The Vertical Slicer: Verticals’ Entry Point to 5G Networks

Claudio Casetti, Carla Fabiana Chiasserini
Politecnico di Torino, Italy

Thomas Deiß
Nokia Solutions and Networks, Germany

Jose Enrique González Blázquez
ATOS, Spain

Giada Landi
Nextworks, Italy

Josep Mangues-Bafalluy
Centre Tecnologic de Telecomunicacions de Catalunya, Spain

Jorge Martín-Pérez, Nuria Molner
IMDEA Networks Institute and
Universidad Carlos III de Madrid, Spain

Cao-Thanh Phan, Farouk Messaoudi
BCOM, France

Nicolás Serrano
Telefonica, Spain

Charles Turyagyenda
InterDigital, UK

Abstract— **Network slicing is the emerging framework for the support of services offered by vertical industries in 5G networks. In order to ensure that the service characteristics required by a vertical are met, it is of paramount importance to adequately interpret the vertical’s requirements and map them onto specifications that the system orchestrator can use to deploy the service. In this paper, we take this challenge and present a new entity, called Vertical Slicer, to be incorporated within the OSS/BSS of 5G networks. We describe the architecture of the Vertical Slicer, as well as its main functional blocks and the algorithms therein.**

Keywords—5G system architecture; vertical service support; network services; algorithms

I. Introduction

It is envisioned that upcoming 5G networks will offer an unprecedented degree of integration and support for the operational requirements of vertical industries (car manufactures, media & entertainment industries, e-health service provides…). Indeed, one of the strong suits of 5G networks is the capability of tailoring the infrastructure to satisfy such higher-layer requirements, leveraging the softwarization and virtualization of the infrastructure. Unfortunately, the performance improvement achieved with reconfigurability and flexibility of virtual networks entails an increased management complexity, which is often outside the skills of many verticals. It is thus necessary to devise simplified, yet effective, means that allow the verticals to interact with the provider network, so as to enable an effective deployment of the services that the verticals intend to offer to the mobile end users.

The main tool that enables this vision is the concept of network *slice*. In simple terms, a slice is defined as a softwarized, programmable, dedicated logical infrastructure equipped with applications and virtual functions in support of vertical or network services. A slice can span one or more (possibly federated) administrative domains, using their networking and computing resources in order to meet the Service Level Agreement (SLA) established between the vertical and the network provider. Once instantiated, a slice can nominally expose its capabilities through a set of APIs that allow the vertical to monitor (and even operate) the slice.

In order to streamline the creation of slices tailored to vertical SLAs, a crucial piece of the 5G architecture is the *Vertical Slicer,* to be integrated in the Operation and Business Support System (OSS/BSS) of the administrative domain of a provider. In the following, the Vertical Slicer will be referred to as 5GT-VS, as its definition is one of the main tasks undertaken by the 5G-TRANSFORMER project [1] [2].

In this paper, we first describe the purpose and the architecture of the 5GT-VS, in, respectively, Section II and Section III. We then introduce its inner functionalities, namely, the vertical service mapping in Section IV, the arbitration in Section V, and the service/slice management and monitoring in Section VI and Section VII, respectively. Finally, we draw some conclusions and future developments of the 5GT-VS, in Section VIII.

II. Vertical Slicer Overview

The 5G-TRANSFORMER project [1] explores how network slicing can help verticals and mobile (virtual) network operators (M(V)NO), acting as customers, to deploy their services more quickly. The project aims to ease the definition of services and their deployment by the verticals without requiring knowledge of orchestration. The system hides unnecessary details from the verticals, allowing them to focus on the definition of the services and the required SLAs. The *vertical slicer* (5GT-VS) is a key component to achieve this simplification. Other key components are the *service orchestrator* (5GT-SO) [3], responsible to orchestrate the network services and to federate with other

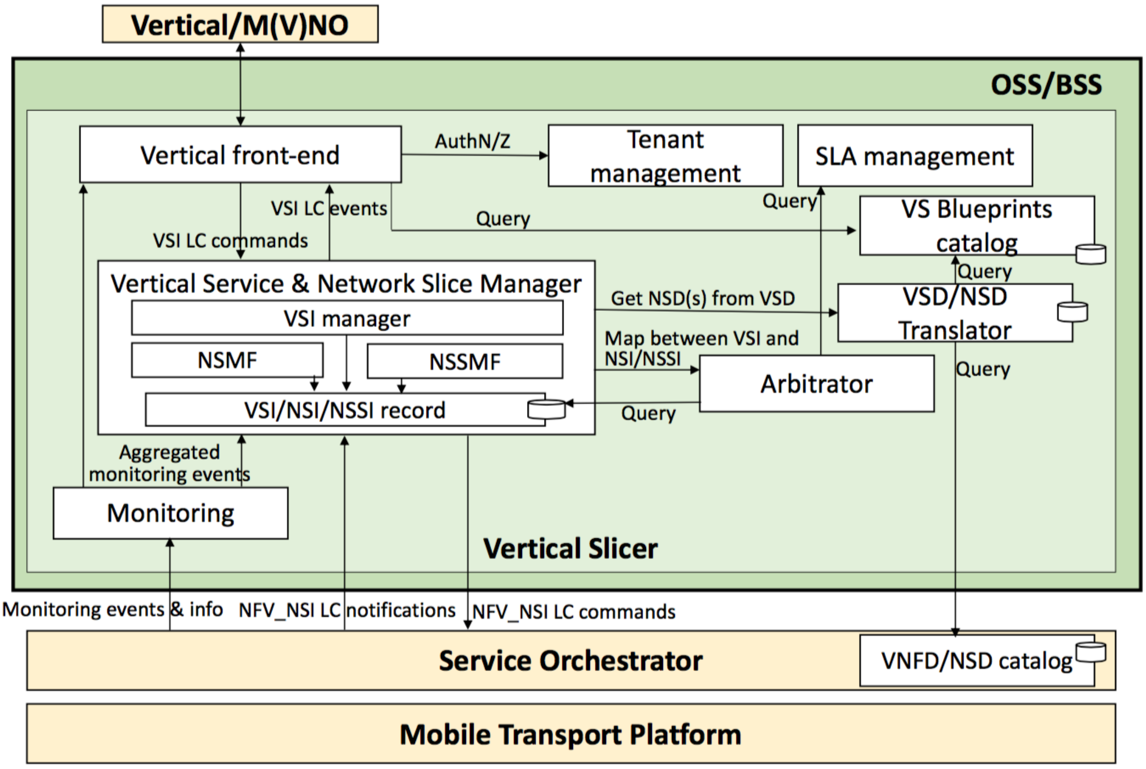

Figure 1: The 5G-TRANSFORMER logical architecture, and internal architecture of the Vertical Slicer

infrastructure providers, and the *mobile transport and computing platform* (5GT-MTP) [4], providing the underlying physical mobile transport network, computing and storage infrastructure and orchestrating the resources of this infrastructure. Figure 1 provides a global view of the 5G-TRANSFORMER logical architecture, showing the location of the 5GT-VS therein.

The 5GT-VS is the common entry point for all verticals into the 5G-TRANSFORMER system, being part of the OSS/BSS of the administrative domain of a provider. The 5GT-VS coordinates and arbitrates the requests for vertical services. Vertical services are offered through a high-level interface focusing on the service logic and needs of vertical services. It allows composing a vertical service from a set of vertical-oriented service blueprints, which, along with instantiation parameters, results in a vertical service descriptor (VSD). The 5GT-VS maps vertical service descriptions and requirements on vertical application level to a network slice. Network slices are implemented as ETSI NFV network services (NFV_NS) and are described correspondingly by network service descriptors (NSD) [5] [6]. NSDs define forwarding graphs composed of a set of virtual (network) functions (V(N)F) connected with each other with virtual links. The 5GT-VS allows to map, possibly several, vertical service instances to one network slice, taking care of necessary capacity changes. Vice versa, one vertical service instance may be mapped to several network slices. In summary, the 5GT-VS provides the functionality for (i) creating the VSDs from service blueprints, (ii) mapping VSDs to NSDs (iii) managing the lifecycle of vertical service instances, and (iv) managing the corresponding network slice instances.

In addition to handling vertical service and network slice instances, 5GT-VS provides arbitration among several service instances of the same vertical, as well as of different verticals. In particular, the arbitration component in the 5GT-VS maps priority of services and SLA requirements to ranges of cardinalities of resources. These cardinalities are then used by the service orchestrator, e.g., in case of actual shortage, to assign resources to the most important vertical service instances.

III. Vertical Slicer Architecture

The architecture of the 5GT-VS is shown in Figure 1. As mentioned, the 5GT-VS is part of the provider's OSS/BSS and interacts with the Vertical (or the M(V)NO) through its North Bound Interface (NBI), and with the service orchestrator through its South Bound Interface (SBI).

Looking at Figure 1, at the top we find the Vertical Front-end, which provides an entry point to receive requests from the verticals/M(V)NO about the service management and monitoring. Importantly, the Vertical Front-end presents the vertical with VS Blueprints stored by the provider in the related catalogue, and supports the vertical in providing the parameters therein, so as to obtain the corresponding vertical service descriptor (VSD).

The component called "Vertical Service and Network Slice Manager" implements the necessary functionality to handle the lifecycle (LC) of Vertical Service Instances (VSIs), and that to handle the Network Slice Instances (NSIs) and Network Slice

Subnet Instances (NSSIs). This module, described in Section VI, also implements the centralized engine of the 5GT-VS: it manages the association between VSIs and NSIs, regulating the sharing of network slices among different vertical services and their decomposition in network slice subnets. Moreover, it also handles the finite state machines for VSIs and NSIs lifecycle, coordinating commands and events associated to both logical entities. The network slice management is handled through requests for instantiation, modification and termination of the corresponding NFV_NSs, interacting with the service orchestrator. The status and the current characteristics of VSIs and NSIs are persistent in the VSI/NSI record.

While the Vertical Service and Network Slice Manager module handles the lifecycle of vertical services and network slices, the main 5GT-VS decision logic is implemented in the VSD/NSD Translator and in the Arbitrator. The former, described in Section IV, selects the NSDs that are able to support the requested vertical services. Furthermore, the Translator identifies the network service Deployment Flavours (e.g., number of VNFs, amount of vCPU or RAM for VNFs, bandwidth of virtual links) [7] most suitable to guarantee the performance and the characteristics defined in the vertical service descriptor. The Arbitrator, described in Section V, makes decisions about the sharing of network slices among different vertical services and the scaling of vertical services based on service priority and resource budget in the verticals' SLAs. Accordingly, the Arbitrator takes final decisions about the Deployment Flavours that will be used at the service orchestrator to instantiate the NFV_NSs.

Finally, the 5GT-VS Monitoring service, described in Section VII, interacts with the service orchestrator to collect monitoring parameters about the established NFV network services and correlates, or aggregates, these data in order to produce metrics and KPIs for network slices and vertical services. These metrics can be used as input for SLA verification, or to make decisions about the lifecycle of a network slice, for example triggering a scale up action in the case of decreasing performance.

The 5GT-VS NBI is based on a REST API and implements the Ve-Vs reference point between the 5GT-VS and the vertical, [6]. In particular, the Ve-Vs reference point provides mechanisms for VS Blueprint queries and operation (e.g., instantiation, termination, queries and update), or monitoring (e.g., queries and subscriptions-notifications) of vertical services.

The 5GT-VS SBI allows the interaction with the service orchestrator. Its definition is based on the ETSI NFV IFA 13 [6], which defines the NBI of an NFVO. The 5GT-VS SBI implements the Vs-So reference point and provides the following functionalities:

- Lifecycle management, for the operation of NFV services. It offers methods to instantiate, terminate, query, update and re-configure services or receive notifications about their lifecycle.
- Monitoring, for the monitoring of NFV network services and VNFs, through queries or subscriptions and notifications about performance metrics and failures.
- Management of catalogues for Network Service Descriptors (NSDs) [5] and VNF Package Descriptors [7], including mechanisms for their on-boarding, removal, updates and queries.

IV. The Translator Module

The Translator module is an entity within the 5GT-VS that translates the verticals' requirements into technical specifications demanded by the service orchestrator, so that a NFV network service can be effectively deployed.
To define a vertical service, or prepare it for deployment, the vertical first selects a VS Blueprint from the corresponding catalogue, and completes it with the requirements it demands. By providing the missing parts in the VS Blueprint, the vertical creates a Vertical Service Descriptor (VSD): a high-level specification of a NFV_NS. The VSD includes the Virtual Network Function Forwarding Graph (VNFFG), which details the needed VNFs to deploy the service, and how they should be connected, i.e., the Virtual Links (VLs) between VNFs.
The VSD is then mapped into a Network Service Descriptor (NSD) through the following steps: (1) the Translator queries the Virtual Network Function Descriptor (VNFD) catalogue for the VNFs referenced in the VSD; (2) the Translator queries the NSD catalogue for the NSDs referenced in the VSD; (3) the Translator fills the fields present in the Deployment Flavour of each VNFD and of each Virtual Link Descriptor (VLD), using the QoS restrictions of the VSD (such fields will be finalized by the Arbitrator later on).

A relevant example of vertical service is the video streaming service. Among the VSD fields the vertical fills in, there are:

- Service type: video streaming
- No_users: K
- Time_availability: 24 h/day – 7days /week.

Upon receiving the VSD, the Translator looks for the VNFDs of firewall, load balancer and streaming servers. Such VNFDs are referenced in the NSD and connected with the VLs composing the VNFFG shown in Figure 2. Some of the VLs and VNFs are backup components to fulfil the availability requirement, while the number of streaming VNF instances is set to a suitable value to satisfy both availability and number of users to serve. With a less stringent availability requirement, the Translator would choose a VNFFG without redundant components and paths. The NSD also specifies the initial values of VNF instances and VLs Deployment Flavours, as well as monitoring and healing fields indicating the start and reconnect mode for VNFs, in the case some of them go down. Then this NSD is given as input to the Arbitrator, in order to finalize the CPU and bandwidth requirements specified in the aforementioned Deployment Flavours.

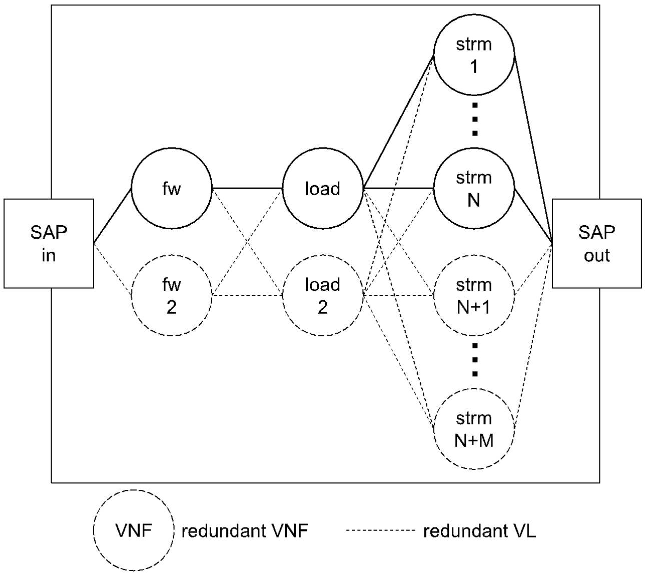

Figure 2: Video streaming VNFFG

V. The Arbitrator Module

SLA management will become a key aspect in the provisioning of services to the vertical on top of 5G networks. Any affection on the SLA negotiated between the provider's OSS/BSS and the vertical, can impact not only on the technical behavior of the service but also on the reputation or business leadership of the vertical itself. Finally, yet importantly, legal consequences could occur depending on the nature of the service being provided. SLAs, then, become more than just sales agreements.

The deployment of a vertical service is a multi-dimensional problem, where various objectives, resource types and autonomous providers can be involved. Thus, there will be interdependencies in the different classes of resources to be provisioned and configured.

The vertical, as consumer, will specify some Service Level Objectives (SLOs) adapted to its service need. Examples of SLOs can be, e.g., 2s of response time from the system. On the contrary, the slice provider can state different Service Level Agreement Templates (SLATs), representing distinct service level classes and guarantees on resource availability. The convergence from SLOs desired by the vertical and SLATs offered by a provider will define an SLA. The contents and terms of an SLA will be inevitably used as concrete directives for configuration and orchestration of services and resources in the provisioned slice. A slice operation is effective if an acceptable trade-off is found between satisfying the terms in the SLA and satisfying the rest of the service requests in the system. It is here where arbitration mechanisms are required for managing such trade-offs.

The Arbitrator within the 5GT-VS is the entity where these mechanisms are implemented. It operates based on the SLA, as well as on the information on each service provided by the vertical, namely: (1) the service priority level, (2) the VNFFG representing the service, (3) the relative CPU requirements of the VNFs in the VNFFG, as well as their memory/storage requirements, and (4) the vertical's quality of service (QoS) requirements (e.g., end-to-end latency, service availability or reliability level). Note that such information is described in the NSD created by the Translator for the received VSD.

We consider that verticals can be categorized into different classes based on their SLAs, e.g., Gold, Silver, Bronze verticals. The 5GT-VS will thus process the service requests by verticals according to such classification. With regard to services by the same vertical instead, the Arbitrator's task is to determine the Deployment Flavours associated with each service, such that the vertical's QoS requirements are met, while accounting for the services priority level. Note that, upon the request of a new service by the vertical, the Arbitrator may need also to update the Deployment Flavours associated with previously allocated services.

We envision that the Arbitrator acts as follows. Let us denote by C, B, M and S the total amount of, respectively, CPU, bandwidth, memory and storage that the vertical's services can use per SLA. First, the Arbitrator orders the services that the vertical wants to deploy, or it has deployed, from the highest priority level to the lowest. It then considers the highest-priority service, say, s, and allocates memory and storage based on the needs exhibited by the VNFs in the VNFFG representing s. A more complicated procedure is instead required for CPU and bandwidth allocation, since this depends on the VNF placement decisions made by the service orchestrator. For clarity of presentation, let us focus on the service latency as the main performance metric, and denote by D_s the maximum latency that service s is allowed to experience. Two components contribute to the service latency: (i) the processing time, due to the execution of the VNFs in the VNFFG, and (ii) the network travel time, which is due to the time needed to transfer data from one VNF to the next in the VNFFG, when adjacent VNFs are deployed in different servers. While the former depends on the CPU allocated to the VNFs execution, the latter depends on the deployment decisions made by the service orchestrator, and on the bandwidth associated with the virtual links (VL) connecting the servers when the service orchestrator resorts to different servers for the implementation of the VNFs.

In light of the above considerations, the Arbitrator considers the best and worst possible cases that may occur depending on the service orchestrator's later decisions. The best case happens when all VNFs in the VNFFG, whose set is denoted by V, are deployed within the same server. In this case, the bandwidth required for data transfers over VLs for service s, β^b, can be set to zero, while the allocated CPU, μ^b, can be computed so that:

$$\sum\nolimits_{v\in V} \frac{1}{f_v\mu^b-\lambda_v} \le D_s \quad \text{and} \quad \mu^b \le C \tag{1}$$

where, for simplicity, we neglected the delay due to memory/storage access. In (1), f_v is the relative computational requirement of VNF v, such that $\sum_{v\in V} f_v = 1$; moreover,

- the first inequality imposes that the latency experienced by the service does not exceed the maximum value. The total latency is given by the sum of the latency contributions due to all VNFs in the VNFFG. Each contribution is computed by modelling the generic VNF v as a FIFO M/M/1 queue with $f_v \mu^b$ as the output rate and λ_v as the service request rate input to v;

- the second inequality imposes that the CPU allocation does not exceed the vertical's CPU budget C.

Conversely, in the worst case each VNF in V is deployed in a different server, thus implying the need for bandwidth allocation. Specifically, the allocated CPU, μ^w, and bandwidth, β^w, now should satisfy the following conditions:

$$\sum_{v\in V} \frac{1}{f_v \mu^w - \lambda_v} + \sum_{(u,v)\in E} \frac{d_{v,w}}{f_{u,v} \beta^w} \leq D_s \qquad (2)$$

$$\mu^w \leq C \; ; \;\; \beta^w \leq B \qquad (3)$$

where $f_{u,v}$ is the relative bandwidth requirement for the VL connecting VNFs u and v. Also,

- (2) accounts for the latency due to both the VNF execution and the travel time over the VLs connecting any two adjacent VNFs (E denotes the set of edges in the VNFFG);
- (3) imposes that both the total CPU and bandwidth allocations do not exceed the corresponding budget available to the vertical.

Once the Arbitrator has determined the values (μ^b,0) and (μ^w, β^w) for the tagged service, it proceeds with the second service in the list, following the same steps as above but using the remaining budget available to the vertical in terms of CPU (C) and bandwidth (B). The Arbitrator can then determine the corresponding per-VNF and per-VL values based on the relative "weight" that, respectively, VNFs and VLs have in the NSD initial setting. The Arbitrator will thus update the Deployment Flavours in the VNF Descriptors (VNFDs) and in the Virtual Link Descriptors (VLDs) of the NSD created by the Translator, by using the worst-case values as default Deployment Flavours and the best-case values as optional. The Arbitrator also defines per-service auto-scaling rules that the service orchestrator will follow to automatically adjust the CPU and bandwidth based on the real resource utilization, which may be affected by the allocation decisions taken by the service orchestrator during the deployment phase. The updated NSD is then returned to the Vertical Service & Network Slice Manager, which will send it to the service orchestrator. After the service orchestrator has taken the deployment decisions, it will notify the 5GT-VS about the current resource allocation (e.g., in terms of characteristics of the instantiated VNFs) so that the Arbitrator can compute the new budget available to the vertical. It is clear that, in case of resource shortage, some lower-priority services may not be accommodated. Under these conditions, it is also important to stress that the Arbitrator will accommodate services according to the SLA between the vertical and the OSS/BSS, favouring, e.g., golden over silver verticals. In particular, to ensure that verticals belonging to the same category are treated fairly, the Arbitrator will select the vertical to be processed first, v, with probability:

$$P(v) = \frac{1}{|I_v|} \qquad (4)$$

where I_v is the set of verticals in the considered category. Additionally, the verticals that have been already extracted, are removed from I_v until all verticals of that category have been served at least once. This guarantees that random selection does not benefit a vertical over another, e.g., that a vertical is selected twice while others not even once.

At last, it is worth noticing that, thanks to the Arbitrator module, the service orchestrator can make deployment decisions meeting the vertical's indications even if (i) it is unaware of higher-layer information like the SLA between the vertical and OSS/BSS, and (ii) the total amount of available resources is not sufficient to adequately deploy all requested services.

VI. The Vertical Service and Network Slice Manager

The Vertical Service (VS) and Network Slice (NS) Manager is the module that is responsible for the sharing of network slices among different vertical services and their decomposition in network slice subnets. It also manages the lifecycle of VS instances (VSI), NS instances (NSI), and their nested Network Slice Subnets (NSSI). Finally, it interacts with the service orchestrator to control the service instances. The control consists in collecting the state of the NSIs (metrics, alarms, and usage records) so that the expected performance and capacity can be verified, and charging can be put in place in agreement with the pricing schemes.

Focusing on the Network Slice Manager, this is composed of two management functions, namely, Network Slice Management Function (NSMF), and Network Slice Subnet Management Function (NSSMF), as defined in [8] [9].

VII. The Monitoring Module

The 5GT-VS monitoring component collects performance metrics or failure events about VNFs or services interacting with the monitoring module of the service orchestrator, and it correlates or aggregates these data generating monitoring parameters related to network slices or vertical services.

Examples of performance metrics retrieved from the service orchestrator are the amount of vCPU and RAM consumed in a time interval by a specific VNF, or by an entire service, the number of packets lost on a virtual link, or VNF-specific indicators. The correlation between these data allows the 5GT-VS monitoring module to elaborate information associated to network slices and vertical services, based on monitoring rules that may be encoded in vertical service descriptors or network service descriptors. The monitoring information and performance metrics produced by the 5GT-VS monitoring component can then be used internally at the 5GT-VS, in order to determine the status and the performance of the managed entities, i.e., network slices and vertical services.

In more detail, the monitoring module provides the inputs that inform and influence management operations such as configuration management (CM), fault management (FM), security management (SM), performance management (PM) and life cycle management (LCM). The module gathers the monitoring information by querying SO Monitoring Service for performance parameters. The derived monitoring information is subsequently conveyed to the vertical front-end and the Vertical Service & Network Slice Manager modules of the 5GT-VS.

In the context of fault management and security management, the monitoring module receives an abstracted view of the following types of notifications: communications alarm, processing error alarm, environmental alarm, quality of service alarm, integrity violation, equipment alarm, operational violation, physical violation, security service or mechanism violation and time domain violation [10]. In the context of lifecycle management, the monitoring module listens for notifications pertaining to the following events: network service and/or VNF instantiation; network service and/or VNF scaling; network service and/or VNF updating, e.g., change of the Deployment Flavour of the VNF instance; network service and/or VNF termination [11]. Additionally, VNFFG lifecycle events are monitored, e.g., create, query, update and delete. In the context of performance management, the monitoring module requests and/or receives notifications related to the usage of the virtualized resources, e.g., vCPU power consumption, VM memory usage oversubscription, VM disk latency, etc. [12]. This kind of information can be used for continuously validating the SLAs for the active service and detect potential SLA violations, to feed the Arbitrator's decisions and, where needed, to trigger actions related to the lifecycle management of vertical services or network slices (e.g., scaling or healing procedures). Moreover, the 5GT-VS monitoring service exposes a NBI towards the verticals, supporting monitoring queries or notifications for authorized consumers.

VIII. Conclusions

We presented the concept and the design of the Vertical Slicer, an entity aimed at enabling the mapping between the vertical's requirements and the specifications that should be followed by the system orchestrator while deploying the vertical's services. We detailed the internal architecture of the Vertical Slicer, its functional blocks, and its logic, which is included in the Translator and Arbitrator modules.

Our future work within the 5G-TRANSFORMER project will aim at refining the algorithms representing the logic of the Vertical Slicer, e.g., by defining how the auto-scaling rules should be set at the Arbitrator for the different services. Importantly, we will implement the Vertical Slicer and verify its ability to efficiently support the services offered by various vertical industries (automotive, content delivery, e-health and robotics), as well as its interaction with the service orchestrator.

IX. Acknowledgment

This work has been partially funded by the EU H2020 5G-TRANSFORMER Project (grant no. 761536).

References

[1] The 5G-TRANSFORMER project, 5G-transformer.eu.

[2] C. Casetti et al., "Network Slices for Vertical Industries," IEEE WCNC Workshop Compass, Barcelona, Spain, Apr. 2018.

[3] Xi Li et. al., "Service Orchestration and Federation for Verticals", IEEE WCNC Workshop Compass, Barcelona, Spain, Apr. 2018.

[4] P. Iovanna, "5G Mobile Transport and Computing Platform for Verticals," WCNC Workshop Compass, Barcelona, Spain, Apr. 2018.

[5] ETSI GS NFV-IFA 014, V2.3.1, Management and Orchestration, Network Service Templates Specification, 2017.

[6] ETSI GS NFV-IFA 013, V2.3.1, Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification, 2017.

[7] ETSI GS NFV-IFA 011, V2.3.1, Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; VNF Packaging Specification, 2017.

[8] ETSI EVE 12, Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework.

[9] 3GPP TR 28.801 V15.1.0, Study on management and orchestration of network slicing for next generation network, Rel. 15, 2018.

[10] ITU-T Recommendation X.733 (02/92): "Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function".

[11] ETSI GS NFV-IFA 008 (V2.1.1): "Network Function Virtualisation (NFV); Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specification".

[12] ETSI GS NFV-IFA 006 (V2.1.0): "Network Function Virtualisation (NFV); Management and Orchestration; ViVnfm Reference Point - Interface and Information Model Specification".