Network Slices for Vertical Industries

Claudio Casetti, Carla Fabiana Chiasserini
Politecnico di Torino, Italy

Pantelis A. Frangoudis, Adlen Ksentini
EURECOM, France

Xi Li
NEC, Germany

Nuria Molner
IMDEA Networks Institute and Universidad Carlos III de Madrid, Spain

Thomas Deiß
Nokia Solutions and Networks, Germany

Giada Landi
Nextworks, Italy

Josep Mangues
Centre Tecnologic de Telecomunicacions de Catalunya, Spain

***Abstract*—Network Slicing allows to simultaneously support the specific needs of vertical industries with a diverse range of networking and computing requirements. Network Functions Virtualization (NFV) has been defined to deploy multiple network services on a common infrastructure. We extend the NFV concept to vertical services, i.e. services implemented on top of network services and providing the applications of the verticals. We present a component of the 5G-Transformer system, named vertical slicer, which acts as the interface to verticals. The vertical slicer has two main functionalities: allowing verticals to define vertical services based on a set of service blueprints and arbitrating among several vertical services in case of resource shortage.**

Keywords—network slicing, vertical service, arbitration

I. INTRODUCTION

Networking slicing is an inherent concept in the definition of 5G networks. Three slice types are supported: enhanced mobile broadband (eMBB), ultra-reliable low-latency communication (URLLC), and massive IOT (mIOT) [1]. There can be several slices of each type and a UE can signal when establishing a PDU session to which slice this should be connected. The network functions in a slice can be deployed differently depending on the requirements of the service. E.g., an eMBB core network function, such as a user plane function (UPF), can be deployed in a central cloud to increase scalability, whereas for a slice supporting URLLC the UPF can be deployed in an edge cloud to reduce latencies. For verticals with different needs different network slices can be provided for each of the slice types.

To ease operation of the slices this should be automated as much as possible. Ideally, a vertical itself would define its vertical service as virtual functions (VF) connected by virtual links (VL) to a forwarding graph (FG). Once a vertical service is defined, the vertical should be able to trigger its instantiation on an infrastructure, monitor it while it is operating, update it, and eventually terminate it. All these operations should be possible without detailed knowledge of the infrastructure, service orchestration, etc. Ultimately, a new service could be rolled out within minutes or hours as compared to weeks or months if manual operation is needed.

In this paper we focus on defining services and arbitrating among them in case of resource shortage, assuming that orchestrators and managers for virtual functions and for the infrastructure are in place. In Section II we present an overview of the 5G-Transformer system and relate it in Section III with the ETSI NFV framework. In Sections IV and V we present the main functionalities of the VS and in Section VI we present conclusions.

II. 5G-TRANSFORMER SYSTEM

The 5G-Transformer project [2] 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 Service Level Agreements (SLAs). The system will also allow infrastructure providers to share the 5G mobile transport and computing infrastructure efficiently among verticals and M(V)NOs.

We envision a system of three major components: *vertical slicer* (VS), *service orchestrator* (SO) and *mobile transport and computing platform* (MTP), see Figure 1.

The VS is the common entry point for all verticals into the 5G-Transformer system, being part of the operating and business support systems (OSS/BSS) of the administrative domain of a provider. The 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 vertical services from a set of vertical-oriented service blueprints, which along with instantiation parameters will result in a vertical service instantiation request. Then, the VS maps vertical service descriptions and requirements on vertical application level onto a network service descriptor (NSD), which is a service graph composed of a set of virtual (network) functions (V(N)F) chained with each other and fine-grained instantiation parameters (e.g., deployment flavor) that are sent to the SO.

This work has been partially funded by the EU H2020 5G-Transformer Project (grant no. 761536). * Corresponding author email: thomas.deiss@nokia.com

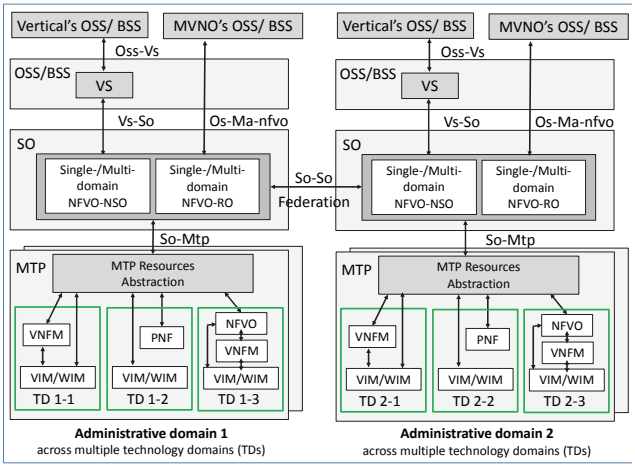

Figure 1: 5G-Transformer basic system architecture

The SO [3] provides end-to-end orchestration of services accross multiple administrative domains by interacting with the local MTP and with the SOs of other administrative domains. It receives requests from the VS or directly from the M(V)NO. Depending on the use case, both network service (NFV-NSO) and resource (NFVO-RO) orchestration may be used for single and multi-domains [6]. Based on the request, the SO may decide to create a new network slice instance or to reuse one previously created to be shared. Therefore, it manages the monitoring and allocation of virtual resources to network slices, be it for vertical services or for network services of an M(V)NO. If needed (e.g., not enough local resources), the SO interacts with SOs of other administrative domains (federation) to take decisions on the end-to-end (de)composition of virtual services and their most suitable execution environment. Even if a service is deployed across several administrative domains, e.g., if roaming is required, a vertical still uses one VS to access the system, and so, the SO hides this federation from the vertical. The NFVO-RO functionality of the SO handles resources coming from the local MTP (real or abstracted) and from the SOs of other administrative domains (abstracted). The orchestration decision for creating or updating a network slice includes the placement of V(N)Fs as well as the resources to be allocated. The SO will then request the MTP to create the slice instance.

The MTP [4] is responsible for orchestration of resources and the instantiation of V(N)Fs over the infrastructure under its control, as well as managing the underlying physical mobile transport network, computing and storage infrastructure. This is done through the MTP resource abstraction building block, which in turn acts as end-to-end resource orchestrator for the various technology domains of the MTP. The computing and storage infrastructure may be deployed in central data centers as well as distributed, as in Multi-Access Edge Computing (MEC) [5]. The MTP provides support for slicing, enforces slice requirements from the SO and provides physical infrastructure monitoring and analytics services. Depending on the use case, the MTP may offer different levels of resource abstraction to the SO via the MTP resources abstraction component, which in turn forwards the SO requests to the right entity accordingly (as single point of contact): Virtual/WAN Infrastructure Manager (VIM/WIM), VNF Manager (VNFM) Physical Network Function (PNF), or NFV Orchestrator (NFVO) [6].

1) The MTP exposes virtual resources and the possibility to instantiate entire VNFs through the VNFM.
2) The MTP exposes PNFs that can be configured but not instantiated (e.g. a physical BTS). At the VIM/WIM level the MTP instantiates virtual networking resources.
3) The MTP abstracts an entire network service to the SO and it takes care internally about how to orchestrate it, through the NFVO – VNFM - VIM/WIM stack.

III. Vertical Slicer in Relation to ETSI NFV

In the ETSI NFV architecture [6] the VS acts as a client of the NFVO, where NFVO functionalities can be mapped onto the SO component. The VS can be considered as an internal function of the OSS that helps the vertical to request and manage its services, mediating the interaction with the NFV management and orchestration (MANO) platform.

The VS provides the vertical with an interactive interface to access a service blueprints catalogue and a programmable interface that simplifies the instantiation, monitoring and operations of services, using a technology- and resource-independent information model. The two main algorithmic blocks that we envision within the VS are the *NSD selector* and the *arbiter*.

The *NSD selector* maps vertical service descriptors, based on application level requirements, to NFV-NS descriptors (NSD). These NSDs are selected from the catalogue shared between VS and SO. The NSD format follows the specification of Network Service Templates [8], where the Network Service is defined in terms of VNFs and/or PNFs, virtual links among them and VNF Forwarding Graphs (VNFFG) for traffic steering. Similarly, VNFs are described following the VNF Packaging Specification [13]. The *NSD selector* translates the requests from the verticals into lifecycle actions to be performed on the less abstract entities of the ETSI NFV Network Services (NFV-NS). In the mapping between service descriptor and NSD, the application level requirements are translated in a resource-centric view (e.g. number of VNF instances, vCPU and RAM of a VNF instance, QoS properties of a virtual link, etc.). The resource-related aspects of the service are then managed entirely at the NFVO in the SO, without the need to know the details of the service logic that is kept hidden at the VS level.

The *arbiter* component arbitrates among services competing for resources. As a result of arbitration, the VS requests operations on the NFV-NS instances that may involve the reconfiguration of VNFs or VNFFGs or the scaling of VNFs. These reconfigurations are still within the limit of the configurable parameters and maximum/minimum number of VNF instances and sizes declared in the VNF descriptors. However, this kind of decisions is triggered by service-related events only. On the other hand, resource-related decisions, like scaling out due to high usage of the vCPU in a VNF, should be fully delegated to the SO. This delegation is usually described in the NSD as a list of "Auto Scaling Rules". The rules are based on values of virtual resource performance

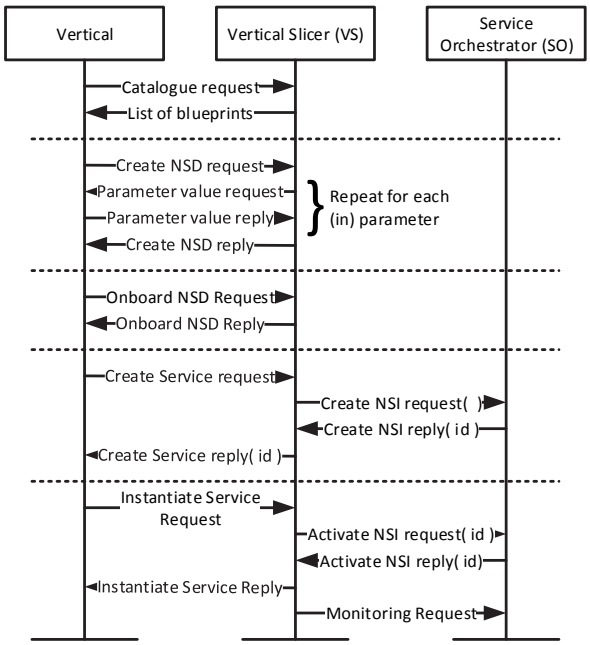

Figure 2: VS workflow

metrics and VNF indicators that the SO needs to monitor, interacting with the VIMs or the VNFs (through their VNFM).

The interface between VS and SO relies massively on the modelling of the Os-Ma-Nfvo reference point [14]. This interface is used by the VS to request the instantiation and termination of the NFV-NS instances that implement the services requested by the verticals. The VS can also request additional operations during the lifecycle of an NFV-NS instance, when triggered by service level events like an SLA update, the need to share an existing NFV-NS instance with a new vertical service or potential conflicts among several services belonging to the same vertical and competing for a limited set of resources declared in the SLA.

The same Os-Ma-Nfvo reference point is also used to exchange monitoring information between SO and VS. The list of monitoring parameters to be collected at the SO is specified in the NSD and VNF Descriptor (VNFD). Part of this monitoring data can be used internally within the SO, e.g. as input for the auto-scaling decisions. However, the VS may also request to receive monitoring data, aggregated at the NFV-NS level, using on-demand queries or subscriptions for threshold-based events. The Os-Ma-Nfvo provides two dedicated interfaces for monitoring issues, one for performance and one for fault management.

IV. VERTICAL SERVICE DESCRIPTION

The 5G-Transformer considers vertical services from the automotive, eHealth, and entertainment domains. The vertical services have different requirements e.g. regarding bandwidth, latency, and availability. In addition to the requirements on the services themselves, there are requirements on the interaction among the verticals and the 5G-Transformer system, e.g. the

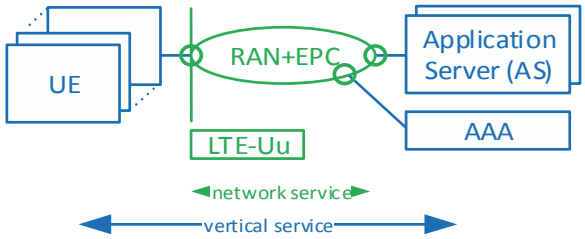

Figure 3: Vertical service blueprint to collect sensor data

possibility to define a priority for a vertical service or to define a common resource budget for several services of one vertical.

Network services are defined as sets of virtual network functions and endpoints, connected by virtual links. Conceptually, VNFs perform network functionality, either in the control or data plane, see [7]. A vertical service is similar to a network service, just relaxing the conceptual restriction of VNFs to networking functions. In a vertical service the virtual functions may perform arbitrary functionality in the application domain. A vertical service may also include the end user devices or applications within them. E.g., in a vertical service in the automotive domain, applications within a vehicle may be part of the service. The end user devices or applications can be considered as physical or virtual functions.

Vertical services can be described similar to networking services by service descriptors. To ease the definition of vertical services for the verticals we propose the use of service blueprints. A blueprint can be seen as a parameterized service definition where information such as coverage area, amount of devices to be served, or specific virtual machine images, still have to be provided. A vertical can select from a catalogue of service blueprints a suitable blueprint for its service, provide the missing information, and let the VS turn this into a service descriptor, which would then be used by the SO for actual deployment of the service, see Figure 2.

A vertical may also request a set of resources, which it manages and orchestrates on its own and, an M(V)NO may define directly a network service. Both use cases are within the scope of the 5G-Transformer project, but are not covered in this paper, in which we focus on the definition of a vertical service by a vertical.

As an example of a vertical service, consider a service to collect monitoring data of sensors in a production plant via LTE. The vertical service consists of the sensors, an application server (AS) to collect the monitoring data, and an AAA server to control whether a sensor is granted access to this service. See the diagram in Figure 3. These functions are connected with a network service with 3 endpoints. This network service represents an LTE radio access network (RAN) and evolved packet core (EPC).

The diagram in Figure 3 can be considered a service blueprint, as it still lacks information to be provided by the vertical, e.g. which virtual machine image is to be used as application server. A vertical service blueprint as a parameterized vertical service descriptor can have a wide range of possible parameters. These parameters are used to

express requirements of the vertical service, but also management related parameters such as file locations of virtual machine images or the priority of a service. A subset of parameters to express requirements, based on a use case analysis of 5G-Transformer are:

- Bitrate of VFs and the connecting links.
- Number of UEs and their traffic volume.
- One-way latency or round-trip time (RTT) among two VFs or a VF and an endpoint.
- Geographical area to be covered by the vertical service, i.e. the location of UEs.

Note, these parameters are different to the parameters which can be given to VNF instances at instantiation time [13]. Such a parameter could be the IP address of an element management system of the vertical, to which the AAA and the application server connect, whereas here we are interested in parameters defining the service itself

The actual values for the parameters in the blueprint are mapped by the VS to a complete service descriptor. This service descriptor is passed to the SO with the usual operations at the Os-Ma-Nfvo reference point [14]. In the sensor data collection example these parameters could be mapped as follows:

- The bitrate of the AS is mapped to a bandwidth requirement of a corresponding virtual link.
- The number of sensors and the message rate is mapped to the necessary amount of processing cores for the AS.
- One-way latency or RTT are not relevant in this specific example. For other vertical services, e.g. remote control applications, this information can be used by the SO in placement decisions for VFs.
- The geographical area is used by the SO to decide which eNbs are needed for this service and need to be connected to the AS and AAA server.
- The virtual machine imageof the AS and AAA servers are needed for later instantiation by the SO.
- The priority information is used by the arbitration function of the VS itself, see Section V, to prioritize among vertical services in case of resource shortage.

We present further examples of parameters and how the system could use them with a second use case, which is related to media distribution and provides a content delivery network (CDN) as a service. In this example, a vertical wants to deploy an eMBB CDN service for HD video distribution to mobile users in several geographically distributed regions.

A CDN service includes virtual components from two categories: (i) Network-level services, and (ii) application services. Network-level services are related with network connectivity and virtual computation and storage resources, and include RAN-level VNFs, e.g. vEPC. Application services implement the CDN service logic and involve virtual (network and other) functions, such as content provider and end user interface modules, origin servers, DNS resolvers, request redirection services, caches, media transcoders, service-level monitoring components, and others.

The vertical can create a CDN service definition by composing virtual functions and network services made available by an operator. The resulting vertical service definition includes specific requirements. The service description is used eventually for service dimensioning and placement decisions, such as which user-plane functions to place at the edge, how much storage space and vCPU resources to allocate per cache/video server, and how much capacity to allocate to virtual links as a function of the targeted end user demand per region. Some input parameters and how they are used are described in the following:

- Targeted regions: Passed by VS to SO to decide to which MEC hosts or points of presence (PoP) cache instances are deployed to.
- Minimum, maximum, and average number of UEs and video streams per region: used by VS in bandwidth definition of links and definition of cores needed. Information on streams per region is also passed to the SO for allocation of cache and video server instances.
- Video resolution and required quality of experience: same as number of UEs and video streams before.
- Minimum service availability: used by the VS to map to different numbers of VNFs as replicas.
- Content origin server information: launch-time configuration of caches to appropriately retrieve content from the vertical's external content servers.

The above input to the VS is also relevant for CDN-service runtime management and arbitration, see Section V. Such decisions cannot typically be taken autonomously by the SO, given that the SO is agnostic to service-specific functionality. At service instantiation time, the VS defines monitoring parameters to be collected by the SO at the resource level and by specialized VFs at the service level. Such VFs are responsible for translating the monitoring data collected to CDN-specific service management actions. For example, the radio network interface service (RNIS) service [16] of the MEC platform can convey per-UE radio channel quality measurements. These can be translated to achievable data rates and used as input to estimate the QoE enjoyed by each user, without needing direct access to the UE platform and application. Combined with the minimum QoE threshold defined by the vertical, and with awareness of the current infrastructure conditions (at the MTP level) as reported by the SO, user-perceived service quality can be estimated, and the root causes of potential QoE degradation can be identified (e.g., poor radio conditions vs. increased workload on specific cache instances). The function responsible for this decision can then either signal the VS arbiter to reconfigure the NSD, with the latter, in turn, requesting the SO to scale out specific VNF instances, or it can instruct the Traffic Rules Control MEC service to redirect traffic for specific users to a transcoder instance running on the edge to reduce the video bitrate to match the current channel conditions.

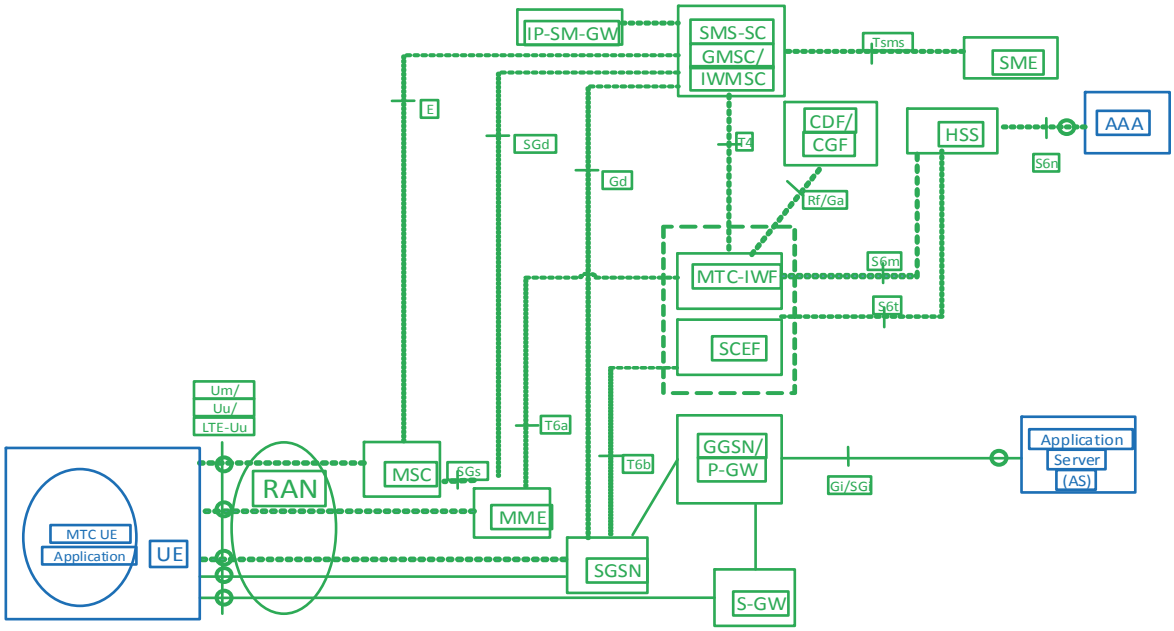

Figure 4: vertical service expanded to network service descriptor

As illustrated by these examples, the parameters provided by the vertical are used in different ways; some can be mapped directly to parameters of the service descriptor, some are mapped to different choices of network services or network service flavors, and some are passed to the SO, e.g. for placement decisions. Blueprints may refer to other VNFs or NFV-NSs, e.g. the vEPC in the sensor data example. The VS expands the refered VNFs or NFV-NSs to a completely described network service. In Figure 4 the EPC part has been expanded as defined in [15]. For the sake of brevity we omitted the expansion of the RAN part, i.e. the eNBs, and an element management system for the VNFs.

The expanded service descriptor, including the mapping or translation of the parameters of the blueprint, is passed to the SO for the actual orchestration of the service. The initial interaction of the vertical with the vertical slicer is depicted in Figure 2. Once the network service has been instantiated, the VS may request monitoring of the service instance and may use this information in arbitration, see Section V.

We plan to describe vertical services in a similar way as network services, using ETSI NFV NSDs [8], or an extension thereof. NSDs might also be expressed in TOSCA. Both notations have been used in literature. [10] proposes a model to describe network services for VNF orchestration leveraging SDN interfaces, which uses ETSI specifications for NFV orchestration and business features between consumers and providers. [11] uses TOSCA notation in OpenStack to orchestrate the deployment of multi-cloud applications on an architecture for the development of multi-component applications across federated cloud providers. [12] uses the ETSI notation to create NSD templates which considers the requirements of tenants and presents an environment to generate NSDs automatically.

So far, we have considered parameters of blueprints, for which the values are provided by the vertical. We envision also the need for out parameters for information provided by the VS to the vertical. As an example, when instantiating a sensor monitoring service, the SO should return, via the VS, the 5G network slice selection assistance information (NSSAI) or the 4G dedicated core network identifier, i.e. a slice identifier at the air interface. The vertical may then use this value to configure the sensors.

V. NSD SELECTION AND ARBITRATION AMONG SERVICES

The NSD selector takes care of mapping the vertical service blueprint into the appropriate NSD. As mentioned in Section IV, the selected NSD will define the VNFFG, deployment flavors and possibly other VNF and VL attributes that meet the vertical's requirements. Such a decision may also account for the performance metrics monitored by the SO and reported to the VS. The arbiter instead arbitrates among services (namely, NSDs) that compete for resources. There are two scenarios where a vertical may not get as many resources as needed. Firstly, no more resource of a specific type is available. Secondly, a vertical may have a resource budget across several of its resources and this budget is exhausted. In both cases, the VS arbitrates resources among services requested by the same or by different verticals in order to meet the desired SLAs while not exceeding the resources available or the budget assigned to the vertical.

As an example, consider an automotive vertical requesting both a vehicle Overtaking Assist Service (OAS) and an Improved Mobility Service (IMS). OAS provides a driver with the occupancy state of the road ahead, while IMS provides a driver with a synthetic vision of a, possibly far-away geographical area so that the driver can become aware of the

traffic conditions therein. Both can exploit either videos or Cooperative Awareness Messages (CAMs) sent by neighboring vehicles, i.e., they can work in either video- or CAM-mode. While working in the same mode, the two services may share the same VNFFG, or portions of it; however, OAS has typically higher priority than IMS. Hence, the VS will select two different NSDs for the two services, one allocating computational capacity with higher-priority to the composing VNFs than the other, to ensure that OAS has lower latency and higher reliability. The logic of priorisation is encoded by the VS into the NSDs as far as possible, such that the SO can autonomously and coherently scale the resources allocated to the two services if needed. Similarly, we expect the SO to migrate VNFs to other less-utilized VNFI-PoPs when appropriate and without having to contact the VS.

However, when an emergency occurs in a geographical area, it becomes important to deliver the IMS to all vehicles approaching the area, while overtaking assistance becomes less critical. This causes a swap of the priority levels associated to the two services: accordingly, the VS arbiter will have to update the NSD of OAS and IMS, and request an NSD update to the SO. In particular, the maximum/minimum number of VNF instances and sizes declared in the VNF Descriptors, as well as the "Auto scaling rules" defined in the NSD can be changed in favor of the IMS [14].

Consider now the aforementioned OAS and IMS and assume that the network radio segment is congested due to high vehicle density. In this case, it would be advisable to switch the lower priority service, say IMS, from video-mode to CAM-mode, i.e., to exploiting the vehicles' CAMs instead of the output of vehicle cameras so that no video has to be transferred over radio links. In other words, the VS should update the IMS VNFFGs by terminating some VNFs and adding others. Again, by using the Os-Ma-Nfvo interface [14], the VS arbiter can request the SO to put in place these operations. Conversely, if a specific resource is no longer scarce, the VS can relax or remove previously imposed restrictions and inform the SO about the new settings.

The above actions can be realized by the VS only if the SO alerts the VS about resource shortage and, in case of a resource shortage, which parts of a vertical service is using these resources. Monitoring and reporting by the SO are therefore fundamental operations that need to be implemented. The ETSI framework in [8] foresees that the NSD itself supports the capability to provide the SO with monitoring parameters to be tracked during the lifetime of an NS instance. Specifically, the VS can define the performance metric of interest and the VNFs, or other virtual resources, for which they should be reported.

The 5G-Transformer project will identify the monitoring parameters to be reported by the SO to the VS for different vertical services, and will extend the Os-Ma-Nfvo interface when needed. Importantly, for each vertical service, it will define the monitoring mode to implement, to be selected, among, e.g., periodically, threshold-based, and query-based. Additionally, the project will devise and evaluate techniques to establish when arbitration at the VS should be triggered. On this regard, it should be noted that resolving resource shortage is in the order of seconds. Therefore the SO has to trigger the VS before all resources are used up: e,.g. triggering the VS when 90% of a specific resource is in use, would still allow high-priority services to be scaled out quickly enough.

VI. CONCLUSION

In the paper we presented the VS and its main functionalities, mapping service descriptors and requirements on services to network service descriptors and arbitrating among services. The VS can use the services provided by the SO through the interfaces at the Os-Ma-Nfvo reference point, although we expect that some extensions are needed, e.g. in case of resource shortage to indicate which resource has been used by which part of a service. We plan to extend the VS to cases, where the vertical request more control over the network services or even requests a network slice. In this case the vertical is expected to orchestrate the service and the VFs on its own. The examples used in this paper will be deployed in testbeds to evaluate the proposed VS and the overall 5G-Transformer system.

REFERENCES

[1] 3GPP TS 23.501, v1.2.0, System Architecture for the 5G-System.

[2] 5G-transformer.eu

[3] Xi Li et.al., Service Orchestration and Federation for Verticals, 1st Workshop on Control and management of Vertical slicing including the Edge and Fog Systems (COMPASS), Barcelona, IEEE, 2018

[4] P. Iovanna et.al., 5G Mobile Transport and Computing Platform for verticals, 1st Workshop on Control and management of Vertical slicing including the Edge and Fog Systems (COMPASS), Barcelona, IEEE, 2018

[5] ETSI MEC, [Online], http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing

[6] ETSI GS NFV-MAN 001, V1.1.1, Management and Orchestration, 2014

[7] ETSI GS NFV 002, V1.2.1, Architectural Framework, 2014

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

[9] Topology and Orchestration Specification for Cloud Applications Version 1.0. 25 November 2013. OASIS Standard. http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html.

[10] F. Paganelli, F. Paradiso, M. Gherardelli, G. Galletti, "Network service description model for VNF orchestration leveraging Intent-based SDN Interfaces", Conf. on Network Softwarization (NetSoft), Bologna, 2017, IEEE

[11] G. Tricomi, et.al., "Orchestrated multi-cloud application deployment in OpenStack with TOSCA", Conf. on Smart Computing (SMARTCOMP), Hong Kong, China, 2017 IEEE

[12] S. Mustafiz, N. Nazarzadeoghaz, G. Dupont, F. Khendek, M. Toeroe "A Model-Driven Process Enactment Approach for Network Service Design". In: Csöndes T., Kovács G., Réthy G. (eds) SDL 2017: Model-Driven Engineering for Future Internet. LNCS 10567. Springer

[13] ETSI GS NFV-IFA 011, V2.3.1, Management and Orchestration, VNF Packaging Specification, 2017

[14] ETSI GS NFV-IFA 013, V2.3.1, Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification, 2017.

[15] 3G-PPP GS 23.682, V15.1.0, Architecture enhancements to facilitate communications with packet data networks and applications

[16] ETSI GS MEC 012 V1.1.1, Mobile Edge Computing (MEC); Radio Network Information API, July 2017.