H2020 5G-TRANSFORMER Project
Grant No. 761536

Final design and implementation report on service orchestration, federation and monitoring platform (report)

Abstract

This deliverable provides the final specification of the Service Orchestrator architecture. In particular, it describes its key features and functional components, workflows, and specification of interfaces including the northbound interface towards the vertical slicer, southbound interface towards the mobile transport and computing platform, and east/westbound interface to the federated service orchestrators. Moreover, this deliverable describes the software design and the implementations of the Service Orchestrator along with the Service Monitoring Platform prototypes.

Document properties

Document number	D4.3
Document title	Final design and implementation report on service orchestration, federation and monitoring platform
Document responsible	Oleksii Kolodiazhnyi (MIRANTIS)
Document editor	Oleksii Kolodiazhnyi (MIRANTIS)
Editorial team	Arturo Zurita (ATOS), José Enrique Gonzalez (ATOS), Barbara Martini (SSSA), Kiril Antevski (UC3M), Samer Talat (ITRI), Josep Mangues (CTTC) Jorge Baranda (CTTC), Marco Capitani (NXW), Giada Landi (NXW), Oleksii Kolodiazhnyi (MIRANTIS), Xi Li (NECLE), Carla Fabiana Chiasserini (POLITO), Pantelis Frangoudis (EURECOM)
Target dissemination level	Public
Status of the document	Final
Version	1.0

Production properties

Reviewers	Li Xi (NEC), Barbara Martini (SSSA), Thomas Deiss (NOK-N), Carlos J. Bernardos (UC3M)

Disclaimer

This document has been produced in the context of the 5G-TRANSFORMER Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement Nº H2020-761536.

Table of Contents

List of Contributors

Partner Short Name	Contributors
ATOS	Arturo Zurita, José Enrique Gonzalez
CTTC	Jorge Baranda, Josep Mangues, Ricardo Martínez, Luca Vettori, Manuel Requena, Javier Vílchez, Engin Zeydan
Eurocom	Adlen Ksentini, Pantelis Frangoudis
ITRI	Samer Talat
Mirantis	Oleksii Kolodiazhnyi, Konstantin Tomakh
NEC	Xi Li, Andres Garcia-Saavedra, Josep Xavier Salvat Lozano
NXW	Giada Landi, Marco Capitani, Francesca Moscatelli
Polito	Carla Fabiana Chiasserini
SSSA	Luca Valcarenghi, Barbara Martini, Silvia Fichera
UC3M	Kiril Antevski, Carlos J. Bernardos, Antonio de la Oliva, Jaime Garcia, Jorge Martin, Borja Nogales

List of Figures

List of Tables

List of Acronyms

Acronym	Description
5GC	5G Core
5GT-MTP	Mobile Transport and Computing Platform
5GT-SO	Service Orchestrator
5GT-VS	Vertical Slicer
AAA	Authentication, Authorization, Accounting
AN	Access Network
API	Application Programming Interface
AppD	Application Descriptor
BSS	Business Support System
CAT	Catalogue
CN	Core Network
CSMF	Communication Service Management Function
DB	Database
DF	Deployment Flavor
E2E	End to end
EBI	Eastbound Interface
EM	Element Management
EPC	Evolved Packet Core
EPCaaS	EPC as a Service
ETSI	European Telecommunication Standardization Institute
GRE	Generic Routing Encapsulation
GS	Group Specification
HSS	Home Subscriber Server
IaaS	Infrastructure as a Service
IFA	Interfaces and Architecture
KPI	Key Performance Indicator
LC	Lifecycle
LCid	Lifecycle Operation Occurrence Id
LCM	Lifecycle Management
M&E	Media and Entertainment
M(V)NO	Mobile (Virtual) Network Operator
MANO	Management and Orchestration
MEC	Multi-access Edge Computing
MEO	Multi-access Edge Orchestrator
MEP	Multi-access Edge Platform
MILP	Mixed Integer-Linear Programming
MIoT	Massive Internet of Things
MLPOC	Multiple Logical Point of Contact
MME	Mobility Management Entity
MNO	Mobile Network Operator
MON	Monitoring
MVNE	Mobile Network Enabler
NaaS	Network as a Service
NBI	Northbound Interface
NF	Network Function
NFP	Network Forwarding Path
NFV	Network Function Virtualization

NFVI	Network Functions Virtualisation Infrastructure
NFVIaaS	NFVI as a Service
NFV-NS	NFV Network Service
NFV-NSaaS	Network Service as a Service
NFV-NSO	Network Service Orchestrator
NFVO	NFV Orchestrator
NFVO-RO	Resource Orchestrator
NS	Network Slice
NSD	Network Service Descriptor
NS-DF	Network Service Deployment Flavor
NSI	Network Slice Instance
NSMF	Network Slice Management Function
NS-OE	NFV-NS Orchestration Engine
NSSI	Network Slice Subnet Instance
NSSMF	Network Slice Subnet Management Function
NST	Network Slice Template
ONAP	Open Network Automation Platform
OSM	Open Source MANO
OSS	Operating Support System
PM	Performance Management
PMON	Performance Monitoring
PNF	Physical Network Function
PNFD	PNF Descriptor
PoP	Point of Presence
QoS	Quality of Service
RAM	Resource Advertisement Management
RAN	Radio Access Network
REST	Representational State Transfer
RM	Resource Management
RMM	Resource Monitoring Management
RO	Resource Orchestration
RO-EE	RO Execution Entity
RO-OE	RO Orchestration Engine
SAP	Service Access Point
SBI	Southbound Interface
SDK	Software Development Kit
SDO	Standard Developing Organization
SLA	Service Level Agreement
SLPOC	Single Logical Point of Contact
SLPOC-F	Single Logical Point of Contact for Federation
SO	Service Orchestrator
SPGW-C	Serving/Packet Data Network Gateway Control Plane
SPGW-U	Serving/Packet Data Network Gateway User Plane
SST	Slice/Service Type
TD	Technology Domain
TETRA	Terrestrial Trunked Radio
TN	Transport Network
TOSCA	Topology and Orchestration Specification for Cloud Applications
TSP	5G-TRANSFORMER Service Provider
UC	Use Case

UE	User Equipment
UPF	User Plane Function
VA	Virtual Application
vEPC	virtual Evolved Packet Core
VIM	Virtual Infrastructure Manager
VL	Virtual Link
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	VNF Descriptor
VNFFG	VNF Forwarding Graph
VNFFGD	VNFFG Descriptor
VSD	Vertical Service Descriptor
VSI	Vertical Service Instance
WBI	Westbound Interface
WIM	Wide area network Infrastructure Manager
YAML	YAML Ain't Markup Language

Executive Summary and Key Contributions

This deliverable provides the final specification of the 5G-TRANSFORMER Service Orchestrator (5GT-SO) architecture, key features and functional components, and workflows, extending from the initial design provided in D4.1 [13]. The 5GT-SO related interfaces also have been extended to support these new features and interaction between the 5GT-SO and the Vertical Slicer (5GT-VS), and the Mobile Transport and Computing Platform (5GT-MTP). These new features and extension of interfaces along with the information models have been implemented and included to the new 5GT-SO release (R2) along with D4.4 [14].

To summarize, the main contributions in this deliverable include the following:

- The final 5GT-SO architecture (Section 2.1), in terms of key features, functional components, management and operational interfaces, internal procedures for modelling. The final design is refined based on the initial design from D4.1. The major refinements include: (i) use of monitoring services from the developed monitoring platform in WP4 to support service scaling and SLA management; (ii) service and resource federation capabilities; and (iii) interface specification towards end-to-end service SLA management.
- The specification of novel or new extended features introduced in the 5GT-SO system (Section 2.2), including service scaling, network service composition, service federation, enhanced placement algorithms considering location constraint and MEC support, and enhanced service monitoring platform which provides monitoring data to the 5GT-SO for automated service scaling and SLA management.
- The introduction of main functional components in the implemented architecture of the 5GT-SO system (Section 2.3), based on the 5GT-SO functional architecture shown in Section 2.2, but presenting the software implementation of the system. The 5GT-SO software development follows a modular design, where the service orchestrator platform is structured in functional module components that interact with each other internally via REST APIs. The 5GT-SO consists of a “Service Manager (SM)” module connecting to two MANO platforms as orchestration engines for managing compute resources，Open Source MANO (OSM) and Cloudify. The SM is the “brain” of the 5GT-SO to integrate the different MANO platforms with individual wrappers tailored to each MANO platform providing translation of formats and workflows, in this way allowing multi-MANO platform compatibility. The implementation of the 5GT-SO prototypes is reported in Section 6.8, highlighting the detailed functionalities developed in its second release (R2) and the full implementation is available for download at: https://github.com/5g-transformer/5gt-so.
- The description of updated 5GT-SO workflows (Section 3 and Section 6.7) for describing the basic service management operations: service on-boarding, service instantiation and termination, service scaling, service federation, and service monitoring, based on the ones reported in D4.1 [13].

These workflows illustrate the interaction among the different components inside the 5GT-SO internally, and also between them and 5GT-VS, 5GT-MTP or external 5GT-SOs.The changes reflect both the architectural concept updates and feedback from the implementation phase.

- The introduction of the developed 5GT-SO monitoring platform design (Section 2.5) and prototype implementation (Section 4). The overall architecture of the service monitoring framework is still aligned with the major design principles and functional components defined in D4.1 [13]. The monitoring platform prototype has been implemented by integrating a Prometheus backend with the 5GT-SO through the development of an adapter called Monitoring Configuration Manager, which acts as a unified entry point for all the configuration functionalities offered by the monitoring platform. The full implementation is available for download at: https://github.com/5g-transformer/5gt-mon.

It should be noted that Section 2 provides an overview of the new functionalities introduced in the 5GT-SO R2. However, in order to provide also a complete and self-contained document that does not require the previous knowledge of D4.1, in appendix (Section 6) we also provide the final specification of the 5GT-SO that integrates all the updates from D4.1 and refinements defined after the first release of the Service Orchestrator. Thus, this appendix can be considered as an independent document that defines the full architecture of the 5GT-SO.
The 5GT-SO reference implementation will be adopted in the public demonstrations of the 5G-TRANSFORMER use cases in WP5. The SO prototype has already been showcased in several public events (EuCNC 2018, MWC 2018, and ICT 2018), with ongoing preparation for demonstrations at EuCNC 2019 and it will be also taken as the baseline service orchestration platform to be extended and integrated into the platform for the upcoming 5G-PPP phase 3 5GROWTH project.

1 Introduction

The 5GT Service Orchestrator (5GT-SO) is one of the key components of the 5G-TRANSFORMER (5GT) system, in charge of instantiation and management of vertical services in terms of deploying NFV-Network Services (NFV-NS) and managing its service life-cycle. Depending on the service requests received from the Vertical Slicer (5GT-VS), the 5GT-SO offers service and resource orchestration and federation across one or multiple administrative domains. These functions include all tasks related to coordinating and providing the vertical with an integrated view of services and resources from one or multiple administrative domains. In particular, service orchestration entails managing end-to-end services that may split into various network domains (RAN, transport, and mobile core) or technology domains (of various wireless and wired technologies, data centers, MEC domains) based on requirements and availability. Federation entails managing relations at the interfaces between the 5GT-SOs belonging to different administrative domains, which are owned by different 5GT-Service Providers.

The main function of the 5GT-SO is based on the NFV Orchestrator (NFVO) as defined in ETSI NFV. Depending on the requests from verticals, both network service (NFVO-NSO) and resource (NFVO-RO) orchestration may be used for both single and multiple domains. The NFV-NSO functionality of the 5GT-SO is responsible of deploying and managing the NFV network services (NFV-NS) requested by 5GT-VS, possibly across multiple domains, and it is also responsible for the network service lifecycle management including operations such as service on-boarding, instantiation, scaling, termination, and management of the VNF forwarding graphs associated to the network services. Moreover, it also decomposes the end-to-end network services into constituent network service segments and decides to deploy them either in the local administrative domain or federating with the peering administrative domain. On the other hand, the NFVO-RO functionality of the 5GT-SO orchestrates the infrastructure resources (physical or virtual) provided in the local domain. More specifically, it decides the optimum placement of Virtual Network Functions (VNFs) and/or Virtual Applications (VAs) and the required resources to be allocated, based on the abstracted resource view of the local domain as well as of the ones from peering administrative domains.

The initial specification of the 5GT-SO has been reported in D4.1 [13], which introduced the 5GT-SO internal design and main functionalities, its related Northbound and Southbound Interface (NBI and SBI) as well as the Eastbound and Westbound Interface(EBI and WBI), and its workflows for basic service life cycle management operations such as service instantiation and termination. This D4.3 deliverable reports the final design and implementation of 5GT-SO which was extended from the initial design presented in D4.1 [13] and D4.2 [15] with a number of added new features for SLA and service management. The main extended features of the 5GT-SO are service scaling, network service composition, service federation, enhanced placement algorithms considering location constraint and MEC support, and enhanced service monitoring platform which provides monitoring data to the 5GT-SO for service monitoring and SLA management. In addition, the 5GT-SO interfaces also have been enhanced to support these new features and interaction at the 5GT-SO NBI towards the Vertical Slicer (5GT-VS), and at the SBI towards the Mobile Transport and Computing Platform (5GT-MTP). These new features and extension of interfaces along with the information models have been implemented and included to the new 5GT-SO release (R2), see D4.4 [14], on top of the first 5GT-SO release (R1). The evolution of the 5GT-SO architecture is reported in Section 2.1. The new 5GT-SO features and summarized in Section 2.2, and Section 2.3 describes the implemented functional

components and information models in the 5GT-SO. Section 2.4 reports the updates on the interfaces of 5GT-SO on its NBI, SBI and E/WBI, and Section 2.5 reports the updated monitoring platform design. Section 3 summarizes the updates on the basic workflows focusing on the ones being implemented. Section.4 . reports the final 5GT-SO monitoring platform prototype implementation.

The Annex in Section 6.8 provides the details on the reference implementation of the 5GT-SO software prototypes developed in WP4 and released in D4.4 [14]. The reference implementation of the 5GT-SO software prototype will be further integrated in the demonstration of the use cases in the context of WP5 activities and the related 5GT-SO performance KPIs will be also evaluated in different vertical use cases within WP5.

In addition to the reference implementation, one additional 5GT-SO implementation has been developed by ITRI, which is also aligned with the 5GT-SO functional architecture while the implementation focusing on only basic operations for deployment of communication services based on standardized Slice/Service Types (SSTs) defined in 3GPP specifications. It is described in the Annex Section 7.

In this deliverable, the main body of the document is mainly focused on providing a summary of the updates introduced in the second release of the 5GT-SO design and implementation. However, in appendix (Section 6) we include a completed and detailed updated version of D4.1 that integrates and extensively discusses these modifications within the overall architecture specification and prototype implementations. This annex can be considered as a self-contained document that provides the full specification of 5GT-SO architecture and software prototypes.

2 Final 5GT-SO Architecture

2.1 Final architecture overview

The final 5GT-SO architecture is aligned with the overall design principles and subsystems specified in R1 and described in D4.1 [13]. Some refinements have been applied to the new 5GT-SO release R2 according to an incremental approach in order to specify more details to the supported mechanisms, foreseen features and functionalities, and to show the evolution of some building blocks and interactions as result of lesson learnt from the implementation. The revised high-level 5GT-SO R2 architecture is shown in Figure 1.

More specifically, major refinements of the 5GT-SO R2 architecture regard the (i) use of monitoring services from the monitoring platform for service scaling and SLA assurance; (ii) service and resource federation capabilities; and (iii) interface specification toward SLA management. Each of these points is elaborated in the follows.

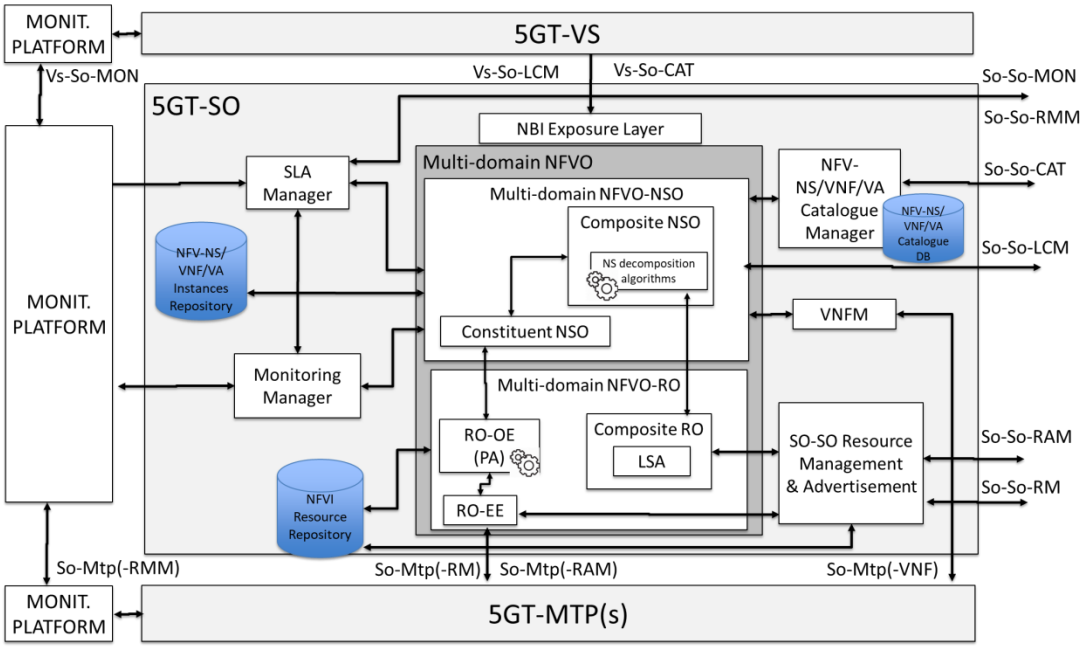


FIGURE 1: 5GT-SO ARCHITECTURE – BUILDING BLOCKS AND INTERACTIONS

- *Use of monitoring services toward SLA assurance*

Concerning the way the 5GT-SO uses the monitoring reports to support SLA operations (i.e., scaling) also based on the collected monitoring data provided by MTP, the following architectural refinements have been introduced:

Firstly, the Monitoring Platform is explicitly shown as a separated platform providing monitoring services to the 5GT-SO as well as to the 5GT-MTP and 5GT-VS. More specifically, the Monitoring Platform has been reported as three different functional blocks to indicate also that three different level of monitoring data are provided to 5G-T systems, i.e., at the (virtual) network service level to the 5GT-SO, at the infrastructural level to the 5GT-MTP, at the vertical slice level to the 5GT-VS.

Secondly, the interactions between the 5GT-SO and the Monitoring Platform have been refined to reflect the actual use of monitoring reports from the 5GT-SO itself. Indeed, the Monitoring Manager is not the only functional entity acting as monitoring data consumer (to carry out required monitoring configuration operations). Also, the SLA Manager exploits relevant information from the Monitoring Platform (i.e., alerts) to detect SLA breaches and/or anomalous behaviors.

Thirdly, the So-Mtp-(RMM) and Vs-So-MON have been specified more precisely considering the functional split of the Monitoring Platform in three parts. As result, the So-Mtp-(RMM) indicates the reference point for monitoring data exchanges between the 5GT-MTP Monitoring Platform and 5GT-SO Monitoring Platform. Similarly, the Vs-So-MON indicates the reference point for monitoring data exchanges between the 5GT-SO Monitoring Platform and 5GT-VS Monitoring Platform.

- *Service and Resource federation*

In terms of federation capabilities supported by the 5GT-SO the following updates have been applied:

At service federation level, the functional block in charge of decomposing the Network Service according to specified algorithms has been more clearly specified, both in terms of naming (i.e., NS decomposition algorithm) and in terms of functional scope. Indeed, within the Composite NSO more extended features are supported through the handling and decomposition of multiple nested NFV-NSs and by deciding where to deploy each of them, either locally or in a federated domain.

At the resource federation level, the functional blocks and interactions involved in the resource advertisement, allocation and release operations across federated domains have been better specified. SO-SO Resource Management & Advertisement has been extended to also include resource management (i.e., allocation and release) operations, in addition to advertisement operations to/from federated domains. Therefore, the RO-EE is now able to exploit the extended SO-SO Resource Management & Advertisement block to trigger resource provisioning operations and to coordinate any correlated operations. Moreover, the SO-SO Resource Management & Advertisement also consolidates abstract resources views from other domains and stores it in the NFVI Resource Repository. As result of this extension, the former block involved in the resource federation (i.e., SO-SO Resource Federation) has been re-focused to deal with nested NFV-NSs and re-named as Composite RO, accordingly. Indeed, in case of nested NFV-NSs it supports the Composite NSO to implement them into other federated domains. In particular, the LSA block is added to select and trigger the creation of links/paths between the deployed nested NFV-NSs across federated domains.

- *Interface specification toward SLA management*

In terms of interface specification to support SLA management across domains, interactions have been foreseen at the inter-SLA Manager level to exchange monitoring data and support the SLA assurance end to end when federated services are deployed. Thus, the existing reference points, So-So-MON and So-So-RMM, supporting the exchange of network service and resource monitoring data, respectively, have now been moved to the SLA Manager block.

A detailed description of the 5GT-SO high-level architecture is presented in the Annex (Section 6.2)

2.2 5GT-SO features

The second release (R2) of the 5GT-SO introduces new features with respect to the initial architecture functional specifications, reported in D4.1 [13], and the first release (R1) of the prototype, delivered in D4.2 [15].

In particular, the following features have been introduced:

- Placement algorithms: this feature allows evaluating the most suitable and efficient way to place the services.
- Service scaling: this feature describes the scaling of the Vertical Services to meet all the requirements for the services work and to avoid issues caused by network issues, resource shortage etc.
- Service composition: enables a feature of joining already instantiated NFV-NS instances with new ones that need to be instantiated, providing the connectivity for the NSs.
- Service federation: this feature provides the overall process of establishing, consuming or providing NFV-NS by an administrative domain from/to other (peering/partner) administrative domain.
- Resource federation: is the overall process of consuming or providing NFVI resources from or to external federated domains.

More details of the mentioned features are provided in Annex I, i.e., Section 6.6.

2.3 5GT-SO functional components

D4.1 introduced the high level architecture of the 5GT-SO in which the presented building blocks were defined considering their core functionalities. This high-level architecture is updated in section 6.2 not only based on high-level architectural decisions, but also on the feedback received during the implementation period of the 5GT-SO. Figure 2 presents the implemented architecture. As for the former, a key design decision that determines the architecture of the 5GT-SO functional components is the aim of integrating widely used available MANO platforms (e.g., OSM or Cloudify) to be able to exploit their features and also making the 5GT-SO platform independent of which MANO platform selected or deployed. At the same time, the 5GT-SO embeds additional advanced functionality (e.g., service composition, service federation) not currently offered by these platforms, inside a building block of the 5GT-SO called Service Manager (SM). In this sense, the SM is the brain of the 5GT-SO and is in charge of handling service and resource orchestration, by exploiting the OSM/Cloudify functionalities when needed for the cloud part. Setting up of network paths between datacenters through Wide Area Networks (WANs) for the interconnection of VNFs involved in NFV-NSs is carried out by the SM.

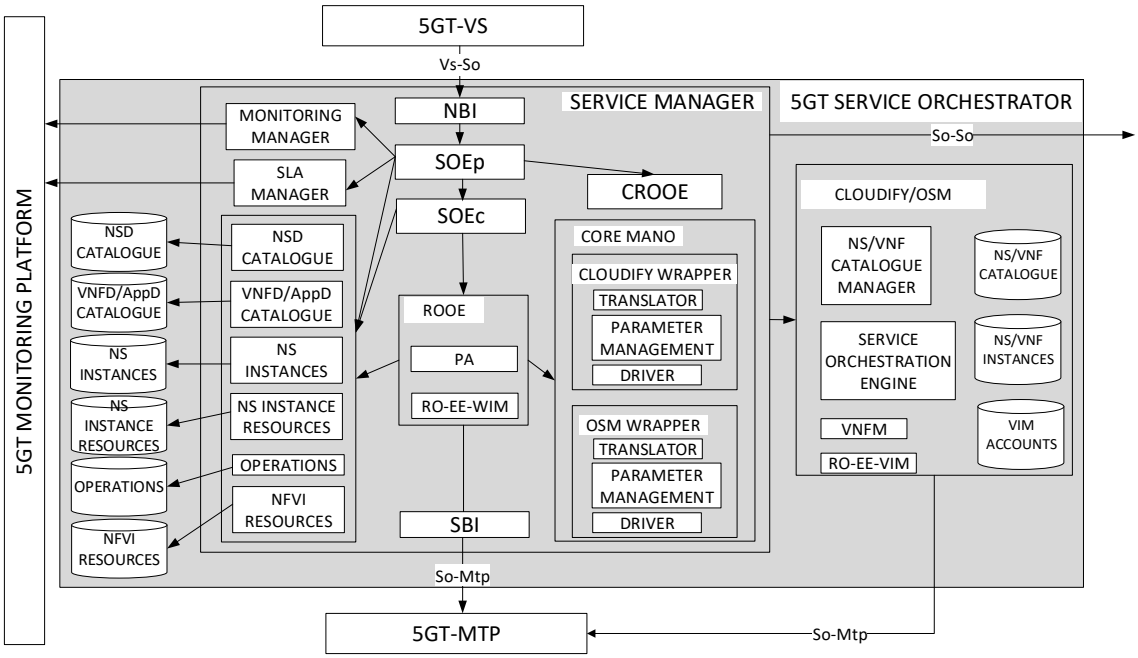


FIGURE 2: 5GT-SO IMPLEMENTED COMPONENTS

As for implementation, the high-level functionality of Figure 1 translates into the building blocks in Figure 2 which are the ones implemented. In the following paragraphs, we explain the mapping between high-level functionality and the building blocks that implement it, which are described in detail in Section 6.8. The full implementation is available for download at: https://github.com/5g-transformer/5gt-so.

Both network service lifecycle management and resource management are handled by the service manager with the help of OSM/Cloudify (for the cloud part). The SM interacts with OSM/Cloudify through a wrapper, which is generalized for any MANO in the form of the core MANO building block that adapts the lifecycle management processing done at the SM to the specific calls of each of these platforms. The composite NSO functionality is carried out by the Service Orchestration Engine parent (SOEp). The constituent NSO functionality is carried out by the Service Orchestration Engine child (SOEc).

As for resource management, the RO-OE functionality is carried out by the building block of the same name. The RO-EE functionality is split into RO-EE-WIM (for networking) inside the SM and RO-EE-VIM (for cloud), as part of Cloudify or OSM. The composite RO functionality is implemented in the composite ROOE (CROOE) building block, which directly interacts with its peer CROOE in another administrative domain, which is needed for service federation. That is, the SO-SO resource Management & Advertisement functionality in the high-level architecture would apply to the implementation of resource federation.

Finally, the interaction with the monitoring platform is carried out through the monitoring manager and the SLA manager, which also have the same name in the implementation, and the VNFM is that of Cloudify and OSM.

Additionally, underlying all the above processes, there are a series of databases for storing service- and resource-related information (e.g., catalogues, instance repository, per-instance resource repository, operations), which are fundamental for the maintenance of state information of the 5GT-SO, hence for its correct operation.

2.4 5GT-SO Interfaces

The interfaces defined in D4.1 [13] remain as the main interfaces where the 5GT-SO communicates with the 5GT-VS on the North-Bound Interface (NBI), with the 5GT-MTP on the Southbound Interface (SBI) and with peering/federated 5GT-SOs on the East/West-Bound Interface.

2.4.1 NBI

The NBI defined in D4.1 consists of the three reference points:

- **Vs-So (-LCM)**
- **Vs-So (-MON)**
- **Vs-So (-CAT)**

An additional reference point is added:

- **Vs-So (-POL)** is used for the management of policies. It offers primitives to transfer, delete, activate, deactivate, associate, disassociate policies and receive notifications about policy conflicts.

By adding the additional reference point, the NBI additionally supports the policy management based on the ETSI NFV IFA 013 policy management.

The proposed extension of the NFV-NSDs according ETSI NFV IFA 014 in D4.1., has been defined to extend or modify the ETSI NSD such that the 5GT-SO supports deployment of NFV-NS that contain AppDs and define information elements within the ETSI NSD for expressing location constraints.

More detailed information on the changes of the NBI is covered in Section 6.4.1.

2.4.2 SBI

In R2, the MTP interfaces have been updated considering the new features and the use cases requirements.

The SBI defined in D4.1 consists of the three reference points:

- **So-Mtp(-Resource Advertising Management).**
- **So-Mtp(-Resource Management).**
- **So-Mtp(-Resource Monitoring Management).**
- **So-Mtp(-VNF).**

The following update has been introduced:

- MTP NBI to include the instantiation of MEC App and Radio functions (e.g., gNB), to trigger monitoring jobs, to allocate/terminate intra-PoP connectivity in DC as well inter-PoP connectivity across DCs and to explicitly start/stop the allocated VNF resources;

More information regarding the SBI can be found in Section 6.5.2.

2.4.3 E/WBI

The E/WBI remains aligned with the proposed reference points as in D4.1 [13]. In R2, the end-points for the defined six reference points are already defined. The Composite NSO (Section 6.4.3.1) uses the reference points (1) So-So-LCM and (2) So-So-CAT. The SLA Manager extracts monitoring information with a peering/federated SLA

Manager using the (3) So-So-MON and (4) So-So-RMM. The SO-SO Resource Management & Advertising block is operating over the (5) So-So-RM and (6) So-So-RAM.

More information regarding the E/WBI can be found in Section 6.5.3.

2.5 Service Monitoring

The overall architecture of the 5G-TRANSFORMER monitoring framework is still aligned with the major design principles and functional components defined in R1 and reported in D4.1. The evolution in R2 is based on an incremental approach that allows to support an additional feature, driven by verticals' requirements, to enable the visualization of monitoring data on a vertical service basis directly from the main dashboard of the 5G-TRANSFORMER system.

In particular, in order to increase the vertical's awareness of service status and performance, the service monitoring architecture has been enriched with respect to what has been reported in D4.1 [13] with the addition of a Reporting component, as shown in Figure 3. This component automatically generates a dashboard concisely reporting the service's monitored data to be provided to the vertical.

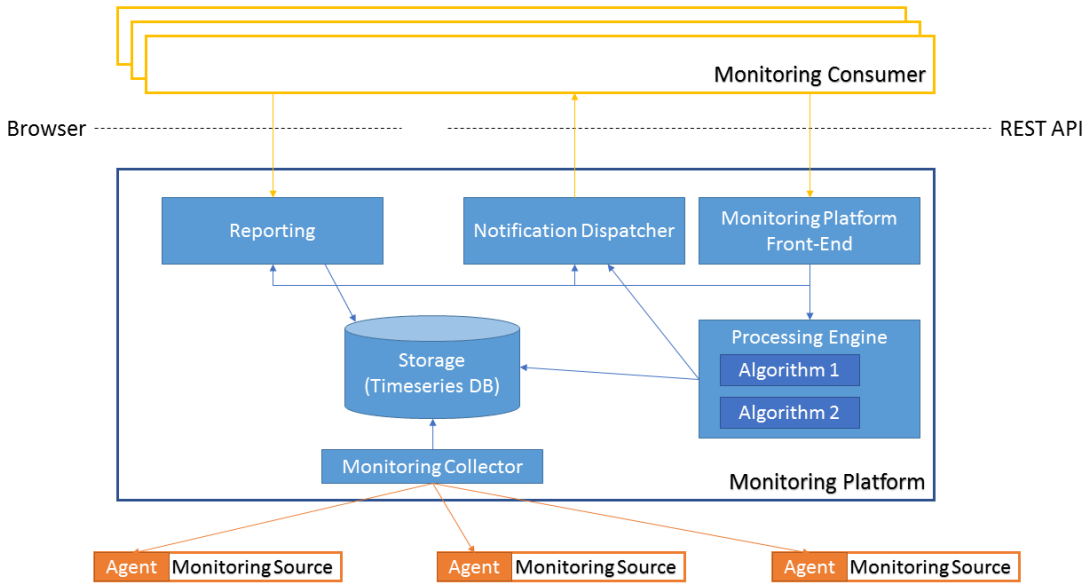


FIGURE 3: FUNCTIONAL ARCHITECTURE OF THE SERVICE MONITORING FRAMEWORK

As depicted in Figure 3, the reporting component is configured via REST API through the unified Monitoring Platform Front-End, providing the description of the "per-service" dashboard(s) to be made available to the users (e.g. in terms of type and format of monitoring parameters to be visualized). When the vertical connects to the URL of a generated dashboard, the reporting component then generates the graphs on-demand by querying the monitoring timeseries DB.

The prototype of the Monitoring Platform developed in R2 extends the R1 version, implementing the mechanisms for alerts management in support of SLA verification and automated scaling. The details about the design and implementation of the Monitoring Platform prototype are provided in section 6.9.

3 5GT-SO workflows

The 5GT-SO workflows have been incrementally revised for R2 to reflect both the architectural upgrades and the lessons learnt from the implementation. Indeed, in this deliverable the workflows specify more details on the supported orchestration mechanisms in terms of service composition and federation and in terms of service scaling (for service assurance) as well as in terms of NFV-NS instantiation with the support of MEC applications.

In terms of service composition and federation workflow, the 5GT-SO through the revised Composite NSO component is able to differentiate requests for single/regular NFV-NSs and composite NFV-NSs and to perform dynamic decomposition on regular NFV-NSs. Moreover, the added component Composite RO is able to take care of inter-nesting composite NFV-NS in the process of service federation.

In terms of service assurance, we better specify the workflows for service monitoring and for scaling by (i) including revised architectural blocks, namely Monitoring Manager and SLA Manager; (ii) considering both the case of NFV-NS scaling triggered by the 5GT-VS and the case of scaling triggered internally at the 5GT-SO (i.e., SLA Manager) when the monitoring data collected for that NFV-NS instance exceed a given threshold specified in the autoscaling rules.

In terms of providing MEC support, the NFV-NS instantiation workflow has been extended to orchestrate the MEC applications in the 5GT-SO R2 in terms of deciding the optimum placement of the MEC applications into the NFVI in the 5GT-MTP (e.g., based on MEC hosting constraints).

The selected set of 5GT-SO workflows are presented in the Annex (Section 6.7).

4 Conclusions

This deliverable provides the final specification of the 5GT-SO architectural design, which has been revised and updated from the initial design reported in D4.1, according to an incremental approach to cope with new and more specific requirements and to specify with more details the supported orchestration mechanisms and functionalities as result of lesson learnt from the implementation. More specifically, orchestration capabilities have been enhanced in terms of dynamicity thanks to enhanced service composition and federation features, enhanced placement decisions to consider location constraints, enhanced service monitoring platform, auto-scaling capabilities, and support of MEC applications.

This deliverable has also documented the detailed implementation of the 5GT-SO software prototype and service monitoring prototype. In particular, the 5GT-SO reference implementation will be adopted in the public demonstrations of the 5G-TRANSFORMER use cases in WP5. The SO prototype has already been showcased in several public events (EuCNC 2018, MWC 2018, and ICT 2018), with ongoing preparation for demonstrations at EuCNC 2019 and it will be also taken as the baseline service orchestration platform to be extended and integrated into the platform for the upcoming 5G-PPP phase 3 5GROWTH project. This guarantees the continuous evolution and enhancement of the 5GT-SO component itself and, as a consequence, of 5G-TRANSFORMER concepts about vertical services and network slices as a whole.

To conclude, in the main body of this deliverable, namely from the section 2 to the section 4, the major changes regarding the main functional and software building blocks, monitoring capabilities and workflows supported by the 5GT-SO have been identified and described, also providing a brief rationale of each change. Finally, in the Annex a full self-contained description of the updated 5GT-SO system is provided. This organization prevents that readers have to cross-check the previous deliverable to get a comprehensive view of the current design and implementation as well as the whole perception of the refinements of the 5GT-SO system.

5 References

[1] 5G-TRANSFORMER project code repository, available at: http://github.com/5g-transformer

[2] ETSI ISG NFV-IFA 005, “Network Function Virtualisation (NFV), Management and Orchestration; OR-VI reference point- Interface and Information Model Specification”, v2.1.1, Apr. 2016

[3] ETSI ISG NFV-IFA 014, Network Functions Virtualisation (NFV); Management and Orchestration; Network Service Templates Specification, August 2017.

[4] ETSI ISG NFV-IFA 011, Network Functions Virtualisation (NFV); Management and Orchestration; VNF Packaging Specification, Oct. 2016.

[5] ETSI ISG NFV-IFA 013, Network Functions Virtualisation (NFV); Management and Orchestration; OS-MA-NFVO reference point - Interface and Information Model Specification, Aug. 2018.

[6] ESTI ISG NFV-IFA 009, Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options, Jul. 2016.

[7] Canonical, Juju. [Online]. Available: https://docs.jujucharms.com/2.4/en/about-juju

[8] Haversine formula. [Online]. Available: https://en.wikipedia.org/wiki/Haversine_formula

[9] M. Mitchel, *An Introduction to Genetic Algorithms*, 5th ed. Cambridge, MA, USA: MIT Press, 1999.

[10] F. Murtagh, "A Survey of Recent Advances in Hierarchical Clustering Algorithms," *The Computer Journal*, 1983.

[11] S. Fichera et al., "Latency-aware resource orchestration in SDN-based packet over optical flexi-grid transport networks," in *IEEE/OSA Journal of Optical Communications and Networking*, vol. 11, no. 4, pp. B83-B96, April 2019.

[12] 5G-TRANSFORMER, D3.1, Definition of vertical service descriptors and SO NBI, March 2018.

[13] 5G-TRANSFORMER, D4.1, Definition of service orchestration and federation algorithms, service monitoring algorithms, March 2018.

[14] 5G-TRANSFORMER, D4.4, Final design and implementation report on service orchestration, federation and monitoring platform (reference implementation), May 2019.

[15] 5G-TRANSFORMER, D4.2, Initial Service Orchestrator Reference Implementation, November 2018.

[16] ETSI GS NFV-MAN 001, “Network Functions Virtualisation (NFV); Management and Orchestration”, v1.1.1, 2014.

[17] 5GEx deliverable D2.1, public version “Initial System Requirements and Architecture, available at https://drive.google.com/file/d/0B4O_JVjsvab9VXItUDJqMUR6d28/view

[18] Costa-Pérez, Xavier; García-Saavedra, Andrés; Li, Xi; Deiss, Thomas; Oliva, Antonio de la; Di Giglio, Andrea; Iovanna, Paola, and Mourad, Alain. 5G-Crosshaul: An SDN/NFV Integrated Fronthaul/Backhaul Transport Network Architecture. IEEE Wireless Communications, 24(1), pp. 38-45, February 2017

[19] 3GPP Specifications official site: https://www.3gpp.org/specifications

[20] 5G-TRANSFORMER, D1.1, Report on vertical requirements and use cases, November 2017.

[21] 5G-TRANSFORMER, D1.2, 5G-TRANSFORMER initial system design, May 2017.
[22] 5G-TRANSFORMER, D3.3, Final design and implementation report on the Vertical Slicer (report), May 2019.
[23] ETSI GS MEC 010-2, “Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management”, v1.1.1, July 2017.
[24] 5G-TRANSFORMER, D2.1, Definition of the Mobile Transport and Computing Platform, March 2018.
[25] ETSI GS NFV-IFA 006, “Network Function Virtualisation (NFV); Management and Orchestration; Vi-Vnfm reference point – Interface and Information Model Specification”, v2.1.1, 2016.
[26] ETSI GS NFV-IFA 008, “Network Functions Virtualisation (NFV); Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specification”, v2.1.1, 2016.
[27] 5G-TRANSFORMER, D2.3, Final design and implementation report on the MTP, May 2019.
[28] ETSI GS NFV-IFA 028, Management and Orchestration; Report on architecture options to support multiple administrative domains”, v3.1.1, 2018
[29] Prometheus website: https://prometheus.io/
[30] 5G-TRANSFORMER, D3.4, Final design and implementation report on the Vertical Slicer (reference implementation), May 2019.
[31] OpenStack Tracker project webpage: https://wiki.openstack.org/wiki/Tacker
[32] OpenStack Heat project: https://wiki.openstack.org/wiki/Heat
[33] OpenStack Neutron project: https://wiki.openstack.org/wiki/Neutron
[34] OpenVswitch project: https://www.openvswitch.org/
[35] OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA): https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca
[36] 5G-TRANSFORMER, D1.3, 5G-TRANSFORMER Refined Architecture, May 2019.
[37] https://specs.openstack.org/openstack/tacker-specs/specs/newton/event_logging.html.

6 Annex I – 5GT-SO Design and Implementation

6.1 Requirements to the 5GT-SO

Technical requirements on the overall 5G-TRANSFORMER system have been defined in D1.1 [20]. The requirements covered in D1.1 [20] focus on properties related to vertical services and relevant use cases. General requirements related to the overall system are described in D1.3 [21]. In this section, we define business (Section 6.1.1) and functional (Section 6.1.2) requirements specific to 5GT-SO. The notation used to refer to the different requirements is described in Annex III D4.1 [13].

6.1.1 Business Requirements

5G-TRANSFORMER platform is designed to foster business relationships with other administrative domains via service and/or resource federation. The 5GT-SO is the entity responsible to manage these business relationships and to achieve so, the following table enumerates the business requirements we consider to be fulfilled by the 5GT-SO within the 5G-TRANSFORMER architecture.

TABLE 1: BUSINESS REQUIREMENTS

ID	Requirement	F/NF
ReqSO.B01	The 5GT-SO shall include a REST interface with the 5GT-VS.	F
ReqSO.B02	The 5GT-SO should be able to accept a network service specification from the 5GT-VS including both functional and non-functional requirements expected for the requested service.	F
ReqSO.B03	The 5GT-SO should manage the collaboration of federated 5G-TRANSFORMER administrative domains for the completion of the NFV NS	F
ReqSO.B04	The 5GT-SO shall expose appropriate interfaces to federated 5GT administrative domains and to the 5GT-MTP.	F
ReqSO.B05	The 5GT-SO shall orchestrate the services requested by the 5GT-VS using the resources exposed by the 5GT-MTP and/or the resources exposed by federated 5GT administrative domains.	F
ReqSO.B06	The 5GT-SO shall expose a subset of the resources coming from the 5GT-MTP to federated 5GT administrative domains.	F
ReqSO.B07	The 5GT-SO shall expose a subset of its own network services to federated 5G-TRANSFORMER administrative domains	F
ReqSO.B08	The 5GT-SO shall use the PNFs exposed by the 5GT-MTP for service completion.	F
ReqSO.B09	The 5GT-SO must adhere to industry multi-tenancy requirements including isolation, scalability, elasticity and security.	NF
ReqSO.B10	The 5GT-SO shall allow monitoring with an appropriate granularity according to the network service characteristics.	F
ReqSO.B11	The 5GT-SO must provide the 5GT-VS with a network service catalogue with information about available service	F

	offers and capabilities, in order to facilitate the automated provision of services.	
ReqSO.B12	The 5GT-SO must provide a mechanism to perform network service accounting and charging. This information should be available internally and externally (for the 5GT-VS).	F
ReqSO.B13	The 5GT-SO should be able to support long-live and short-lived services.	F
ReqSO.B14	The 5GT-SO should be reliable.	NF
ReqSO.B15	The 5GT-SO should be available (as carrier class component).	NF
ReqSO.B16	The 5GT-SO should keep responsiveness for the 5GT-VS and federated 5GT domains.	NF
ReqSO.B17	The 5GT-SO shall allow defining composed network services.	F
ReqSO.B18	The 5GT-SO shall support the selection of preferred, non-preferred, and prohibited virtual infrastructure providers for the network service instantiation.	F
ReqSO.B19	The 5GT-SO shall support to select the deployment area based on KPIs[1] of another service.	F

6.1.2 Functional Requirements

The 5GT-SO is involved in the service lifecycle at different stages. Thus, different requirements can be considered according to each stage, namely (1) Discovery, (2) Fulfilment, (3) Assurance, and (4) Decommissioning.

6.1.2.1 Discovery

The discovery phase facilitates the 5GT-SO to understand which capabilities and services (in terms of descriptors) are supported by 5GT-SO. That information will be exposed to the 5GT-VS and to federated 5GT domains for 5G-TRANSFORMER service offering.

The following requirements are identified:

TABLE 2: REQUIREMENTS ON THE DISCOVERY PHASE

ID	Requirement	F/NF
ReqSO.Di.01	The 5GT-SO must provide the 5GT-VS with the means to send detailed requests including information regarding the placement of resources, the location of service points, QoS, charging options.	F
ReqSO.Di.02	The NFV-NS/VNF catalogue entries may contain a service manifest and a price tag (or an indicative price range from which the exact price can be extracted at run-time).	F
ReqSO.Di.03	The 5GT-SO shall keep an up-to-date network service catalogue with the network services and VNFs received from other federated 5GT administrative domains.	F

[1] As an example, Extended Virtual Sensing should cover critical intersections, where 'critical' is defined in terms of occurrence of abrupt braking manoeuvres in the past.

ReqSO.Di.04	The 5GT-SO should provide a mechanism to set-up, re-size and terminate network services.	F
ReqSO.Di.05	The 5GT-SO shall allow a 5G-TRANSFORMER service providers (TSP) to store network service descriptors persistently and to: retrieve, update, and delete them.	F
ReqSO.Di.06	The 5GT-SO shall keep an up-to-date catalogue with the resources, links, connection points and PNFs as exposed by the 5GT-MTP abstraction.	NF
ReqSO.Di.07	The 5GT-SO shall keep an up-to-date catalogue with the resources exposed by other federated 5GT administrative domains.	NF

6.1.2.2 Fulfilment

During the service fulfilment phase, the 5GT-SO orchestrates (namely, creates and instantiates) network services requested by 5GT-VS.

The following requirements are identified:

TABLE 3: REQUIREMENTS ON THE FULFILMENT PHASE

ID	Requirement	F/NF
ReqSO.Fu.01	The 5GT-SO should allow scaling (up / down) as part of the lifecycle management	F
ReqSO.Fu.02	The 5GT-SO shall support to manage the lifecycle of each of the service instances separately.	F
ReqSO.Fu.03	The 5GT-SO shall support the management of NSDs	F
ReqSO.Fu.04	The 5GT-SO shall support the management of VNFDs.	F
ReqSO.Fu.05	The 5GT-SO shall allow connecting network service instances.	F
ReqSO.Fu.06	The 5GT-SO shall be able to translate the network service request to one or more resource allocation or lifecycle actions reflecting the agreed service levels in each case.	F
ReqSO.Fu.07	The 5GT-SO shall allow creating several instances of the same network service.	F

6.1.2.3 Assurance

5GT-SO is responsible to guarantee the service level performance agreements made with 5GT-VS for orchestrated NFV network services and to provide 5GT-VS with sufficient monitoring information of said network services.

The following requirements are identified:

TABLE 4: REQUIREMENTS ON THE ASSURANCE PHASE

ID	Requirement	F/NF
ReqSO.As.01	The 5GT-SO must provide the vertical slicer with APIs to monitor the QoS attained for the requested service.	F
ReqSO.As.02	The 5GT-SO should provide isolation among service requests.	NF
ReqSO.As.03	The 5GT-SO could provide resources from other 5GT-SO using federation.	F

ReqSO.As.04	The 5GT-SO shall be able to collect and provide performance information related with the NSs through the monitoring platform.	F
ReqSO.As.05	The 5GT-SO shall manage fault information, reacting when necessary and generating alarms to the 5GT-VS when the fault cannot be solved at the 5GT-SO level.	F

6.1.2.4 Decommissioning

Once a network service is decommissioned, 5GT-SO shall properly release the used resources and terminate the required VNFs.

The following requirements are identified:

TABLE 5: REQUIREMENTS ON THE DECOMMISSIONING PHASE

ID	Requirement	F/NF
ReqSO.De.01	The 5GT-SO should be able to identify the monitoring mechanisms to be de-activated as a result of a service termination.	F
ReqSO.De.02	The 5GT-SO should have means for receiving acknowledgement of successful actions of resources release.	F
ReqSO.De.03	The 5GT-SO should be able to notify the vertical slicer about a service termination.	F

6.2 5GT-SO Architecture

The 5GT-SO is in charge of end-to-end (E2E) orchestration of the NFV-NS across single or multiple administrative domains by interacting with the local 5GT-MTP and/or with other 5GT-SOs of neighbour administrative domains, respectively. 5GT-SO receives the service requirements from 5GT-VS via the Vs-So interface in the shape of a Network Service Descriptors (NSD). A NSD describes a Network Service (NFV-NS) that 5GT-SO should provide (either on its own or by leveraging neighboring SOs); it is expressed in terms of chaining of VNF components (i.e., constituent VNFs) described by VNF Descriptors (VNFD). A VNFD describes a VNF in terms of its deployment and operational behavior as well as specifies the resource and the connectivity (i.e., virtual links) requirements.

The 5GT-SO processes the NSD in order to decide the optimum placement of the VNFs/VAs and assign virtual networking, computing and storage resources across a single or multiple local/remote domains while meeting the service requirements specified in the NSD. To this purpose, the 5GT-SO embeds a series of modules, most notably, the Network Service Orchestrator (NFV-NSO) and the Resource Orchestrator (NFV-RO) with functionalities equivalent to those of ETSI NFV Orchestrator [2]. The NFV-NSO has the responsibility of coordinating the deployment of NFV-NSs along with their lifecycle management. The NFV-RO is in charge of deciding function placement and orchestrating virtual resources under the control of the underlying 5GT-MTP(s). Additionally, the 5GT-SO provides multi-domain network service orchestration through service and/or resource federation. In this direction, the 5GT-SO interacts with 5GT-SOs of other administrative domains through the So-So interface (federation) on the

end-to-end deployment of network services and placing them in most suitable execution environment based on the offered services and/or available resources in each administrative domain.

Service orchestration involves the management and instantiation of VNFs and/or VAs at local, edge and cloud NFVIs. The problem of mapping VNFs to (virtual) computing entities (nodes, NFVI-PoPs) and the mapping of virtual links between VNFs into (virtual) paths, depending on the granularity of abstraction offered by the 5GT-MTP, can be tackled by different optimization strategies, e.g., heuristics or mixed-integer linear programming. Moreover, automatic network service management and self-configuration algorithms (e.g., NFV-NS auto-scaling) are also required to adapt service deployments to network changes and/or the infrastructure resource utilization (e.g., virtual machine CPU load) in order to prevent service degradations or SLA violations due to concurrent usage of resources. To this purpose, the collection and aggregation of monitoring data is foreseen to trigger self-adaption actions.

Summarizing, the 5GT-SO key functionalities are the following. The 5GT-SO:

- decides the optimal service (de)composition for the whole NFV-NS based on service availability as well as the resource capabilities exposed by local 5GT-MTP and by the other administrative domains;
- performs the lifecycle management of the whole NFV-NS as well as of each VNF composing the NFV-NS through the VNFM, including service catalogue management;
- decides the optimal placement of VNFs/VAs[2] along with the optimal deployment of virtual links connecting VNFs through mapping operations, thereby enabling the execution of the NFV-NS or a portion of NFV-NS on the local 5GT-MTP;
- requests the needed services to federated 5GT-SOs to address the execution of portions of the NFV-NS in other administrative domains;
- performs monitoring tasks and SLA management functions to enable the triggering of self-adaptation actions (e.g., healing and scaling operations) thereby preventing service performance degradations or SLA violations.

Figure 4 presents the functional architecture of 5GT-SO building blocks and their interactions designed to achieve the essential 5GT-SO operation described before. The described 5GT-SO architecture follows ETSI guidelines [16] and is in line with orchestration system designs developed in related EU projects (i.e., 5GEx [17] and 5G-Crosshaul [18]). It results from refinements performed to the 5GT-SO architecture shown in D4.1 to cope with new requirements on the service management and to consider more functionalities emerged by vertical use case discussions and after 5GT-SO implementation started.

[2] The granularity that 5GT-SO has when placing functions (NFVI-PoPs, servers, etc.) depends on the level of abstraction offered by the 5GT-MTP.

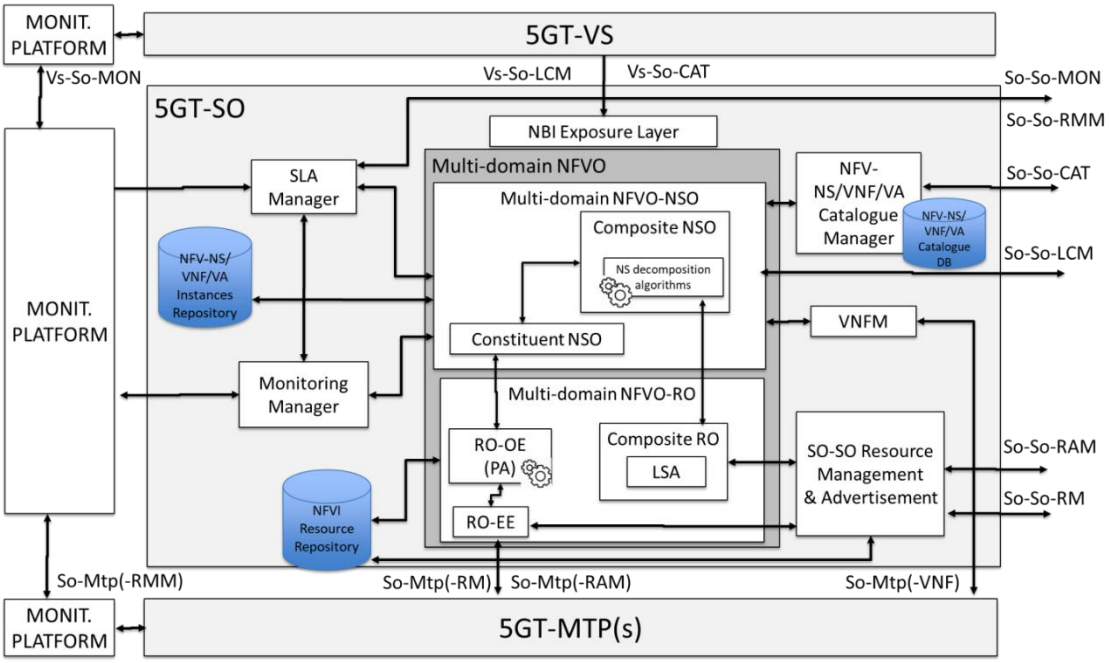


FIGURE 4: 5GT-SO Architecture – Building Blocks and interactions

The main building blocks comprising 5GT-SO are the following:

- **NBI Exposure Layer:** This layer offers a Northbound API towards the 5GT-VS to support requests for service on-boarding, service instantiation, service modification (e.g., scaling), and service termination. Details on the supported API are given in subsection 6.8.4.1.
- **NFV-NS/VNF/VA Catalogue DB/Manager:** Catalogue DB is the repository of all usable NFV Network Service (NFV-NS), Virtual Network Function (VNF) and VA descriptors that can be accessed through the Catalogue Manager. Such descriptors are used by the 5GT-SO in the process of NFV-NS/VNF/VA instantiation and its lifecycle management to obtain relevant information, e.g., deployment flavors or auto-scaling rules. The Catalogue Manager also contains the aggregated information on the available NFV-NSs/VNFs/VAs offered to the external/federated domains for federation. The aggregated information is stored by the Composite NSO.
- **Multi-Domain NFV Orchestrator (NFVO):** NFVO has the responsibility of orchestrating virtual resources across multiple domains, fulfilling the Resource Orchestration (NFVO-RO) functions, as well as of coordinating the deployment of NFV-NSs along with their lifecycle management, thus fulfilling the Network Service Orchestration (NFVO-NSO) functions. More specifically:
  - **Multi-Domain NFVO-NSO** coordinates all the NFV-NS deployment operations as well as formal checks of service requests based on attributes retrieved from NSDs and VNFDs. In particular, the Composite NSO, using the NS decomposition algorithms decomposes the NSDs into several nested NFV-NSs or does not decompose the NFV-NS (keeping it as a single compact NFV-NS). Note that this is the case when the received NSD is not explicitly a composite NSD. In case of explicit composite NSD, the

decomposition is executed by default as it is configured in the NSD. Then the Composite NSO decides where to deploy each nested NFV-NS either locally or in a federated domain, i.e., whether using a local 5GT-MTP or leveraging neighbor 5GT-SOs. Accordingly, the Composite NSO requests (i) the Constituent NSO and then the local NFVO-RO to deploy a local nested NFV-NS into its local administrative domain; and/or (ii) the federated NFVO-NSO to deploy a nested NFV-NS into other federated administrative domains. In the latter case, the Composite Resource Orchestration (Composite RO) within the Multi-Domain NFVO-RO, which includes a LSA (Link Selection Algorithm/Entity), is in charge of selecting and triggering the creation of the links/paths between the deployed nested NFV-NSs either locally or across the federated domains. Finally, the NFVO-NSO is responsible for the network service lifecycle management including operations such as service on-boarding, instantiation, scaling, termination, and management of the network services according to their associated VNF forwarding graphs.

  - **Multi-Domain NFVO-RO** maps the nested NFV-NS onto a set of virtual resources through the RO Orchestration Engine (**RO-OE**) by deciding the placement of each VNF within the virtual infrastructure, based on specified computational, storage and networking (e.g., bandwidth) requirements. The decision is made by the RO-OE element leveraging specified Placement Algorithms (PAs) and it is based on the available virtual resources that are exposed by the 5GT-MTP via the So-Mtp Southbound Interface (SBI) or from other domains via the So-So/East-Westbound Interface (EBI/WBI) and stored into the NFVI Resource Repository. In the latter case, the sharing of abstract views is needed to build-up a comprehensive view of resources available from different domains which is carried out and stored by the SO-SO Resource Management & Advertisement element. Then, the RO Execution Entity (**RO-EE**) takes care of resource provisioning by managing the coordination of correlated actions to execute/forward the allocation/release requests to either local 5GT-MTP or to the 5GT-SO NFVO-RO of other domains, in the latter case leveraging the SO-SO Resource Advertisement & Management building block.

- **VNF Manager (VNFM):** the VNFM is in charge of the lifecycle management of the VNFs deployed by the 5GT-SO. It receives relevant VNF lifecycle events from the local NFVO and provides reconfiguration according to specified counteractions decided by the NFVO based on VNFDs (e.g., auto-scaling).
- **SO-SO Resource Management & Advertisement:** in resource federation, this block is in charge of exchanging abstract resource views (e.g., abstract topologies, computing and storage capabilities) with other domains consolidating inputs and storing federated resources into the NFVI Resource Repository. Either static or dynamic approach can be used to this purpose, e.g., static views with topological information only or dynamic views with also operational information (like current virtual resource load) that is updated periodically. This block is also in charge of issuing virtual resource allocation/release requests commands to/from federated domains that are triggered by the RO-EE within a resource federation process. For service federation, it simply acts as a proxy for the inter-connection of Composite

RO, to exchange resource-related information for inter-domain inter-nested path setup (i.e., set-up of paths between two nested NFV-NSs that are deployed in different domains).

- **NFVI Resource Repository:** This repository stores consolidated abstract resource views received from the underlying 5GT-MTPs, either from the So-Mtp Southbound Interface (SBI) or from the SO-SO Resource Management & Advertisement block in case of abstract resource views received from other 5GT-SOs/domains through the So-So/East-Westbound Interface (EBI/WBI).
- **NS/VNF/VA Instance Repository**: This repository stores the instances of VNFs, NFV-NSs, and VAs that have previously been instantiated.
- **SO Monitoring Manager**: This block is the one responsible for translating the high-level, service-centric monitoring requirements of instantiated NFV-NSs into low-level, resource-centric jobs to be activated within the Monitoring Platform to retrieve the necessary information from the virtual infrastructure elements (e.g., CPU usage of the caches" requires the "average CPU usage" of every cache VM). After activating the required jobs, this block is also in charge of configuring and managing the monitoring jobs lifecycle by interacting with the Monitoring Platform. It makes the details of these jobs available to the SLA Manager for threshold configuration and, finally, it is also notified whenever a service is scaled or terminated in order to update or remove the monitoring jobs related to the service.
- **SLA Manager:** This block assures that the agreed SLAs between the 5GT-VS and the 5GT-SO on the network service deployed are continuously satisfied through on-line SLA verification. To this purpose, this block leverages information about trends of relevant monitoring data that may indicate potential SLA breaches or anomalous behaviours that may lead to system under-performance. For this reason, the SLA Manager interacts with the Monitoring Platform following a subscription-notification paradigm, in order to promptly react to any alert associated to the target monitoring data. In particular, the SLA Manager subscribes with the Monitoring Platform (Config Manager), registering monitoring parameter queries (which may include arbitrarily complex expressions on many different time series) and setting threshold values for notifications, such that whenever the given threshold is passed, a notification is sent to the SLA Manager, which will then trigger the needed reactions at the 5GT-SO. One such potential action could be scaling. This scaling action is referred to as "Automated NFV-NS scaling" detailed in Section 6.6.2.4.

6.3 5GT-SO key operations

The functionalities described in Section 6.2 cooperate to support a number of operations including the ETSI defined operations of managing a NFV-NS. Such operations are described below.

1. *Interconnection to other 5GT-SOs for federation*

This operation is a prerequisite to achieve federation; that is, 5GT-SO shall be made aware of the existence of other 5GT-SOs, and thus become part of the service ecosystem. To this purpose, the 5GT-SO (i) exchanges resource/service capabilities with neighbouring SOs, and (ii) cooperates with other 5GT-SOs to accomplish service deployments across multiple administrative domains. The interconnection to neighboring SOs can be the result of either a static (i.e., manual pre-configuration of

inter-links) or automatic mechanisms (i.e., auto-discovery and interconnection). Without loss of generality, the static case is followed in 5G-TRANSFORMER. The automatic mechanisms are left for future work to explore.

2. *Resource/Service capabilities advertisement*

With this operation, 5GT-SO becomes aware of the resource and service capabilities of SOs in other administrative domains in order to make resource and service orchestration decisions. The exchanged capabilities are related both to supported VNFs and services (i.e., catalogues) and to topology and other resource-related information (e.g., abstract network topology, cloud resource capabilities). The service capabilities are more static and less frequent, therefore, by default the information is exchanged offline on a business-level communication. The resources for federation are not permanently available or time-constant which defines the dynamic advertisement mechanism as a requirement.

3. *Deployment*

This operation is triggered as a request for a NFV-NS arrives either from the Vertical Slicer or from neighbor SOs. The deployment operation is carried out by the orchestration engine (i.e., NFV Orchestrator (NFVO)) both at the *Network Service* level, handled by the Network Service Orchestrator (NFVO-NSO), and at the *Resource* level, handled by the Resource Orchestrator (NFVO-RO), according to the ETSI MANO guidelines [16]. At the service level, composite NSDs are decomposed according to specified NS decomposition algorithms. At the resource level, the decision where to allocate resources is made by leveraging specified Placement Algorithms (PAs) and it is based on the available virtual resources exposed by the 5GT-MTP. Different PAs make different level of decisions depending on the objective to address (e.g., minimization of end-to-end latency). The deployment operation may involve either the 5GT-MTP(s) in the local administrative domain of the 5GT-SO (single-domain case) or other SOs in different administrative domains (federation case).

4. *SLA assurance and scaling*

Once a network service is deployed, its assurance phase begins. This phase consists of collecting and processing monitoring data during the service lifecycle in order to check if the agreed SLAs between the 5GT-VS and 5GT-SO are continuously met (i.e., on-line SLA verification). The assurance phase is carried out by an SLA manager supported by the Monitoring Manager and by the Monitoring Platform that collects performance metrics from virtual resources. In case SLA parameters are not satisfied the 5GT-SO trigger either scaling actions by means of proper resource/service reconfigurations.

5. *Service composition*

In case complex services need to be deployed, which are based on the composition of multiple simpler ones, the 5GT-SO, through the composite NSO will manage the lifecycle of the composite NFV-NS by commanding the constituent NSOs (and underlying RO-OEs) for the individual nested NFV-NSs. Once all nested NFV-NSs are deployed, together with the Composite RO-OE, the composite NSO will set up the interconnection of nested NFV-NSs (through inter-nested paths) by triggering the corresponding actions towards the local entities in coordination with the equivalent ones in other domains.

The interfaces to other SOs, 5GT-MTPs and 5GT-VS are described in Section 6.5.

6.4 5GT-SO building blocks detailed overview

This section focuses on a more detailed explanation of the functional architecture explained in Section 6.2, introducing the 5GT-SO main building blocks.

6.4.1 NBI Exposure Layer

The northbound APIs exposed by the 5GT-SO follow the ETSI NFV IFA 013 specification [25] and they are specified in D3.3 [26], section 5.3.5.

6.4.2 NFV-NS/VNF/VA Catalogue DB/Manager

Catalogue DB is the repository of all usable Network Service Descriptors (NSDs) and VNF Descriptors (VNFDs) that can be accessed through the Catalogue Manager. The NSD/VNFD is used by the 5GT-SO in the process of NFV-NS/VNF instantiation and its lifecycle management to obtain relevant information, e.g., deployment flavors or auto-scaling rules. The Catalogue Manager also takes care of the advertising of NFV-NSs for federation purpose.

6.4.3 NFVO-NSO

The NFVO-NSO is in charge of coordinating all the NFV-NS deployment, orchestration and (service) federation operations. The NFVO-NSO can receive service requests on the NBI and E/WBI interfaces and based on each request proceeds with triggering certain operations (e.g., NFV-NS instantiation).

The NFVO-NSO orchestrates incoming requests for a single NFV-NSs and composite NFV-NSs (see Figure 5) A single NFV-NS can be instantiated as it is compact, without any change similar to a nested NFV-NS or it can be further decomposed in a set of nested NFV-NSs using a NS decomposition algorithm. The decomposition may be already done at the 5GT-VS, in which case the 5GT-SO receives a composite NFV-NS request with an NSD that explicitly defines the decomposition into several nested NFV-NSs and there is no need to run the NS decomposition algorithm.. The difference between the nested NFV-NS and the single NFV-NS is that the nested NFV-NS cannot be additionally decomposed and it must be instantiated as it is, as a compact NFV-NS in a specific domain (local or federated). The composite NFV-NSs are explicitly composed of several nested NFV-NSs (defined in their NSDs) and they can contain already active (up-and-running) nested NFV-NS that are referenced at instantiation time. For orchestrating NFV-NSs in external domains, the NFVO-NSO can only provide/consume nested NFV-NS to/from external federated domains.

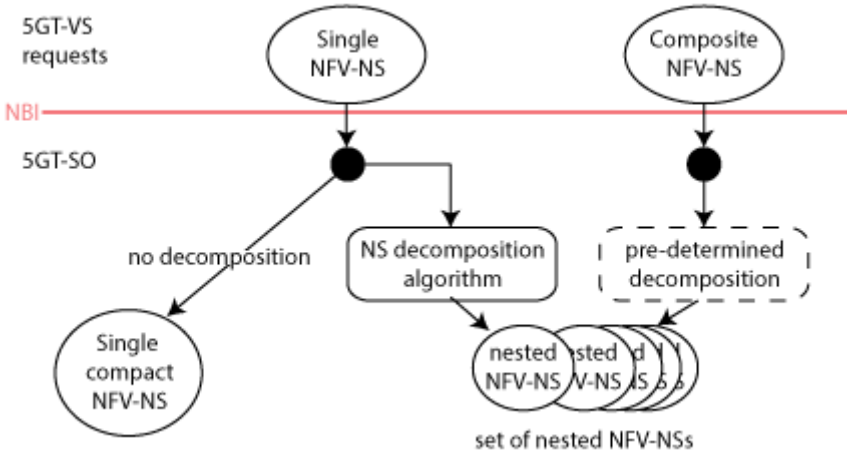


FIGURE 5: DECOMPOSITION OF SINGLE AND COMPOSITE NFV-NSS

The service requests that the NFVO-NSO can receive on both interfaces are:

- On-boarding of NFV-NS
- Instantiation of a single NFV-NS
- Instantiation of a composite NFV-NS (with no active nested NFV-NSs)
- Instantiation of a composite NFV-NS (with referenced active nested NFV-NSs)
- Modification of a single/composite NFV-NS
- Termination of a single/composite NFV-NS

The NFVO-NSO is composed of two modules: the Composite NSO and the Constituent NSO. The Composite NSO is the first module to process all received requests. Hierarchically, the Constituent NSO acts as a child module to the parent – Composite NSO.

6.4.3.1 Composite NSO

The Composite NSO is the module in charge of receiving requests, processing them (e.g., decomposition) and making orchestration/federation decisions for NFV-NSs.

In particular, upon reception of an instantiation request for an NFV-NS, the Composite NSO checks if the request is for a single or composite NFV-NS. If the request is a single NFV-NS, based on the internal policies of the Composite NSO it may be decomposed using the NS decomposition algorithms (and the result would be a set of decomposed nested NFV-NS) or it may pass it to the constituent NSO. In the former case, a new NSD is generated for each decomposed nested NFV-NS. For example, if a single NFV-NS is decomposed into four nested NFV-NSs, the Composite NSO will automatically generate a set of four new nested NSDs. In case the NS decomposition algorithms are not used, the NFV-NS is not decomposed (the NSD remains compact). Requests received from external/federated domain can be only be a deployment of single/nested NFV-NSs, the Composite NSO does not apply decomposition to requested NFV-NS of external/federated domains (decomposition is already done in the consumer external/federated domain).

In case of reception of an instantiation request for a composite NFV-NS, the decomposition is already done and the Composite NSO checks if the set of nested NFV-NS contain referenced active NFV-NS instances. The composite NFV-NS can be received only on the NBI from a 5GT-VS.

Based on the processing of the request (in this case instantiation request), the Composite NSO makes orchestration/federation decisions for each single or nested NFV-NS. The decision is whether each nested NFV-NS is to be deployed locally or in a federated domain. The decision may be based on service or resource availability, deployment cost/benefit, etc. For local deployment, the Composite NSO issues instantiation requests towards the Constituent NSO. For federated deployment, the request is sent to a federated 5GT-SO on the E/WBI. During the deployment process, the Composite NSO orchestrates the instantiation of each nested NFV-NS that composes the full end-to-end composite NFV-NS (in the case of a single compact NFV-NS it is similar to a single nested NFV-NS). Once the deployment of the set of nested NFV-NSs is finished, the Composite NSO orchestrates the interconnection between the nested NFV-NSs through the Composite RO for both local and federated domains.

6.4.3.2 NS decomposition algorithms

NS decomposition algorithms are capable to transform/decompose a single NFV-NS into several components or several nested NFV-NSs. By decomposing the NFV-NS, the deployment and orchestration of the smaller nested NFV-NSs aims to produce more optimal usage of the underlying infrastructure, while the nested NFV-NSs joined together (stitched) maintain the end-to-end functionalities and performance of the single/requested NFV-NS.

6.4.3.3 Constituent NSO

The constituent NSO is in charge of orchestrating and deploying a single or nested NFV-NS in the local domain. That is, in case the 5GT-VS requests a single/regular NFV-NS, the composite NSO just forwards the request to the constituent NSO, which handles all the required service orchestration. Therefore, the instantiation of a single/nested NFV-NS starts with receiving the requests from the Composite NSO. Then, the Constituent NSO obtains (queries) information from the NFV-NS/VNF Catalogue DB, such as the NSD and VNFDs contained in the NSD. Then, the information is processed (e.g., associating the NFV-NS instance ID, etc.) and redirected to the ROOE to follow with a the regular instantiation workflow.

If the 5GT-VS requests the instantiation of a composite NFV-NS, the composite NSO will parse the NSD and send one by one each of the nested NSs (which become single/regular NFV-NSs) to the constituent NSO, which will handle them as described in the previous paragraph. Once a nested NFV-NS is instantiated, the Constituent NSO confirms the deployment to the Composite NSO, which can proceed with the following one.

6.4.4 VNF Manager (VNFM)

The VNFM is in charge of the lifecycle management of the VNFs deployed by the 5GT-SO. It receives relevant VNF lifecycle events from the local NFVO and provides reconfiguration according to specified counteractions decided by the NFVO based on VNFDs (e.g., auto-scaling). The VNFM in the 5GT architecture is that of the MANO platform integrated in the 5GT-SO.

6.4.5 NFVO-RO

The NFVO-RO orchestrates the deployment of NFV-NSs over the 5GT-MTP resources and handles resource-related operations in the local 5GT-MTP and in a federated domain (the latter only for resource federation). The NFVO-RO is composed of several modules: Resource Orchestration Engine (RO-OE) including Placement Algorithm module (PA), Composite Resource Orchestration (Composite RO) and Resource Orchestration Execution Engine (RO-EE).

The NFVO-RO receives NFV-NS requests from the NFVO-NSO, processes the requests and issues new resource related requests towards the 5GT-MTP on the SBI or towards another federated NFVO-RO on the E/WBI.

6.4.5.1 Composite Resource Orchestration (Composite RO)

The Composite Resource Orchestration module is in charge of inter-connecting (stitching) several nested NFV-NSs to form/compose the full end-to-end (composite) NFV-NS. It is the main module of setting the inter-nested paths, the inter-connection between two nested NFV-NS instances. The Composite RO contains the Link Selection Algorithm/Entity (LSA) module, which is provisioning the inter-PoP links and inter-domain links so that the stitching procedure between all nested NFV-NSs is executed successfully.

The module receives requests for setting up the inter-nested paths directly from the Composite NSO once all nested NFV-NS are instantiated (locally or in a federated domain (for service federation)). First, the Composite NSO establishes the inter-PoP connectivity and stitching of local NFV-NSs in the local domain. Sequentially, the Composite RO communicates with the provider/federated domain on the E/WBI (via the SO-SO Management & Advertisement module) to establish all the tunneling connections between the domains with the aim to inter-connect federated nested NFV-NS (More details in the federation workflow – Section 6.7.7). The communication is between consumer Composite RO and provider Composite RO in the federated domain.

6.4.5.2 Resource Orchestration Engine (RO-OE)

The Resource Orchestration Engine (RO-OE) handles the requests that are related to 5GT-MTP resources management, i.e., NFV-NS instantiation and termination, for regular NFV-NSs (i.e., non-composite). It interacts with the Placement Algorithm (PA) submodule and the Resource Orchestration Execution Engine (RO-EE) submodule to accomplish the requests.

Upon receiving an instantiation request, the RO-OE extracts information from the referenced NSD and VNFDs and information on available resources from the NFVI Resource Repository or by requesting the 5GT-MTP. Based on the obtained information and their required allocation of resources, the Placement Algorithm (PA) determines the placement of the VNFs and selects the inter-PoP logical links over the available 5GT-MTP resources. Once the placement is determined, the RO-OE passes the information towards the RO-EE in order to execute the deployment of the VNFs that form a single NFV-NS over the 5GT-MTP resources. Note that the available resources are a combined view of the local 5GT-MTP resources and federated domain resources (when doing resource federation). In case the PA calculates the deployment over external (federated) resources, the RO-EE executes resource federation.

6.4.5.3 RO Execution Entity (RO-EE)

The Resource Orchestration Execution Engine (RO-EE) is the NFVO-RO module in charge of communicating with the SBI to handle network resources management operations that involve the 5GT-MTP and the module that enables resource federation via the SO-SO Resource Management & Advertisement module. Based on the decisions made by the RO-OE, the RO-EE module issues requests for computing resources & networking allocations on the local 5GT-MTP via the SBI. In case the allocation of the resources needs to be executed in a federated external domain, the request is redirected towards the SO-SO Resource Management & Advertisement module. The Composite RO orchestrates the set-up of inter-nested paths by directing the requests towards the RO-EE once the decision on what logical links to use is taken by the LSA. The RO-EE adds additional network-related information needed to set up the links at the MTP and redirects the requests towards the SBI, and the MTP.

6.4.6 SO-SO Resource Management & Advertising

This module oversees two operations: (i) advertising/accumulating available resources (resource abstractions, topologies, etc.) to/from other federated domains and (ii) resource management in the federation procedures (both for service and resource federation). The SO-SO Resource Management is connected to the Composite RO, the RO-EE, NFVI Resource Repository and to the external federated NFVO-ROs on the E/WBI (So-So-RAM, So-So-RM).

In the advertisement case, the SO-SO Resource Management & Advertisement shares information regarding the available resources from the local 5GT-MTP. The information is obtained from the NFVI Resource Repository and shared among the interconnected federated NFVO-ROs. Note that depending on what will be exposed towards the external domains according to the agreed SLAs among federated domains, additional abstractions may be applied prior to sharing the information for security reasons. The advertising procedure can be static, without changes on the available resource for federation, and dynamic, where the availability of the resources changes over time. More details regarding advertising procedure can be found in the Section 6.6.6. All accumulated information regarding the resources from external domains, are stored by the SO-SO Management & Advertisement module in the NFVI Resource Repository.

In case of service federation, the Composite RO orchestrates the interconnection of the federated nested NFV-NS. The SO-SO Resource Management & Advertisement module acts as a proxy module by receiving requests from the Composite RO and redirects them towards the federated provider domain for establishing the tunneling interconnection to the federated NFV-NSs. On the provider side, the module passively redirects the messages towards the Composite RO.

In the resource federation case, the SO-SO Resource Management & Advertisement block has the translation role as:

- Provider - translating incoming requests on the E/WBI for providing available resources and issue the translated request towards the RO-EE;
- Consumer – redirect the requests for federation of resources from the local RO-EE towards the specific federated domain on the E/WBI.

6.4.7 NFVI Resource Repository

NFVI Resource Repository is a database updated by the 5GT-MTP and the SO-SO Resource Management & Advertisement modules. It contains the global view of available resources both from the local and external federated domains. Based on the information contained in the database, the RO-OE decides where to place the VNFs that form a NFV-NS.

6.4.8 NFV-NS/VNF/VA Instance Repository

A database for containing information on NFV-NS instances, VNF instances that are part of the NFV-NSs as well as the VA instances.

6.4.9 SO Monitoring Manager

The Monitoring Manager component is the functional module responsible for translating the high-level, service-centric monitoring requirements managed at the 5GT-SO into the low-level, resource-centric monitoring jobs managed at the Monitoring Platform. The Monitoring Manager is thus responsible to handle the lifecycle of such monitoring jobs, interacting with the Monitoring Platform through the mediation of its Config Manager, according to the lifecycle of the associated NFV-NS instances.

Whenever an NFV-NS is instantiated through the NFV-NSO, the Monitoring Manager inspects the NSD, looking for any data that the service might require to be monitored. For each of these monitoring data (expressed at the service level), the Monitoring Manager determines the corresponding, elementary, resource-related monitoring data and how they need to be combined. For example, the “average CPU usage of the caches” for a virtual Content Delivery Network (vCDN) service requires the “average CPU usage” of every cache VM allocated for that specific service instance, combined together with an “average” operation.

After having determined which low level time series should be collected, the Monitoring Manager sends a request to the Monitoring Platform to create the jobs for retrieving the

necessary information from the virtual infrastructure elements. It then makes the details of these jobs available to the SLA Manager for threshold configuration.

After the configuration of this job, the Monitoring Manager requests the creation of a monitoring dashboard for data reporting, by specifying appropriate aggregation functions composing the metrics to be visualized. The dashboard is provided in the form of a web page containing several plots and charts which report the value of the monitoring time series for the target NFV-NS instance. The URL of the dashboard is also stored in the record of the NFV-NS instance (within the NFV-NS Instance Repository – see section 6.4.8) and they are made available at the 5GT-SO NBI for hands-on monitoring and management of the service, or to use as a diagnostic tool. In particular, the 5GT-VS will use that URL to visualize the monitoring data of the vertical service associated to the network slice realized through the given NFV-NS instance.

The Monitoring Manager is also notified whenever a service is scaled or terminated, in order to update or remove the monitoring jobs related to the service, following a similar workflow.

6.4.10 SLA Manager

The SLA Manager component is a functional module of the 5GT-SO responsible for alerts configuration and analysis of the monitoring notifications received from the Monitoring Platform in search of potential SLA breaches. The SLA Manager interacts with the NFVO-NSO, through the NBI, to trigger automated scaling actions according to the scaling rules encoded in the NSD of the NFV-NS instance.

During the instantiation of an NFV-NS with active autoscaling rules, the SLA Manager configures the alerts on the Monitoring Platform, interacting with its Config Manager. This configuration allows the SLA Manager to receive notifications from the Monitoring Platform (in particular from its Alert Manager) during the service runtime, whenever one of the target monitoring parameters exceeds the given thresholds. Once an Alert is raised, the SLA Manager analyzes it considering the active autoscaling rules and the current status of the NFV-NS instance. If needed, it sends a ScaleNS request to the NFVO-NSO, through the NBI, to request the scaling of the NFV-NS instance to the target instantiation level.

6.5 5GT-SO Interfaces and reference points

This section introduces the 5GT-SO interfaces and references points towards other building blocks of the 5G-TRANSFORMER system, including the northbound interface towards the 5GT-VS, southbound interface towards the 5GT-MTP, and east/westbound interface to the federated 5GT-SOs.

6.5.1 5GT-SO NBI

The sequel briefly introduces the design of the 5GT-SO's northbound interface (NBI); it is solely based on the content found in D3.3 [22]. The NBI enables the interaction between the 5GT-VS and the local 5GT-SO, which is based on the following four reference points (see Figure 6).

- **Vs-So(-Life Cycle Management)** is used for the operation of NFV network services. It offers primitives to instantiate, terminate, query, update or re-configure network services or receive notifications about their lifecycle;
- **Vs-So(-MONitoring)** is used for the monitoring of network services and VNFs through queries or subscriptions/notifications about performance metrics, VNF indicators and network services or VNFs failures;
- **Vs-So(-Catalogue)** is used for the management of Network Service Descriptors (NSDs), VNF packages, including their VNF Descriptors (VNFDs), and MEC Application Packages, including their Application Descriptors (AppDs). This reference point offers primitives for on-boarding, removal, updates, queries and enabling/disabling of descriptors and packages.
- **Vs-So(-POLicy)** is used for the management of policies providing a series of operations such as activation, deactivation, association, disassociation, transfer and deletion of policies and it allows notifications about policy conflicts.

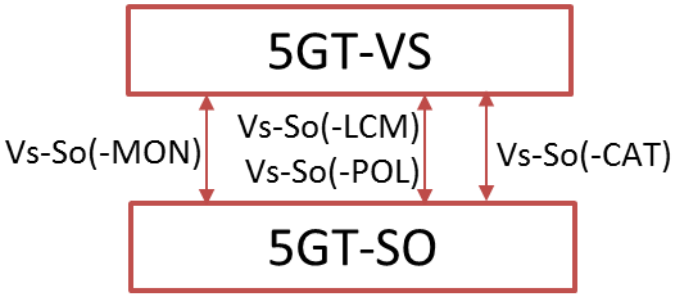


FIGURE 6: REFERENCE POINTS BETWEEN 5GT-VS AND 5GT-SO

The implementation of the 5GT-SO NBI is mostly based on ETSI GS NFV-IFA 013 specification [5], which defines the NBI of an NFVO. However, the 5GT-SO NBI implements further methods to allow the 5GT-SO to handle Network Services extended to include also MEC Applications. In terms of interfaces, this has an impact mostly on the management of the catalogues and on the queries of the NFV-NS instances, where both of them should support the information related to the MEC applications. In particular:

- The Vs-So(-LCM) reference point is implemented through the **ETSI GS NFV-IFA 013 [5] NFV-NS Lifecycle Management Interface**. This interface provides messages for creation and deletion of network service identifiers as well as instantiation, scaling, update, querying and termination of a network service instance;
- The Vs-So(-MON) reference point is implemented through the **ETSI GS NFV-IFA 013 [5] NFV-NS Performance Management Interface** and the **ETSI GS NFV-IFA 013 [5] NFV-NS Fault Management Interface**. The Performance Management Interface provides messages for creating, deleting and querying Performance Monitoring jobs (PMON jobs) as well as subscriptions and notifications to receive monitoring reports. The Fault Management Interface

provides subscriptions, notifications and queries about alarms related to network services and VNFs;

- The Vs-So(-CAT) reference point is implemented through the **ETSI GS NFV-IFA 013 [5] NSD Management Interface**, the **ETSI GS NFV-IFA 013 [5] VNF Package Management Interface** and the **ETSI MEC 10-2 Application Package Management Interface** [23], where the 5GT-SO implements the role of the Mobile Edge Orchestrator (MEO). Each of them provides mechanisms to on-board, enable, disable, update, delete and query NSDs (ETSI GS NFV-IFA 013 [5]), PNFDs (ETSI GS NFV-IFA 013 [5]), VNF Packages (ETSI GS NFV-IFA 013 [5]) and MEC Application Packages (MEC 10-2), as well as subscriptions and notifications to notify new on-boarding actions or changes in existing descriptors.

NSD and VNF packages follow the information models defined in the ETSI GS NFV-IFA 014 [3] and ETSI GS NFV-IFA 011 [4] respectively, while the AppD follows the information model defined in ETSI MEC 10-2. However, in 5G-TRANSFORMER, the NSD is extended to include references to the AppDs, as shown in Figure 7 and Figure 8. Moreover, in order to specify the geographical location where the NFV network service should be placed, the instantiation request in the ETSI GS NFV-IFA 013 is extended with an additional "location constraint" attribute in the definition of the Service Access Points that connects the network service instance to the external networks.

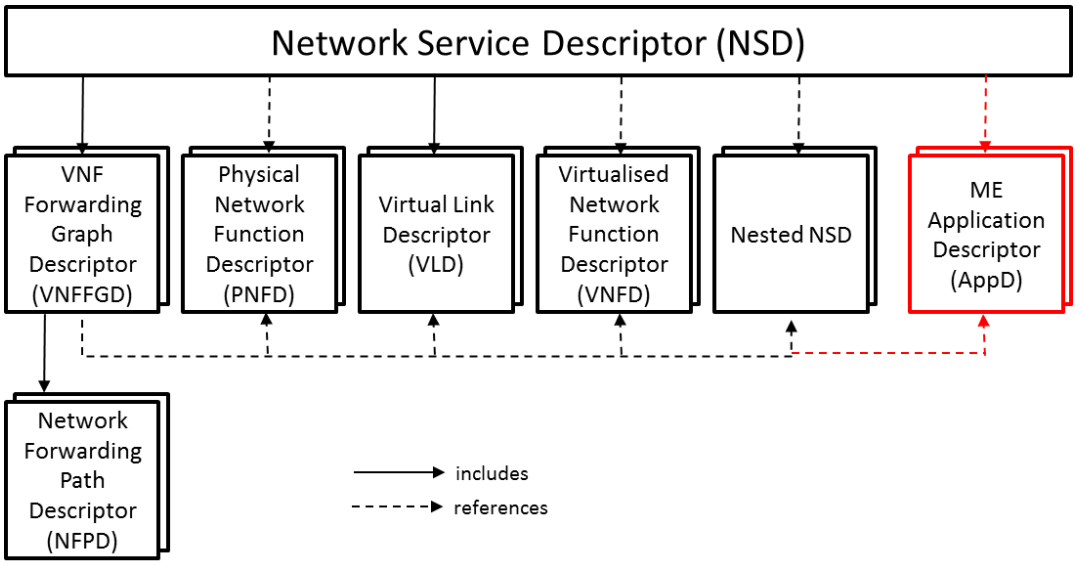


FIGURE 7: NSD EXTENDED WITH APPD REFERENCES SCHEMA 1

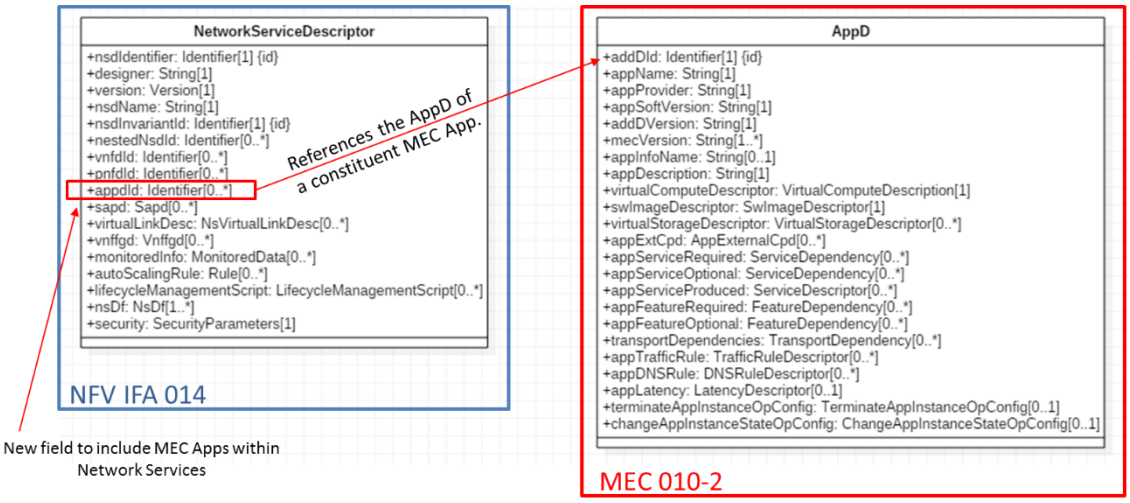


FIGURE 8: NSD EXTENDED WITH APPD REFERENCES SCHEMA 2

6.5.2 5GT-SO SBI

We next summarize the design of the 5GT-SO's southbound interface (SBI); additional details can be found in D2.1 [24]. The SBI addresses the interworking between the 5GT-SO and 5GT-MTP building blocks of the 5G-TRANSFORMER architecture [24]. It is worth mentioning that 5GT-SO and 5GT-MTP may follow a 1:1 relationship. The 5GT-MTP, which handle the configuration and programmability of a number of technological domains including heterogeneous virtualized resources for compute, storage and networking. In the following we are also assuming that a 5GT-MTP is managed by a single 5GT-SO. Besides managing the utilization (i.e., de/allocation) of the virtualized resources, the 5GT-SO SBI also encompasses the required functionalities for deploying (updating and terminating) demanded VNFs by a given NFV-NS. In the 5G-TRANFORMER project all these operations are supported at So-Mtp interface.

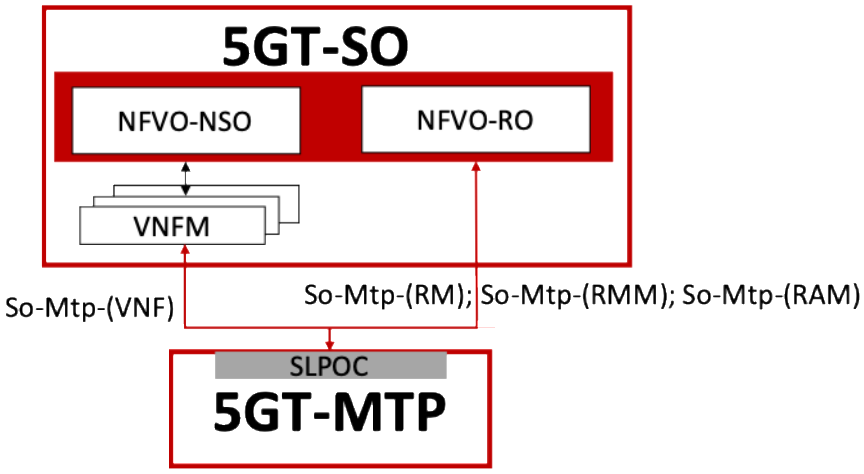


FIGURE 9: REFERENCE POINTS FOR 5GT-SO SBI (I.E., SO-MTP INTERFACE)

Figure 9 illustrates the targeted 5GT-SO SBI and its key reference points. Similar to the 5GT-SO NBI, the 5GT-SO SBI is mostly based on a set of standard documents being produced within the ETSI NFV framework, namely ETSI GS NFV-IFA 005 [2], ETSI GS NFV-IFA 006 [25] and ETSI GS NFV-IFA 008 [26]. In a nutshell, the 5GT-SO SBI shall

provide the operations and functions for: (i), providing abstracted information (e.g., capacities, availability, connectivity, etc.) of the virtualized resources managed by each 5GT-MTP; (ii) managing (i.e., instantiation, reservation, allocation, scaling up/down and release) of the virtualized resources required to support an NFV-NS; (iii) enabling the fault management and performance monitoring aiming at recovering interrupted services or ensuring the targeted SLAs demanded by each NFV-NS; and (iv) supporting the lifecycle management (i.e., creation, configuration, modification and termination) along with related performance and fault management of the VNFs instantiated over the virtualized (radio, compute and storage) resources. As shown in the figure, the So-Mtp interface enables the specific entities of the 5GT-SO (VNFM and NFVO-RO) to communicate with a single logical point of contact (SLPOC) at each 5GT-MTP entity. Accordingly, four reference points for the 5GT-SO SBI are conceived:

- **So-Mtp(-Resource Advertising Management).** It provides the Resource Advertisement Management functions. That is, it allows feeding the 5GT-SO's NFVI repository with information regarding the virtualized resources that will accommodate requested NFV-NSs. Such information, modelled in 5G-TRANSFORMER D2.3 [27], can be delivered using different levels of details/abstraction. Thus, the adopted abstraction, and view of the resources, will notably impact on the 5GT-SO NFVO-RO algorithms used for the VNF placement and/or networking computation. The mechanism used by the 5GT-MTP(s) to update the 5GT-SO's NFVI could be achieved also via different mechanisms such as immediate update when a change in any (abstracted) virtualized resource occurs (e.g., allocation or reservation), upon an explicitly demand sent by the 5GT-SO, or even periodically.
- **So-Mtp(-Resource Management).** This encompasses the Resource Management operations over the virtualized resources. Basically, it contains the set of operations used for reserving, allocating, updating (in terms of scaling up or down) and terminating (i.e., release) the resources handled by each 5GT-MTP. In short, and according to the abstracted information managed by the 5GT-SO, the So-Mtp(-RM) interface allows to coordinate the involved 5GT-MTPs to manage the utilization of a selected set of virtualized resources. This entails the VNF placement and triggering the reservation / allocation of the network resources constituting the demanded NFV-NS.
- **So-Mtp(-Resource Monitoring Management).** This provides the Resource Monitoring Management operations. Basically, it provides the required interworking procedures including the primitives and parameters for supporting the 5GT-SO Monitoring Service capability. This entails a twofold functionality: i) fault management to recover / restore the interrupted NFV-NS by a failure (e.g., link failure, VNF host crashes, etc.); ii) permanent performance monitoring crucial to ensure the demanded SLA for the existing NFV-NS. Obviously, the involved information needed to such Monitoring Service functions are also related with the adopted granularity and abstracted information (i.e., level of detail) within the 5GT-SO.
- **So-Mtp(-VNF).** This provides the general VNF lifecycle management (e.g., scaling up/down a particular VNF instance, fixing VNF malfunctions, etc.) commanded by the 5GT-SO VNFM. Moreover, the VNF configuration (i.e.,

specifying targeted parameters defining the targeted VNF behaviour) is also supported over this reference point. Last but not least, the So-Mtp-VNF reference point supports performance and fault management functionalities to monitor the VNF operations and adopt/apply (if needed) necessary actions to revert/solve VNF failures and performance degradations.

As anticipated above, the implementation of the 5GT-SO SBI operations and, particularly, those driven into the four reference points, leverage the procedures (i.e., interworking between entities, messages and basic contents) described in ETSI GS NFV-IFA 005 [2], 006 [25] and 008 [26] resp., as much as possible. Nonetheless, focusing exclusively on the operations currently supported in [2], [25], and [26], the following operations are identified as essential for encompassing the implementation of the 5GT-SO SBI reference points:

- For the **So-Mtp(-RAM),** a set of pairs of request/response messages is considered. This is divided into two main sets/groups, namely, Virtualized Resources Information Management and Virtualized Resources Capacity Management. These two subsets of messages (see [2] and [25]) grant 5GT-SO with multiple functionalities such as subscription to specific (filtered) resource information, query of and update (for changes) resource information, specific resource capacity, etc. Specifically, the resource capacity can be provided with respect to its current status (i.e., available, allocated or reserved) as well as specifying the amount of resources. Moreover, the resource information can be retrieved for a particular 5GT-MTP governing a certain Resource Zone (e.g., geographic NFVI-PoP specifying the reachable endpoints). With respect to the amount of resources, this clearly depends not only on the type of virtualized resource but also on the abstraction policy. An abstraction model is planned (for the sake of scalability and/or confidentiality purposes) to keep track of selected and relevant information about the virtualized resources (i.e., compute, storage and networking) without having a complete awareness of all the features and capabilities intrinsic to such resources. As described in D2.3 [27], this abstraction may rely on basically knowing the maximum, available and allocated amount of each resource type, or having a summarized topology view enabling basic connectivity between remote network elements hiding details of the underlying transport infrastructure supporting such connectivity. In this regard, for instance, for the compute resources, it can be delivered the total amount of available or allocated virtual memory and virtual CPU; for storage, the information is related to the size of storage and type (e.g., volume, object) or the support of remote direct memory access; for networking resources, the provided attributes regarding the link type (e.g., VLAN or GRE), the supported link QoS parameters (e.g., latency), the total link bandwidth, the IP addressing for a specific (sub-)network, the port types connected to specific network elements (e.g., Layer 1, 2 or 3), etc.
- For the **So-Mtp(-RM)**, in ETSI GS NFV-IFA 005 [2], sets of request/response messages to allocate, query, update, migrate, and terminate virtualized resources are specified. For instance, for compute resources the set named Virtualized Compute Management Interface is defined. This provides the specific allocation of compute resources over a particular Resource Zone, with a

determined set of virtual CPUs and memory, as well as informing about the software image to be set on the Virtual Machine. For network resources, the set Virtualized Network Resource Management Interface takes over of all the operations to be made over the network resources. Non-exhaustively, this includes the allocation of selected bandwidth over a network entity (e.g., link) or the utilization of an entire data port. In addition, the operations to create Network Forwarding Paths (NFPs) to deploy VNFFG are supported. The request messages should include the list of virtual networks and ports forming the NFP. In addition, the Virtualized Storage Resource Management Interface entails the set of operations (mapped to pair of messages) handling the storage resources. Similar to the compute and networking resources, these operations enable the selection of an amount of storage to be allocated over, e.g., a particular Resource Zone. Finally, ETSI GS NFV-IFA 005 [2] also defines a interface to reserve virtualized resources referred to as Virtualized Resource Reservation Interfaces. This interface is used to (pre-)book virtualized resources which eventually may be needed and used.

- For the **So-Mtp(-RMM)**, ETSI GS NFV-IFA 005 [2] also specifies the interfaces supporting fault management and performance monitoring. Specifically, the Virtualized Resource Fault Management Interface defines the messages enabling the 5GT-SO to subscribe for notifications from 5GT-MTP about containers or virtual machines crashes, virtual network port errors or reserved resources unavailable or exhausted. To this end, such interface supports detailed alarms. The Virtualized Resources Performance Management Interface describes a set of messages used for collecting measurements within notifications that will feed the 5GT-SO's Monitoring Service. These messages include resource consumption, memory oversubscription, disk latency, etc. In general, the collection of such information is controlled by a Performance Monitoring (PMON) job. The interface is oriented towards handling the management of PMON jobs (creation, subscribe, update, query, etc.). For a given PMON job, it can be specified the object to be monitored (e.g., CPU power consumption in VM), the performance metric, the frequency for capturing the measurements, threshold to send notifications, etc.
- For the **So-Mtp(-VNF)**, ETSI GS NFV-IFA 008 [26] describes the messages and contents supporting the operations for the creation / configuration / termination, scaling (up / down), monitoring and fault management of VNFs. In this context, [26] firstly addresses the necessary set of messages (as request/respond pairs) used for both initially configuring and modifying (e.g., deleting) a VNF (or Component). In general, the messages providing a VNF operation must carry a unique identifier to unambiguously determine over which particular VNF (or Component) the action will be conducted. Moreover, configuration data or parameters are also included specifying the amount of required memory, CPU capacity, storage size, connection points (address and ports), software image of the VNF container, etc. Another interworking operation supported by **So-Mtp(-VNF)** is the VNFM Indication action. These indicators are used to notify the 5GT-SO VNFM about a VNF behaviour which can be eventually used by the VNFM to trigger auto-scaling operations. Finally, the performance and fault

management interworking enable creating PMON jobs to impose the generation of notifications sending specific VNF parameters status (e.g. CPU) which are then gathered and processed. To this end, the interface entails the creation of thresholds to manage the notification message creation or the periodicity when such notifications are actually composed and sent. Last but not least, for the fault management purposes, the 5GT-SO VNFM is able to indicate the subscription demanding for specific alarms generated by the VNFs when, for instance, to react when a fault occurs.

6.5.3 5GT-SO EBI/WBI

The 5GT-SO eastbound/westbound interface (EBI/WBI) provides federation of services and federation of resources. 5GT-SO EBI/WBI enables interaction between a local 5GT-SO and other SOs that are part of other administrative domains. The approach of the So-So interface is adopted from the SLPOC for federation (SLPOC-F) solution of the ETSI GS NFV-IFA 028 [28]. The SLPOC-F solution offers two modes of operation: direct and indirect. The difference is on the NFVIaaS (resource federation) use case. In the SLPOC-F direct mode the consumer VNFM can directly invoke resource management operations on the provider 5GT-SO NFVO-RO. In SLPOC-F indirect mode, the consumer VNFM uses the consumer 5GT-SO NFVO-RO as a proxy (or forwarding point) to invoke resource management operations on the provider 5GT-SO NFVO-RO. Both approaches are suitable for the 5GT architecture, where the indirect mode was finally selected. In that sense, the 5GT-SO EBI/WBI is based on six reference points (see Figure 10

- **So-So(-Life Cycle Management)**, used for the operation of federated nested NFV-NSs. The reference point is used to instantiate, terminate, query, update or re-configure or receive notifications for federated nested NFV-NSs in both directions as provider or consumer role. 5GT-SO's Composite NSO is using this reference point;
- **So-So(-Catalogue)**, used for the management of Network Service Descriptors (NSDs) flavors and MEC Application Packages, including their Application Descriptors (AppDs). This reference point offers primitives for on-boarding, removal, updates, queries and enabling/disabling of descriptors and packages. 5GT- SO's Composite NSO is using this reference point;
- **So-So(-MON**itoring**)**, used for the monitoring of network services through queries or subscriptions/notifications about performance metrics, VNF indicators and network service failures. SLA Manager is using this reference point;
- **So-So(-R**esource **M**onitoring **M**anagement**),** used for monitoring of different resources, computing power, network bandwidth or latency, storage capacity, VMs, MEC hosts provided by the peering administrative domain. The details level depends on the agreed abstraction level. SLA Manager is using this reference point.
- **So-So(-R**esource **M**anagement**),** used for the operation of performing allocation, configuration, updates and release of resources. The Resource Management reference point offers operations such as configuration of the resources, configuration of the network paths for connectivity of VNFs. These operations mainly depend of the level of abstraction applied to the actual resources. 5GT-SO's SO-SO Resource Management & Advertising is using this reference point;
- **So-So(-R**esource **A**dvertising **M**anagement**),** used for advertising available resource abstractions to/from other SOs. Broadcast of available resources or resource abstractions upon capability calculation and periodic updates for near

real-time availability of resources. The SO-SO Resource Management & Advertising is using this reference point.

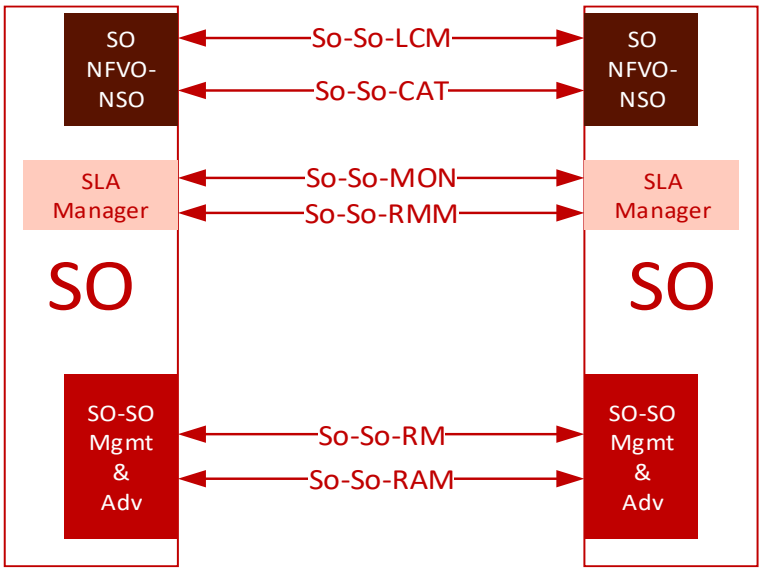


FIGURE 10: 5GT-SO EBI/WBI reference points

The 5GT-SO EBI/WBI implementation is based on ETSI GS NFV-IFA 013 [5], ETSI GS NFV-IFA 005 [2], ETSI GS NFV-IFA 006 [25] and ETSI GS NFV-IFA 008 [26]. Service federation provides nested NFV-NSs through the reference points of the 5GT-SO NFVO-NSO and the SLA Manager, mainly by implementing and extending interfaces of ETSI GS NFV-IFA 013 [5]. In particular:

- The So-So-LCM (Lifecycle Management) is implemented through the **ETSI GS NFV-IFA 013** [5] **NFV-NS Lifecycle Management Interface**. The interface provides request messages for instantiation, creation, scaling, update, querying, termination and deletion of network services;
- The So-So-CAT (Service Catalogue) is implemented through the **ETSI GS NFV-IFA 013** [5] **NSD Management Interface** and the **ETSI MEC 10-2 Application Package Management Interface**. The reference point implements mechanisms for providing updates of flavors and querying NSDs (ETSI GS NFV-IFA 013 [5]) as well as on-boarding or applying changes of MEC Application Packages (MEC 10-2).
- The So-So-MON (Service Monitoring) is implemented through the **ETSI GS NFV-IFA 013** [5] **NFV-NS Performance Management Interface** and the **ETSI GS NFV-IFA 013 NFV-NS Fault Management Interface**. They provide messages for monitoring of performances as well as notifications for alarms related to NFV-NSs;

NFVIaaS or resource federation provides resources or resource abstractions through the reference points of 5GT-SO NFVO-RO or the SO-SO Resource Management & Advertising module and the SLA Manager, generally by implementing interfaces of ETSI GS NFV-IFA 005/006/008 [2][25][26]. In particular:

- The So-So-RMM (Resource Monitoring Management) implementation is based on **ETSI GS NFV-IFA 005/006 [2][25] Virtualized Resource Performance Management Interface, ETSI GS NFV-IFA 005/006 [2][25] Virtualized Resource Fault Management Interface, ETSI GS NFV-IFA 008 [26] VNF Lifecycle Change**

Notification and ETSI GS NFV-IFA 008 [26] VNF Performance & Fault Management. The limited implementation provides information about the performance metrics and monitoring reports of the allocated federated resources or abstractions as well as alarms or notifications of resource failures or VNFs;

- The So-So-RM (Resource Management) implementation is based on the **ETSI GS NFV-IFA 005/006 [2][25] Software Image Management Interface**, **ETSI GS NFV-IFA 005/006 [2][25] Virtualized Compute/ Network/Storage Management Interface, ETSI GS NFV-IFA 008 [26] VNF Lifecycle Management, ETSI GS NFV-IFA 008 [26] VNF Configuration Management**. The implementation is limited and directly dependable of the abstraction level and federation level. Operations for managing VNFs or VMs, software images as well as basic operation and management of allocated compute/network/storage resources;
- The So-So-RAM (Resource Advertising Management) implementation is based **on ETSI GS NFV-IFA 005/006 [2][25] Virtualized Resource Quota Interfaces**. The implementation is limited to the query requests and broadcasting updates with purpose of advertising available resources or resource abstractions.

It is important to note that each relation on the 5GT-SO EBI/WBI (between two peering SOs) has different limitations (e.g. advertising and/or monitoring storage capacity, without vendor indicator or type of storage parameters, etc.) on each reference point due to hiding details of the NFV-NSs or abstractions of federated NFVI resources. These limitations depend of the business agreement levels and policies applied by each administrative domain. 5GT-SO EBI/WBI contains similar functionalities enabled through NBI and SBI with expected limitations.

6.6 Service Orchestrator Features

In this section, we provide a detailed presentation of core features of the 5GT-SO. We begin by presenting a common framework for modeling 5GT-SO-level VNF placement algorithms, as well as four such algorithms optimizing for different criteria, which we have designed, implemented, and evaluated. We then present the runtime service scaling capabilities of 5GT-SO: scaling can be triggered by the vertical, by the 5GT-VS, or automatically by the 5GT-SO via auto-scaling rules. Service composition, assurance and SLA management are also discussed. The rest of the section focuses on the federation features of the 5GT-SO, and in particular service (cross-domain NFV-NS establishment) and resource (NFVIaaS) federation.

6.6.1 Placement algorithms: framework, description, and evaluation

In this section, we provide four different VNF placement algorithms that operate at the SO level. These algorithms build on a common framework that defines a set of optimization goals and constraints. Each algorithm has different features and capabilities, optimizes for potentially different criteria, and may support different constraints.

Table 6 presents a qualitative comparison of these algorithms, focusing on their distinctive characteristics. A detailed description of each algorithm follows.

TABLE 6: 5GT-SO PLACEMENT ALGORITHM FEATURES

PA	Features	Optimization criteria
A	Genetic algorithm. Can be configured to optimize for one of three different objectives.	Either cost, latency, or service availability
B	Minimizes latency between VNFs placed at remote NFVI-PoPs. Also considers inter-PoP link physical distance.	Latency
C	Considers penalties associated with SLAs. Uses traffic forecasting. Reconfigures resource (re)allocation dynamically in timeslots.	Cost/revenue, resource utilization
D	Clustering techniques to group VNFs and NFVI-PoPs according to VNF link requirements and inter-PoP link capacity, respectively. Heuristic to minimize cost and delay in two separate steps.	Cost vs. latency tradeoff

6.6.1.1 Framework and system model

The different placement algorithms that have been designed inside the 5GT-SO follow (unless otherwise noted) a common system model, which is presented in this section. Table 7 summarizes the notation used in our model.

TABLE 7: SUMMARY OF THE NOTATION USED IN OUR PLACEMENT ALGORITHM FRAMEWORK

Symbol	Definition
V	Set of VNFs.
R	Set of resource types.
H	Set of physical points where a VNF can be placed. Can be physical hosts or their aggregation, e.g., NFVI-PoPs or data centers (depending on the abstraction provided by the 5GT-MTP). We use the terms host and NFVI-PoP interchangeably.
S	Set of services to deploy in a single PA request.
$r(v,\rho)$	Amount of resource type ρ requested by VNF v.
$d(v)$	Delay incurred by VNF v to process a request.
$f(v_1,v_2)$	Capacity (traffic volume) requirements for VNFFG edge $v_1 \rightarrow v_2$.
$n(s,v)$	Number of times a requests of service s visits VNF v.
$P(v_2\|v_1,s)$	Probability that a request of service s visits VNF v_2 immediately after v_1.
$D^{max}(s)$	Maximum acceptable delay for service s, it corresponds to the end-to-end delay requirement encoded in the NSD.
$T(h_1,h_2)$	Capacity of the virtual link from host h_1 to host h_2.
$\delta(h_1,h_2)$	Latency of the virtual link from host h_1 to host h_2.
$C(h,\rho)$	Capacity of host h in terms of resource type ρ.
$m(v)$	Binary parameter indicating that v is a MEC application.
$M(h)$	Binary parameter indicating that h is a MEC-capable host.
$\kappa^{max}(s)$	Maximum deployment cost allowed for service s.
$x(h,v)$	Binary decision variable indicating that VNF v is placed at host h.
$l(h,v)$	Physical distance[3] between the location of host h and the center of the area requested to place VNF v.
$\Re(v)$	Radius of the area around a specified location where VNF v is allowed to be placed.

[3] Geographical locations are expressed as GPS coordinates. This distance is therefore the great-circle distance calculated using the haversine formula [8].

We consider that the 5GT-SO receives two main pieces of information, upon which it can leverage to make orchestration decisions. The former is provided by the 5GT-VS and is given in the form of the service VNF Forwarding Graph (VNFFG), i.e., the set of VNFs and edges connecting them, and the deployment flavors representing the service to be deployed and the associated requirements. The latter is provided by the 5GT-MTP and refers to the available resources. The representation of the resources depends on the abstraction level, although our algorithms can work with multiple different levels of abstraction. Thus, in the following we will refer to the resource representation as a host graph, where hosts (i.e., vertices) can be either physical machines, or NFVI-PoPs, depending on the assumed abstraction level, and edges are virtual links (VLs) connecting hosts.

As far as the **VNFFG** is concerned, we denote its VNFs (i.e., vertices) by $v \in V$, each requiring an amount $r(v,\rho) \geq 0$ of resource type $\rho \in R$. Resource typeset R includes resources such as CPU, memory, and storage. In detail, r-values account for both the quantity of traffic each VNF has to process (e.g., in Mbits), and the amount of computational resources needed to process each unit of traffic (e.g., in CPU cycles per Mbit). Each time data traverse a VNF v, this incurs a delay $d(v)$. For each pair of VNFs $v_1, v_2 \in V$, i.e., for each edge of the VNFFG, we know the amount of traffic $f(v_1, v_2) \geq 0$ flowing from v_1 to v_2. Clearly, $f(v_1, v_2) = 0$ means that there is no traffic between those VNFs.

The 5GT-SO also knows the set of **services** to be deployed, $S = \{s_1, s_2, \dots\}$, the number of times $n(s,v)$ requests of service s visit VNF v, the probabilities $P(v_2|v_1,s)$ that they visit v_2 immediately after v_1, and the maximum acceptable delay for that service $D^{max}(s)$. If $n(s,v) = 0$, VNF v is not part of service s. The aforementioned probabilities indicate how branching in the VNFFG is implemented, and are mainly used when calculating an estimate of the end-to-end network latency for a given service. For example, a load balancer v_1 may be present to split the traffic load equally towards two identical VNFs v_2 and v_3. In this case, $P(v_2|v_1,s) = P(v_3|v_1,s) = 0.5$.

The **host graph** has hosts $h \in H$ as vertices, each with capabilities $C(h,\rho) > 0$ for each resource type. Links (VLs) between hosts have capacities $T(h_1, h_2)$, expressing the maximum *total* quantity of traffic that can flow per second from VNFs hosted at h_1 to VNFs hosted at h_2. Similarly, requests traversing a link incur a delay $\delta(h_1, h_2)$.

It should be noted that a VNF may in fact be a MEC application, which is a constraint the algorithm needs to consider by placing it at a MEC host. The requirement to be placed at the edge is indicated by m, where $m(v) = 1$ if the VNF is a MEC application and 0 otherwise. In the same spirit, $M(h) = 1$ if a host is MEC-capable, i.e., it is located at the mobile edge and can host MEC applications, and 0 otherwise.

The main **decision** to make at the 5GT-SO is whether to place an instance of VNF v at host h, expressed through a binary variable $x(h,v) \in \{0,1\}$. Each VNF placement incurs a cost $\kappa(h,v,op)$. A very relevant factor contributing to κ is represented by the fees charged by each different infrastructure provider, *op,* for the usage of their infrastructure by placing VNF v at host h. The fees are pre-determined and defined by each provider. The maximum cost per service is denoted by $\kappa^{max}(s)$.

Two sets of **constraints** must be satisfied, concerning the **capabilities** of hosts and the **capacity** of links, i.e.,

$$\sum_{v \in V} r(v,\rho)x(h,v) \leq C(h,\rho), \qquad \forall h \in H, \rho \in R,$$

$$\sum_{v_1,v_2 \in V} x(h_1,v_1)x(h_2,v_2)f(v_1,v_2) \leq T(h_1,h_2), \qquad \forall h_1,h_2 \in H.$$

In addition, **latency** constraints, accounting for both request processing and propagation delays, as well as **cost** constraints, have to be met. For each service $s \in S$, the following must hold.

- The sum of processing delays and delays to traverse links should be below the maximum end-to-end latency threshold indicated by the vertical:

$$\sum_{v \in V} n(s,v)d(v) + \sum_{v_1,v_2 \in V} n(s,v_1)P(v_2|v_1,s) \cdot \sum_{h_1,h_2 \in H} x(h_1,v_1)x(h_2,v_2)\delta(h_1,h_2) \leq D^{max}(s).$$

The first term of the left hand side of the above expression represents the latency incurred to process a request at the service's traversed VNFs. Although in many practical scenarios a given service request would normally traverse each VNF once ($n(s,v)=1$), there could be cases[4] where a request has to pass a VNF multiple times, resulting in higher processing delay. The second term is an estimate of the latency a request experiences as it traverses the edges of the VNFFG. Each such edge contributes to network latency proportionally to its traversal probability. The third sum in the expression amounts to zero for all host pairs other than the one where VNFs v_1 and v_2 have been placed.

- The sum of fees for the deployment of all the VNFs composing a service should be below the service's maximum allowed cost:

$$\sum_{v \in V, h \in H, op} k(h,v,op)x(h,v)n(s,v) \leq \kappa^{max}(s).$$

MEC-related constraints are captured by the following expression:

$$x(h,v)m(v) \leq M(h), \qquad \forall h \in H, v \in V.$$

The above constraints guarantee that a MEC application v (i.e., for which $m(v)=1$) cannot be placed at a non-MEC host (i.e., for which $M(h)=0$). Note that a regular VNF is allowed to be placed at a MEC host.

Finally, the **location** of a VNF can be constrained to a specific geographical area defined as a circle around given coordinates and with a given radius, as provided by the vertical. At the same time, location information about each host/NFVI-PoP (depending on the abstraction) is exposed by the 5GT-MTP. If a location constraint is in place for a VNF, the latter can be deployed only at one of the locations that are in the area specified by the vertical. In this case, the following should hold:

$$x(h,v)l(h,v) \leq \Re(v), \qquad \forall h \in H, v \in V,$$

where $l(h,v)$ is the geographical distance between the center of the desired area specified by the vertical for VNF v and the location exposed by the 5GT-MTP for host/NFVI-PoP h, and $\Re(v)$ is the radius of the desired area.

Based on this system model, different placement algorithms developed in this project are introduced along with the performance evaluations in the following sections, while

[4] For example, a VNFFG may specify that packets reaching a VNF needs to be passed to another VNF performing deep packet inspection, and then sent back to the former to continue their route.

each of the algorithms look into different optimization criteria and applied to different use cases, as mentioned at the beginning.

6.6.1.2 Placement Algorithm A

6.6.1.2.1 Description

Placement Algorithm A follows the Genetic Algorithm (GA) paradigm [9]. Genetic Algorithms are a generic heuristic methodology for solving optimization problems by efficiently exploring the space of potential solutions. A solution is represented as a *chromosome*, which is in turn composed of a number of *genes*, each corresponding to a specific property.

A genetic algorithm operates as follows: Starting from a pool of initial solutions, it iteratively applies genetic operations to selected chromosomes to produce *offspring* (i.e., new chromosomes) of better quality according to a *fitness* function, in an attempt to drive search towards the optimal solution. The main genetic operations are *crossover* and *mutation,* as explained below. In each iteration of the algorithm (a *generation*, in the GA terminology), a new solution pool is generated including the highest quality chromosomes out of the population of existing ones and their offspring. After a fixed number of generations or if it is detected that new chromosomes do not significantly improve the quality of the solution, the algorithm terminates returning the chromosome with the highest fitness function value. We have adapted the GA methodology to our problem context as follows:

Encoding: In our case, a chromosome is a specific placement solution. A gene corresponds to a specific NFVI-PoP, together with the set of VNF instances placed at it.[5]

Fitness function and selection: Our genetic algorithm can be adapted to optimize for different criteria. A cost or delay-minimizing algorithm uses as fitness functions directly the expressions defined in Section 6.6.1.1 for calculating the overall cost and the end-to-end latency of a service, respectively.

At the end of a generation, the fitness function is calculated for each chromosome and the top-*S* of them form the new solution pool.

If VNF, host, and link failure rates are available, a service reliability-aware fitness function can be defined, where the aim is to find a placement which minimizes the probability that any VNFFG component fails.

For the case of service availability, the system model of Section 6.6.1.1 is extended as follows: Let $q_i^{(V)}$, $q_j^{(H)}$, and $q_{(h1,h2)}^{(L)}$ denote the failure rates of the i-th VNF, the j-th host and the host VL (h_1, h_2), respectively. We assume that the $q_i^{(V)}$ values are available by the vertical providing the VNFs and the latter values are exposed by the 5GT-MTP. We define service availability as the probability that *all of the VNFs composing the service instance are accessible*, which implies that (i) no VNF instance fails due to internal reasons, (ii) no host where any of the VNFs deployed fails, and (iii) no host VL involved in the service instance fails. Although VNF-hosted software, hosts, and links are assumed to fail independently, VNF failures can be correlated: For a single gene, if the corresponding host fails, all VNFs of the gene become inaccessible at the same time. A

[5] However, assuming a different level of abstraction where the placement algorithm would decide to place VNFs directly per host, a gene would represent a host instead of an NFVI-PoP.

gene's availability, that is, the probability that the VNF group instantiated at host *h* is available, is defined as follows:

$$a_h(y) = \Pr\{all\ VNFs\ are\ functional\ and\ h\ is\ up\} = \left(1 - q_h^{(H)}\right) \prod_{v \in V | y(h,v)=1} \left(1 - q_v^{(V)}\right).$$

The overall availability of a placement is thus given by the probability that all genes (correlated VNF groups) are available *and* no VNF link fails (due to the underlying host link):

$$A(y) = \prod_{h \in H | \{\exists v \in V | y(h,v)=1\}} a_h(y) \prod_{h_1,h_2 \in H | \{\exists v_1,v_2 \in V | y(h_1,v_1)=1, y(h_2,v_2)=1\}} (1 - q_{(h_1,h_2)}^{(L)}) \ .$$

Initial population generation: Our GA begins by creating an initial pool of S chromosomes by performing random VNF-to-host assignment. For each initial solution, the capacity, delay and availability constraints (if present) are checked, and, if violated, the chromosome is rejected and a new random placement is attempted.

Crossover: The crossover rate (r_c) parameter determines the number of offspring to produce per generation. The selection of r_c involves a trade-off between execution time and solution quality; typical values range between 0.5 and 1. Each crossover operation selects two random chromosomes from the pool and combines their genes to derive a new one. To improve the quality of the produced offspring, we do not select the genes to combine blindly. Rather, we introduce specific gene efficiency metrics per optimization objective, and select the genes of the two parents with the highest efficiencies. After the two parent solutions are selected, their genes are sorted by their efficiency values. The crossover operation proceeds by selecting the genes with the highest efficiencies until all VNFs have been placed. In the event that a VNF of a gene that is about to be added to the new chromosome has already been placed at another more efficient host, it is ignored. If a gene's host has already been used, the gene is ignored. This however can cause a situation where, after all genes of the two parents have been checked (and have either been included in the new chromosome or ignored), there are still some VNFs to be placed. If this is the case, the remaining VNFs are placed in the first available host in a first-fit manner.

Efficiency functions: Depending on the optimization criterion, one of the following efficiency metrics is used to evaluate a gene during crossover:

Cost efficiency: For each gene representing host *h*, its cost efficiency is given by $E_c(h,y) = \sum_{v \in V | y(h,v)=1} \kappa(h,v)$, i.e., the sum of the costs of the VNF instances placed at it.

Latency efficiency: To characterize a gene representing host *h* from a latency perspective, we use the average delays of *h*'s incoming and outgoing host edges ($\overline{d_{in}}(h)$ and $\overline{d_{out}}(h)$, respectively). A VNF contributes to the gene's latency efficiency via the VNFFG edges it is incident to. For example, if v_1 is placed at *h*, edge $(v1, v2)$ contributes $\overline{d_{out}}(h)P(v_2|v_1,s)$. Our rationale is to mix hosts with better delay properties in the new chromosome. Note that we ignore processing delays in this calculation, since, according to our system model, the $d(v)$ values are independent of the underlying hosts. The latency efficiency for a specific service is thus given by:

$$E_l(h,y,s) = \sum_{(v_1,v_2)|y(h,v1)=1,y(h,v2=0)} \overline{d_{out}}(h)P(v_2|v_1,s) + \sum_{(v_1,v_2)|y(h,v1)=1,y(h,v2=0)} \overline{d_{in}}(h)P(v_2|v_1,s).$$

This function promotes genes with many VNFs (since edges between them are zero-latency ones) and/or genes of "fast" hosts (i.e., those which have low-latency incoming and outgoing links). If there are more than one services, the latency efficiency of a gene is defined as the average of its E_l values across all services.

Service availability efficiency: To characterize the availability of a gene y, we directly use the expression for $a_h(y)$.

Mutation: At each generation, and with low probability, each chromosome is subject to a random change in its genes. In our case, we implement mutation by selecting two random hosts of a solution and exchanging their VNFs. If the resulting chromosome violates any constraint, it is rejected. The purpose of mutation is to introduce diversity in the chromosome pool in order to avoid being trapped into local optima. On the other hand, the mutation probability r_m should be very low so that the fitness-driven evolution of the population is maintained.

Termination: The algorithm terminates either when the value of the objective function does not improve by more than δ for a number of consecutive generations (convergence criterion) or after a fixed number of generations.

6.6.1.2.2 Performance evaluation

We have carried out a set of experiments to evaluate the convergence properties of Placement Algorithm A and to compare its behavior under different optimization criteria. We execute the algorithm on a simulated full mesh topology with 16 NFVI-PoPs, where the algorithm places three services, each composed of 5 to 10 VNFs. The algorithm terminates if for 30 consecutive generations the value of the best solution does not improve by more than $\delta = 10^{-4}$.

Table 8 presents the performance of the GA in terms of the cost and (network) latency objectives, as well as its execution time, for two different configurations (i.e., optimizing for cost and optimizing for latency). The reported results are average values across 30 iterations on the same topology[6]. 95% confidence intervals are shown in parentheses. The experiments executed on an Intel i5 machine with 16 GB of RAM, running Ubuntu 16.04.

TABLE 8: AVERAGE PERFORMANCE OF PLACEMENT ALGORITHM A (GENETIC ALGORITHM)

Configuration	Cost	Network latency (s)	Execution time (s)
Cost-driven	**110.86** (107.84, 113.88)	**0.0314** (0.03, 0.032)	**75.53** (63.73, 87.34)
Latency-driven	**232.2** (223.64, 240.76)	**0.019** (0.018, 0.02)	**84.43** (74.98, 93.88)

Figure 11 and Figure 12 present the behavior of the cost- and latency-minimizing genetic algorithms, respectively, as generations progress. Each figure shows the evolution of the objective function value for a single execution of the algorithm on the

[6] This is necessary due to the stochastic nature of the algorithm.

same topology and service scenario. Notably, in both cases, after a small number of generations, the value of the objective function is practically the same and the algorithm has converged to a solution.

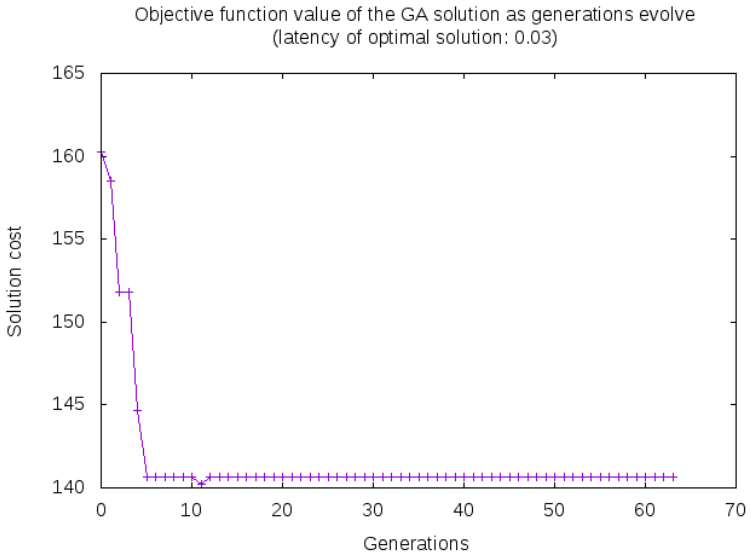


FIGURE 11: COST-MINIMIZING GENETIC ALGORITHM CONVERGENCE BEHAVIOR

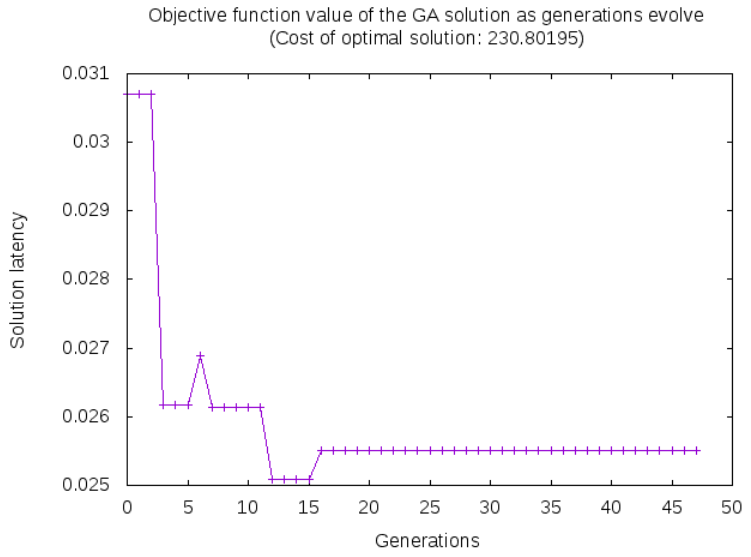


FIGURE 12: LATENCY-MINIMIZING GENETIC ALGORITHM CONVERGENCE BEHAVIOR

6.6.1.3 Placement Algorithm B

6.6.1.3.1 Description

The Placement Algorithm realized by SSSA and CTTC (a.k.a. Placement Algorithm B), is made of two different heuristics that with different objective functions. In the first Algorithm (i.e., Placement Algorithm B-1), the objective function is the minimization of the distance (MD) between the selected NFVI-PoPs. If we consider the network latency proportional to the length of a **logical link** (LL), the approach is equivalent in minimizing

the network latency of the requested NS. The second algorithm proposed (i.e., Placement Algorithm B-2) wants to minimize the overall latency (ML). In fact, in this algorithm, we also consider the NFVI-PoP internal latency parameter of each selected NFVI-PoP. To explain our devised algorithms, the following statement is considered:

$$min\left|\sum x_i\right| \leq \sum min|x_i| \qquad (1)$$

Thus, according to (1), to minimize the distance (the latency) of the overall end-to-end service, we can proceed by minimizing the distance (the latency) of a single segment/hop forming the entire service. This is done by the MD (ML) algorithms based on a set of input information provided by the RO-OE starting from (*i*) the NSD specifying the service requirements, and (*ii*) the abstract view of the NFVI specifying underlying available resources. Accordingly, the algorithms are provided with the VNFs to be deployed and their characteristics (e.g., required virtual links to connect them, etc.), and with the required details of the underlying (abstracted) NFVI infrastructure (e.g., NFVI-PoP capabilities such as aggregated available CPU, RAM and Storage, Logical Link (LL) attributes and related available resources, etc.).

The MD algorithm (Placement Algorithm -B-1) aims at minimizing the propagation latency experienced by the data while traversing DC interconnections, and thus the VNF placement decisions are performed according to the distance between a given pair of NFVI-PoPs, i.e., length of LL. Hence, for each NFVI-PoP selected for a VNF composing the service, the MD algorithm seeks for a NFVI-PoP with the minimum delay from the previously selected one, provided that it has sufficient available computing resources (i.e., CPU, RAM, storage), and the inter-NFVI-PoP interconnection (LL) meet both the bandwidth and the latency requirements expressed for the VNF virtual links. The pseudocode of MD algorithm is shown below.

We have defined the function CHECKRESOURCESAVAILABILITY (lines 1-5) to verify whether a particular NFVI-PoP has enough available computational (i.e., CPU, RAM, and Storage) capabilities to host/accommodate the requested VNFs. In lines 6-7 we verify if the input NSD specifies the presence of a Service Access Point (SAP).

For each VNF of the NSD that need to be allocated (line 8), the MD algorithm checks if ihe VNF is connected to the same VNFLink of the SAP and set the SAP parameter to True (lines 9-12). After that we also check if the VNF is a MEC_app, so it has to be placed in a MEC capable NFVI-PoP. If so, we set the MEC parameter to True (lines 13-14). Then the algorithm invokes CHECKRESOURCESAVAILABILITY to check for resource availability (line 15). If the NFVI-PoPs cannot deal with the VNF requirements, the NS request is blocked (line 16-17).

If we have to place the first VNF (line 18), if the SAP parameter is True, we look for a NFVI-PoP (from the list of the available NFVI-PoP) whose coordinates are within the SAP area and we add it to a list *possibleNFVI-PoP* (lines 18-22). If the SAP parameter is False, the list *possibleNFVI-PoP* correspond to the list *available_NFVI-PoP* (lines 23-24). The MD algorithm selects randomly the NFVI-PoP where to allocate the VNFfrom the *possible_NFVI-PoP* list (line 25) and adds it to the *usedNFVI-PoP* list (line 26). Otherwise, if the VNF is not the first one from the targeted NSD to be allocated, both *available_NFVI-PoP* and *possibleNFVI-PoP* are empty (line28) and the CHECKRESOURCESAVAILABILITY (line 29) function is invoked removing the previous *selectedNFVI-PoP* (i.e., the last NFVI-PoP appearing in the *usedNFVI-PoP* list). Indeed, such a previous chosen NFVI-PoP is considered to be the source endpoint of the logical link (LLsrc) that eventually will allow the remote connectivity between the

resulting NFVI-PoPs. In this case, for each NFVI-PoP in *available_NFVI-PoP* (line 33) , if the MEC parameter is set to True, we add to *possibleNFVI-PoP* all the NFVI-PoP Mec capable, otherwise, if MEC is False, *possibleNFVI-PoP* correspond to the list *available_NFVI-PoP* (Lines 33-37). From the *VNFLink* structure (within the NSD), it is extracted the networking requirements in terms of both the bandwidth (*Bw*) and latency (*l*) that the virtual link connecting a pair of VNFs must satisfy (line 38).

In the *candidateLL* list it is added all the LLs that have the specific selected source NFVI-PoP, previously allocated (line 45), and do guarantee the *VNFLink* service requirements (i.e., *Bw* and the *l*). Among all the elements in the *candidateLL* list, the shortest one (line 47) with respect to the distance is chosen. For the sake of clarification, it is worth noting that among the set of attributes describing every LL, there is a parameter related to the real LL distance (e.g., expressed in km) for each pair of interconnected NFVI-PoPs. The chosen source LL (referred to as *LL_s*) is then added to the *usedLL* (line 48). The remote *LL_s* endpoint, that is the destination NFVI-PoP (*LLdst_s*) is then added as well in the *usedNFVI-PoP* (line 30). Finally, it is updated the resulting *TotalLatency* with the latency associated to the *LL_s* (*latency_s*) and the internal latency of the selected NFVI-PoP (line 50).

Algorithm B-1: Minimum Distance (MD) Algorithm

Input: *NSD*, *NFVI*
Output: response=[usedNFVI-PoPs,usedLLs, TotalLatency] **OR** BlockVNF-FG_Req
Inizialization: Parse the request and extract VNFs parameter to be allocated
available_NFVI-PoP = Ø
candidate_NFVI-PoP = Ø
candidate_LL = Ø
usedNFVI-PoP = Ø
usedLL = Ø
TotalLatency = 0

```
1: FUNCTION CHECKRESOURCESAVAILABILITY:
2:        for each NFVI-PoP in NFVI:
3:                  check_resource_availability (CPU, RAM, Storage)
4:                  if resource_Available:
5:                            available_NFVI-PoP.append(NFVI-PoP_i)
6:if SAP in NSD:
7:   SAP_info=[coordinates, VNFLink_id]
8:for each VNF in NSD:

9:    for each CP in VNF:
10:      if CP['VNFLinkID']==SAP_info[1]
11:      //VNF has to be placed in a NFVI-PoP within the coordinates SAP_info[0]
12:        SAP=True
13:   if 'MEC' in VNF['requirements']:
14:      MEC=True
15:   CHECKRESOURCESAVAILABILITY
16:      if available_NFVI-PoP = Ø:
17:                            return BlockVNF-FG_Req
18:      elif usedNFVI-PoP = Ø:
19:      if SAP==True:
20:        for each NFVI-PoP in available_NFVI-PoP:
21:          if NFVI-POP['coordinates'] in SAP_info[0]:
22:              possibleNFVI-PoP.append(NFVI-PoP)
23:      else:
```

```
24:         possibleNFVI-PoP= available_NFVI-PoP
25:         selectedNFVI-PoP=random(possibleNFVI-PoP)
26:         usedNFVI-PoP.append(selectedNFVI-PoP)
27:       else:
28:         available_NFVI-PoP = Ø, possible NFVI-PoP= Ø
29:         CHECKRESOURCESAVAILABILITY
30:         if available_NFVI-PoP = Ø:
31:               return BlockVNF-FG_Req
32:       else:
33:         for each NFVI-PoP in available_NFVI-PoP:
34:           if MEC==True:
35:             possible_NFVI_PoP.append(MEC_capable_NFVI-PoP)
36:           else:
37:             possible_NFVI-PoP= available_NFVI-PoP
38:                   for each VNFLink:
39:                         get(Bw, l)
40:                         for each LL in NFVI:
41:                               i=length(usedNFVI-PoP)-1
42:                               if LLsrc = usedNFVI-PoP[i]:
43:                                     if LLbw>=Bw and LLl<=l:
44:                                           if LLdst in possible_NFVI_PoP ⅼLLsrc:
45:                                                 candidateLL.append([LL,
                                                    length, LLdst, latency])
46:                               for each candidateLL:
47:                                     min(candidateLL[i][2]} //compare length and
                                    select the shortest one
48:                                     usedLL.append(LL_s)
49:                                     usedNFVI-PoP.append(LLdst_s)
50:                               TotalLatency=+latency_s+NFVI-PoPsInternalLatency
51: return response
```

The ML algorithm (Placement Algorithm B-2) aims at at minimizing the overall latency which is defined as the sum of the NFVI-PoP internal latency and the incurred LL latency that connects them. Thus, VNF placement decisions are performed according to the processing latency featured by NFVI-PoPs and based on the distance between a given pair of NFVI-PoPs, i.e., length of LL. Hence, for each NFVI-PoP selected for a VNF composing the service, the ML algorithm seeks for a NFVI-PoP that minimize the sum of the processing latency at hosts and the propagation latency at the LL reaching that NFVI-PoPs, provided that it has sufficient available computing resources (i.e., CPU, RAM, storage), and the inter-NFVI-PoP interconnection (LL) meet both the bandwidth and the latency requirements expressed for the VNF virtual links. The pseudocode of the ML algorithm is shown below.

Similarly to the MD algorithm, we have defined the function CHECKRESOURCESAVAILABILITY (lines 1-5) to verify whether a particular NFVI-PoP has enough available computational (i.e., CPU, RAM, and Storage) capabilities to host/accommodate the requested VNFs. Also the ML algortihm also checks if in the NSD is specified the SAP location (Lines 6-7).

From line 8 to line 24, the procedure is the same as the MD Algorithm. Conversely to the MD algorithm, in the ML algorithm the NFVI-PoP where to allocate the first VNF in the one with the minimum Internal latency (line 25), that is added to the *usedNFVI-PoP* List (Line 26). The TotalLatency value is updated with the NFVI-PoP Internal latency (line 27).

Otherwise, if a VNF was already allocated, likewise the MD algorithm, the CHECKRESOURCESAVAILABILITY (line 30) function is triggered, where the previously selected NFVI-PoP (the last from the *usedNFVI-PoP* list) is removed. Recall that, as said before, such previously selected NFVI-PoP becomes the source endpoint of the LL (i.e., *LLsrc*). Also in this Algorithm the MEC parameter (if present) of a VNF has to match the MEC capability of the NFVI-PoP where to place it (Lines 35-38). From NSD's *VNFLink* structure, both the *Bw* and *l* requirements are extracted (line 40).

In the *candidateLL* list, it is added those existing LLs outgoing the chosen source NFVI-PoP that can accommodate the required *Bw* and the *l* of the *VNFLink*. For each of the resulting LL, the algorithm computes the *tempLatency* as the sum of the *LL_Latency* (delay associated to the LL) and the *src/dstNFVI-PoPInternalLatency*. Among all the elements in the *candidateLL* list, the one with the smallest computed *tempLatency* is chosen (line 49). This LL (called *LL_s*) is then added to the *usedLL* (line 50), where its destination NFVI-PoP (*LLdst_s*) is included into the *usedNFVI-PoP* (line 51). Last but not least the *TotalLatency* is updated with the previously computed *tempLatency* (line 52).

Algorithm B-2: Minimum Latency (ML) Algorithm

Input: *NSD, NFVI*
Output: response=[usedNFVI-PoPs,usedLLs, TotalLatency] **OR** BlockVNF-FG_Req
Inizialization: Parse the request and extract VNFs parameter to be allocated
available_NFVI-PoP = Ø
candidate_NFVI-PoP = Ø
candidate_LL = Ø
usedNFVI-PoP = Ø
usedLL = Ø
TotalLatency = 0

```
1: FUNCTION CHECKRESOURCESAVAILABILITY:
2:         for each NFVI-PoP in NFVI:
3:                 check_resource_availability (CPU, RAM, Storage)
4:                 if resource_Available:
5:                         available_NFVI-PoP.append(NFVI-PoP_i)
6:if SAP in NSD:
7:   SAP_info=[coordinates, VNFLink_id]
8:for each VNF in NSD:
9:   for each CP in VNF:
10:      if CP['VNFLink']==SAP_info[1]
11:        VNF has to be placed in a NFVI-PoP within the coordinates SAP_info[0]
12:        SAP=True
13:  if 'MEC' in VNF['requirements']:
14:     MEC=True
15:  CHECKRESOURCESAVAILABILITY
16:  if available_NFVI-PoP = Ø:
17:        return BlockVNF-FG_Req
18:  elif usedNFVI-PoP = Ø:
19:     if SAP==True:
20:       for each NFVI-PoP in available_NFVI-PoP:
21:          if NFVI-POP['coordinates'] in SAP_info[0]:
22:            possibleNFVI-PoP.append(NFVI-PoP)
23:     else:
24:       possibleNFVI-PoP= available_NFVI-PoP
25:       selectedNFVI-PoP=min(possibleNFVI-PoP['internal_latency'])
26:                 usedNFVI-PoP.append(selectedNFVI-PoP)
27:       TotalLatency=+NFVI-PoPInternalLatency
```

```
28:  else:
29:        available_NFVI-PoP = Ø, possible NFVI-PoP= Ø
30:        CHECKRESOURCESAVAILABILITY
31:        if available_NFVI-PoP = Ø:
32:              return BlockVNF-FG_Req
33:        else:
34:        for each NFVI-PoP in available_NFVI-PoP:
35:          if MEC==True:
36:             possible_NFVI_PoP.append(MEC_capable_NFVI-PoP)
37:          else:
38:             possible_NFVI-PoP= available_NFVI-PoP
39:                                for each VNFLink:
40:                                      get(Bw, l)
41:                                for each LL in NFVI:
42:                                      i=length(usedNFVI-PoP)-1
43:                                      if LLsrc = usedNFVI-PoP[i]:
44:                                            if LLbw>=Bw and LLl<=l:
45:                                                  if LLdst in possible_NFVI_PoP ⅼLLsrc:
46:                                                        tempLatency =
                                                    latency+LLdstInternalLatency
47:                                                        candidateLL.append([LL, LLdst,
                                                    tempLatency])
48:                                for each candidateLL:
49:                                      min(candidateLL[i][2]) //compare tempLatency and select
                    // the smallest one
50:                                      usedLL.append(LL_s)
51:                                      usedNFVI-PoP.append(LLdst_s)
52:                                      TotalLatency = +tempLatency
53: return response
```

6.6.1.3.2 Performance evaluation

The experimental performance evaluation is performed using the CTTC ADRENALINE testbed (represented in Figure 13) encompassing a cloud/network infrastructure. For the sake of clearness, the building block termed Cloud & Network Orchestrator (Allocator) in the picture takes into account a specific set of functions mostly related to the resource orchestration which in the framework of 5GT, these functions are performed at both the SO and the MTP. Moreover, T-SDN (PCECC) controller used for the experimental setup provides the operations and functions that a WIM plugin (within the context of 5GT architecture) provides. A detailed description of the testbed and of the experiment setting is reported in [11].

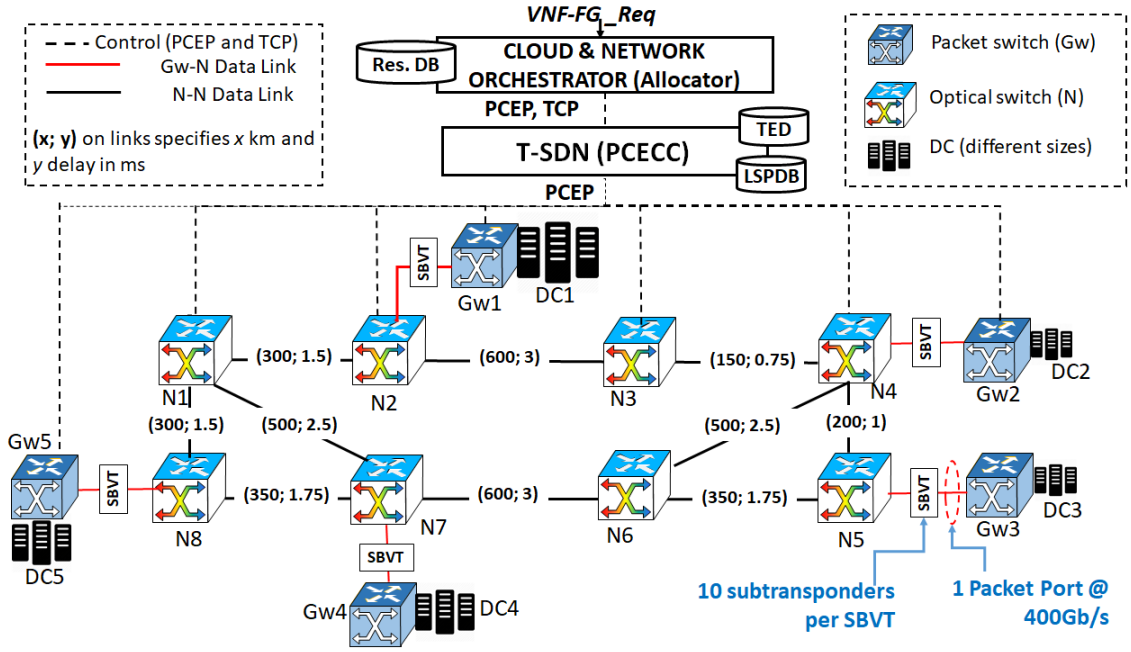


FIGURE 13: ADRENALINE TESTBED SETUP

The performance evaluation has been conducted by considering a network service composed of two VNFs. During the test we have not considered the location constraints related to the SAP and the MEC constraints.

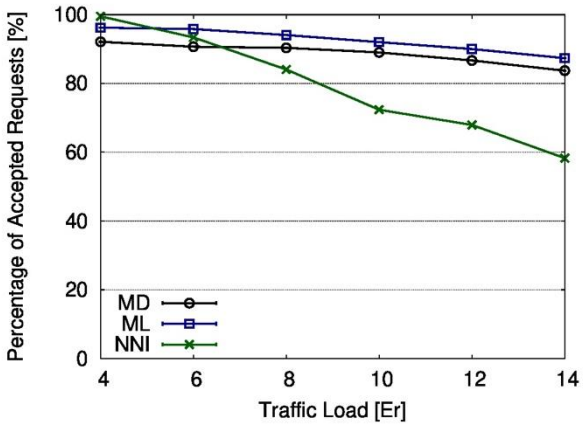


FIGURE 14: ACCEPTED REQUESTS

In Figure 14, the VNF-FG_Req's acceptance ratio is illustrated as a function of the traffic load. At the lowest traffic load, the NNI algorithm presents comparable acceptance ratio values with respect to both ML and MD algorithms. . The NNI algorithm is the one without orchestration capabilities. In fact, in this case, the Cloud and Network Orchestrator has no information (knowledge) about the underlying network. We recall that NNI performs separated cloud and networking resource selections and does not address the minimization of the distance (in km) and the latency (in ms) as done in MD and ML. This result in fewer constraints to be fulfilled thereby obscuring the less efficient resource selection due to the separation. This

explains the high values of acceptance rates for NNI, even higher than MD and ML for low traffic Load values (4 and 6 Er) where NNI achieves more than 99% of accepted requests. However, as traffic load increases, the performance of NNI significantly degrades (e.g., at traffic load set to 14 Er the acceptance rate is around 55%), whilst MD and ML algorithms present acceptance ratio values smoothly decreasing, being stabilized around 85%. The rationale behind this is at higher offered traffic load, cloud and networking resources become more occupied and the separated cloud and networking selections made by NNI encounters more difficulties to be satisfied (especially for the cloud) despite fewer constraints need to be fulfilled. On the other hand, the advantage of joint selections becomes evident for ML and MD where they allow attaining a more efficient use of the cloud and network resources thanks to the fact that both are in somehow jointly selected at the Allocator.

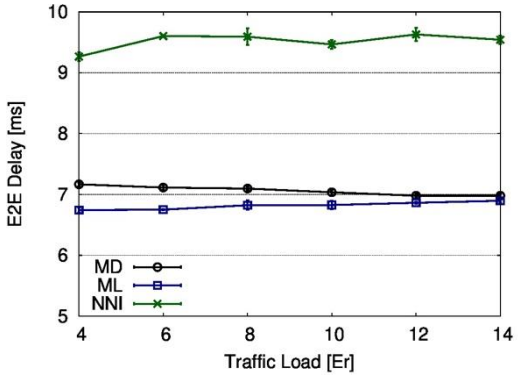


FIGURE 15: END-TO-END DELAY

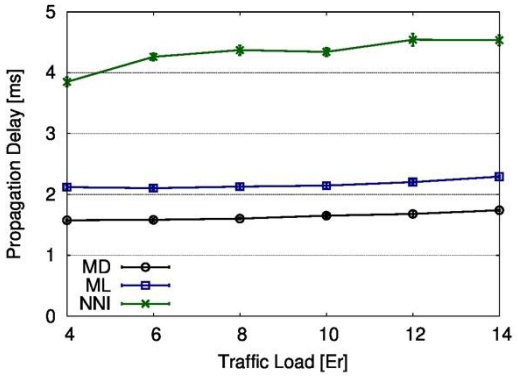


FIGURE 16: PROPAGATION DELAY

Figure 15 and Figure 16 depict the propagation delay (due to fiber links) and the end-to-end delay (i.e., propagation delay plus DC processing latency) for the orchestration approaches. As shown in Figure 15, ML approach does minimize the end-to-end experienced latency (i.e., sum of the propagation and processing delays). Conversely, as shown in Figure 16, MD approach, aiming at minimizing distance between source and destination DCs, attains the lowest propagation delay. As expected, NNI achieves the worst performance since it focuses on simply satisfying the VNF-FG_Req's latency requirement without conducting an optimization of any delay source.

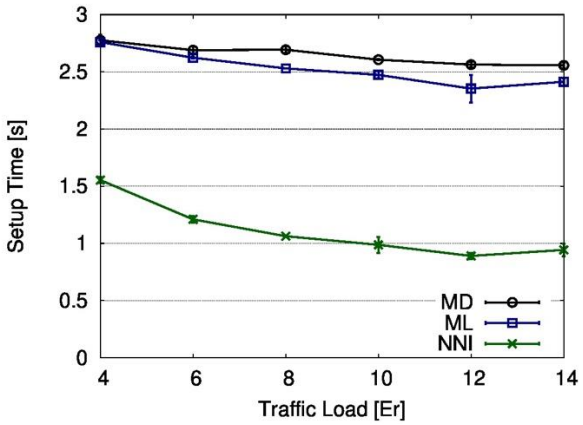


FIGURE 17: SETUP TIME

Figure 17 depicts the average setup time (in s) with respect to the offered traffic load. The setup time is computed as the elapsed time between the reception of an incoming VNF-FG_Req and when it is successfully set up. This indicator is always higher in both MD and ML orchestration strategies than NNI because it results the price to pay for a truly joint cloud and net-work orchestration requiring further operations with the T-SDN Controller to acquire aggregated network resource (at packet layer) information. The setup time overhead introduced by ML and MD approaches with respect to the NNI strategy is slightly above 2 seconds which remains almost steady as traffic load grows. However, we observe increasing traffic load the average setup time performance for all the resource orchestration strategies tends to be reduced. Traffic load being increased entails that larger resources (both compute and networking such as Gw's packet port bandwidth) are more occupied.

6.6.1.4 Placement Algorithm C

6.6.1.4.1 Description

The key idea of the Placement Algorithm C is to overbook the amount of NFV-NSs through driving the reservation of the resource closer to the actual usage. This enables dynamic resource sharing among vertical services and thus increasing the resource efficiency and boosting the revenue. The algorithm constantly receives the monitoring report through the monitoring platform on the resource usage for each NFV-NS and leverages the ability of 5G-TRANSFORMER Service Orchestrator (5GT-SO) to reallocate the resources of its deployed network services during configurable time windows. Intuitively, we allocate less resource to network services during the periods that exhibit a low activity allocating those spare resources to the network services that are having high activity peaks. In fact, the algorithm assumes that verticals have an agreement with the 5GT service provider on the SLA and its price (service reward). In detail, the assumption is that the price is fixed during the total service time of the deployed service. However, resource underprovising has a risk of SLA violation. Thus, the algorithm balances the risk of service disruption reimbursing the verticals back an amount (service penalty) proportional to the service disruption.

We formulate the placement algorithm as an optimization problem running periodically on a slotted-time system. That is, we run our placement algorithm at the beginning of each slot and we only reconfigure our system only at the beginning of the next slot (e.g. each slot might have a duration of an hour). We design the following objective function:

$$\min_{x\in\{0,1\}^S,\, z\in\mathbb{R}_+^S} \Psi := \sum_{\tau\in\mathcal{T}} \sum_{\substack{p\in\mathcal{P}_{b,c} \\ \forall b\in\mathcal{B}, c\in C}} \overbrace{K_\tau \rho(z_{\tau,p}, \hat{\sigma}_{\tau,p}, L_\tau) x_{\tau,p}}^{\text{Estimated penalty}} - \overbrace{R_\tau x_{\tau,p}}^{\text{Reward}}$$

The objective function tries to minimize and recomput the overall cost of the system subtracting the potential penalties (K_τ) and the reward (R_τ) among all the network service requests ($\mathcal{T}$), paths ($\mathcal{P}$), base stations ($\mathcal{B}$) and cloud units (C). The objective function is composed of two kinds of optimization variables. The set $\boldsymbol{x}$ contains the binary variables managing the admission control while $\boldsymbol{z}$ contains a set of continuous variables managing the resource allocation. Furthermore, ρ function estimates the potential service disruption weighting a forecast of the resource deficit with the risk of a wrong prediction. Finally, the optimization function is subject to two sets of constrains. On the one hand, the system is subject to a decoupled set of constrains such as total resource capacity and maximum delay. On the other hand, the system is subject to a couple set of constrains which indicated that the resources granted for an accepted NFV-NS should be bigger than the forecasted utilization and less than the vertical SLA. We solve this optimization problem using Bender's decomposition and a heuristic algorithm.

6.6.1.4.2 Performance evaluation

We evaluated our approach against a no-overbooking scenario considering three types of network services: eMBB (enhanced Mobile Broad Band), mMTC (massive Machine type communications) and uRLLC (Low-latency communications). We simulate how these service behave when running our placement algorithm. In Figure 18 we observe how overbooking leads to a better resource utilization and boosts the revenue since the system has more room to accept NFV-NSs. We also observe how accurate predictions are key to take more risks when under-provisioning a network service accepted into our system.

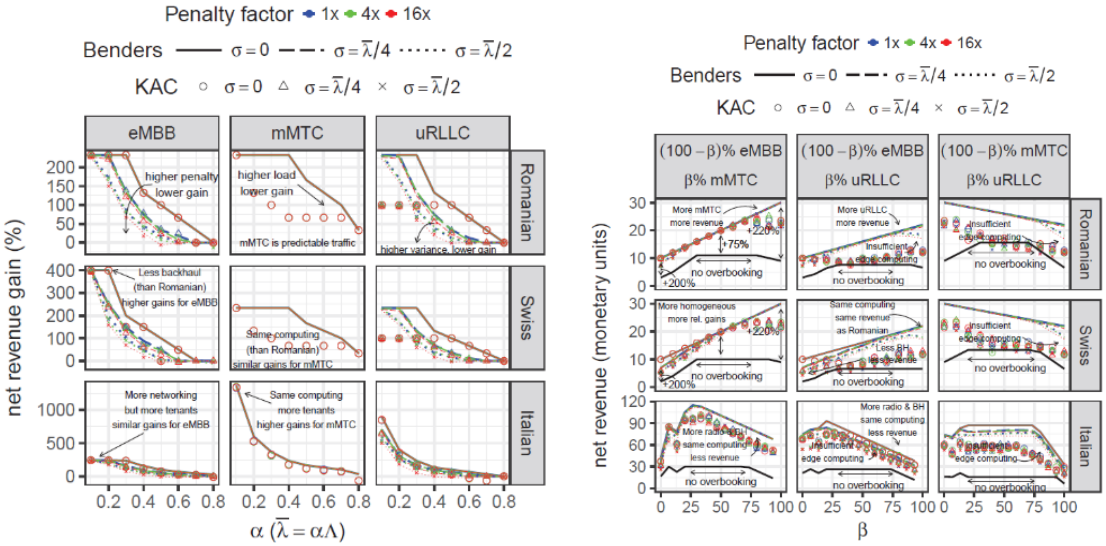


FIGURE 18: PERFORMANCE OF ALGORITHM C

6.6.1.5 Placement Algorithm D

6.6.1.5.1 Description

The high-level goal of Algorithm D, based on a cluster-based approach, is to find the best tradeoff between the cost for the operator, as expressed by the κ-parameters, and the service latency. The algorithm follows a two-step approach, dealing with delay constraints and cost separately, in two different stages:

1. We divide both the VNFFG and the host graph into *clusters* in such a way to guarantee low network delays;
2. we assign VNFs in each VNFFG cluster to hosts in the corresponding host cluster so as to ensure low costs $\kappa(h, v, op)$.

Clustering stage. The intuition behind this stage is that, in order to meet service delay constraints, we must keep network delays low, and this in turn means having as little traffic as possible flowing on high-delay links between hosts. Therefore, we *cluster* both the VNFFG and the host graph in the same number of clusters, ensuring that: (i) in the VNF graph, high-traffic edges connect VNFs of the same cluster and low-traffic edges connect VNFs of different clusters; (ii) in the host graph, low-delay links connect hosts of the same cluster and high-delay links connect hosts of the same cluster and high-delay links connect hosts of different clusters.

We adopt an iterative, hierarchical clustering technique [10]: at the first iteration, each node starts in its own cluster (*singleton*). At subsequent iterations, the two clusters connected by the highest-traffic edge in the VNFFG, and the two connected by the lowest-delay edge in the host graph, are joined.

Assignment stage. In this stage, we have to decide at *which host* each VNF shall run. Due to the previous clustering stage, network delays can be ignored, while processing delays d only depends on the VNF and not on the host at which it runs. Therefore, we can assign VNFs to hosts with the sole purpose of minimizing costs; specifically, we start from the VNF with the largest delay and place it at the cheapest host with enough spare resources to run it.

In many situations, multiple hosts will be associated with the same cost. In these cases, we break ties by trying to *balance* the load across different hosts. Formally, we choose the VNF to place and the host at which it should be placed so that the following quantity is minimized:

$$\max_{h \in H} \max_{\rho \in R} \frac{\sum_{v \in V} x(h, v) r(\rho, v)}{C(h, \rho)}.$$

In the expression above, the fraction represents how close to exhaustion resource ρ is at host h. We seek to minimize the maximum of such ratios among all resources and hosts, thus reducing the risk to have, e.g., hosts with plenty of spare CPU but no free memory.

Importantly, each step of our approach has linear time complexity in the number of VNFs, hosts, and links; therefore, the global complexity is linear as well. This guarantees very quick decisions, enabling the 5GT-SO to quickly react to new requests and changed conditions.

Furthermore, the approach can be easily extended to *multi-domain scenarios* where federation can be exploited in case of lack of resources in the domain controlled by the 5GT-SO that is in charge of deploying the service requested by the 5GT-VS. In particular, the algorithm can be extended as follows:

- in the clustering stage, edges connecting hosts belonging to different domains should be assigned higher weight, so as to limit the amount of traffic flowing across different domains;
- in the assignment stage, hosts belonging to foreign domains should be assigned higher costs, to model the fact that resources from foreign domains ought to be used only when necessary.

6.6.1.5.2 Performance evaluation

We compare our algorithm against Placement Algorithm A (GA) in its cost- and delay-minimizing configurations and for the same topology and service scenario as in the experiments of Section 6.6.1.2.2. Figure 19 shows the cost and delay associated with our algorithm; each yellow dot therein corresponds to a specific number of clusters, varying from 1 to 7. It is easy to see that changing the number of clusters leads to different cost/delay trade-offs.

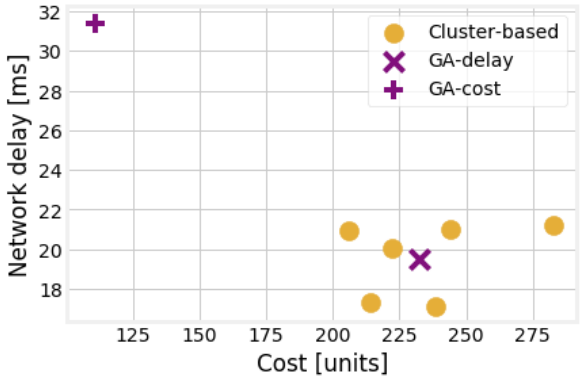


FIGURE 19: PERFORMANCE OF ALGORITHM D AND OF ALGORITHM A

The two purple markers correspond to the results of the GA-based benchmark under the two objectives it supports: when it is set to minimize delay, the resulting configuration is similar to the one generated by the cluster-based approach. Setting the GA algorithm to minimize costs results in significantly lower costs than the cluster-based approach, but in a much higher delay.

Finally, it is worth mentioning that the execution time is below one second for the cluster-based approach, against one minute for the GA benchmark.

6.6.2 Service Scaling

6.6.2.1 Motivation

5GT Vertical Services are designed to be deployed and used in highly dynamic environments. Here the volume of service requests and the mobile traffic might change drastically and unexpectedly. Moreover, users might connect and disconnect from the service at a very high rate.

Provisioning a service with the size required to serve the highest anticipated traffic might be overly expensive, for example, the traffic might be as much as 10x higher during the day than at night. Moreover, such strategy might still lead to interruptions of the service or degradation of the QoS during unforeseen peaks.

To mitigate these issues, the 5GT-SO offers the capability of scaling the service at runtime, without service interruption, in order to be able to accommodate a very high

number of peak users without wasting resources in periods where the service demand is lower.

In 5G-TRANSFORMER three different types of service scaling have been defined, Vertical-driven VSI Scaling, Automated VSI Scaling and Automated NFV-NS Scaling, as summarized in the following sections.

6.6.2.2 Vertical-driven VSI Scaling

The Vertical-driven VSI Scaling occurs whenever the scaling of a vertical service instance is triggered by the vertical, either manually through the GUI of the 5GT-VS or through the 5GT-VS REST API. The REST API is automatically triggered by some other tool under the vertical's control (e.g. embedded in its own applications) and external to the 5GT platform. In both cases, a message is sent to the 5GT-VS, which, after checking that the request is acceptable, identifies the NFV-NS instance(s) to be scaled and invokes the 5GT-SO by sending a ScaleNsRequest (according to IFA 013, 7.3.4.2 [5]) to its NBI.

The 5GT-SO NBI dispatches the request to the SOE, which identifies the changes that should be applied to the NFV-NS instance. Then, for each resource identified, either to be created, to be removed or to be updated, the SOE contacts the ROOE with the resource request, which in turn manages the low-level requests towards the MANO and WIM components.

6.6.2.3 Automated VSI Scaling

The Automated VSI Scaling occurs whenever the 5GT-VS, usually its Arbitrator component, encounters the need to resize a vertical service or a network slice. This leads the 5GT-VS to trigger a scaling action on the corresponding NFV-NS instance. The most likely scenario is if the vertical customer has exhausted its resources quota, but it needs to instantiate a high priority service. In that case the 5GT-VS scales down another – lower priority – service in order to free the resources required to instantiate the high priority service.

From the point of view of the 5GT-SO, this case is equal to the Vertical-driven scaling, as in both cases the 5GT-VS will send a ScaleNsRequest to the 5GT-SO, and the 5GT-SO will react in exactly the same way.

6.6.2.4 Automated NFV-NS Scaling

The Automated NFV-NS scaling is driven by the auto-scaling rules expressed in the NFV-NSD and it is managed entirely at the 5GT-SO level. Each auto-scaling rule contains a condition, based on the service's monitored data, and a reaction, i.e. a scaling action. Whenever the condition is verified, the scaling reaction is performed by the 5GT-SO.

The whole procedure is coordinated by the SLA Manager (see section 6.4.10). During the NFV-NS instantiation phase, the SLA Manager subscribes to the Monitoring Platform according to the "conditions" encoded in the auto-scaling rules of the NSD, in order to receive the required alerts during the service runtime. Whenever the SLA Manager is notified that one of the conditions is verified, it will send to the 5GT-SO a ScaleNsRequest based on the "reaction" specified for that condition in the auto-scaling rule. From then on, everything proceeds as in the other cases.

6.6.3 Service Composition

Service composition is the process of joining already instantiated NFV-NS instances with new NFV-NS instances that need to be instantiated. The service composition as a

joining process can be executed as concatenation of NFV-NS or composition of NFV-NS.

The concatenation of NFV-NS consists of instantiating a new NFV-NS instance and establishing connections with already existing (up-and-running) NFV-NSs. Each of the concatenated NFV-NS can operate independently but being concatenated can provide different features and/or new E2E services. In case one of the NFV-NS instances is terminated, the rest NFV-NS instances would remain active and provide their service features.

The composition of NFV-NS consists of creating a new composite NFV-NS instance that would contain a set of nested NFV-NSs. The set of NFV-NSs can be combination of already instantiated and active NFV-NS instances and new NFV-NS instances. The composite NFV-NS is orchestrated as a single instance. Applying modification to the composite NFV-NS instance affects each of the nested NFV-NS. The nested NFV-NS are not independent and the connections between each nested NFV-NS are not dynamically changed.

6.6.4 Service Assurance and SLA Management

Service assurance is the application of policies and processes with a goal to ensure that services offered over networks and computational resources meet a pre-defined service quality level for an optimal subscriber experience.

Each NFV-NS is instantiated with a certain configuration, which contains scaling, monitoring and alerting rules. These configurations are controlled by the Configuration, Monitoring and SLA Managers to meet certain SLA for the service.

The SLA Management capabilities are covered by SLA Manager features which are described in section 6.4.10.

6.6.5 Service Federation

Service federation is the overall process of establishing, consuming or providing NFV network services (NFV-NS) by an administrative domain from/to other (peering/partner) administrative domain. The administrative domain that requests service is referred to as consumer domain while the peering administrative domain capable of providing service is a provider domain. The service federation is the combination of establishing business/service level agreements among the administrative domains and consuming/providing federated nested NFV-NSs that are part of composite End-to-end NFV-NS. The details about the business/service agreements are out of scope of 5G-TRANSFORMER. Section 6.6.5.1 focuses on the definition and procedure of service federation. The Section 6.6.5.2 focuses on the ETSI GS NFV-IFA 028 [28] solution for multi-domain orchestration and decomposition of composite NFV-NSs. The last section, Section 6.6.5.3 briefly explains the exchange of information regarding available services for federation among different domains.

6.6.5.1 Federation Assumptions: Service delegation & Service federation

The service federation, as a concept of deploying a nested NFV-NS of a composite NFV-NS in an external domain or of providing a nested NFV-NS to an external domain, is a broad concept. It appears in two forms: service delegation & service federation. Both procedures are orchestrated by the Composite NSO module of the 5GT-SO.

Service delegation is the process of delegating the request for a service instantiation to an external (federated) domain. Generally, when a 5GT-VS issues a request for instantiation of a single NFV-NS (not composite), the request is received by a 5GT-SO

(Composite NSO) on the NBI. The Composite NSO analyses the request and decides to delegate to an external/peering domain. The Composite NSO simply redirects the request on the E/WBI (So-So-LCM). The peering 5GT-SO (Composite NSO) accepts the request and instantiates the single NFV-NS. Note that federation of composite NFV-NS or further decomposition by the provider domain Composite NSO is not foreseen. More information regarding decomposition of NFV-NSs can be found in Section 6.6.5.2. Upon instantiation of the single NFV-NS, it sends back all the information to the consumer 5GT-SO (Composite NSO). The consumer Composite NSO redirects the response back to the 5GT-VS on the NBI. The local/consumer 5GT-SO role is to act as a proxy between the 5GT-VS and the federated/provider 5GT-SO. The 5GT-VS is not aware of the setup, hence service delegation is transparent for the 5GT-VS. The life-cycle management is handled by the federated/provider 5GT-SO. The monitoring information is delivered to the 5GT-VS using the local/consumer 5GT-SO as proxy as well.

Federation of services in 5GT-SO follows a similar approach as in ETSI GS NFV-IFA 028 [28]. Federation of services is used for composite NFV-NSs or single NFV-NSs that are decomposed by the NS decomposition algorithms. The composite NFV-NS has defined a set of nested NFV-NSs that by instantiating each of them and stitching them together form the composite End-to-end NFV-NS. On the other hand, the NS decomposition algorithm runs at instantiation time and generates a set of nested NFV-NS out of a single NFV-NS. In both cases, the Composite NSO decides for each of the nested NFV-NS whether to deploy it locally or in a peering/federated domain. The process of instantiating part of a composite NFV-NS (one or more nested NFV-NSs) in a federated domain and another part in a local domain in order to form the end-to-end NFV-NS is defined as service federation. The main difference with respect to service delegation is that the consumer Composite NSO has an active role in orchestrating the composite End-to-end NFV-NS.

The 5GT-SO receives a request for instantiation of a composite NFV-NS/ single NFV-NS from 5GT-VS on the NBI. In the case of single NFV-NS, the Composite NSO first makes decomposition of the requested NFV-NS into several nested NFV-NSs using the NS decomposition algorithm module. More on the decomposition are described in Section 6.6.5.2. Afterwards, the Composite NSO decides where to instantiate each of the nested NFV-NS, in the local domain or in a federated domain using service federation. The reasons can be diverse, such as: lack of resources provided by the local 5GT-MTP, different target location that is not supported by the administrative footprint, extension of services, unsupported NFV-NSs, lack of on-boarded NSDs for the nested NFV-NS, etc.

Upon the decision for service federation, the Composite NSO queries a peering administrative domain for availability of providing the requested nested NFV-NS. (Alternatively, the set of offered services could be pre-negotiated offline among administrative domains.) Note that the information of availability is previously stored in the local Catalogue Database (DB) and the network connections to all peering 5GT-SOs are already established. Optionally, for the dynamic decomposition case, the consumer Composite NSO can provide an NSD to the peering (provider) 5GT-SO for the requested NFV-NS for federation (e.g., for more specific deployment flavor). This optional case is for further study. The provider Composite NSO checks & confirms the availability and starts with instantiation of the federated NFV-NS. The instantiation procedure in the provider domain is similar to instantiating local nested NFV-NS (through the Constituent NSO, RO-OE, RO-EE, etc.). However, the process is marked as “federated NFV-NS” at starting time. Once the provider domain instantiates the

federated nested NFV-NS, the provider Composite NSO confirms the instantiation to the consumer Composite NSO. The Composite NSO checks instantiation of both local nested NFV-NSs and federated NFV-NSs before proceeding with the stitching procedure. In the stitching procedure the Composite NSO instructs the Composite RO to exchange information and setup interconnections with the provider domain Composite RO on the E/WBI (So-So-RM). Once all the interconnections are set locally and with the provider domain, the composite End-to-end NFV-NS is up and running.

The implementation workflow of establishing service federation is covered in Section 6.7.7.

As in ETSI GS NFV-IFA 028 [28], the consumer Composite NSO is not aware of the resources and VNFs that the provider Composite NSO is using to provide the federated NFV-NS. The consumer 5GT-SO NFVO-NSO is using the So-So-LCM to send requests for lifecycle operations of the provided NFV network service. The lifecycle operations are performed by the provider 5GT-SO NFVO-NSO. The So-So-MON allows exchange of limited information consisting of performance indicators and fault alarms. According to the already negotiated terms and conditions, some performance indicators may be hidden from the consumer domain. The 5GT-SO NFVO-NSO is using the So-So-CAT reference point to exchange catalogue updates, querying for NSDs or on-boarding MEC AppDs.

6.6.5.2 NFV-NS decomposition and NFVO Multi-domain orchestration

The use case of offering NFV-NS to other administrative domains is covered in ETSI GS NFV-IFA 028 [28]. In the reference document, it is described how composite NFV-NS can span onto separate administrative domains. NFVO Multi-domain orchestration of NFV-NSs is done upon decomposition of composite NFV-NS received from the 5GT-VS by the 5GT-SO NFVO or by handling those composite NFV-NSs that come from the 5GT-VS. The result of decomposing a composite NFV-NS is a set of multiple nested NFV-NS. The decomposition of the requested NFV-NS can be static or dynamic. Static decomposition is prearranged when the composite NSD is on-boarded. On the other hand, dynamic decomposition generates the set of nested NFV-NSs during instantiation time as a result of applying decomposition algorithms in the NS decomposition algorithm module. These algorithms may optimize the decomposition process according to the current overview of available resources at instantiation time or based on other metrics. Upon decomposition, each nested NFV-NS can be instantiated on a different administrative domain as a federated nested NFV-NS and by chaining it with the constituent nested NFV-NSs, together, they form the requested composite NFV-NS. According to the ETSI GS NFV-IFA 028 architectural approach [28] the NFVO Multi-domain orchestration of NFV-NSs can be established using the Or-Or interface between the NFVO belonging to a consumer domain and the NFVO belonging to a provider domain. The consumer NFVO is requesting nested NFV-NS to be federated at the provider NFVO. Once the federated NFV-NS is instantiated, the consumer NFVO can continue requesting Lifecycle Management (LCM) operations to the provider NFVO. However, the provider domain is responsible for all NFVO functionalities and LCM operations over the established federated nested NFV-NS. The consumer NFVO is aware neither of the resources that are part of the provider NFVO nor of the exact placement of the VNFs and corresponding interactions with its underlying MTP for their eventual deployment.

6.6.5.3 Catalogue of services

The catalogue of services can be formed dynamically or statically. In the dynamic approach, each administrative domain (5GT-SO) locally creates a list of NFV-NSs that

can provide as federated NFV-NS to other peering administrative domains based on the capabilities and previously on-boarded NSDs. Each list is broadcasted to peering administrative domains and stored in their Catalogue DB. Note that we assume that the network of peering administrative domains is already established. Optionally, the catalogue list can be dynamically extended due to on-boarding of a new NSDs or by a request from peering 5GT-SOs (other administrative domains). The dynamic approach is intended for further study.

The static approach is simple and time endurable. Each administrative domain implementing the 5G TRANSFORMER system estimates its capabilities and generates a list of possible services that it can offer as provider of NFV-NSaaS (service federation) to other partnering administrative domains. This list of NFV-NSs is later negotiated (e.g., on business agreements) with each peering domain. Depending on the federation level and business/service level agreements, it is reduced or extended. The final and agreed modifications of the list of services represent the catalogue of (offered) services that are delivered to the specific partnering administrative domain. The partnering administrative domain receives the catalogue of services and stores it in the NFV-NS/VNF Catalogue Database of the 5GT-SO. All available services that can be consumed from an external partnering administrative domains are stored in this database.

6.6.6 Resource Federation

Resource federation is the overall process of consuming or providing NFVI resources from or to external federated domains. The resource federation process is divided between establishing business/service level agreements and the allocation process of providing/consuming NFVI resources. In this section we are focusing on the allocation process. Throughout the document, the resource federation is also referred to as NFVI as a Service (NFVIaaS). Based on the agreed terms and conditions, a share of the resource capacity is available to be federated in external domains.

Similar to service federation, resource federation could be exploited when there is lack of resources in the local 5GT-MTP domain, due to different location constraints, extended footprint, etc. However, in the case of resource federation, the orchestration of federated resources is done by the consumer domain. NFV-NSs, VNFs, VAs or VMs can be deployed over the federated resources by the consumer domain, and the deployed components are transparent to the provider domain. Since resource federation requires a high level of trust, the administrative domains should set-up precise agreements with various trust/business levels. Based on these levels, the administrative domains define the amount of resources (capacity) and quality of information related to the resources that they share with external domains. For this reason, resource abstractions are employed. The RO-EE and the SO-SO Resource Management & Advertisement modules carry out the resource federation procedures.

The allocation of NFVI resources in resource federation includes two phases: 1) advertising phase and 2) allocation and management of federated resources. In the next sections, the motivation for using federation of resources is covered. The advertisement phase is covered in Section 6.6.6.1 and the allocation and management of federated resources is covered in Section 6.6.6.2.

6.6.6.1 Advertisement phase

From a resource federation perspective, the business/service level agreements define only the relationship between two administrative domains. Only the general terms and conditions for sharing NFVI resources are defined. A finite list of available resources for federation that is similar to the catalogue of services, does not exist. The 5GT-SOs need to perform an advertisement phase where they exchange information about the NFVI resources that they can offer to other potential consumer 5GT-SOs. In this phase, all 5GT-SOs have an equal role without being consumer or provider. The aim of the advertisement procedure is to allow each 5GT-SO to broadcast its available resources for federation to the external domains and, at the same time, to discover or accumulate all the potentially available resources for federation owned by the external domains. The advertisement phase consists of two local operations and an outbound/inbound exchange of information:

- Calculating available resources – local operation performed by 5GT-MTP and stored into the NFVI Resource Repository. The available resources are stored and tagged as “local resources” in the NFVI Resource Repository database.
- Generating resource abstractions – local operation performed by SO-SO Resource Management & Advertisement module. According to the federation levels, specified service level agreements, etc., first the SO-SO Resource Management & Advertisment extracts the available resources intended for federation. Then, from the extracted set of resources, it generates the resource abstractions and stores them in the NFVI Resource Repository.
- Exchanging resource abstractions with peering 5GT-SOs – performed by SO-SO Resource Abstractions via the So-So-RAM reference point of the EBI/WBI. The exchange of information occurs in both directions. The generated resource abstractions, from the “Generation resource abstractions” operation, are broadcasted in outbound direction to the peering 5GT-SOs. In inbound direction, the received abstractions from the peering 5GT-SOs are accumulated and stored in the local cache or in the NFVI Resource Repository.

The advertising phase occurs periodically or event-based. Upon NFV-NS instantiation, modification (scaling up), performed by the 5GT-SO, that directly changes the state of the available NFVI resources provided by the local 5GT-MTP, the advertisement phase should be repeated in order the peering domains to get the updated information.

6.6.6.2 Allocation & management phase

The allocation of federated resources or the trigger for resource federation begins when the RO-OE decides for VNF placement over a set of resources that belong to an external domain. The RO-OE by default makes transparent decision over the Aggregated view from both local and federated domains, without distinguishing among both. Once the VNF placement (performed by the PA) is done, the result is delivered to the RO-EE. The RO-EE, by querying the NFVI Resource Repository, is aware of the location of each resources contained in the VNF placement decision. For allocating the external resources from federated domain, the RO-EE issues a request towards the SO-SO Resource Management and Advertisement module, which redirects the request to the specific domain or peering/provider SO-SO Resource Management and Advertisement module on the E/WBI (So-So-RM). The provider SO-SO Resource Management and Advertisement module translates the request (e.g., the request

contains certain level of resource abstraction), which is de-abstracted and translated into a request for local resources. The translated request is sent to the provider RO-EE module which generates allocation requests for the resources at the provider 5GT-MTP and prepares the end-points to grant access to the consumer domain. In the final stages, resources are allocated and the provider RO-EE via the provider SO-SO Resource Management and Advertisement module exchange information regarding connectivity and usage of the resources within the consumer domain. At the final stage of the allocation phase, the consumer RO-EE is in charge of inclusion, orchestration and deployment of VNFs over the federated resources in the provider domain, meaning that it operates over the federated resources as over local ones.

During the management phase, the RO-EE may trigger requests for deployment of VMs or other resource-related lifecycle management operations towards the federated resources on the So-So-RM of the E/WBI via the SO-SO Resource Management and Advertisement module (used as proxy). On the provider domain side, the provider SO-SO Resource Management and Advertisement and the provider RO-EE module are in charge of delivering the requests to the provider 5GT-MTP and deliver back responses. The limitations of the lifecycle operations directly depend of the federation level established with the provider domain. The So-So-RMM reference point is used for receiving monitoring information of the federated resources at the SLA Manager. The monitoring information (performance parameters and fault alarms) is limited as well and directly depending on the federation level.

6.7 5GT-SO Workflows

In this section we detail the main workflows that are implemented in the 5GT-SO, namely: (1) Service On-boarding, (2) Service Instantiation, (3) Service instantiation include MEC applications, (4) Service Termination, (5) Service Monitoring, (6) Service Scaling, and (7) Service Federation.

6.7.1 Service Onboarding

Description: The workflow describes the on-boarding of VNFs (initiated by the 5GT-SO admin), VAs (initiated by the vertical) and network services (initiated by the 5GT-VS). A similar procedure applies regarding on-boarding MEC applications.[7]

Prerequisites: None.

Assumptions: The vertical service can be deployed in one NFV-NS. Also, the service is deployed in one new NFV-NSI. We assume the mapping between vertical service and the NSD has been already performed. Note that the on-boarding of a MEC application, which is packaged as a Virtual Application (VA), will be initiated by the verticals.

Workflow:

This workflow includes three parts that are initiated at separate times by different actors, as set forth below.

[7] MEC applications are described using the AppD as specified by ETSI MEC. The AppD has similar information as the VNFD, but with specific fields reflecting the MEC applications requirements (e.g., traffic redirection, latency requirement, required MEC service, etc.). In 5G-TRANSFORMER we assume that the NSD is extended to integrate AppD(s).

The first part of the workflow describes the on-boarding of VNFs:

- The process is triggered by the 5GT-SO administrator or operator, requesting the on-board to the 5GT-SO NFVO-NSO (1) and providing the VNF package – including the VNFD as well as additional configuration files – as an input;
- The 5GT-SO NFVO-NSO requests the VNF package (2) from the software provider[8], which replies (3) by sending the VNF package;
- The 5GT-NFVO-NSO extracts the VNFD from the VNF package (4) and stores it in its VNFD/AppD catalogue (5-7);
- The 5GT-NFVO-NSO can confirm the successful on-boarding to the 5GT-SO administrator[9] (8).

The second part of the workflow describes the on-boarding of VAs:

- The vertical sends to the 5GT-VS an on-board request (9), including the VA application package path (see [12] for more details);
- The 5GT-VS sends to the 5GT-SO NFVO-NSO a request to on-board the application package, including the package path (10). The 5GT-SO NFVO-NSO then issues a request for the application package itself (using the path) to the 5GT-VS (11). The 5GT-VS replies (12) by sending the application package, including the corresponding VNFD or AppD as well as additional configuration files;
- The 5GT-SO NFVO-NSO extracts the VNFD/AppD from the VA application package (13), and then stores the VNFD/AppD into the 5GT-SO VNFD/AppD catalogue (14-16);
- If the VA package contains an AppD, the 5GT-SO NFVO-NSO on-boards the VA (17) via the SBI to the 5GT-MTP, including in the request the corresponding AppD. The on-boarding is confirmed by the 5GT-MTP (18).
- The 5GT-SO NFVO-NSO confirms (19) the successful uploading to the 5GT-VS. Then the 5GT-VS sends a response (20) to the original on-board request from the vertical.

The third part of the workflow describes the on-boarding of network services:

- The 5GT-VS sends an on-boarding request to the 5GT-SO NFVO-NSO (21), including the NSD info;
- The 5GT-SO NFVO-NSO checks with the 5GT-SO VNFD/AppD catalogue that all the VNFs referenced in the NSD are already correctly stored (22-24);
- The 5GT-SO NFVO-NSO stores the NSD in the 5GT-SO NSD catalogue (25-27);
- The 5GT-SO NFVO-NSO replies with a success message (28) to the original request from the 5GT-VS.

[8] A software provider is an entity that assists 5G-TRANSFORMER's service provider with VNFs.

[9] The 5GT-SO administrator is the entity in charge of the management of 5G-TRANSFORMER service orchestrator.

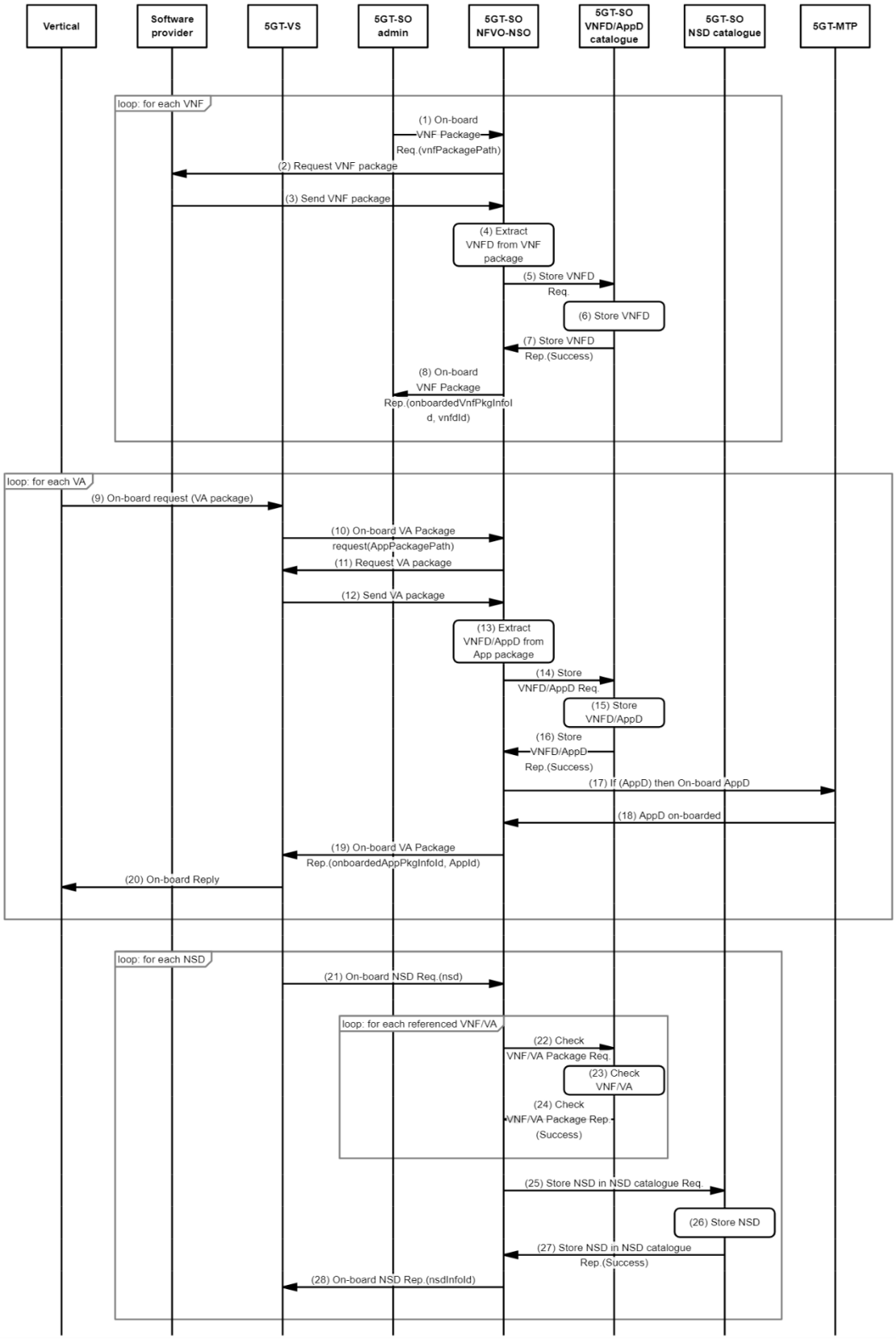


FIGURE 20: SERVICE ONBOARDING WORKFLOW

6.7.2 Service instantiation

Description: The workflow describes the instantiation of a network service triggered by the vertical on instantiation of a vertical service.

Prerequisites: The vertical has prepared a vertical service description and the 5GT-VS sends a service instantiation requests to the SO.

Assumptions: The vertical service is a simple one, meaning it can be deployed on one NFV network service instance. We also assume that this service is deployed in a new NFV network service instance. We assume that (1) VSD has been on-boarded into the 5GT-VS; (2) and the NSD including its associated VNFDs/AppDs, to which the vertical service is mapped to and which describes the network service, has been previously on-boarded to the 5GT-SO.

Workflow:

1. The 5GT-VS requests the SOE-parent of the 5GT-SO, through its NBI, to create a new NFV-NS identifier, specifying its NSD id.
2. The SOE-parent checks in the DB if the NSD of the requested NFV-NS is on-boarded already.
3. The DB module confirms that NSD_id exist in the database.
4. The SOE-parent generates & stores “ns instance ID” in the DB.
5. The DB confirms the stored operation, and the confirmation is redirected back to the 5GT-VS with the “ns instance ID” included in the response confirmation.
6. The 5GT-VS sends request for instantiation of the NFV-NS, specifying the “ns instance id”, “flavor Id”, “deploymentLevel”, etc.
7. The SOE-parent initiates the operation. Generates & stores operationID in the DB and responds to the 5GT-VS with the operationID included in the message.
8. The SOE-parent extracts the descriptors from the DB module: NSD and VNFD.
9. The DB module returns all the descriptors in JSON format.
10. The SOE-parent processes the NSD to check if it is a composite or regular descriptor. In case of composite descriptor, it needs to iterate and extract all the nested NSDs contained in the descriptor. In this workflow we are focusing in the case of regular/single descriptor.
11. The SOE-parent sends instantiation request to the SOE-child.
12. The SOE-child receives the request and issues new request to the ROOE including the JSON descriptors, deployment flavors, deployment levels, etc.
13. The ROOE extract all the information from the descriptors.
14. At the same time, issues request for available aggregated resources to the 5GT-MTP on the SBI interface.
15. The 5GT-MTP responds with information of the available aggregated resources on the underlying infrastructure.
16. The ROOE stores the information in the DB
17. The DB module confirms the updates on the aggregated resources
18. The ROOE sends the request for computation of VNF placement to the PA module, including the extracted information from the NSD/VNF descriptors plus the newly updated information of the aggregated resources.
19. The PA module computes the placement using one of the placement algorithms. The result is returned to the ROOE.

20. Meanwhile, the 5GT-VS polls the status of the instantiation operation, using the operation ID.
21. The SOE-parent responds with the status (“processing”).
22. Based on the PA output, the ROOE sends a request to the OSM wrapper for allocation & instantiation of computing resources/VMs and establishing intra-PoP connectivity.
23. The OSM wrapper translates the incoming information from the request to the API procedures of the OSM and triggers the instantiation procedures in the OSM.
24. The OSM using internal flows, allocates & instantiates the computing resources and establishes the intra-connectivity (intra-PoP connections).
25. The OSM via the OSM wrapper sends back to the ROOE, the allocation & instantiation confirmation along with the IP addresses of the computing resources.
26. The ROOE sends multiple requests to the RO-EE-WIM module, for each Virtual Link (VL) that needs to be established between the instantiated VNFs (inter-PoP connectivity). Each of the requests is processed and redirected from the RO-EE-WIM to the 5GT-MTP via the SBI.
27. The 5GT-MTP sends VL instantiation confirmation back to the ROOE, via the SBI and the RO-EE-WIM. All information per VL is included in the confirmation.
28. The ROOE stores all the information related to computing and networking resources the instantiated NFV-NS in the DB.
29. The ROOE stores additional optional information regarding the deployment in the DB.
30. The ROOE sends NFV-NS instantiation confirmation to the SOE-child including the IP addresses to access the VNFs/VMs.
31. The SOE-child updates the DB with the operation “processing” → “instantiated NFV-NS”.
32. The SOE-child sends confirmation to the SOE-parent.
33. The 5GT-VS polls the SOE-parent for the operation (using the operationID).
34. The SOE-parent sends back the status “instantiated NFV-NS”.

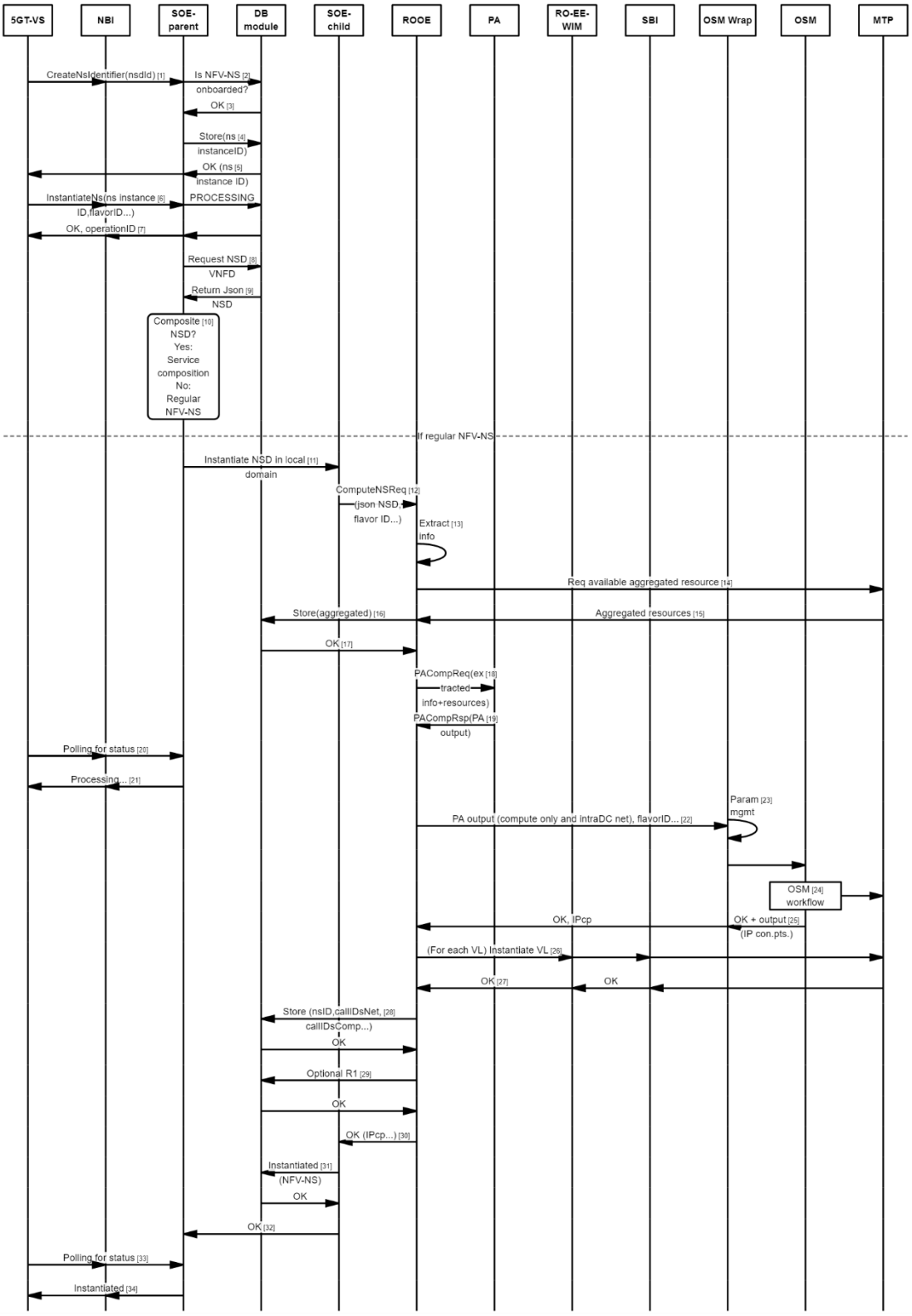


FIGURE 21: SERVICE INSTANTIATION WORKFLOW

6.7.3 Service instantiation including MEC applications

5G-TRANSFORMER considers three scenarios to deploy a MEC application: (i) a MEC application requiring traffic redirection and consuming MEC services; (ii) a MEC application requiring only MEC services; (iii) MEC application requiring only traffic redirection. For the first scenario, the MEC application requires that also an EPC (i.e., MME, HSS, SPGW-C), SPGW-U, eNodeB(s) and Mobile Edge Platform (MEP) are deployed. Specifically, the SPGW-U and the MEP should be run at the edge with the MEC application. For the second scenario, the MEC application needs eNodeB(s) and MEP. For the third scenario, the MEC application can be deployed without the need of other components.

Description: The workflow describes the instantiation of a NFV-NS instance including AppD (i.e. MEC applications). This workflow is based on the normal NFV-NS instantiation workflow as described in Section 6.7.2, but adding the additional steps and parameters for the MEC specific support and orchestrations, which are highlighted in blue in the text in the workflow in Figure 22.

Prerequisites: None.

Assumptions: The NSD includes AppD.

Workflow:

1. The 5GT-VS requests the SOE-parent of the 5GT-SO, through its NBI, to create a new NFV-NS identifier, specifying its NSD id.
2. The SOE-parent checks in the DB if the NSD of the requested NFV-NS is on-boarded already.
3. The DB module confirms that NSD_id exist in the database.
4. The SOE-parent generates & stores “ns instance ID” in the DB.
5. The DB confirms the stored operation, and the confirmation is redirected back to the 5GT-VS with the “ns instance ID” included in the response confirmation.
6. The 5GT-VS sends request for instantiation of the NFV-NS, specifying the “ns instance id”, “flavor Id”, “deploymentLevel”, as well as the location requirements for placing the AppDs, etc.
7. The SOE-parent initiates the operation. Generates & stores operationID in the DB and responds to the 5GT-VS with the operationID included in the message.
8. The SOE-parent extracts the descriptors from the DB module: NSD and VNFD and AppD.
9. The DB module returns all the descriptors in JSON format.
10. The SOE-parent processes the NSD to check if it is a composite or regular descriptor. In case of composite descriptor, it needs to iterate and extract all the nested NSDs contained in the descriptor. In this workflow we are focusing in the case of regular/single descriptor.
11. The SOE-parent sends instantiation request to the SOE-child.
12. The SOE-child receives the request and issues new request to the ROOE including the JSON descriptors, deployment flavors, deployment levels, etc.
13. The ROOE extract all the information from the descriptors, i.e., NSD and AppD.
14. At the same time, issues request for available aggregated resources including MEC and the availability of the MEC support in the 5GT-MTP on the SBI interface.

15. The 5GT-MTP responds with information of the available aggregated resources, and also a list of the MEC capable NFVI-PoPs with their associated locations in the underlying infrastructure.
16. The ROOE stores the information in the DB
17. The DB module confirms the updates on the aggregated resources
18. The ROOE sends the request for computation of VNF placement to the PA module, including the extracted information from the NSD/VNFD/AppD descriptors and the location constraint information plus the newly updated information of the aggregated resources including MEC part.
19. The PA module computes the placement using one of the placement algorithms to decide where to place the VNFs and the AppDs of the associated NFV-NS. The result is returned to the ROOE.
20. Meanwhile, the 5GT-VS polls the status of the instantiation operation, using the operation ID.
21. The SOE-parent responds with the status ("processing").
22. Based on the PA output, the ROOE sends a request to the OSM wrapper for allocation & instantiation of computing resources/VMs and establishing intra-PoP connectivity.
23. The OSM wrapper translates the incoming information from the request to the API procedures of the OSM and triggers the instantiation procedures in the OSM.
24. The OSM using internal flows, allocates & instantiates the computing resources and establishes the intra-connectivity (intra-PoP connections).
25. The OSM via the OSM wrapper sends back to the ROOE, the allocation & instantiation confirmation along with the IP addresses of the computing resources.
26. The ROOE sends a request to the 5GT-MTP to instantiate the AppD providing the AppD ID and its associated compute resources.
27. The ROOE sends multiple requests to the RO-EE-WIM module, for each Virtual Link (VL) that needs to be established between the instantiated VNFs (inter-PoP connectivity). Each of the requests is processed and redirected from the RO-EE-WIM to the 5GT-MTP via the SBI.
28. The 5GT-MTP sends VL instantiation confirmation back to the ROOE, via the SBI and the RO-EE-WIM. All information per VL is included in the confirmation.
29. The ROOE stores all the information related to computing and networking resources the instantiated NFV-NS in the DB.
30. The ROOE stores additional optional information regarding the deployment in the DB.
31. The ROOE sends NFV-NS instantiation confirmation to the SOE-child including the IP addresses to access the VNFs/VMs.
32. The SOE-child updates the DB with the operation "processing" → "instantiated NFV-NS".
33. The SOE-child sends confirmation to the SOE-parent.
34. The 5GT-VS polls the SOE-parent for the operation (using the operationID).
35. The SOE-parent sends back the status "instantiated NFV-NS".

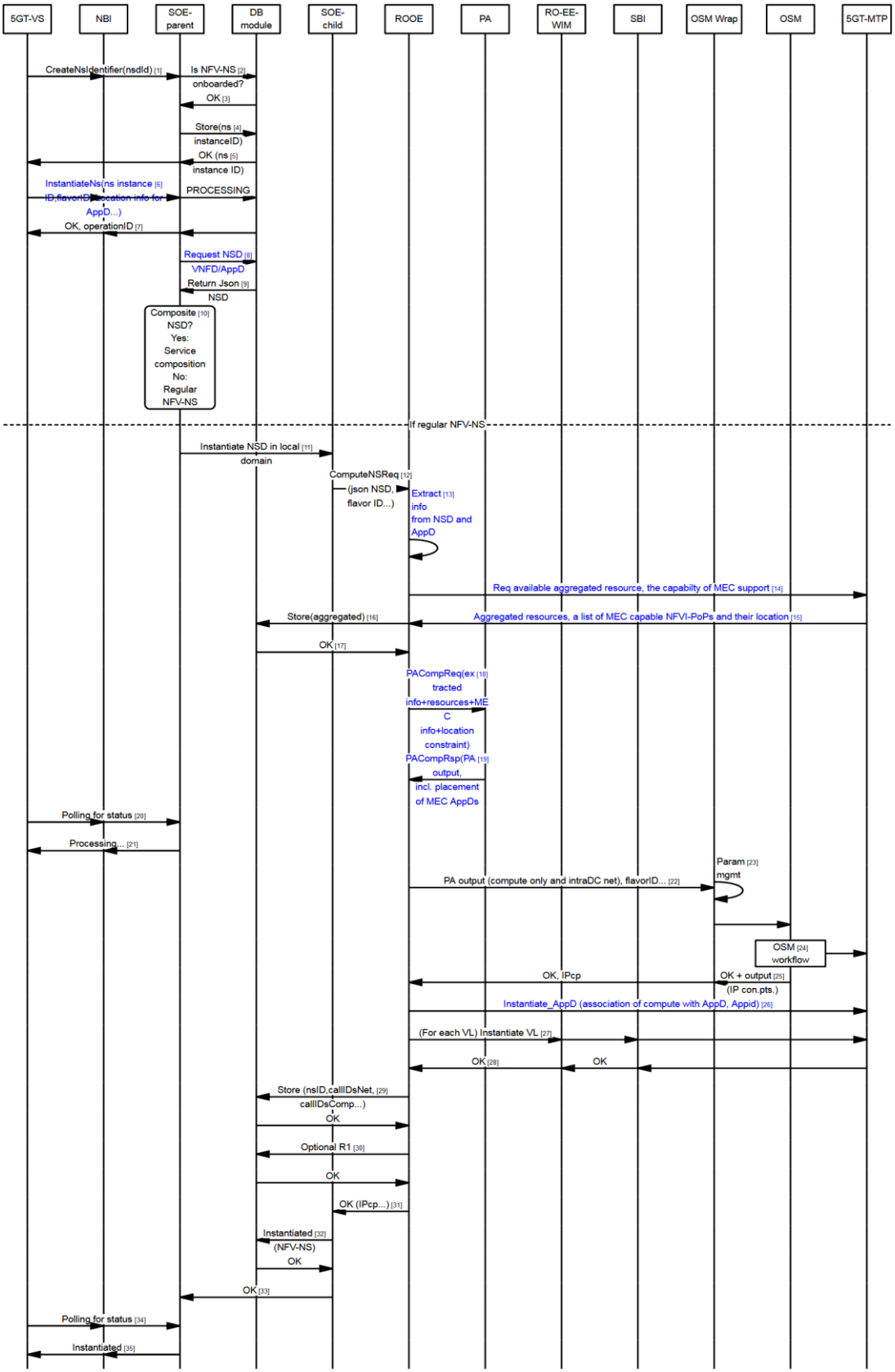


FIGURE 22: MEC SERVICE INSTANTIATION WORKFLOW

6.7.4 Service termination

Description: The workflow describes the termination of a networks service, triggered by the vertical on termination of a vertical service.

Prerequisites: The vertical service instance has been instantiated.

Assumptions: The vertical service is a simple one, meaning it can be deployed in one network slice. We also assume, that this service is deployed in its own network slice, the service does not share the slice with another slice.

Workflow:

1. The 5GT-VS requests the SOE-parent of the 5GT-SO, through its NBI, to terminate a NFV-NS identifier, using the “ns instance ID”.
2. The SOE-parent generates new termination “operation ID” and stores into the DB.
3. The SOE-parent confirms the start of the termination procedure to the 5GT-VS including the “operation ID” in the confirmation.
4. The SOE-parent sends request for termination to the SOE-child using the “ns instance id”.
5. The SOE-child redirects the request to the ROOE.
6. Meanwhile, the 5GT-VS periodically polls the SOE-parent for the status of the termination operation using the “operation ID”
7. The SOE-parent sends response o
8. The ROOE based on the “ns instance id”, extracts the information of the resources used for the NFV-NS instance.
9. The DB returns the information for all computing and networking resources used for the NFV-NS instance.
10. The ROOE sends termination request to the OSM wrapper for each of the allocated & instantiated computing resources and termination of the intra-PoP connectivity. The OSM wrapper translates the incoming request and adjusts to the OSM API.
11. The OSM runs internal workflow for termination of the computing and networking resources, interacting with the 5GT-MTP.
12. The OSM sends termination confirmation to the ROOE via the OSM wrapper, after all computing and intra-PoP connectivity are terminated.
13. The ROOE sends multiple requests to the RO-EE-WIM, one for each VL that needs to be terminated. The RO-EE-WIM redirects the request to the 5GT-MTP via the SBI.
14. The 5GT-MTP confirms the termination of the VLs via the SBI and the RO-EE-WIM.
15. The ROOE updates the DB by removing all the information regarding the allocated & instantiated computing and/or networking resources of the NFV-NS instance.
16. The ROOE sends termination confirmation to the SOE-child including the “ns instance ID”
17. The SOE-child redirects the confirmation to the SOE-parent.
18. The SOE-parent updates the status of the termination procedure “processing” → “terminated NFV-NS”

19. The DB confirms the update.
20. The 5GT-VS polls for the status of the operation to the SOE-parent, using the “operation ID”
21. The SOE-parent confirms the termination operation with the “terminated NFV-NS (operation ID)”

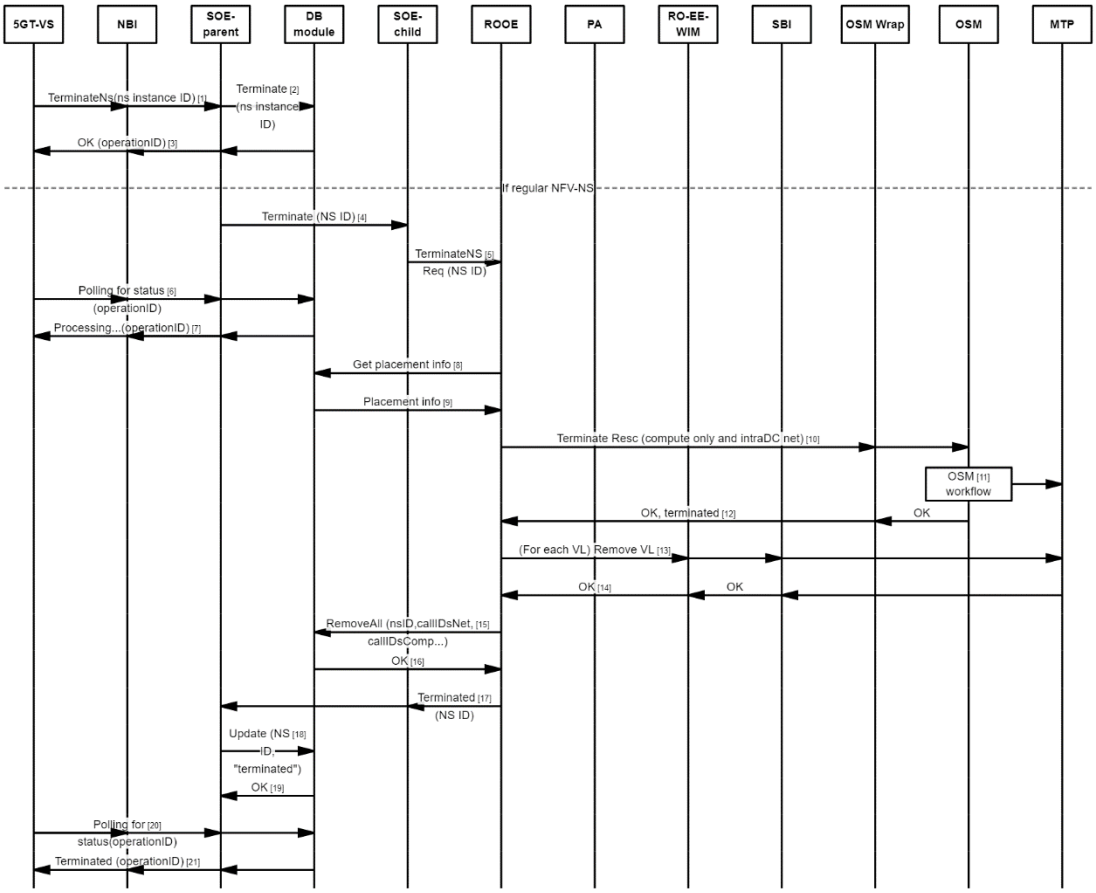


FIGURE 23: SERVICE TERMINATION WORKFLOW

6.7.5 Service monitoring

Description:

The workflow describes the procedures to activate the monitoring of an NFV-NS, based on the monitoring parameters defined in its NSD. Monitoring data are visualized on a dedicated, per-service web GUI implemented through the Grafana tool. The per-service GUI is returned to the 5GT-VS, that integrates the visualization of the monitoring parameters in its own GUI.

Prerequisites: The NSD includes a list of monitoring parameters. The 5GT-SO, the 5GT-VS and the Monitoring Platform are up and running.

Assumption: The instantiation of the NFV-NS proceeds successfully.

Workflow:

1. The 5GT-VS requests the 5GT-SO, through its NBI, to instantiate a new NFV-NS instance, specifying its NSD, Deployment Flavor (DF) and Instantiation Level (IL).
2. A new ID for the requested NFV-NS is generated (in this example NFV-NSI ID 1) and returned to the 5GT-VS.
3. The NBI forwards the request to instantiate the NFV-NSI to the SOE
4. The NFV-NSI is successfully instantiated, following the procedure defined in section 6.7.2.
5. At the end of the instantiation procedure, the SOE activates the monitoring for the new instance, sending a request to the Monitoring Manager.

The configuration of the monitoring for a given NFV-NSI includes two different steps. The former is the creation of the monitoring jobs to collect the required monitoring parameters through specific Prometheus exporters. The latter is the creation of a dashboard to visualize graphs for all the relevant monitoring parameters. In particular:

6. For each monitoring parameter defined in the NSD, the Monitoring Manager sends a request to the Config Manager for the creation of a monitoring job. The details of a new Prometheus exporter are provided whenever a new kind of monitoring collection is required.
7. The Config Manager interacts with Prometheus to request the configuration of the exporters.
8. The Prometheus platform is internally reconfigured and an acknowledgment is sent back to the Config Manager.
9. The Config Manager generates an internal ID for the monitoring job (and the exporter), which is returned back to the Monitoring Manager.
10. When the complete set of monitoring jobs is configured, the Monitoring Manager interacts with the Config Manager to request the creation of a dashboard to visualize the monitoring data through a set of graphs. The request includes a list of queries specifying how the elementary monitoring items must be aggregated to generate the monitoring time series to be represented in each graph.
11. The dashboard configuration request is forwarded to Prometheus.
12. Prometheus verifies the presence of exporters for the elementary monitoring items and interacts with Grafana to request the creation of the dashboard.
13. Grafana generates a new dashboard, available at a certain URL, and returns the URL to Prometheus.
14. An acknowledgement with dashboard identified and URL is provided back to the Config Manager…
15. … and from the Config Manager to the Monitoring Manager.
16. All the monitoring details about the NFV-NSI are persisted in the NS Instances DB, including the IDs of the exporters and the URL of the dashboard.
17. A notification is sent back from the Monitoring Manager to the SOE to notify the successful activation of the monitoring.

During the runtime of the NFV-NSI, its monitoring data can be visualized from the 5GT-VS GUI, in the page of the Vertical Service Instance (VSI) corresponding to the given NFV-NSI. The procedure to visualize the NFV-NSI monitoring dashboard in the 5GT-VS GUI is the following:

18. The 5GT-VS identifies the NFV-NSI associated to the VSI of interest (in this case NFV-NSI 1) and sends a query to the 5GT-SO NBI to retrieve the details of the NFV-NSI.
19. The 5GT-SO NBI queries the NS Instance DB…
20. … and retrieves the NFV-NSI 1 information …
21. … that are returned back to the 5GT-VS.
22. The 5GT-VS processes the NFV-NSI record and extracts the information about the monitoring dashboard URL.
23. This URL is encoded in the 5GT-VS GUI, so that it can be accessed directly from the vertical user.

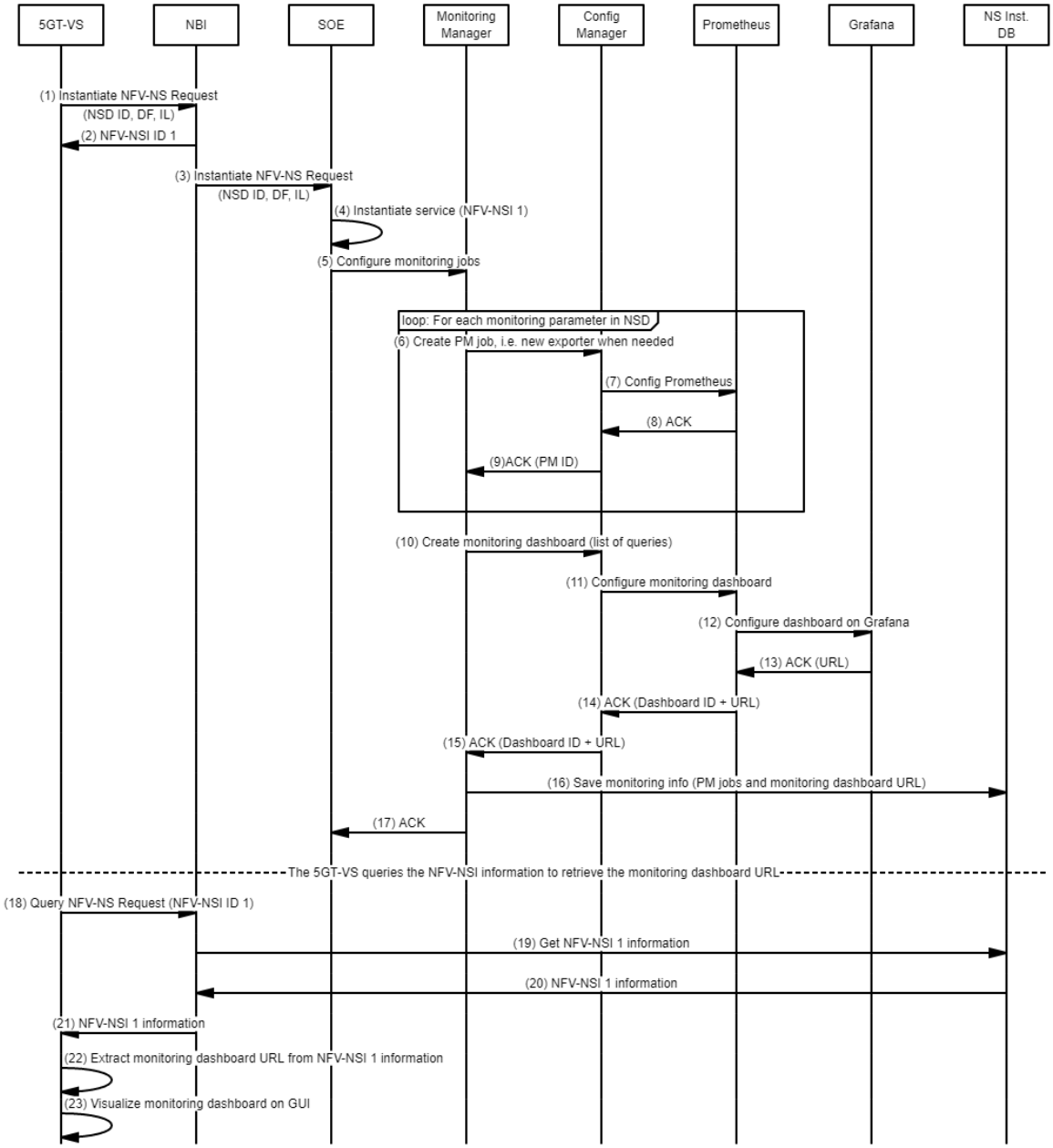


FIGURE 24: Service Monitoring Workflow

6.7.6 Service scaling

6.7.6.1 Vertical-driven scaling

Description: The workflow describes the procedure to scale a NFV network service, triggered by a vertical's request on the scale of a vertical service. This request is processed by the 5GT-VS, which sends the 5GT-SO an explicit request to scale the NFV-NS to a different instantiation level.

Prerequisites: The vertical has instantiated a vertical service instance and the 5GT-VS has sent a ScaleNS requests to the SO.

Assumptions: We assume that the service is deployed successfully. The vertical sends a request to the 5GT-VS to scale out the service from the current deployment toward a higher instantiation level.

Workflow:

Since the vertical-scaling workflow is very large, it is hard to present it on a single diagram. In order to mitigate this issue, the whole workflow is split on to 4 phases:

- Request phase
- Execution phase
- Monitoring upgrade phase
- 5GT-VS notification phase

Each phase workflow diagram is shown after the actions list, that describes it.

1. After the vertical has triggered the VSI Scale workflow by requesting the 5GT-VS to modify an existing VSI, the 5GT-VS sends a ScaleNS request to the 5GT-SO NBI. This request includes information about the type of scaling to be performed (in this case the scaling of an NFV-NS towards a different instantiation level), the ID of the NFV-NSI to be scaled and the target instantiation level. Such request is forwarded to the SOE.
2. The SOE invokes the NS inst DB to check if the NFV-NSI with the given ID is correctly instantiated.
3. The SOE receives information about the target NFV-NSI from NS inst DB.
4. The SOE requests the NSD and the VNFDs of the target NFV-NSI to DB Catalog.
5. The SOE receives the requested information from DB Catalog.
6. The SOE verifies the feasibility of the scaling request and, if the operation is possible, it changes the NFV-NSI status in NS inst DB for to "INSTANTIATION"
7. The SOE generates an ID that is assigned to the current scaling operation and it returns it to the 5GT-VS.
8. The SOE invokes the ROOE requesting to scale the NFV-NSI. This request contains information nsId, nsd, vnfds, current df, current instantiation level, requested instantiation level.
9. The 5GT-VS periodically requests the information about the scaling operation to the 5GT-SO, in order to get updates about the status of the scaling process.
10. The status of the scaling operation (PROCESSING) is returned to the 5GT-VS.

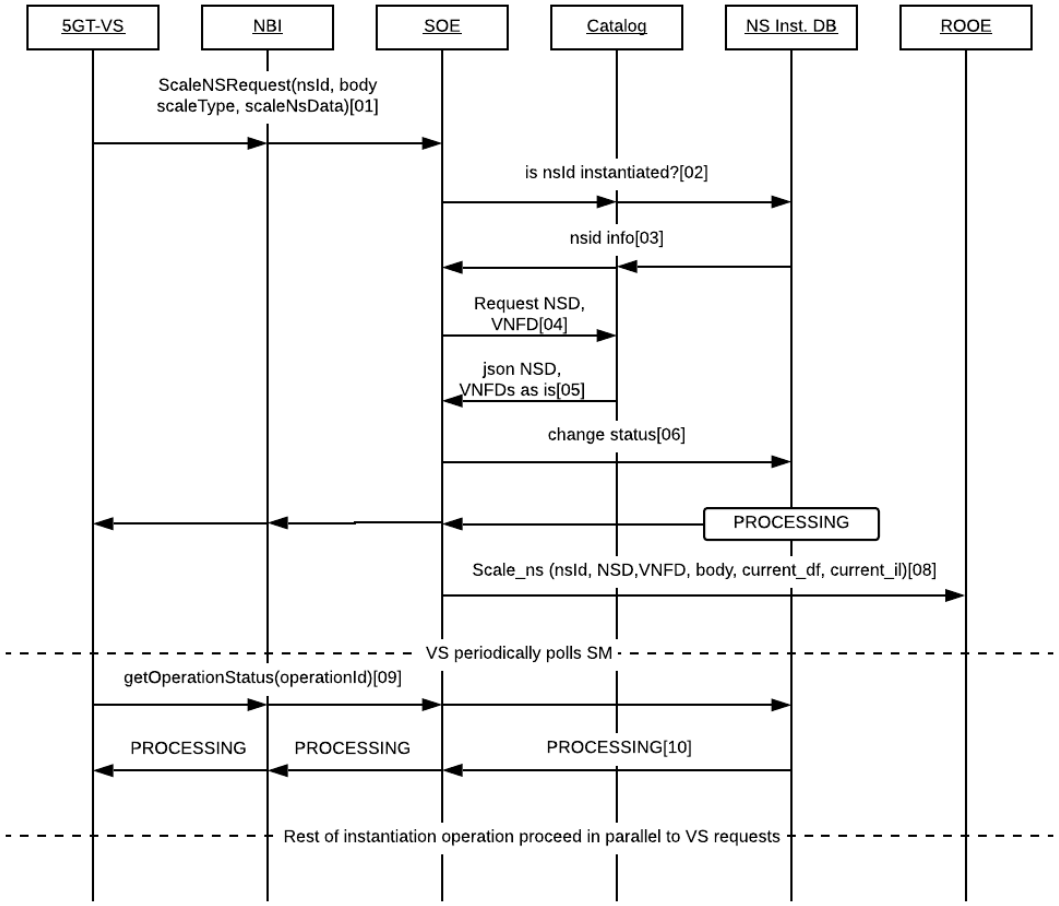


FIGURE 25: VERTICAL-DRIVEN SCALING - REQUEST PHASE

11. The ROOE sends a ScaleNS request to the CoreMANOWrap. This request contains the following information: ID of the NFV-NSI to be scaled, its NSD and VNFDs, its current and target deployment flavours (DF) and instantiation levels (IL). The CoreMANOWrap forwards this request to The Cloudify/OSM Wrapper.
12. The Cloudify/OSM Wrapper processes the received information and calculates the additional number of VNF instances to be instantiated and the links between them. Then, the Cloudify/OSM Wrapper sends a request to Cloudify/OSM, transmitting all the necessary NFVO-specific information to deploy the new VNF instances.
13. Cloudify/OSM requests the 5GT-MTP to deploy the new VNF instances and, when the allocation is completed, it receives back the needed information about links and IP addresses assigned to the VNF instances.
14. Cloudify/OSM returns the deployment status and the information about the IP addresses to ROOE through the Cloudify/OSM Wrapper.
15. The ROOE identifies the new required VLs.
16. If needed, the ROOE invokes the SO-EE-WIM to request the creation of the new VLs.

17. The SO-EE-WIM forward the request to the 5GT-MTP.
18. The operation result from the 5GT-MTP is returned to the ROOE.
19. The ROOE updates the information in the NS Inst. Res. DB,
20. which confirms the success of the operation.
21. The ROOE updates the information in the NFVI Resources DB.

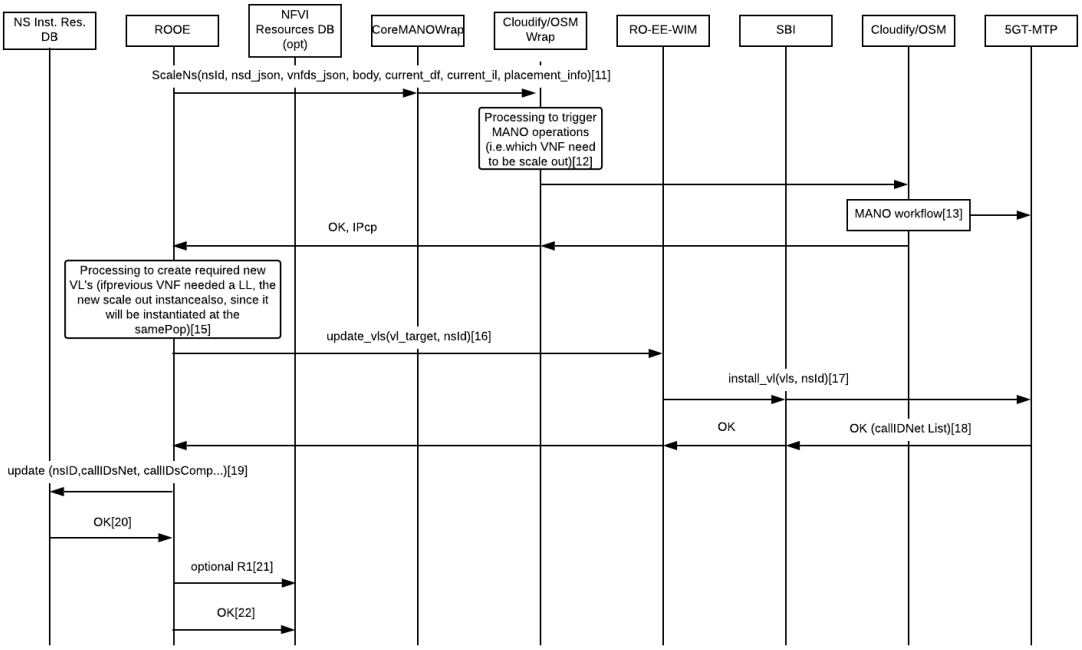


FIGURE 26: VERTICAL-DRIVEN SCALING – SCALING EXECUTION PHASE

22. The ROOE returns the information about the status and the IP addresses of the new VNF instances to the SOE.
23. The SOE requests the Monitoring Manager to create exporters on the Configuration Manager for the new VNF instances (if needed).
24. The SOE requests the SLA Manager to create alerts on the Configuration Manager for new VNF instances (if needed).
25. The SOE changes the status to SUCCESSFULLY_DONE for the current Scale process in NS Inst. DB.

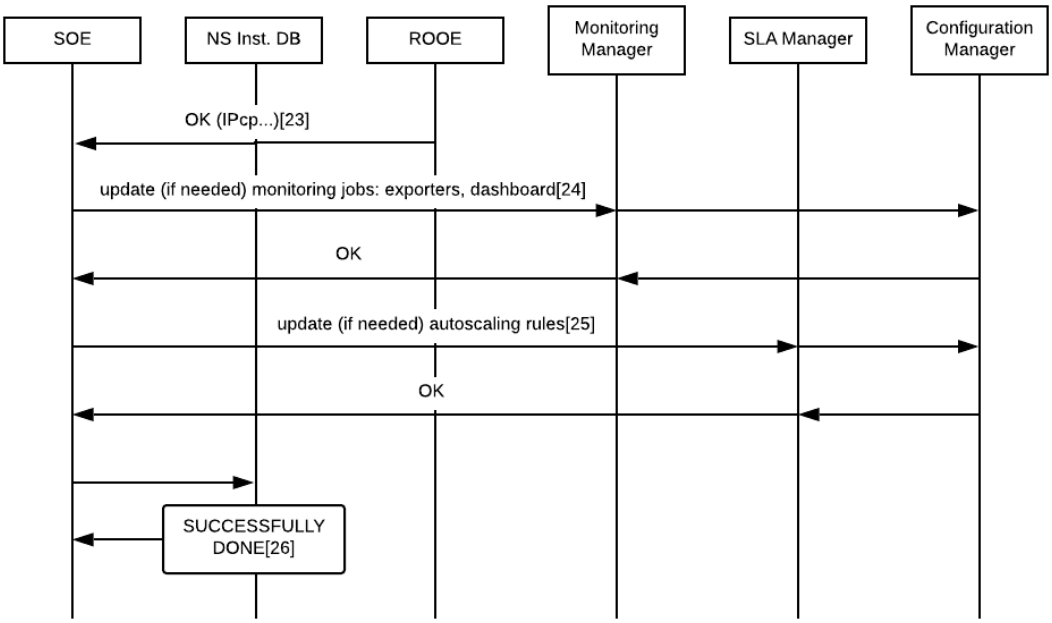


FIGURE 27: VERTICAL-DRIVEN SCALING – MONITORING UPDATE PHASE

26. The 5GT-VS periodically polls the 5GT-SO about the status of the scaling operation.
27. When the scaling operation is successfully completed, the 5GT-SO returns the "SUCCESSFULLY_DONE" code for the requested Scale operation.
28. When the 5GT-VS receives the status SUCCESSFULLY_DONE, it updates its internal database with the new status of the corresponding network slice and vertical service instances.
29. The 5GT-VS sends a queryNs request to the 5GT-SO to receive updated information about the elements of the NFV-NS instance that has been scaled.
30. The 5GT-VS receives such information and updates its internal structures (e.g. updating the amount of resources currently in use for the involved tenant).

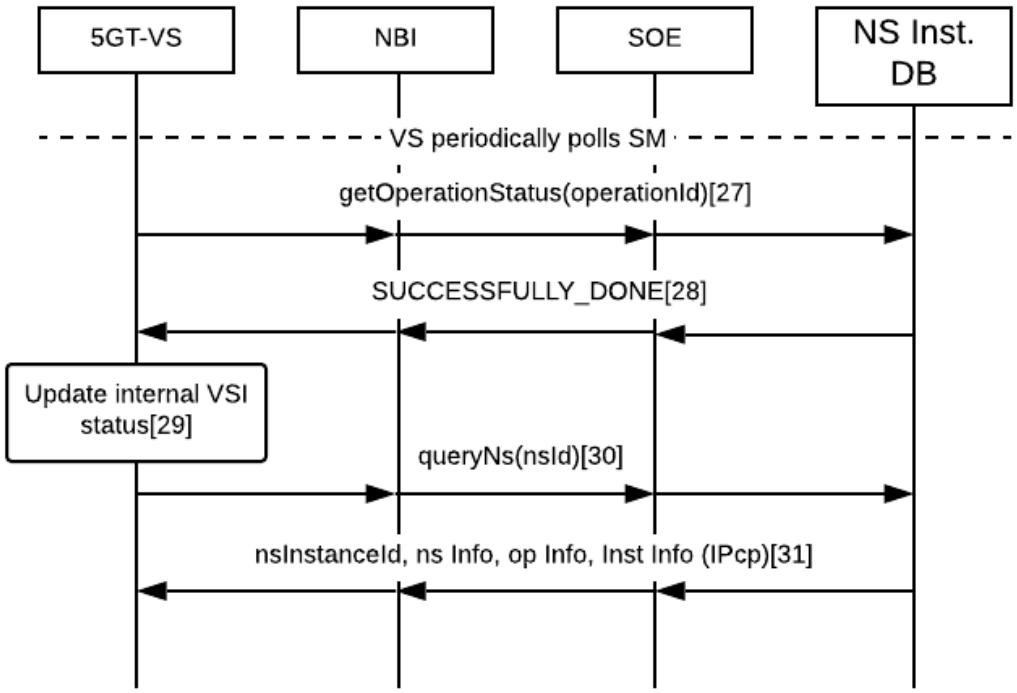


FIGURE 28: VERTICAL-DRIVEN SCALING – 5GT-VS NOTIFICATION PHASE

6.7.6.2 Auto-scaling

Description: The workflow describes the procedure to scale automatically an NFV-NS instance, based on autoscaling rules defined in its NSD. The autoscaling procedure is triggered by the SLA Manager when the monitoring data collected for that NFV-NS instance exceed a given threshold specified in the autoscaling rules.

Prerequisites: The NSD with monitoring parameters and autoscaling rules has been correctly on-boarded.

Workflow:

1. The 5GT-VS sends an “instantiate NS” request to the 5GT-SO NBI. The request specifies the NSD, the initial deployment flavor DF and the initial instantiation level (IL A).
2. The 5GT-SO NBI sends the “instantiate NS” request to the SOE.
3. The SOE instantiates the requested NFV-NS following the procedure described in section 6.7.2 (the detailed messages are omitted for this part of the procedure).
4. Since the NSD contains a list of “monitoredInfo”, each representing a parameter to be monitored, the SOE request the Monitoring Manager to configure the performance monitoring jobs required for the NFV-NS instance.
5. The 5GT-SO Monitoring Manager determines the type of monitoring jobs necessary to retrieve the information related to the NFV-NS instance. For each monitoring parameter defined in its NSD (in section “monitoredInfo”), the

Monitoring Manager invokes the Config Manager to request the creation of the related job.
6. The Config Manager translates the monitoring job specification into Prometheus-specific parameters and changes the Prometheus configuration.
7. The new Prometheus configuration is successfully loaded.
8. The Config Manager returns the ID of the performance monitoring job to the Monitoring Manager.
9. When all the monitoring jobs are created, the Monitoring Manager updates the information of the NFV-NS instance in the 5GT-SO NS Inst. DB.
10. The Monitoring Manager notifies the SOE about the successful creation of the performance monitoring jobs.
11. The SOE invokes the SLA Manager for activating the configuration needed to receive the alerts associated to the NFV-NS instance.
12. The SLA Manager queries the 5GT-SO NS Inst. DB to retrieve the list of performance monitoring jobs activated for the given NFV-NSI.
13. Such details are returned by the NS Inst. DB.
14. For each autoscaling rule defined in the NSD (in the “autoScalingRule” section, the SLA Manager requests the Config Manager to create an alert subscription, specifying a query that defines the metric to be evaluated and the threshold.
15. The Config Manager re-configure Prometheus to create the new alert configuration.
16. This configuration is also applied to the Prometheus Alert Manager.
17. The Config Manager notifies the SLA Manager about the successful creation of the alert configuration, providing details about the alert and its ID.
18. The SLA Manager saves the information about the configured alerts in the NS Inst. DB.
19. The SLA Manager notifies the SOE about the successful configuration of the alerts. At this stage the monitoring for the NFV-NS instance is fully active.

Let’s see now assume that one of the performance metrics specified in the autoscaling rules exceeds the related threshold. The detection of this condition starts the procedures for an autoscaling action.

20. Prometheus detects that some monitoring parameter is higher than the related threshold.
21. The Alert Manager is notified by Prometheus.
22. The Alert Manager sends an alert message to the SLA Manager.
23. The SLA Manager receives the alert message, reads the Alert ID and, based on that, it decides that a scaling action is required.
24. The SLA Manager sends a request to the 5GT-SO NBI to start the NFV-NS scaling procedure.

From this step on, the 5GT-SO internal processing of this scaling action follows exactly the same workflow reported in section 6.7.6.1.

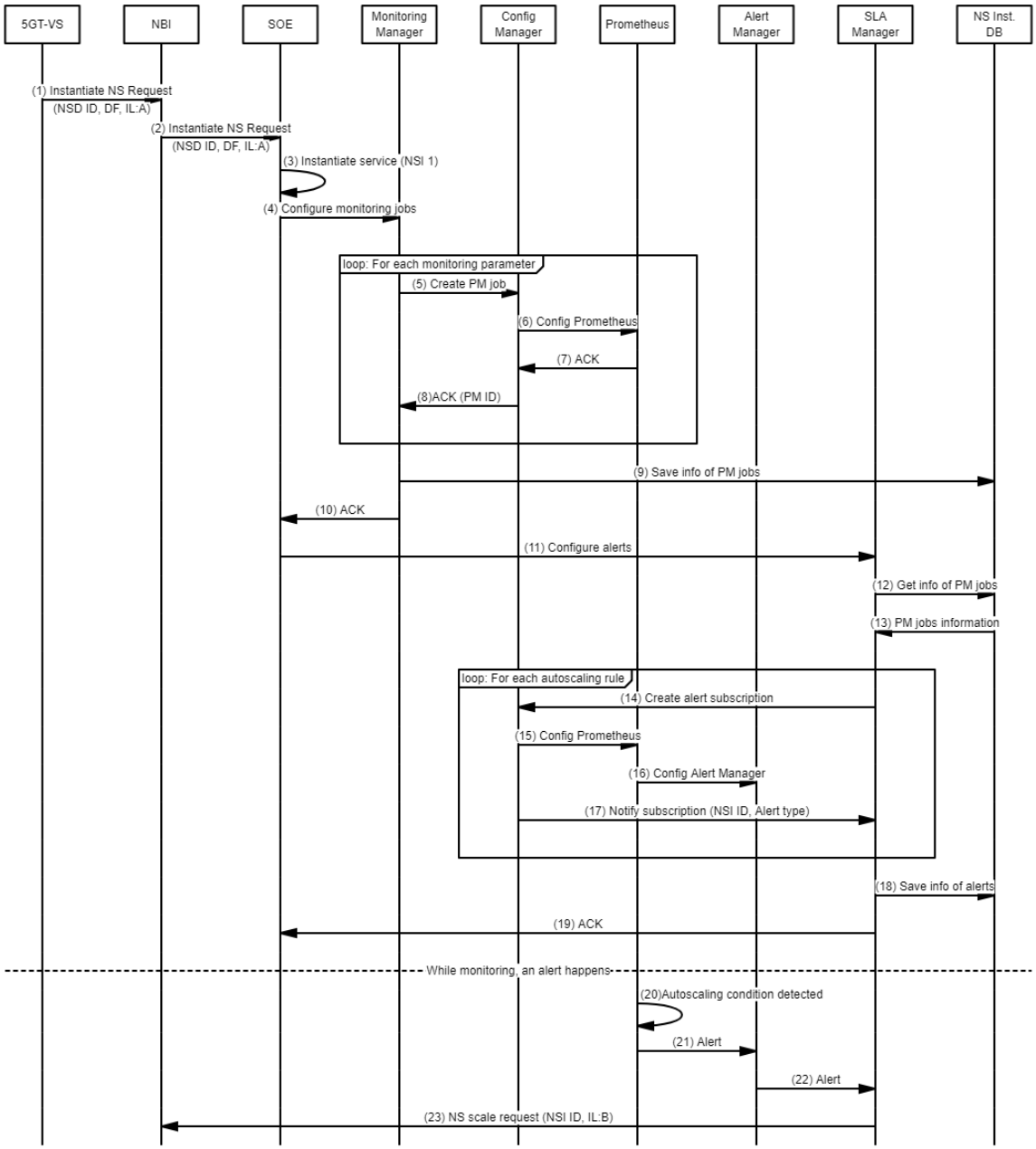


FIGURE 29: NFV-NS AUTOSCALING WORKFLOW

6.7.7 Service Federation

Description: The workflow describes network service federation through instantiation of composite NFV-NSs locally and in different administrative domains.

Prerequisites: Different administrative domains need to have some business relations.

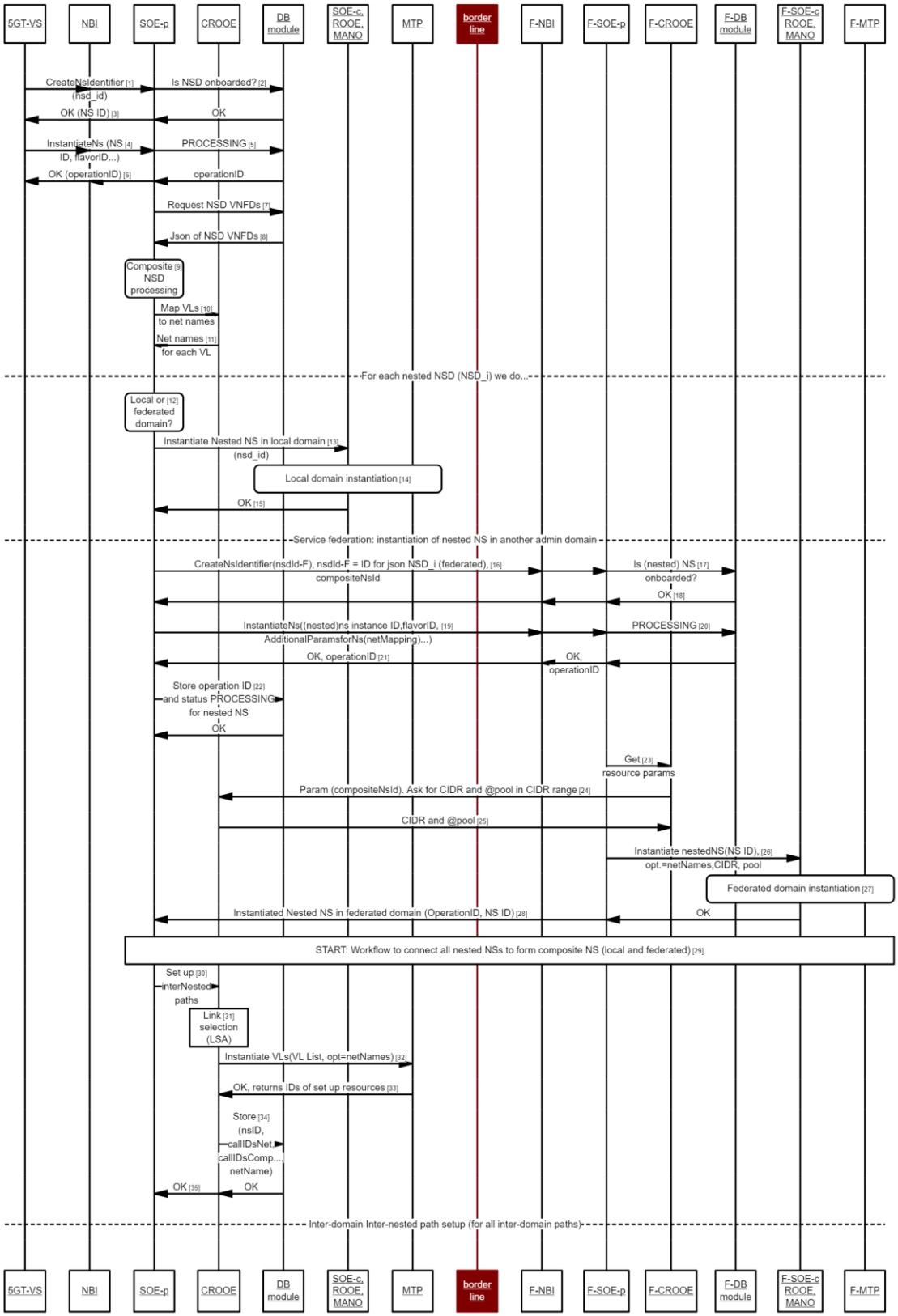
Assumptions: 5GT-SO receives a request for instantiation of a certain composite NFV-NS (with the NSD) that should be instantiated in multiple administrative domains, it can exploit the service federation for accommodating and instantiating the end-to-end NFV-NS instance. Only complete nested NFV-NSs can be instantiated in different (external) administrative domains.

Workflow:

1. The 5GT-VS requests the SOE-parent of the 5GT-SO, through its NBI, to create a new NFV-NS identifier for a composite NFV-NS, specifying its NSD id.
2. The SOE-p checks in the DB if the NSD of the requested NFV-NS is on-boarded already. The DB module confirms that NSD_id exist in the database. The SOE-p generates & stores “NS ID” in the DB.
3. The SOE-p sends confirmation to the 5GT-VS including the NS ID.
4. 5GT-VS sends instantiation of NFV-NS request to the SOE-p via the NBI. The request contains the NS ID, deployment flavor, instantiation levels, etc.
5. The SOE-p initiates the operation. Generates & stores operationID along with a status parameter (“PROCESSING”) in the DB
6. The SOE-p responds to the 5GT-VS with the operationID included in the message.
7. The SOE-p extracts the NSD/VNFD from the DB.
8. The DB responds with all the descriptors delivered as a JSON format
9. The SOE-p processes the NSD, extracts each of the nested NSDs
10. The SOE-p sends request to the CROOE to map Virtual Links (VLs) of composite NSDs and those of the corresponding nested NSDs and to assign a unique network name to each of them, which will eventually be used at the infrastructure level when creating the networks to which VNFs attach
11. The CROOE generates the net names and responds to the SOE-p with the mappings of the VLs
12. The SOE-p starts iteration for each nested NSD, to check if it should be instantiated locally or in a federated domain.
13. If the NSD should be instantiated in the local domain, the SOE-p sends instantiation request to the SOE-c
14. The SOE-c continues with the regular service instantiation workflow (Section 6.7.2) including the rest of the modules ROOE, RO-EE-WIM, OSM/Cloudify wrappers, OSM/Cloudify and MTP.
15. The nested NFV-NS is instantiated locally, SOE-c sends confirmation to SOE-p for the instantiation. The SOE-p continues with iteration and instantiation of local nested NFV-NSs using the NSD_i for the NFV-NS instance i.
16. After the local iteration finishes for instantiation of nested NFV-NS in another federated domain, the SOE-p sends request for creating new NFV-NS identifier using the nsdId-F of the nested (federated) NFV-NS.
17. The (federated) F-SOE-p receives the request and checks if NSD with nsdId-F has been already on-boarded in the F-DB (i.e., there was a previous agreement between domains and the agreed NSD was onboarded in the F-DB). If it is true, the F-SOE-p generates and stores new NS ID in the F-DB.
18. The F-SOE-p responds to the SOE-p including the NS_ID of the nested NFV-NS.
19. The SOE-p sends instantiation request to the F-SOE-p to instantiate the nested NFV-NS using the NS ID, nsdId-F, the deployment flavor, additional parameters (such as the network mappings previously performed by the CROOE).
20. The F-SOE-p receives the request and starts the operation by assigning & storing operation ID and status (“PROCESSING”) in the F-DB.

21. The F-SOE-p confirms the start of the instantiation to the SOE-p, including the reference operation ID.
22. The SOE-p stores the operation ID and the status of the operation in the DB.
23. The F-SOE-p instructs the F-CROOE to obtain information regarding the resource parameters needed for instantiation of the nested (federated) NFV-NS.
24. The F-CROOE sends request for information to CROOE regarding the CIDR and the @pool in the CIDR range.
25. The CROOE returns the CIDR and @pool in the CIDR range to the F-CROOE.
26. The F-SOE-p issues instantiation request to the F-SOE-c including the NS ID of the nested (federated) NS ID, deployment flavors, instantiation levels, as well as the obtained information (network names, CIDR, @pool).
27. The F-SOE-c executes/orchestrates the instantiation of the nested (federated) NFV-NS as an instantiation of regular NFV-NS in local domain (including as optional parameters: network names, CIDR, @pool).
28. The F-SOE-c confirms the instantiation of the nested (federated) NFV-NS to the F-SOE-p. The F-SOE-p redirects the confirmation along with the operation ID to the SOE-p
29. All nested NFV-NSs are instantiated in local and federated domain, the inter-nesting (inter-connecting) to compose the end-to-end composite NFV-NS begins.
30. The SOE-p sends request to the CROOE to set-up all necessary inter-nested paths between nested NFV-NSs of the local domain
31. The CROOE starts the instantiation of the local inter-nested VLs. The Link selection algorithm runs to select the networking resources used to establish the local VLs.
32. The output of the LSA from the CROOE is sent to the MTP to instantiate the local VLs along with the optional parameter of the network names.
33. The MTP confirms the instantiation of the local VLs with all the IDs of the allocated networking resources.
34. Upon instantiation of all local inter-nested VLs, the CROOE stores all the information in the DB.
35. The CROOE sends confirmation to the SOE-p for the completed instantiation of the local inter-nested VLs.
36. The SOE-p instructs the CROOE to set-up inter-domain paths for all inter-domain inter-nested VLs between local and federated nested NFV-NSs.
37. To each federated domain, the CROOE sends request to F-CROOE to get information regarding the IP addresses and VLAN IDs to set up the link towards the nested (federated) NFV-NS.
38. The F-CROOE replies to the CROOE with the information of the IP addresses and VLAN IDs for inter-nesting the instantiated nested (federated) NFV-NS.
39. Based on the collected information from each federated domain, the CROOE starts the instantiation of the inter-domain inter-nested VLs for all nested (federated) NFV-NS. The LSA selects the logical links for all VLs towards the federated domains.
40. The calculated output from the LSA (CROOE) is sent to the MTP along with the collected information (IP addresses, VLAN IDs, network names, etc.)

41. The MTP sets up and allocates all the local networking resources in order to establish the inter-domain inter-nested VLs. The information of the resource IDs is returned back from the MTP to the CROOE.
42. The CROOE stores the returned information regarding the instantiated networking resources in the DB.
43. The DB confirms the operation.
44. The CROOE sends the IP addresses of all VNFs that are connected to the inter-nested links in all domains to all F-CROOEs of federated domains.
45. Each F-CROOE starts the instantiation of all inter-domain inter-nested VLs in order to connect the nested (federated) NFV-NS to the rest of the end-to-end NFV-NSs. The LSA of the F-CROOE calculates the link selections for all VLs.
46. The calculated output is sent from the CROOE to the F-MTP.
47. The F-MTP instantiates the networking resources and establishes the inter-domain inter-nested VLs. Then, the F-MTP sends confirmation and all the information regarding the instantiated networking resources.
48. The F-CROOE stores the information of the instantiated inter-domain inter-nested VLs in the F-DB.
49. Each F-CROOE from each federated domain sends confirmation to the CROOE of successfully instantiated inter-domain inter-nested VLs.
50. The CROOE confirms to the SOE-p the successful instantiation of all inter-domain inter-nested VLs.
51. The SOE-p stores in the DB the successful instantiation of the composite NFV-NS in both local and federated domain.
52. The DB confirms the storing operation.
53. The 5GT-VS polls for information regarding the instantiation of the composite NFV-NS using the operationID.
54. The SOE-p confirms the status of the operation “SUCCESSFULLY_DONE”
55. The 5GT-VS further queries about the information regarding the end-to-end NFV-NS instance.
56. The SOE-p extracts information from the DB and returns all parameters including the NS instance id, IP addresses etc.


5GT-VS
NBI
SOE-p
CROOE
DB module
SOE-c, ROOE, MANO
MTP
border line
F-NBI
F-SOE-p
F-CROOE
F-DB module
F-SOE-c ROOE, MANO
F-MTP
CreateNsIdentifier [1] (nsd_id)
Is NSD onboarded? [2]
OK (NS ID) [3]
OK
InstantiateNs (NS [4] ID, flavorID...)
PROCESSING [5]
OK (operationID) [6]
operationID
Request NSD VNFDs [7]
Json of NSD VNFDs [8]
Composite [9] NSD processing
Map VLs [10] to net names
Net names [11] for each VL
For each nested NSD (NSD_i) we do...
Local or [12] federated domain?
Instantiate Nested NS in local domain [13] (nsd_id)
Local domain instantiation [14]
OK [15]
Service federation: instantiation of nested NS in another admin domain
CreateNsIdentifier(nsdId-F), nsdId-F = ID for json NSD_i (federated), [16] compositeNsId
Is (nested) NS [17] onboarded?
OK [18]
InstantiateNs((nested)ns instance ID,flavorID, [19] AdditionalParamsforNs(netMapping)...)
PROCESSING [20]
OK, operationID [21]
OK, operationID
Store operation ID [22] and status PROCESSING for nested NS
OK
Get [23] resource params
Param (compositeNsId). Ask for CIDR and @pool in CIDR range [24]
CIDR and @pool [25]
Instantiate nestedNS(NS ID), [26] opt.=netNames,CIDR, pool
Federated domain instantiation [27]
Instantiated Nested NS in federated domain (OperationID, NS ID) [28]
OK
START: Workflow to connect all nested NSs to form composite NS (local and federated) [29]
Set up [30] interNested paths
Link [31] selection (LSA)
Instantiate VLs(VL List, opt=netNames) [32]
OK, returns IDs of set up resources [33]
Store [34] (nsID, callIDsNet, callIDsComp..., netName)
OK [35]
OK
Inter-domain Inter-nested path setup (for all inter-domain paths)

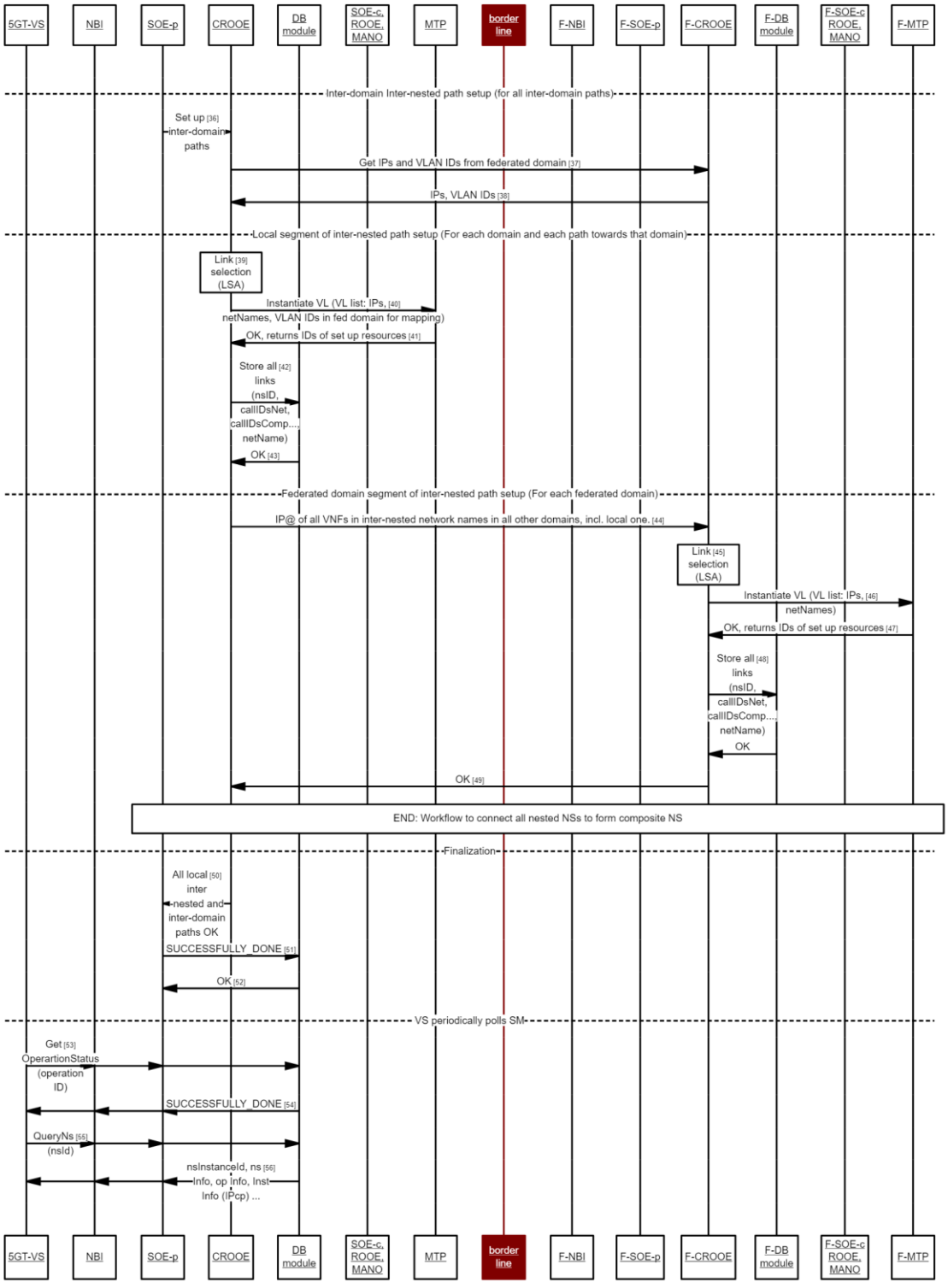


FIGURE 30: SERVICE FEDERATION WORKFLOW

6.8 Details of the 5GT-SO software implementation

This section presents the software architecture and details the internal structure and functionality of the software components that have been implemented in the 5GT-SO.

6.8.1 Mapping of functional architecture to software architecture

In Section 6.2, the functional architecture of the 5GT-SO is introduced. That section presents the concept-level functional architecture with a few more functions than those that were implemented.

In order to distinguish the implementation from the functional architecture, the software architecture is presented below. The mapping from the functional architecture to the software architecture figure is as follows:

- NBI Exposure layer → NBI
- NFV-NS/VNF Catalogue DB/Manager → *not implemented*
- NFV Orchestrator → Service Manager + Core MANO platform
- NFVO-NSO → SOE parent + SOE child
  - Composite NSO → SOE parent (SOE-p)
  - Constituent NSO → SOE child (SOE-c)
  - NS decomposition algorithms → *not implemented*
- NFVO-RO → ROOE + ROEE + CROOE + Core MANO platform
  - RO-OE → ROOE + Core MANO wrappers + Core MANO platform
  - PA → PA
  - RO-EE → ROEE
  - Composite RO → CROOE (incl. LSA)
- VNFM → *Exploiting that of core MANO platform*
- SO-SO advertisement & management block → *not implemented*
- NFVI Resource Repository → NFVI Resources DB
- NS/VNF Instance Repository → NS Instance DB + NS Instance Resources DB
- SO Monitoring Manager → Monitoring manager
- SLA Manager → SLA Manager

6.8.2 Software architecture

The 5GT-SO uses two MANO platforms as orchestration engines for managing compute resources, Cloudify and OSM. Others may be added by implementing a wrapper. The SM component has been designed to integrate the different MANO platforms with the inclusion of a wrapper for each platform providing more functionalities and allowing multi-MANO platform compatibility. The extended functionalities include automatic translation between descriptors of the 5GT-VS component (compliant with ETSI NFV-IFA013) and the modeling language of OSM and Cloudify orchestration platforms (YANG and TOSCA, respectively). It also includes a driver that provides a unified interface between the SM and the MANO platforms.

The architecture extension provides advanced interfacing through the NBI Exposure Layer to OSS/BSS functions represented by the 5GT-VS to provide tailored network service set-up. The interface towards the 5GT-VS supports requests for on-boarding, creation, instantiation, modification and termination of network services. Moreover, SLA support is provided leveraging monitoring services assuring that the agreed SLAs are verified, and the SLA Manager can trigger scaling actions in case of SLA violations.

The 5GT-SO includes the implementation of the PA module that include the placement algorithms in charge of the resource management supporting inter-PoP communications.

6.8.3 5GT-SO components software implementation

The prototype of the 5GT-SO features de functionalities described in section 6.8.2. All the code is open source. In particular, the novel components developed as part of the Service Manager (SM) have been released under Apache 2.0 license and they are available in the project git repository [1] . All its components are presented in Figure 31

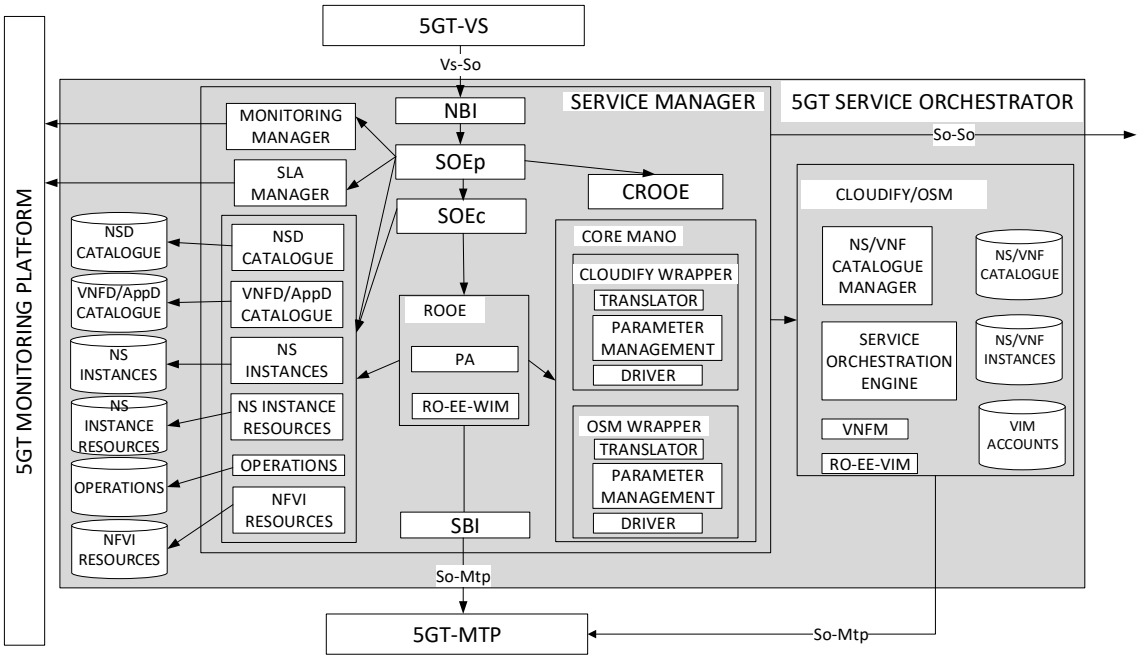


FIGURE 31: 5GT-SO IMPLEMENTED COMPONENTS

6.8.3.1 SM

The SM is implemented in Python 3.5 and adopts MongoDB as database backend for its repositories (Figure 31. The Service Manager is a collection of stateless components, where the status of all the managed entities is maintained through the MongoDB repositories. All the SM modules access the MongoDB repositories to get a shared view of the internal status of the whole system. Concurrent requests about active lifecycle actions on new or existing Network Services are managed in an asynchronous manner, with per-service dedicated threads. The feasibility of each requested action is verified based on the current status of the related service, as reported in the database, and refused in case of conflicts.

The system design of the Service Orchestrator is based on a modular approach, where the entire prototype is structured in several components that interact with each other via REST APIs. The same kind of interaction is also adopted with the external components, like the 5GT-VS and the 5GT-MTP.

In particular, the interaction with the MANO platforms (i.e., Cloudify and OSM) is wrapped by the Core MANO module, which hides the NFVO-specific REST APIs and information models exposing a unified interface towards the other modules of the Service Manager. A similar approach is adopted at the SBI, which is modelled as an extension of the interface specified in ETSI NFV IFA 005 [2] and, internally, VIM- and

WIM-specific drivers translate into the HTTP messages exposed by the 5GT-MTP REST APIs. It should be noted that the standard IFA 005 defines the interaction with a generic VIM, thus including primitives related only to resources managed by cloud platforms like OpenStack, for instance, Virtual Machines (VMs), virtual networks, or virtual storage. In 5GT this concept has been extended to cover also the interaction with WIMs, thus introducing a set of messages for the advertisement of network topology from 5GT-MTP and the request of network paths to the 5GT-MTP to interconnect PoPs with a given Quality of Service.

At the NBI, the SM implements a REST server based on the HTTP protocol and Json language for message encoding. The information models for NSDs, VNFDs, and Network Service lifecycle actions (e.g. instantiation, termination and queries of Network Service instances) follow the ETSI NFV IFA specifications 014 [3], 011 [4] and 013 [5], respectively. This also includes composite network services. Extensions are possible to describe additional characteristics of the Network Services, like location constraints, or specify service constraints or policies to be applied at runtime. Another functionality of the Core MANO component is to translate all the Json-based descriptors exchanged at the Service Manager NBI into the NFVO-specific format when performing onboarding operations. For example, TOSCA-based descriptors are used for Cloudify, while Yaml- or Json-based descriptors are used for OSM, each of them with its own proprietary information models.

REST APIs are also used internally in the Service Manager. The service orchestration engine is split into two, namely parent (SOEp) and child (SOEc). The former handles federation-related service orchestration, while the latter is in charge of orchestration regular or each individual nested service. Another example of the use of REST APIs is allowing the ROOE module to interact with different PAs. This approach allows supporting multiple resource orchestration algorithms which can be developed by third parties and easily integrated in the system. Depending on the specific criteria to be enforced for the selection of the resources, different PAs can be selected, according to operator's policy or constraints provided by the vertical. The ROOE-PA API has been designed using Swagger and it specifies both the Information Model (IM) and the operations that the PAs can perform (e.g., computing where to deploy a requested service). The ROOE and PAs use client and server stubs, respectively, generated with Swagger that implement such operations and IM. Data, such as the resources requested by the service VNFs, and available infrastructure computing and bandwidth resources, are present in the Json that follows the IM and exchanged in every operation. Forcing the ROOE and every PA to implement a common API, eases the integration of new PAs in the system and the development and maintenance of the ROOE component.

Even if they are part of the SM, given their relevance in the SM operation, the SOE and the RO-OE are explained in more detail below.

6.8.3.2 SOE

The service orchestration engine is divided into the SOE-parent (SOEp) and the SOE-child (SOEc). The former deals with service composition and federation, while the former handles service orchestration of regular NFV-NSs (i.e., a non-composite NFV-NS or a single nested NFV-NS included in the composite NFV-NS).

If the service is composite and it must be deployed exclusively in the local domain (i.e., there is no service federation), the SOEp will delegate each nested NFV-NS to the SOEc, and will set up inter-nested NS links by interacting with the composite ROOE, which, in turn, will request path setup to the 5GT-MTP.

If the service is composite and there is at least one nested NFV-NS deployed in another administrative domain, the local SOEp will request the instantiation of this nested NFV-NS to the SOEp of the federated domain through its northbound interface. The internal processing in the federated domain is the same as what was explained above. Furthermore, for those NFV-NSs deployed in the local domain, it will also act as explained above. Finally, the local SOEp will instruct the local CROOE to interact with the federated domain CROOE to eventually set up the inter-domain inter-nested paths, which implies that each of the CROOEs (local and federated) will interact with the respective 5GT-MTPs to request their segment of the inter-domain inter-nested path.

Both SOE are implemented as a Python package. The corresponding SOE exposes a Python package interface to receive the requests from the NBI in case of the SOEp, or the SOEp in case of the SOEc. For the NFV-NS query-related operations, the SOEp interacts with the database to fulfil the request on the instantiation state and NFV-NS-related information of the composite NFV-NS.

On the other hand, any query or processing related with regular NFV-NSs (i.e., non-composite NFV-NSs) is fully delegated to the SOEc, which manages the lifecycle of the service. As far as resource management is concerned, the SOE delegates the operation to the ROOE module, which, interacts with the 5GT-MTP to eventually deploy the VNFs in the form of VMs and to set up the network paths that interconnect the VNFs of the service.

The implementation of the SOE module is located in the folder /5GT-SO/sm/soe.

6.8.3.3 Composite RO-OE (CROOE)

The composite RO-OE is the entity in charge of resource orchestration as far as composite NFV-NSs (federated or not) is concerned. In case of composite NFV-NSs deployed in the local domain, it is in charge of setting up the inter-nested paths by sending the corresponding requests to the 5GT-MTP through the SM-SBI. It follows a very similar procedure to that followed by the RO-OE, which does resource orchestration for regular NFV-NSs (section 6.8.3.4). However, designing the procedure in this way allows a modular architecture that allows to clearly separate the procedures followed for regular and composite NFV-NSs.

In case there is service federation, i.e., at least one of the nested NFV-NSs is deployed in a federated domain, the CROOE will also set up the inter-domain inter-nested path. In this case, when requested by the SOEp, it will interact with the CROOE in the federated domain to request the necessary resource information (e.g., IP addresses, VLAN IDs) to eventually set up the local segment of the inter-domain path (local CROOE) and the segment in the federated domain of the same path (CROOE in the federated domain).

Additionally, every time the CROOE needs to select a logical link to connect two nested NFV-NSs belonging to the same composite NFV-NS, it calls the link selection algorithm (LSA).

6.8.3.4 RO-OE

The ROOE handles the requests that are related to 5GT-MTP resources management, i.e., NFV-NS instantiation and termination, and interacts with the PA module, the coreMANO and the RO-EE-WIM to accomplish the request. The implementation of the ROOE module is located in the folder /5GT-SO/sm/rooe.

6.8.3.4.1 Placement Algorithms (PA) Module and PA APIs

All the algorithms presented in Section 6.6.1 have been implemented and are integrated in the 5GT-SO software distribution. In order to have a unified way to access them, a single REST API over which they can be called has been specified using OpenAPI v2.0. Although all algorithms operate on the same basic system model, each one has its particularities, which are also captured in the API (e.g., Placement Algorithm A, if configured to optimize for availability, needs VNF- and NFVI-PoP-level reliability information; Placement Algorithm C requires gateway IP addresses).

The PAs are expected to be running as micro-services, each exposing a REST API endpoint, which the RO-OE uses to request the calculation of resource allocations during the instantiation of NFV-NSs. All algorithms, which are written in python, use the *python flask* library to implement their API servers.

The interaction between the RO-OE and the PAs is asynchronous: In order to invoke a PA and not to block the RO-OE on PAs execution, the caller (RO-OE) needs to specify in the body of the request a callback URL. The PA API service receives the request, stores this URL internally, responds to the caller notifying it that the request has been received, and launches the PA. When the latter has terminated, the PA server posts the results of the algorithm to the callback URL of the caller. Placement requests are identified by a unique *ReqId* value, which has to be provided by the caller, together with the other request information (NFVI-PoP, topology, and NSD-related information, as well as the callback URL to receive the response). This procedure is presented in Figure 32.

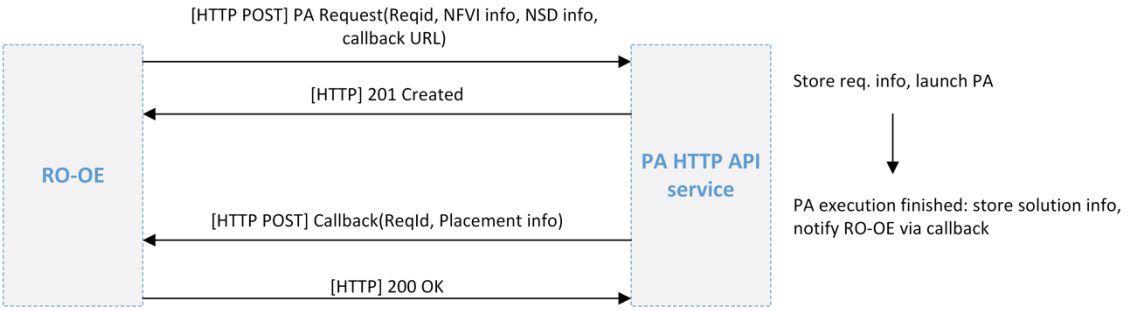


FIGURE 32: INTERACTIONS BETWEEN THE RO-OE AND A PLACEMENT ALGORITHM MODULE.

As shown in Figure 33, the API currently provides three functions:

- Invocation of the PA.
- List of ongoing and past PA execution requests.
- Information about a specific PA execution request identified by ReqId.

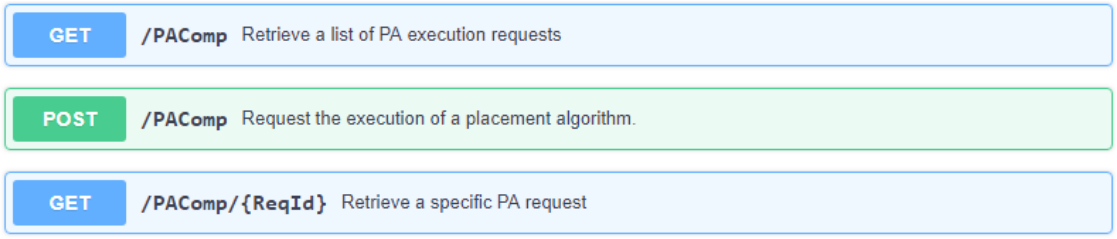


FIGURE 33: AVAILABLE FUNCTION CALLS OF THE PLACEMENT ALGORITHM REST API

The implementation of the PA modules is located in the /pa folder of the service orchestrator software repository. The OpenAPI definition (written in YAML) of the API is available in the same location.

6.8.3.5 CoreMANO wrappers

The coreMANO wrappers are Python modules implemented to provide an abstracted layer between the SM and the MANO platforms enabling the 5GT-SO to be multi-MANO compliant. They include a translator to map ETSI-NFV IFA013 NS descriptors to the MANO platform descriptor format, TOSCA or YANG considering Cloudify and OSM respectively. They also include a parameter management to manage internal operations regarding the relevant orchestration engine, creating NFV-NS identifiers and managing NFV-NS requests including instantiation, modification, termination and updates of the services. The wrappers can support auto-scaling operations and provide a querying engine that interacts with the logging system of the MANO platforms. The driver that implements the integration with the MANO platforms through REST API calls provides functionalities such as PA integration, support for inter-PoP connectivity and service federation.

6.8.3.6 SLA Manager

SLA Manager is the part of Service Orchestrator. SLA Manager creates request for creating alerts to Config Manager and analyze alerts after Alert manager, which are the components of the Monitoring Platform. This interaction is described in the Figure 34. SLA Manager creates the request for creating alerts based on NSD from 5G-VS.

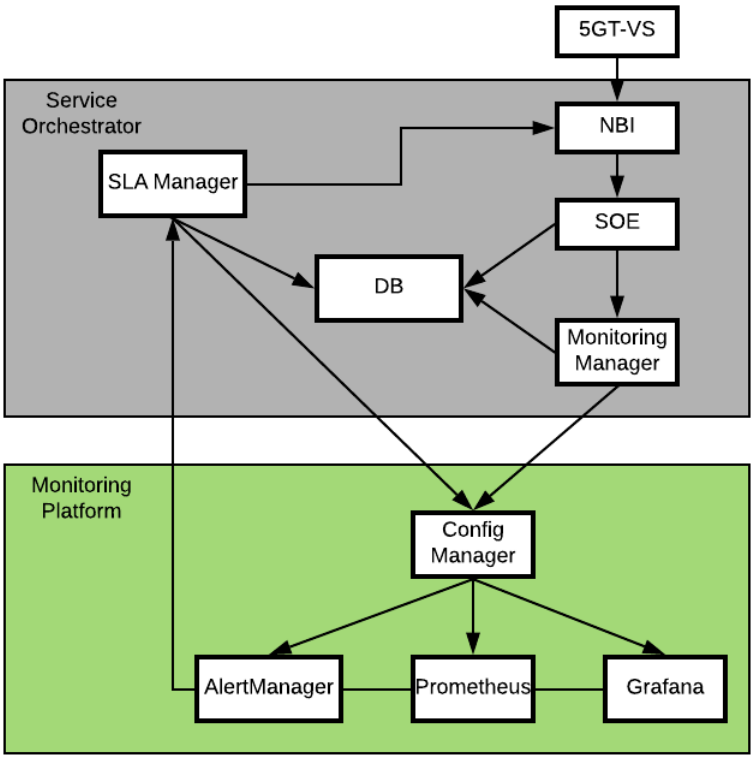


FIGURE 34: 5GT-SO AND MONITORING PLATFORM

In the nutshell, SLA Manager is a program solution, developed on Java programming language. It is run as a process and has an interface to receive messages from Alert Manager, which are basically the system alerts, carrying information about an issue, that can be used by SLA Manager to initiate scaling process. In order to initiate scaling procedure, SLA Manager sends a message to NBI.

6.8.3.7 Monitoring Manager

Monitoring Manager is the part of Service Orchestrator.

Service Orchestration Engine does request to Monitoring Manager to configure monitoring jobs. Monitoring manager receives Network Service Descriptor, Network Service ID of current deployment, set of VNFs' connection points for configure monitoring jobs. Monitoring manager analyses NSD section "monitoredInfo". This section contains information which VNFs should be monitored. Monitoring manager creates requests for Config Manager based on this information and information about VNFs' connection points. When Monitoring Manager does request to Configure Manager, Configure Manager returns JobID for each Monitoring Manager's request. Monitoring Manager save JobID to NS DB. Further this information uses by SLA Manager as mentioned in the previous chapter. Also you can see workflow on Figure 5GT-SO internal workflow for auto-scaling.

6.8.4 5GT-SO Interfaces and reference points implementation

6.8.4.1 5GT SO NBI/SBI

Northbound/Southbound Interface (NBI/SBI) sub-modules, Monitoring Manager and SLA Manager of Figure 31 act as interfaces with the corresponding block of the 5G-TRANSFORMER architecture, namely the 5GT-VS, the 5GT-MTP and the 5GT-Monitoring platform, respectively. In this section, we will provide a review of the NBI interface offered by the 5GT-SO. As mentioned before, this interface is fully aligned with the procedures and information models described at ETSI NFV IFA specifications 013. For a complete specification of the NBI, we refer the reader to the file located in the folder /5GT-SO/nbi/python-flask-server/swagger_server. The port used by this interfaces is 8080 and the base path is 5gt/so/v1/.

The review of the other interfaces of the 5GT-SO are not included in this section since in this case, the 5GT-SO block acts as client to interact with the 5GT-MTP (see deliverable D2.3) and the 5GT-Monitoring platform (sections 6.8.3.6 and 6.8.3.7). It is worth mentioning that the 5GT-SO system communicates with other peering 5GT-SO system by means of the NBI for service federation.

TABLE 9: SO SYSTEM COMMUNICATION OPERATIONS DESCRIPTION

Type of Operation	Path	Brief Description
POST	/ns	Creates and returns a Network Service Identifier (nsId)
GET	/ns/{nsId}	Returns information of the network service referenced by *{nsId}* parameter
PUT	/ns/{nsId}/instantiate	Instantiates the Network Service referenced by *{nsId}* parameter
PUT	/ns/{nsId}/scale	Scales the Network

		Service referenced by *{nsId}* parameter
PUT	/ns/{nsId}/terminate	Terminates the Network Service identified by nsId parameter
GET	/ns/nsd/{nsdId}/{version}	Returns the descriptorn of the network service referenced by *{nsdId}* and *{version}* parameters
DELETE	/ns/nsd/{nsdId}/{version}	Deletes the descriptor of the network service referenced by *{nsdId}* and *{version}* from the nsd catalogue database parameters
GET	/ns/vnfd/{vnfdId}/{version}	Returns the descriptor of the VNF referenced by *{vnfdId}* and *{version}* parameters
DELETE	/ns/vnfd/{vnfdId}/{version}	Deletes the descriptor of the VNF referenced by *{vnfdId}* and *{version}* parameters from the vnfd catalogue database
POST	/ns/vnfdManagement/vnfPackage	Performs and returns information of the onboarded virtual network function
POST	/ns/nsdManagement/nsd	Performs and returns information of the onboarded network service
GET	/operation/{operationId}	Returns the status of the operation specified by *{operationId}* parameter

6.8.5 5GT-SO user guide

The 5GT-SO user guide is available on github, at the following link:

https://github.com/5g-transformer/5gt-so/tree/master/5GT-SO/documentation

6.8.6 How to extend the 5GT-SO

The 5GT-SO software development guide is available on github, at the following link:

https://github.com/5g-transformer/5gt-so/tree/master/5GT-SO/documentation

6.9 Service Monitoring Platform design and implementation

6.9.1 Service Monitoring Platform design

The 5GT-SO Monitoring Platform is in charge of producing monitoring reports related to the performance or to failure events associated to the managed NFV network services

and their VNFs. The generated monitoring reports can be used internally at the 5GT-SO, for example to validate SLAs or as triggers to auto-scaling procedures, according to the auto-scaling rules defined in the NSDs of the instantiated network services. Moreover, the monitoring reports produced by the 5GT-SO Monitoring Service can be also provided to the 5GT-VS through the Vs-So(-MON) reference point (see Section 5.3.5 of D3.3 [22] for details).

The 5GT-SO Monitoring Platform collects elementary monitoring data from different sources and correlates or aggregates them into reports related to the logical entities managed by the 5GT-SO, i.e. NFV network services and VNFs. The sources of elementary monitoring data for the 5GT-SO Monitoring Service are the following:

- Local 5GT-MTP Monitoring Platform, for collecting performance metrics or failure events about the virtualised resources (i.e. virtual computes, virtual networks, virtual storages) that compose the managed VNFs and network services.
- 5GT-SO Monitoring Service of federated 5GT-SOs, for collecting performance metrics or failure events about nested network services or VNFs provided by the federated 5GT-SO, in case of service federation (see section 6.6.5 for further details).
- VNFM(s) or Physical Network Function Managers (PNFMs) for collecting VNF/PNF indicators produced by the managed VNFs/PNFs, or their Element Managers (EMs).

Examples of performance metrics that can be elaborated at the 5GT-SO Monitoring Platform are the amounts of vCPU or RAM consumed in a time interval by specific VNF(s) or by an entire network service, the number of packets lost on a virtual link, or VNF-specific indicators. The rules to correlate the elementary data received by the monitoring sources (e.g. vCPU consumption in single VMs as received by the 5GT-MTP Monitoring Service) are embedded in the monitoring algorithms. These rules allow to obtain the performance record for entire VNFs or NFV network service instances, as specified for the different metrics encoded in the NSDs. Section 6.6.4 provides an example of a workflow to collect monitoring data from the 5GT-MTP and consolidate them in 5GT-SO performance metrics that can feed procedures for SLA validation.

In detail, the scope of the 5GT-SO Monitoring Platform is focused on producing “aggregated” monitoring data that provides relevant information about the status and the performance of a number of logical entities, somehow correlated with the end-to-end NFV-NS instances managed by the 5GT-SO itself. Starting from these monitoring data, the 5GT-SO will take decisions about the lifecycle or the resource orchestration of the NFV-NS instances. In particular, the following logical entities have an impact on the end-to-end NFV-NS instances and, consequently, should be monitored, directly or indirectly, by the 5GT-SO Monitoring Service:

- Virtual resources instantiated from the 5GT-SO in its local domain through the 5GT-MTP.
- Virtual resources from federated 5GT-SO (e.g. VMs, connection services between VMs, etc.) in the NFVIaaS case.

- VNFs, VAs and (nested) NFV Network Service instances managed by the 5GT-SO itself, as well as the ones requested to federated 5GT-SOs in the NFV-NSaaS case.
- PNFs used in scope of the NFV-NS instantiated by the 5GT-SO.
- Optionally, the 5GT-MTP may expose to the 5GT-SO a subset of monitoring data related to the physical resources, for example to enable more efficient resource allocation decisions at the 5GT-SO. This is typically the case where 5GT-SO and 5GT-MTP are managed by the same administrative entity.

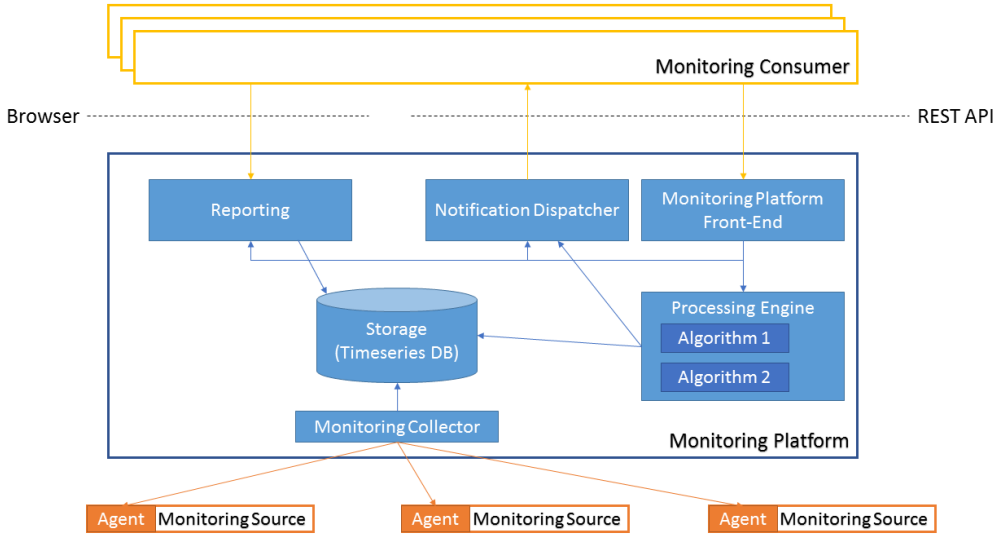


FIGURE 35: FUNCTIONAL ARCHITECTURE OF THE 5GT-SO MONITORING PLATFORM

Figure 35 shows the functional architecture of the 5GT-SO Monitoring Platform. The same architecture can be used for the 5GT-VS and 5GT-MTP Monitoring Platforms, the only difference being that different sources will be used.

At the southbound, the monitoring collector retrieves monitoring data from different monitoring sources through dedicated agents that translates between different information models, monitoring mechanisms or message protocols.

The elementary monitoring data collected by these sources is then stored in the internal storage, typically a time-series DB, to enable more efficient queries. This data can be used as input for further elaboration at the processing engine, which aggregates and correlates elementary data using specialized algorithms, according to the rules and filters specified by the monitoring service consumers. The post-processing data are also stored in the internal DB, so that they are made available for the consumers, which can retrieve them through queries (through the Monitoring Service Front-End) or notifications (through the Notification Dispatcher).

Finally, the monitoring data collected by the platform is also available through monitoring dashboards generated on-demand, which report the status of the requested data via real-time plots populated by querying the monitoring timeseries DB.

6.9.2 Service Monitoring Platform implementation

The 5GT Monitoring platform, as shown in Figure 36 has been implemented by integrating a Prometheus [29] backend with the 5GT-SO through the development of an adapter called Monitoring Configuration Manager (or Config Manager for short). This component acts as a unified entry point for all the configuration functionalities offered by the monitoring platform, simplifying and adapting the several APIs exposing monitoring options to the 5G-T environment.

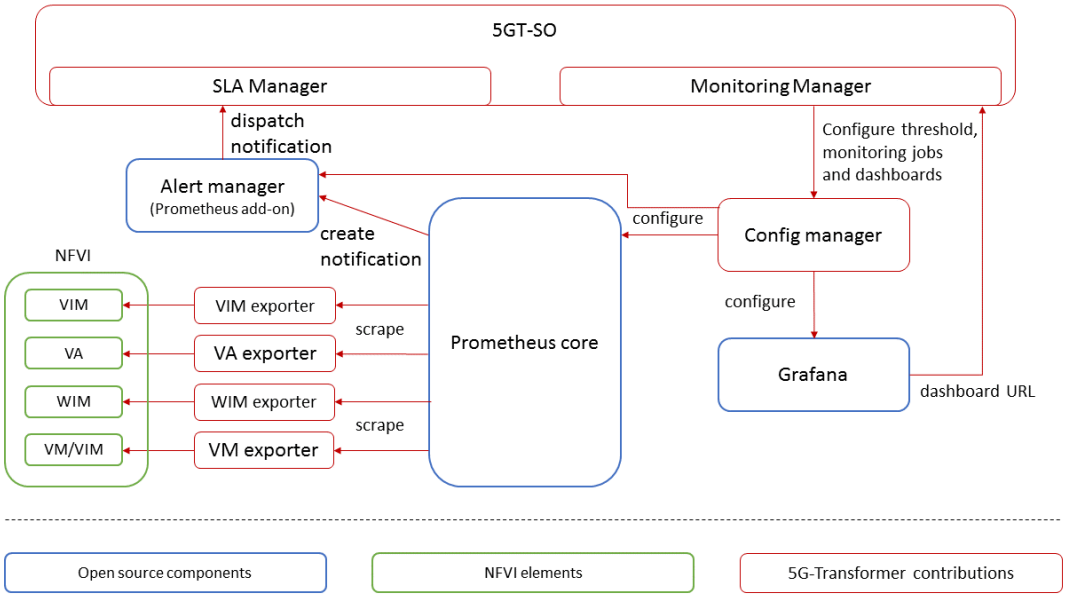


FIGURE 36: MONITORING PLATFORM ARCHITECTURE

In particular, with respect to the functional architecture of the monitoring framework reported in Figure 36, the Agents at the monitoring sources are implemented as Prometheus exporters; the Monitoring collector, the storage and the processing engine components are all implemented by the Prometheus core engine, while PromQL (Prometheus Query Language) expressions take the role of the processing algorithm, aggregating and correlating elementary data as needed. The Notification dispatcher is implemented by the Prometheus Alert manager add-on, while the Reporting component is implemented by a Grafana instance connected to the timeseries database in Prometheus core.

All the Prometheus-based monitoring platform is integrated with the 5GT-SO through the mediation of the Config Manager, a lightweight REST adapter exposing a single, simplified REST API for the configuration of the whole monitoring system (e.g. to create monitoring jobs for target monitoring parameters, or thresholds for monitoring alerts), acting as the “management” NBI of the Monitoring platform. The Config Manager takes care of translating the platform-independent requests for monitoring-related management actions into requests directed to both Prometheus and Grafana, taking care of all the necessary configuration steps.

On The 5GT-SO side, the Config Manager interacts with the Monitoring Manager and the SLA Manager. The Monitoring Manager is the component in the 5GT-SO translating requests for high-level monitoring jobs referred to NFV service instances into low-level requests related to the monitoring of resource-level parameters, which are the data actually collected at the Monitoring Platform.

The SLA Manager deals instead with SLA assurance. This component is thus interested in monitoring parameters, like Key Performance Indicators or real-time measurements mirroring the load of the resources, that may indicate potential SLA breaches or anomalous behaviours that may lead to system under-performance. The SLA Manager interacts with the Monitoring Platform following a subscription-notification paradigm, in order to promptly react to any alert associated to the target monitoring data. In particular, the SLA Manager subscribes with the Config Manager, registering monitoring parameter queries (which may include arbitrarily complex expressions on many different time series) and threshold values for notifications. Such subscription is translated into the appropriate configuration of the Prometheus Alert Manager add-on, so that whenever the given threshold is passed, a notification is sent to the SLA Manager, which will then trigger the needed reactions at the 5GT-SO.

7 Annex II – 5GT-SO SST Implementation

This section introduces a 5GT-SO design that uses a proprietary tool to achieve basic functionalities, if compared to 5GT-SO reference implementation described in Section 6. The basic operations that are supported are Create, Read, Update and Delete (CRUD) for service/slice deployment, as requested from the Vertical Slicer. This implementation is developed to support basic Slices/Service Types (SST) as defined in 3GPP specification [19], and it is intended as a simple SO to support the 5GT-VS SST described in D3.4 [30]. Therefore, this implementation is referred as 5GT-SO SST in this document.

Figure 37 shows the main components in the 5GT-SO SST design. The main components are: *Resource Monitor Module (RMM)* and *Placement Algorithm Module (PAM)*. Furthermore, NBI and SBI interfaces connect the 5GT-SO SST to the 5GT-VS and to the OpenStack *Tacker* orchestrator respectively [31]. The RMM/PAM modules and the corresponding interfaces are developed using Python. *Tacker* is used to either trigger CRUD operations for the requested service according to the command from PAM or to get the resource status of each VIM according to the command from RMM. It is important to note that the orchestrator is not limited to Tacker and it can utilize different platform such as OpenMANO or Cloudify. Also, R2 use the same components of 5GT-SO SST used in R1 with enhancement for PAM to allow the dynamic network slicing and allow adopting several scenarios as elaborated in Section 7.4.

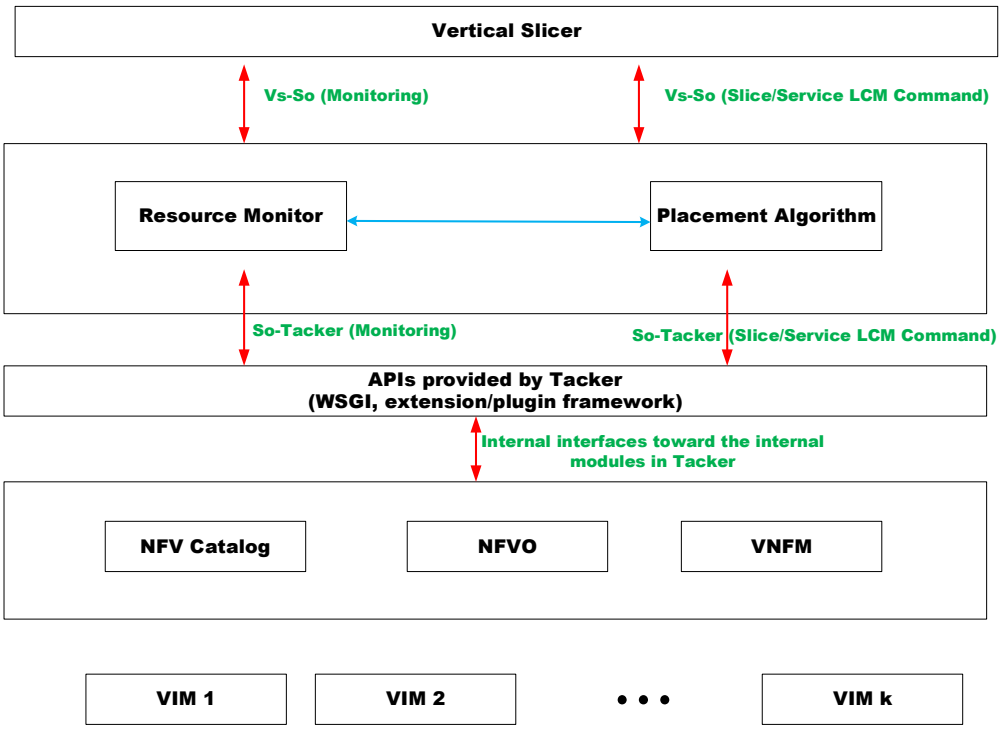


FIGURE 37: 5GT-SO SST ARCHITECTURE

The NBI of RMM and PAM receives the requests from 5GT-VS in terms of Monitoring commands and Slice/Service Life-Cycle Management (LCM) commands. The SBI of RMM and PAM enables the request towards Tacker. Similarly, the messages exchanged between the 5GT-SO STT core components (i.e. RMM and PAM) and

Tacker are Monitoring and Slice/Service LCM commands. The following subsections describe the basic functionalities provided by RMM and PAM.

7.1 RMM

The Resource Monitoring Module (RMM) is responsible for getting the status and the resource consumption of each VIM under the control of the Tacker platform. Additionally, RMM can have the information of current registration/deregistration for each VIM. RMM can collect the information of the resource consumption and forward it towards PAM. PAM can handle the requested service according to the optimized placement for the related VNF placement in each VIM. The RMM module in R1 and R2 is supported by using the interface/plugin provided by Tacker to get the information of resource consumption.

7.2 PAM

The Placement Algorithm Module (PAM) receives the service deployment command from 5GT-VS and starts to estimate the resource requirements for the requested NFV-NS, i.e. calculates the placement of each VNF related to the requested NFV-NS. With the resource consumption information from RMM, PAM can handle and calculate the proper placement for each VNF related to the requested NFV-NS and generate the VNFFG (VNF Forwarding Graph) descriptor in a TOSCA format [XXX] as input for the request to the Tacker platform. Then, Tacker can start to deploy each VNF of the requested service according to the description in the VNFFG. The PAM module in R1 is supported by using the interface/plugin provided by Tacker to use basic SLA checking to see whether the resource is following the SLA constraints and to check whether sufficient resources are available

7.3 Monitoring Platform

In middle of Figure 37, the Monitoring Platform of the 5GT-SO SST implementation is supported by the plugin framework in Tacker. In Tacker, internal interfaces are provided to control and manage each VIM. In each VIM under Tacker's control a plugin, called Tacker slave, is installed to exchange messages with Tacker master. Each Tacker slave is installed into each VIM and is responsible for monitoring the resource status in the controlled VIM. Therefore, RMM in the 5GT-SO SST implementation can communicate with Tacker Master to get resource information from each Tacker slave via the SBI monitoring interface, e.g., to get the resource consumption for a specific VIM or for all VIMs. Then, the resource consumption can be passed to PAM for deployment calculation of the requested NFV-NS. This Monitoring Platform is to use the interface/plugin provided by Tacker to get the status of resource consumption.

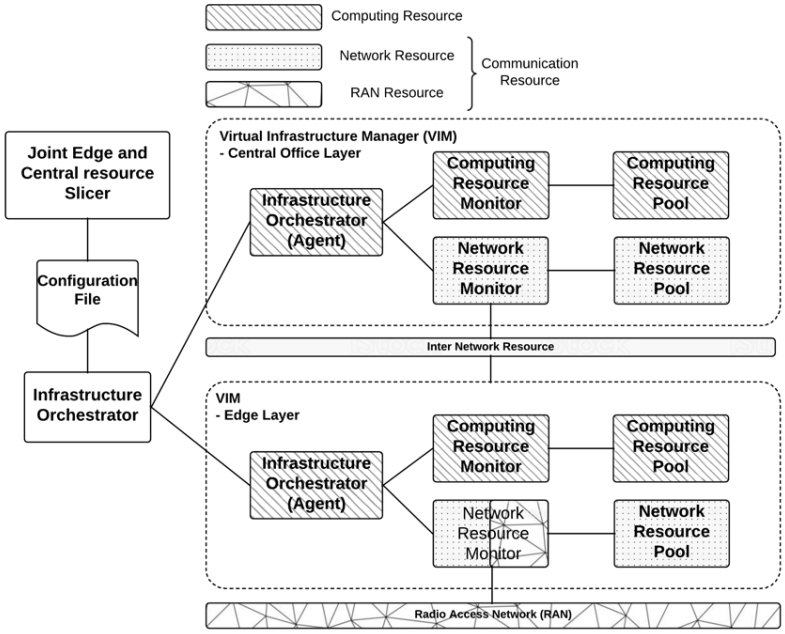
7.4 Two-tier resource allocation scenario

To release the 5GT-SO SST and 5GT- VS SST, the Joint Edge and Central Resource Slicer (JECRS) is developed. The JECRS system contains components of 5GT-SO SST and 5GT- VS SST. The JECRS framework consists of five key elements: an Operations, Administration, and Maintenance (OAM) portal, a resource coordinator, a template database, a configuration file maker and an infrastructure orchestrator communicator. OAM is the implementation of 5GT-SO SST. The OAM portal is implemented as a web user interface (UI) and comprises two modules, namely a slice management module which plays as RRM and a user requirement translator module which plays as the PAM.

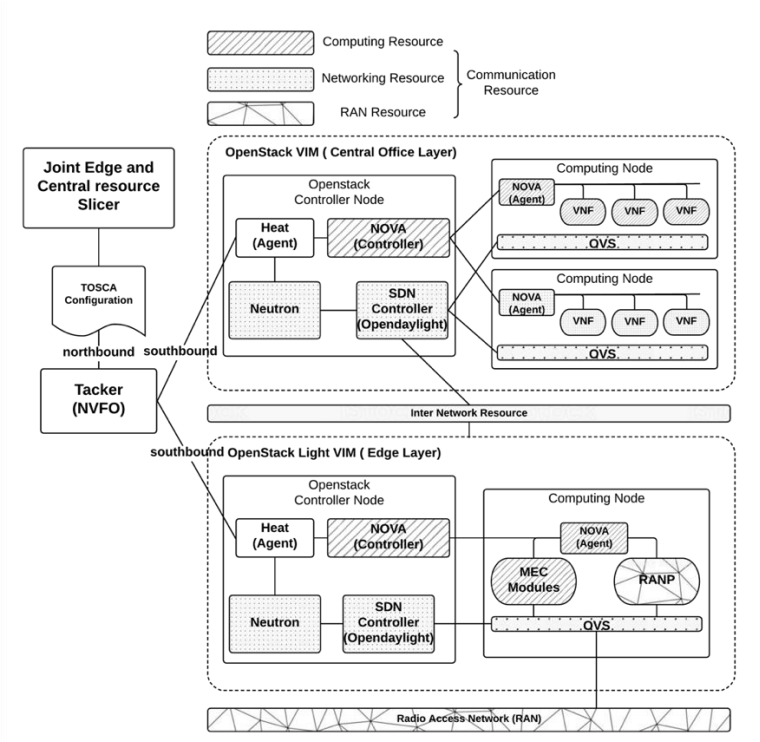
To emulate a real-life scenario, we consider a 2-tier resource allocator that determines the resources to be provided by each tier of the MEC infrastructure. In allocating the resources, the resource allocator chooses one of several pre-configured settings of a resource separation ratio, where this ratio determines the proportion of traffic to be handled by the central office and edge, respectively. The resource allocation result is passed to the configuration file maker provided from VS SST, which creates a corresponding network service descriptor (NSD) with a format dynamically determined by the infrastructure orchestrator. The NSD is then transferred via the infrastructure orchestrator communicator to the orchestrator (e.g., Tacker, OpenMANO, or Cloudify). Upon receiving this NSD, the orchestrator deploys the corresponding slice to the tenant such that they can build their service.

Moreover, Tacker only provides basic life-cycle functions of VNF, which are create, update, and delete. More details of Tacker operation can be found in [37].

Figure 38 illustrates the interface between the proposed JECRS framework and the 2-tier MEC architecture. As shown in Figure 38 (a), the resource orchestrator of the MEC architecture connects to two separate VIMs, where each VIM manages the computing and communication resources of the respective tier. In both tiers, the computing resources are monitored continuously and their status is regularly reported to the resource orchestrator agent of the tier. The agents then report this information back to the resource orchestrator to facilitate the JECRS procedure. Regarding the communication resources, the upper tier controls both the transport network between the two tiers and the intra network in the central office. Similarly, the lower tier controls the intra network in the mobile edge and the RAN sharing between the mobile edge and the user equipment. For both tiers, the communication resources are controlled by an SDN controller within the respective VIM. The controllers periodically report the network resource status to the resource orchestrator agents, which then forward this information to the main resource orchestrator of the MEC architecture. Upon receiving an NSD from the JECRS framework, the orchestrator passes the logical resource allocation to the VIMs, which then map this information to the physical resource pools in order to set up a slice for the corresponding service.



(A) 2-TIER MEC INFRASTRUCTURE WITH COMPUTING AND COMMUNICATION RESOURCES



(B) 2-TIER INFRASTRUCTURE IMPLEMENTATION WITH OPEN SOURCES

FIGURE 38: Overview of 2-tier MEC infrastructure.

The NFVO of the 2-tier infrastructure was implemented using Tacker (see Figure 38 (b)). Moreover, the VIMs in the upper and lower tiers of the architecture were implemented using OpenStack, while the south-bound communications between the NFVO and OpenStack were handled using heat agents [32]. The computing resources were controlled using NOVA; a built-in element in OpenStack. The network resources were controlled by a SDN controller, namely OpenDaylight (ver. Oxygen) cooperating with Neutron [33]. OpenDaylight controls the transport inter-network between the two tiers and the intra-network in the upper tier using Open vSwitch (OVS v2.8.4) [34]. Neutron manages the intra network of the mobile edge based on the OVS of the lower tier and cooperated with a RAN proxy (RANP). Neutron management is a virtual Mobility Management Entity and multiple virtual base station modules are set up inside to process the control plane and then analyze the Non-access Stratum (NAS) messages and identify the operator of each user equipment. Besides, for the date plane, a Tunnel Switching module processes incoming GTP data packets and forwards them to the correct slice, to control the RAN. Finally, the north-bound communication between Tacker and the JECRS framework were handled using TOSCA (Topology and Orchestration Specification for Cloud Applications) [35].