, live migration, configuration of an application-specific component, self-healing). Deprovision regards the release of the allocated resources upon the end of the application provision. The Computing Slice Manager allows the Vertical Orchestrator to interact with the resources in the Telecom infrastructure in order to manage the lifecycle of its slice. In the creation of a 5G slice, the involvement of the Computing Slice Manager is required once a candidate materialized slice has been accepted and all the wide-area paths, attach points and NFV services have been properly reserved and configured. Upon completion of these operations, the Slicing Lifecycle Manager provides the Computing Slice Manager with the list of VIMs and tenant spaces to be made accessible to the vertical.

At this stage of the project, the interactions between the Vertical Orchestrator and the materialized slice resources deployed over the multiple VIMs are made available by the Computing Slice Manager by means of HTTP proxying. Further details on its development will be provided in the next deliverable.

6 The Multi-site NFV Orchestrator (NFVO)

The MATILDA multi-site NFVO supports basic management operations on the instantiated NSs through lifecycle management of NSs and VNFs. In parallel, it is going to interact with the extended OSS to support the placement of NSs over the available programmable physical and virtual resources. The latter are parts of the infrastructures that belong to different Infrastructure Providers. The multi-Site NFVO component will be realized on the basis of existing open source solutions meeting the goals of MATILDA. In particular, the OpenSource MANO (OSM) NFVO is going to be adopted as an official and open-source ETSI-hosted project, aiming to develop an Open Source NFV Management and Orchestration (MANO) software stack aligned with ETSI NFV.

OSM has moved to a common message bus for asynchronous communication between components. That makes the stack more open to integrate with new pluggable modules and

eases the centralization of the common services. In release four, multi-site and monitoring features are mature enough to profit them in MATILDA's architecture, while initial policies management features are also implemented. A set of interesting improvements that match in MATILDA's NFVO requirements are:

- The model driven NBI, according to [9] to implement the OSS-NFVO interface.
- Monitoring improvements that allow us publish metrics and alerts in OSM Kafka bus to be consumed by the OSS or even installing the optional monitoring tools: Prometheus or ELK stack to process these events. Collected monitoring probes can be provided to the Vertical Application Orchestrator based on the available Northbound APIs.
- Policies management functionalities supporting policies enforcement on behalf of the telecom operator.
- OSM deployment is now cloud native based, deploying all the components using Docker. This provides resiliency in case of failure of an OSM component.
- Full charms support that is planned for a release 4 "point" update, for the management of the VNFs deployed.

7 Conclusions

The MATILDA framework is composed of three main layers, corresponding to the applicative, the network platform and the infrastructural levels. The network platform layer, which represents the focus of this document is being designed with the goal of supporting the deployment and orchestration of 5G-ready applications and network services within a multi-tenant and multi-domain environment. Such an environment requires the introduction of a number of features and mechanisms to abstract the physical and logical resources for providing Vertical Industries with the information required for designing a slice intent.

The most relevant building block is represented by the extended Operations and Business Support Systems (OSS/BSS). Its centrality is due not only to its favourable position between the Telecom Service Provider and the Vertical Industries, but also to a microservice-driven design that guarantees a rapid provisioning and deployment of additional functionalities. Among others, these functionalities include exchanging the configuration parameters for the instantiation request/reply of network and computing slices, mapping network slicing types onto a set of network services, configuring the activated network services, and creating tenant domains for edge computing in the selected VIM sites.

The communication of the available resources and their reservation to the above layers is performed by interacting with the VAO (Vertical Application Orchestrator) by means of the slice intent and materialized slice metamodels. Southbound, the OSS/BSS interacts with the NFV Orchestrator (NFVO) to request the activation/deactivation/modification of NFV services, and the VNF instances for configuration purposes.

The multi-site NFV orchestrator is in charge of supporting the deployment and operation of the VNF forwarding graphs. It supports basic management operations on the instantiated VNFs by communicating with the respective VNF manager; in parallel, it interacts with the extended OSS/BSS to support the placement of the network services throughout the wide-area.

The decisions on resources to be allocated across the multi-domain infrastructure are taken by the Wide-Area Infrastructure Manager (WIM) that identifies the potential VIMs according to bandwidth and latency requirements within service components and towards access/UE resources, while supporting infrastructure abstraction and isolation of the multiple tenants.

Verticals are enabled to perform operations on applications deployed over multiple mobile edge hosts thanks to the Computing Slice Manager, which provides the list of VIMs and tenant spaces to be made accessible to the Vertical and, upon completion of a slice materialization, allows interacting with the materialized slice.

References

[1] 3GPP, "5G; System Architecture for the 5G System", Technical specification no. 23.501, version 15.2.0, Release 15, ETSI TS 123 501, June. 2018.

[2] ETSI "Network Functions Virtualisation (NFV), Management and Orchestration," ETSI GS NFV-MAN 001 V1.1.1,
URL: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[3] MATILDA Deliverable D1.1, "MATILDA Framework and Reference Architecture".

[4] MATILDA Deliverable D1.2, "Chainable Application Component & 5G-ready Application Graph Metamodel".

[5] MongoDB, URL: www.mongodb.com/.

[6] ETSI-MEC, URL: https://www.etsi.org/news-events/news/1218-2017-09-news-etsi-multi-access-edge-computing-releases-new-white-paper-and-starts-work-on-interoperability.

[7] MATILDA Deliverable D1.4, "Network-aware Application Graph Metamodel".

[8] https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/013/02.01.01_60/gs_NFV-IFA013v020101p.pdf.

[9] https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/02.04.01_60/gs_NFV-SOL005v020401p.pdf.

[10] ODL website: http://www.opendaylight.org/software.