Automated Service Provisioning and Hierarchical SLA Management in 5G Systems

X.Li, C.F.Chiasserini, *Fellow, IEEE*, J.Mangues-Bafalluy, J.Baranda, *Senior, IEEE*, G.Landi, B.Martini, X.Costa-Perez, *Senior, IEEE*, C.Puligheddu, *Student, IEEE*, L.Valcarenghi, *Senior, IEEE*

***Abstract*—Empowered by *network softwarization*, 5G systems have become the key enabler to foster the digital transformation of the vertical industries by expanding the scope of traditional mobile networks and enriching the network service offerings. To make this a reality, we propose an *automation* solution for vertical services provisioning and hierarchical Service Level Agreement (SLA) management. *Service scaling* is one of the most essential operations to adapt the service deployments and resource allocations to ensure SLA fulfilment. Three different scaling levels are addressed in this work: application-, service- and resource-level. We have implemented our solution in a proof-of-concept of a virtualized mobile network platform, spanning over three geographically-distributed sites. To evaluate our solution, we leverage field tests, focusing on *automotive vertical services* comprising a mission-critical application (collision-avoidance) and an entertainment one (video streaming). The results demonstrate the excellent performance of our solution, and its ability to automatically deploy vertical services and ensure their SLAs through different levels of service scaling.**

***Index Terms*—5G, Vertical Services, Service Provisioning, SLA Management, Scaling, Service Orchestration.**

I. Introduction and motivation

5G systems are envisioned to expand the scope of traditional mobile networks to support various vertical services, such as eHealth, automotive, media, and cloud robotics, hence greatly enriching the telecom network ecosystem. In this new scenario, the imperative for telco service providers is to promptly support vertical industries to deploy their services over the 5G systems, fulfill their diverse requirements, and adjust the service deployments to the dynamic users and traffic demands. To this aim, network softwarization becomes a revolutionary technological shift, thanks to Software-Defined Networking (SDN) and Network Function Virtualization (NFV) enabling the dynamic creation of *Network Slices*, i.e., logically independent network partitions over a shared infrastructure. Network slices are provisioned as end-to-end network services composed of a set of interconnected Virtualized Network Functions (VNFs). Importantly, they are created by properly configuring virtual resources (network, compute, and storage), and tailoring them to address the specific requirements of the vertical services (e.g., bandwidth and end-to-end latency). Such a 5G system is beyond providing mobile radio access network (RAN) and core network functions as defined by 3GPP, but more about providing end-to-end network slicing solutions able to support heterogeneous network and RAN technologies for providing communications suitable for individual vertical services.

Despite the advanced development of network slicing solutions leveraging SDN/NFV, how to automatically deploy a vertical service on a reliable network slice in telco's operational networks is still a real challenge in practice. In particular, what is laboured is for network slices to fulfill at any point in time the required service quality, as per the Service Level Agreement (SLA) established between the vertical and the telco provider. Indeed, SLAs specify a set of service business aspects and quality parameters that telco providers have to guarantee to verticals not to incur in penalties, e.g., required bandwidth, end-to-end latency between service endpoints, mean time to service recovery. Since vertical services are deployed and operated over network slices sharing a common infrastructure, some degradation or violation of the slice service parameters may occur that could impact the performance of the vertical service offered to final users, and, hence, could affect the reputation or business leadership of the vertical itself. For this reason, legal aspects are also regulated in SLAs between telco providers and verticals, identifying which party is responsible for reporting service failures or paying fees.

To meet the vertical SLAs, telco service providers need to map and translate high-level SLA business requirements into network slice- and infrastructure-related requirements, which can be actually handled and addressed at the network level. It follows that other agreements at the network operational level have to be generated in cascade between the telco provider and other parties (e.g., network engineering departments, or cloud infrastructure providers). In an open NFV ecosystem, the SLA management is therefore a multi-dimensional provisioning and management problem, where multiple and interdependent aspects need to be addressed. This calls for a coordinated SLA framework accounting for different levels of performance inter-dependency and obligations, namely, at the application, service, and resource level.

Towards these challenges, the EU H2020 5G-PPP 5G-TRANSFORMER (5GT) project [1] has developed an open and flexible 5G transport and computing platform, able to automatically onboard and deploy vertical services. Importantly, this platform can also manage the service life-cycle and the SLAs, so as to fulfill diverse service requirements. It includes:

- a vertical portal to translate vertical service requirements into network slice-related requirements. This portal also

X. Li and X. Costa are with NEC Labs Europe, Germany. J. Mangues-Bafalluy and Jorge Baranda are with CTTC, Spain. C.F. Chiasserini and C. Puligheddu are with Politecnico di Torino, Italy. G. Landi is with Nextworks, Italy. B. Martini is with CNIT, Italy. L. Valcarenghi is with Scuola Superiore Sant'Anna, Italy.

This work was supported by the European Commission through the 5G-TRANSFORMER and 5GROWTH projects (Grants No. 761536 and 856709).

maps vertical services onto network slices, realizing the latter through Network Services, as defined in NFV (NFV-NS);
- a service orchestration layer to manage the NFV-NSs and construct their logical networks. This is achieved by placing and connecting the service components in the virtual infrastructure, and by allocating the required virtual resources;
- an infrastructure layer that not only manages the underlying infrastructure resources but also handles the actual mapping of a logical network onto the shared physical network. It thus realizes the deployment of the vertical services into slices.

The above three layers may be owned and managed by different providers and entities in the real network scenarios, thus a hierarchical SLA management is essential to provide an automated and coordinated vertical service management throughout the whole stack of the system. As part of the SLA management, service scaling is one of the important operations to automatically adapt the service deployments according to (i) mutable needs of the vertical service and application components deployed on a slice (e.g., varying demand of service instances or of total resources required by the vertical services), (ii) the priorities of different vertical services running into the network slice instances, or (iii) real-time availability of (virtual) resources in the infrastructure underpinning the deployed network slices. Along this line, different levels of service scaling are provided at the different layers of the 5GT platform for such hierarchical SLA management.

In this paper, we present the hierarchical SLA management framework that we have designed and developed on top of the 5GT platform (Sec. II), and we introduce the service scaling mechanisms that we have defined at the different levels, namely, application, service, and resource level (Sec. III). We then describe the proof-of-concept test-bed where we implemented the scaling mechanisms (Sec. IV), which, importantly, have been released as open-source software[1]. Finally, we provide a thorough experimental evaluation in the relevant, practical case of automotive vertical services, as an example to demonstrate the ability of our framework to enable automatic service provisioning and ensure a successful SLA management (Sec. V), and as well as a summary of related work (Sec. VI).

II. The 5G-TRANSFORMER Platform

The 5GT platform consists of three main building blocks [2], as shown in Fig. 1 and described in the following.

The **Vertical Slicer (5GT-VS)** is the aforementioned vertical portal and acts as one-stop shop entry point for the verticals to request a custom network slice, tailored to their needs. A vertical service is a composition of vertical applications as well as network functions, defined by its functional and behavioural specification, as detailed in the Vertical Service Blueprint (VSB). In particular, the vertical requests a service by selecting a VSB from the catalogue offered by the 5GT-VS and customizes it with additional details at the service-level (e.g., expected number of users, coverage area, required SLAs, etc.) thereby defining a Vertical Service Descriptor (VSD). In turn, the 5GT-VS, through its *Translator* module, maps these service requirements into a potential set including Network Service Descriptor (NSD), Deployment Flavour (DF), and Instantiation Level (IL). This triple defines the characteristics of the target network slice (deployed through an NFV-NS) in terms of:

- the functional elements and the structure of an NFV-NS underpinning the network slice able to host the requested vertical service. This is defined through the NSD and the related VNF descriptors;
- the number and capacity of the VNFs, and the virtual links needed to meet the performance requirements of the vertical service. Specifically, the DFs define the different options to instantiate the service, including a min-max range for the number of VNFs to be instantiated, while each IL indicates the specific number of VNF instances and their required computing resources.

In the 5GT system, the definition of network slices is aligned with the latest networking slicing model from 3GPP [3] and has been extended to consider not only the mobile communication segments of the end-to-end service (as in the standard), but also the involved vertical applications. Furthermore, the 5GT network slice model is rather generic and can support any RAN and wireless technologies (although the latter aspects are beyond the scope of 3GPP). Network slices are deployed through NFV-NSs, which are instantiated according to a specific [NSD, DF, IL] set, selected on the basis of the service characteristics. The network slice components can be instantiated through a dedicated NFV-NS, to guarantee the maximum level of isolation, or exploit VNFs already instantiated for other services to optimize resource allocation. Some works in the literature analyse the challenges of RAN and core network slicing and resource sharing (e.g., [4] [5] [6]). In 5GT, the decisions on the sharing strategy applied to each network slice depend on the particular service and slice profile (such as coverage area, resource sharing/isolation policy, and performance requirements), and determined through arbitration at the level of vertical services using those slices.

Specifically, according to the network slice model defined by 3GPP, a network slice can include multiple slice subnets, where each subnet can be shared among multiple end-to-end network slices, thus improving the infrastructure utilization efficiency. The number of network slices sharing a slice subnet and, consequently, the number of vertical services running over a slice subnet has an impact on its resource requirements (e.g., in terms of traffic load to be supported). Also, any decision about re-using a slice subnet instance should be compliant with the isolation requirements of the services using it. As depicted in Fig. 1, the 5GT-VS includes an *Arbitrator* module that, starting from the candidate [NSD, DF, IL] set generated by the *Translator* and taking into account the network slices currently instantiated in the system, decides how to deploy the network slice for the requested vertical service. In particular, the 5GT-VS *Arbitrator* determines: *(i)* the [NSD, DF, IL] of the new network slice to be instantiated and the existing slice

[1] https://github.com/5g-transformer

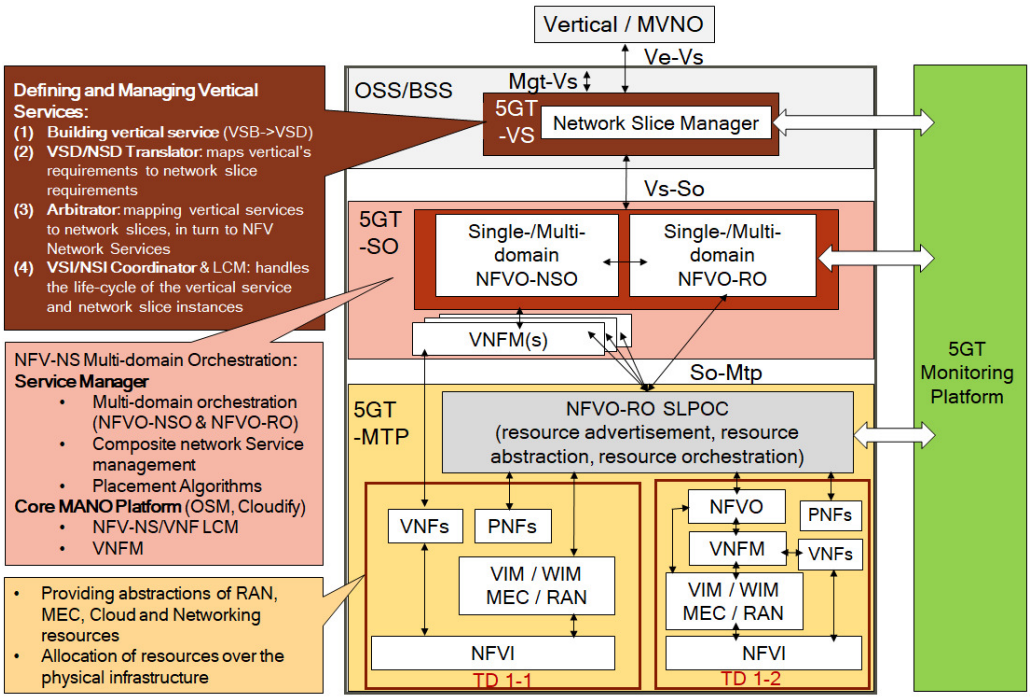

Fig. 1. The 5G-TRANSFORMER (5GT) system architecture

subnets that can be re-used to build the new slice, and *(ii)* for each slice subnet to be re-used, if and how it needs to be scaled to meet the requirements of the additional vertical service. The actions defined as output of the *Arbitrator* involve the instantiation and scaling of NFV-NSs, which are requested to the *Service Orchestrator*.

It is worth to point out that the 5GT-VS actions cover the entire lifecycle of a network slice, as defined by the 3GPP TR28.530 specification [7], from the preparation to the decommissioning phase. In particular, the definition and the on-boarding of VSBs and related NSDs correspond to the design and the on-boarding steps of the network slice preparation, respectively. The definition of the VSD and the instantiation of the service with the related network slice correspond to the creation step of the slice commissioning phase and its activation in the operation phase. Any action related to the scaling of network slice subnets and associated NFV-NSs can be mapped to the modification step during the slice operation, while the service termination actions include the de-activation and termination of the network slice, in the decommissioning phase.

The **Service Orchestrator (5GT-SO)** [8] provides both network service and resource orchestration capabilities in order to instantiate and manage network slices (deployed as NFV-NS instances) over shared resources, across single or multiple administrative domains [9]. This requires an interaction with (i) the 5GT-VS to receive NFV-NS service requests, (ii) the Monitoring Platform to configure metrics and respond to alerts, (iii) the Mobile Transport and Computing Platform to allocate resources, and (iv) other 5GT-SOs in case of multi-administrative domain service orchestration. As such, it is a central point for the coordination of all the architectural entities required to fulfill the SLA requirements of the requested service.

Therefore, the 5GT-SO implements the workflows for the (i) NFV-NS service lifecycle management (including on-boarding, instantiation, scaling, query, termination), (ii) intra-domain and multi-administrative domain orchestration, (iii) selection of VNF placement and inter-VNF links, and (iv) allocation of virtual networking, computing and storage resources through the 5GT-MTP based on service requirements and availability of the resources offered by each administrative domain. The 5GT-SO also integrates core MAN and Orchestration (MANO) platforms, such as Open Source MAMO (OSM) or Cloudify, through wrappers, hence enabling the interworking between different core MANO platforms used by different administrative domains.

The **Mobile Transport and Computing Platform (5GT-MTP)** is responsible for managing the compute, storage, and networking resources (both physical and virtual) in the infrastructure where network slices and services from the above layers are eventually executed. The resources are generally spread in different technological domains (e.g., computing Point of Presence (PoP), Wide Area Network (WAN), RAN) and, hence, the 5GT-MTP provides a coordinated management and orchestration of all these resources toward the fulfilment of 5GT-SO requests. On the one hand, the 5GT-MTP aggregates the underlying resource pool in the infrastructure to be abstracted and exposed as a single coherent whole to the 5GT-SO at Single Logical Point of Contact. On the other hand, the 5GT-MTP translates the 5GT-SO requests from abstract to low-level resource requests to be allocated in each domain. For the 5GT-MTP to interwork with underlying resources, each technology domain exposes the API of its controller (e.g., Openstack,

SDN controller, RAN controller) to the 5GT-MTP. The 5GT-MTP commands each controller through a corresponding plug-in acting as client of such API. This includes the transport WAN Infrastructure Manager (WIM), the Virtual Infrastructure Manager (VIM), the Multi-access Edge Computing (MEC), and the RAN plug-ins.

Transversal to the three aforementioned building blocks, the 5GT architecture includes a cross-layer **Monitoring Platform (5GT-MON)** that collects monitoring data from the 5GT-VS, 5GT-SO, and 5GT-MTP, and generates notifications (alerts) as input for SLA management decisions at the different layers. It is based on the Prometheus and the Grafana software, for the collection/storage/elaboration and the visualization of monitoring data, respectively. The 5GT-MON aggregates metrics and KPIs generated at the different layers, e.g., load of physical infrastructure and virtual resources, performance of network services, metrics associated with vertical applications. Starting from the elaboration of these data, the 5GT-MON recognizes any performance degradation or anomalous conditions on the basis of thresholds defined in the descriptors (e.g., in the NSD) and notifies the 5GT components through asynchronous alerts. These notifications trigger the reaction of the 5GT platform (e.g., scaling or recovery actions) to guarantee the continuous fulfilment of the SLAs established at the different layers.

III. Hierarchical SLA Management and Service Scaling: Concept & Implementation

To automatically manage vertical services through the 5GT system and fulfill the service requirements specified by the vertical, we propose a hierarchical SLA management framework, as illustrated in Fig. 2. From the top down, we define per-layer SLAs, along with the associated management mechanisms, as detailed below.

Vertical SLAs. They are business-level SLAs, which are negotiated between the 5GT telco provider's OSS/BSS (Operations Support System and Business Support System) and the vertical, and are managed by the 5GT-VS. On the one hand, the vertical provides the vertical service requirements in the service request, specifying business-level and service-level parameters (i.e., required service KPIs like maximum service provisioning time, required device density, maximum service latency). On the other hand, the telco provider's OSS/BSS can offer different business service level classes. The matching between the vertical service requirements and the service level classes offered by the 5GT telco provider defines a Vertical SLA. In the 5GT-VS, the Vertical SLA management functions include: (i) mapping the Vertical SLAs to Network Service SLAs (NS SLAs) that will be requested to the 5GT-SO, including network service related policies such as rules for the automatic scaling of a NFV-NS instance; (ii) mapping a vertical service on network slice(s), either instantiating a new network slice or re-using existing slice subnet(s); and (iii) handling service arbitration and service scaling actions to deal with the dynamic changes on the service itself and according to the service Vertical SLAs, the total available resource budget, and the services priority.

NS SLAs. They are defined at the NFV-NS level and are managed by the 5GT-SO. NS SLAs define distinct guarantees on resource availability and KPIs, such as guaranteed data rate, geographical availability, and end-to-end latency. Their management is provided by end-to-end service and resource orchestration, including (i) deciding the optimum placement of the VNFs in certain PoPs/servers and the inter-PoP/inter-server connectivity, and (ii) handling the NFV-NS auto-scaling operation to adapt to the dynamic network conditions following the aforementioned NFV-NS auto-scaling rules defined in the NSD.

Infrastructure SLAs. They are managed by the 5GT-MTP, which is in charge of the actual resource allocation for a specific service, as requested by the 5GT-SO. The Infrastructure SLAs are specified based on the NS SLAs and define different guarantees on infrastructure-level QoS (such as CPU load, network delay, packet losses, link throughput). At the infrastructure level, the resource management function is in charge of the placement of virtual machines (VM) (or containers) in physical servers, and managing their required networking connectivity such as routing and path provisioning.

In summary, each layer is fully responsible for: (i) translating its SLAs to lower-level SLAs and requesting them to the lower layer, and (ii) guaranteeing the corresponding SLAs through its internal SLA management functions. To this end, each layer interacts also with the monitoring platform, which provides monitoring data about SLA-related metrics and triggers alerts whenever a degradation or violation of the SLAs is detected. Finally, each layer may notify the higher layer about the results of the SLAs it is managing.

As highlighted above, one of the crucial SLA management functions is ***scaling***. Depending on the layer at which it is performed, we can define:

1) *Application-level scaling*, which is triggered by the vertical and implies the scaling of a vertical service deployment based on the operational context information [10]. This typically results in a renovated slice service demand to the telco service provider (i.e., the to 5GT-VS), with different service parameters;
2) *Service-level scaling*, which is triggered by the 5GT-VS as a result of an arbitration procedure among service instances and yields the scaling of one or more of them;
3) *Resource-level scaling*, which is triggered by the 5GT-SO after detecting lack of sufficient resources to meet certain NS SLAs. It leads to the scaling of the virtual resources underpinning the network slice deployment.

A. Application-level Scaling

Application-level scaling consists in adjusting vertical service deployments into the network slices during their runtime, according to evolving vertical's business targets (e.g., enlarging the geographical area that is served) or following the dynamics of the application operational context (e.g., average and peak number of user requests). In both cases, the scaling decision results in a renovated slice service request to the 5GT-VS but with different service parameters. Examples of such service parameters could be the number of mobile users or content items for a Content Delivery Network in the case of a multimedia service. For an automotive safety service, a

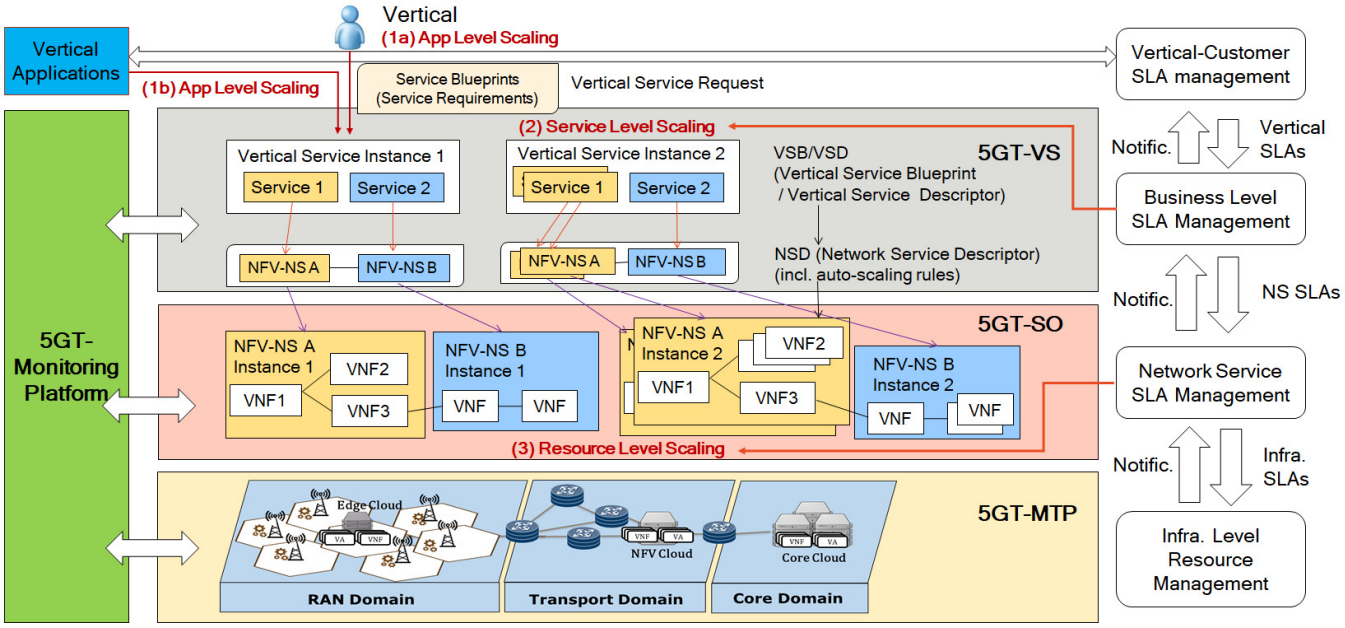

Fig. 2. Hierarchical SLA Management framework

relevant parameter could be the maximum number of cars to be served in a given area. To support the application-level scaling, the 5GT-VS provides a vertical north bound interface (NBI) that allows the verticals to issue slice service requests with revised service parameters specified in the VSD.

The verticals and their applications have thus a fundamental role in the application-based scaling model. They are not only responsible for detecting the need to scale a service and deciding its target size (expressed in terms of service-level parameters within the target VSD), but also for triggering the entire scaling procedure by interacting with the 5GT-VS. In particular, the need for a new VSD with updated service parameters can be stated by the vertical service administrator through a manual configuration (1a in Fig. 2), or can be detected automatically by a service control logic internal to the application (1b in Fig. 2). In the former case, the system administrator uses the 5GT-VS web GUI to manually request the modification. In the latter case, the application itself interacts directly with the NBI of the 5GT-VS, through the REST APIs. The mechanisms for making decisions about application-based scaling are service-dependent and they rely on business considerations or application-level performance metrics. The applications implement their own monitoring procedures to gather and elaborate the required application-level metrics. Their internal logic makes decisions about the required scaling actions, e.g., based on the thresholds defined in compliance with the SLAs established between the verticals and their customers.

At the 5GT-VS level, the enforcement of the vertical service scaling is managed in two phases. In the first phase, the 5GT-VS uses the *Translator* module to map the new VSD into the definition of a network slice able to meet its requirements. This mapping follows the same procedure performed during the instantiation phase and it identifies the characteristics of the target network slice. In particular, the output identifies the kind, size, and capacity of NFV-NS(s) corresponding to the end-to-end network slice, including its slice subnets. If the target network slice differs from the current one (e.g., in terms of DF and/or IL of the correspondent NFV-NS(s)), the current network slice(s) must be updated (i.e., scaled in or out); thus, the 5GT-VS starts a second phase that involves the *Arbitrator* module. Therein the system verifies the compatibility between the new network slice with its new target size/capacity and the SLAs established with the vertical, taking into account the whole set of network slices already active for the given vertical. In this phase, the arbitration algorithm (see Sec. III-B) computes the modifications required for all of the different services owned by the vertical, based on their relative priorities. As output, the *Arbitrator* decides the feasibility of the scaling action and its impact on the existing network slice and slice subnet instances. It thus identifies the NFV-NSs of those services with lower priority and belonging to the same vertical that may need to be scaled down to make some resources available to services with higher priority. The overall resulting set of changes are then applied to the NFV-NSs (realizing the network slices) scaling actions and then requested in an ordered manner to the 5GT-SO, which will modify the NFV-NSs as requested.

B. Service-level Scaling

Service-level scaling consists in adjusting the size (i.e., number of VNF instances) and/or the capacity (i.e., total resource demand) of network slice instances hosting the vertical services, as a result of a decision made by the 5GT-VS Arbitrator. As mentioned before, the *Arbitrator* is the entity responsible for making any decision about network slice sharing and scaling. In particular, upon receiving a vertical's request to deploy a new service instance, the *Arbitrator* looks

up the corresponding Vertical SLA, namely: (1) the priority level of the new, as well as the existing, service instances requested by that vertical, (2) the set of VNFs composing the service and how they are inter-connected, (3) the relative virtual CPU (vCPU) and memory/storage requirements of the involved VNFs as well as the networking requirements for their inter-connectivity, and (4) the vertical's KPI 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.

Given such an input information, the *Arbitrator* makes the following decisions:

1) it determines whether the newly requested service must be created from scratch, thus instantiating all its elements ex-novo, or, instead, one or more already existing slice subnets can be reused;
2) if the service has to be deployed entirely (or partially) ex-novo, it determines (a) the amount of resources that may be needed for the service (or part of it) to be instantiated and to handle the expected traffic load, and (b) whether or not such an amount is compatible with the resource budget available to the vertical, as per the Vertical SLA;
3) when existing slice subnets can be re-used, it decides which (if any) scaling action for any of them is necessary to fit the service requirements, and also feasible as per the Vertical SLA.

Thus, for every new or to-be-reused slice subnet, the *Arbitrator* provides as output the associated pair [DF, IL], so as to ensure that the vertical KPI requirements are met, while accounting for the services priority level and the remaining resource budget available to the vertical, as per the Vertical SLA.

Let us first consider that no existing sub-slices can be reused for the deployment of a newly requested service, and let us denote with C, B, and S the total amount of, respectively, vCPU, bandwidth, and storage that can be allocated for the services of that vertical as per the Vertical SLA. As the first step, the *Arbitrator* orders all service instances, both the one to be deployed and the existing ones, from the highest priority level to the lowest. It then considers the highest-priority service instance, say, s, and allocates storage resources based on the needs exhibited by the VNFs composing s.

A more complicated procedure, however, is required for the vCPU and bandwidth allocation. In particular, let us focus on the service latency as the main performance metric, and denote with D_s the target latency for service s. Two factors contribute to the service latency: (i) the processing time, due to the execution of the VNFs composing the service, and (ii) the network time, i.e., the time it takes to transfer data from a VNF to the next one[11]. While the former depends on the vCPU allocated to the VM or containers running the VNFs, the latter depends on the deployment decisions made by the 5GT-SO, and on the bandwidth associated with the virtual links connecting the servers hosting the VNFs.

In the *best* case, the whole set of VNFs composing service s, denoted by $\mathcal{V}$, can be deployed within the same server. In this scenario, the network time is negligible [12], hence the bandwidth required for data transfers over virtual links for s, β^b, can be set to zero. Additionally, the latency budget, D_s, can be entirely used as processing time, thus reducing the required amount of vCPU, μ^b. To determine such value, we follow a widely adopted approach (see, e.g., [13], [14]) and model each VNF instance as an M/M/1-PS queue. Note that the choice of the processor sharing (PS) policy for the queue model closely emulates the behavior of a multi-threaded application running on a VM. Then μ^b can be computed so as to satisfy the below inequalities:

$$\sum_{v\in\mathcal{V}} \frac{1}{f_v\mu^b - \lambda_v} \leq D_s; \qquad \mu^b \leq C\,. \tag{1}$$

The first inequality imposes that the total latency due to the processing at the service VNFs does not exceed the maximum target value. In particular, the left hand side term represents the total latency due to the processing at every VNF $v \in \mathcal{V}$ [15], with $f_v\mu^b$ being the output rate of the VNF queue v and λ_v being the service request rate input to v. Also, f_v is the relative computational requirement of VNF v, with $\sum_{v\in\mathcal{V}} f_v = 1$. The second inequality, instead, imposes that the vCPU allocation does not exceed the vertical vCPU budget, C.

Assuming that all VNFs run within the same server, however, might be overly restrictive. Thus, the *Arbitrator* also considers a *worst-case* scenario, accounting for the network latency component as well. As a smaller portion of the latency budget would be available for processing, the amount of processing resources required in this case increases. Specifically, in the worst case, each VNF in $\mathcal{V}$ is deployed in a different server, hence the allocated vCPU, μ^w, and bandwidth, β^w, have to satisfy the following constraints:

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

$$\mu^w \leq C \quad \text{and} \quad \beta^w \leq B \tag{3}$$

where $f_{u,v}$ is the relative bandwidth requirement for the virtual link connecting the servers where VNFs u and v are deployed, and $d_{v,v}$ is the amount of data that needs to be transferred from VNF u to VNF v. In (2), the two left hand side terms represent the latency due to, respectively, the VNF execution and the travel time over the virtual links connecting any two adjacent functions in the VNF set ($\mathcal{E}$ denotes the set of edges interconnecting the VNFs composing the service). The constraints in (3), instead, impose that the total vCPU and bandwidth allocations do not exceed the corresponding budgets available to the vertical, as per the Vertical SLA.

Next, given the pairs $(\mu^b, 0)$ and (μ^w, β^w) for service s, the *Arbitrator* can compute the corresponding per-VNF and per-virtual link values, by leveraging the f_v and $f_{u,v}$ values expressing the relative computation and bandwidth requirements of each VNF and virtual link (resp.). The *Arbitrator* then selects an ordered list of [DF, IL] pairs, as encoded in the NSD, with the first pair corresponding to the best-case allocation and the last one to the worst-case allocation; a practical example is provided in Sec. IV.

Once the 5GT-SO receives from the 5GT-VS the instantiation request, it deploys the service selecting the most efficient [DF, IL] pair among the viable ones suggested by the 5GT-VS. The quota of resources used by the vertical is updated

at the 5GT-VS, based on the 5GT-SO's choice. Given such a value, the *Arbitrator* proceeds with the second service in the list, following the same steps as above but replacing C and B (in (1)–(3)) with the amounts of vCPU and bandwidth still available to the vertical. The procedure is repeated for all (newly requested or already deployed) service instances; it is clear that, in case of resource shortage, some lower-priority service instances may not be accommodated, or may be terminated due to the need to reallocate resources to higher priority services.

As a relevant case in terms of scaling, consider now that a service instance requested by the vertical can be deployed by reusing one or more of the existing slice subnets. As mentioned, sharing a slice subnet across multiple end-to-end slices may impact its performance and, consequently, the performance of the vertical services using it. Thus, to guarantee the required performance, the size and/or the capacity of a slice subnet instance may need to be adjusted according to the number and characteristics of the end-to-end slice instances that are sharing it. In this case, the *Arbitrator* adds the traffic load due to the newly requested vertical service to the load of the existing VNFs (λ_v) and virtual links ($d_{u,v}$), and recomputes the necessary vCPU and bandwidth allocation, as described above. Again, the *Arbitrator* uses the vCPU and bandwidth values obtained in the best and worst cases, to update the [DF, IL] pairs associated with the involved VNFs and virtual links.

Finally, we remark that the same procedure is performed when a service instance is terminated: the 5GT-VS updates the amount of resources available to the vertical and recomputes the [DF, IL] pairs for the remaining services, upgrading some of them if needed.

C. Resource-level Scaling

Resource-level scaling involves monitoring and reconfiguration of virtual resources orchestrated by the 5GT-SO (in coordination with the 5GT-MTP), to prevent the performance degradation of VNFs/NFV-NSs. More specifically, it regards the auto-scaling of NFV-NSs according to the scaling rules given in the NSD, by configuring related monitoring jobs and alerts in the monitoring platform, and by properly reacting to such alerts. Scaling rules are defined by the vertical, based on business-related considerations or the application-related operational context, and are part of the VSB definition when on-boarded in the system. These scaling rules are encoded in the NSD and then forwarded from the 5GT-VS to the 5GT-SO.

Each auto-scaling rule contains (a) the conditions to be met by certain metrics, to trigger alerts based on the service monitored data, and (b) a corresponding reaction (i.e., a scaling out/in action). During the NFV-NS instantiation phase, the 5GT-SO configures the 5GT-MON monitoring platform according to the conditions encoded in the NSD auto-scaling rules, in order to receive the required alerts at the NFV-NS runtime. Whenever the 5GT-SO is notified by the 5GT-MON that one of the conditions is met (e.g., exceeded vCPU usage), it triggers the NFV-NS scaling according to the corresponding reaction specified in the auto-scaling rule. To this end, it also coordinates the operation of the core MANO and 5GT-MTP. In particular, in case of scaling out, the 5GT-SO issues a new resource allocation request to the 5GT-MTP for scaling the VNF instances and, hence, to reconfigure the virtual resources towards the new instantiation level specified in the auto-scaling rules. The 5GT-MTP applies all needed settings for the required resource re-allocations, while the 5GT-SO notifies the 5GT-VS about the scaling operation outcome. In case of scaling-out failure, due to, e.g., resource shortage, the 5GT-SO undoes or rolls-back the scaling operation and also informs the 5GT-VS about the failure.

In terms of implementation, the 5GT-SO provides resource-based scaling thanks to two internal submodules, namely the *Monitoring Manager* and the *SLA Manager* [8]. The former configures the monitoring jobs required to measure the resource metrics involved in scaling decisions. The latter requests the configuration of the alerts associated with these metrics in the 5GT-MON and also processes the received alerts to trigger scaling actions according to the auto-scaling rules. The 5GT-MON configuration is then coordinated through a *Configuration Manager* component, which offers REST APIs and wraps the logic of the configuration for the different Prometheus components involved in the monitoring task. Specifically, 5G-MON leverages the following Prometheus components: (i) Monitoring jobs, used to retrieve monitoring data from different sources through the mediation of Prometheus exporters specialized for infrastructure or application metrics, (ii) Thresholds, used to trigger alerts towards the SLA manager, and (iii) Dashboards, used to visualize the monitoring data through Grafana.

IV. Proof-of-concept: Testbed and Scenarios

In this section, we present the proof-of-concept testbed we deployed to evaluate our framework. The testbed implements the whole 5GT platform introduced in Sec. II, and the applications required by an automotive vertical to be provided to mobile users and vehicles. It is worth noting that here we focus on the scaling of NFV-based vertical services rather than on the network functions related to the mobile infrastructure. More importantly, our solution is generic to support any network functions (including RAN or core network functions) and vertical applications, and hence, as also underlined below, our solution can work well with any radio access technology.

Below, we start by introducing the testbed architecture and the vertical targeted services (Sec. IV-A), then we describe the scenario and the experiments that we performed in our field tests (Sec. IV-B). The performed field tests aim to demonstrate the effectiveness of the developed 5GT platform and our proposed solutions, able to: (i) automatically deploy vertical services upon receiving the service requests, and (ii) perform automated management of the vertical services across different layers of the architecture. In particular, SLA assurance is achieved through service-level scaling to handle the arbitration among different services of the same vertical according to their priorities, and by resource-level scaling to handle the scaling of resources (i.e., in terms of the number of VNF instances), according to resource dynamics and load variations.

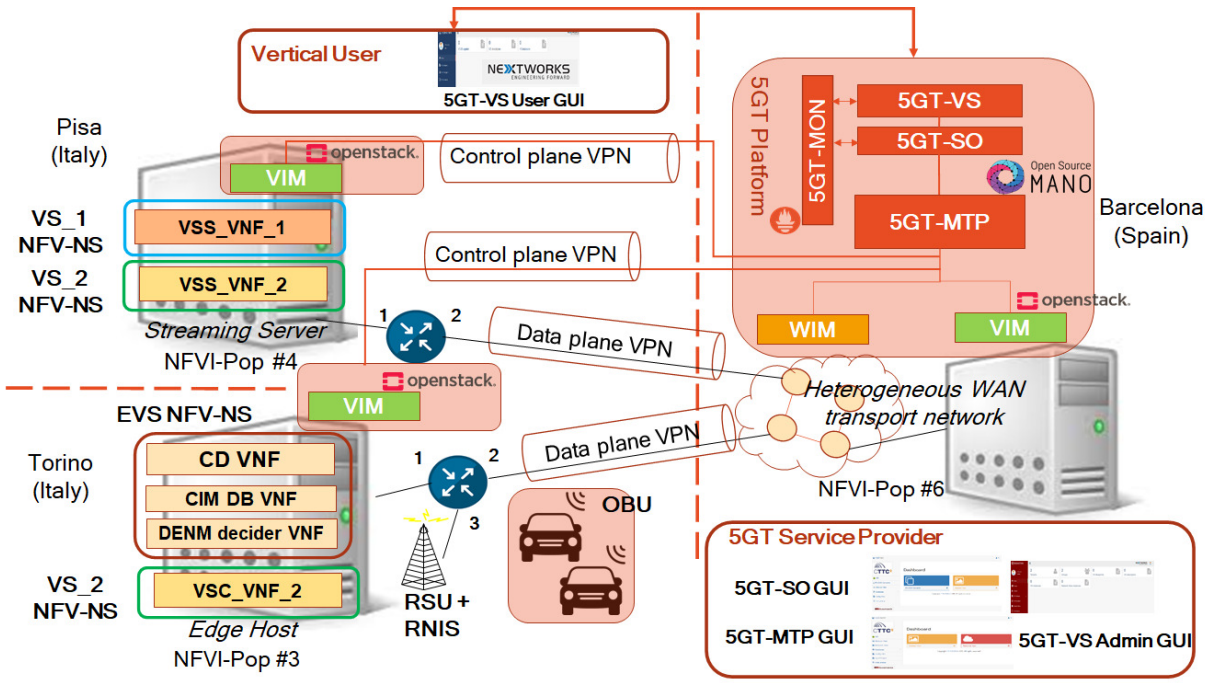

Fig. 3. Testbed architecture and a possible deployment of the considered NFV-NS across the available PoPs

A. Testbed and Supported Services

The testbed, depicted in Fig. 3, spans over three geographical sites, which are connected through VPN tunnels encapsulating both control traffic and data traffic. The Barcelona site, in Spain, hosts all layers of the 5GT platform, including the 5GT-VS, 5GT-SO, 5GT-MTP and 5GT-MON, an instance of an Openstack-based VIM, and an instance of a WIM controlling the core transport network. The 5GT platform uses OSM Release 6 as MANO platform. The WIM, as described in [16], follows an IETF Application-Based Network Operation (ABNO) architecture using the Control Orchestration Protocol (COP) to communicate with the 5GT-MTP and interacts with the forwarding elements of the underlying transport networks by means of open source SDN controllers like Ryu and OpenDaylight. Then, two instances of an Openstack-based VIM are deployed at, respectively, the Pisa and the Torino site, in Italy, with the latter acting as MEC host. The Torino site also hosts an IEEE 802.11p Roadside Unit (RSU) physical network function to provide radio access to vehicles equipped with On-Board Units (OBU), and a Radio Network Information Service (RNIS) providing channel state information to the applications requiring it. An equivalent testbed was deployed in Torino [17], using the open-source Open Air Interface (OAI) [2] implementation of the LTE E-UTRAN and EPC, which is compliant with 3GPP LTE Releases 8/10. Laboratory tests [18] showed that the proposed system architecture and scaling solution work effectively also under these settings. However, during field tests an IEEE 802.11p-based radio access was preferred, so as to deal with vehicular communications and ensure a sufficiently large outdoor coverage.

The testbed supports two services requested by an automotive vertical, namely, (i) vehicle collision avoidance at intersections, and (ii) video content delivery, which are relevant examples of, respectively, safety and entertainment services for vehicular users.

The vehicle collision avoidance service will be referred to as Extended Virtual Sensing (EVS), since it leverages vehicular communications as a virtual sensor collecting data related to vehicle mobility [19], [20], [17]. Specifically, it exploits the Cooperative Awareness Messages (CAMs), defined by ETSI, which are periodically transmitted by vehicles and carry the position, speed, acceleration, and heading of the sender. By processing such data, the EVS service can detect dangerous situations and generate warnings accordingly. These warnings are encoded in the ETSI Decentralized Environmental Notification Messages (DENMs) and delivered to human drivers, or to an emergency braking system aboard vehicles. The VNFs composing the EVS service are as follows:

- the CIM (Cooperative Infrastructure Manager), which receives, decodes, and stores CAMs sent by the vehicles within the area covered by the EVS service;
- the Collision Detector (CD), which queries the CIMs for new CAMs and runs a trajectory-based algorithm (e.g., the one presented in [19], [20]), to detect pairs of vehicles on collision course;
- the DENM Decider, which timely encodes the warning messages and sends them to the vehicles deemed to be on collision course.

The video content delivery service refers to a Video Streaming (VS) service that may be provided in *full-fledged* or *reduced* configuration. The *full-fledged* version consists of the following two VNFs:

- the Video Streaming controller (VSC), which exploits a Radio Network Information Service (RNIS) and a radio

[2]https://openairinterface.org

link manager, recording information on the quality of the user radio channel. This information is given as input to an optimization algorithm [21] that selects the most suitable bit rate for streaming the video to the user;
- the Video Streaming Server (VSS), featuring a Python-based front-end, which applies the selected video bit rate to the video segments to be transmitted. It is based on HTTP streaming and contains a video catalogue, a front-end, the Media Presentation Description (MPD) files, and the media chunks. The front-end receives the selected video bit rate and edits the MPD files with such a rate.

The *reduced* VS service differs from the *full-fledged* version in the fact that it includes the VSS only, i.e., it is unable to adapt video encoding to the user channel conditions. Finally, we remark that both the EVS and the VS services require a mobile transport function that realizes the communication between the network infrastructure and the vehicular users.

B. Evaluation Setup and Experiments

The evaluation setup consists of (i) a vertical service deployment in the form of NFV-NSs, and (ii) a vertical service operation once the NFV-NSs are deployed.

As for the vertical service deployment, upon receiving the automotive vertical request for services, the corresponding VSD is compiled at the 5GT-VS. Importantly, at this stage, the vertical specifies (i) the services' priority (with EVS having higher priority than VS), (ii) the services' configuration, (iii) the storage requirements for the VS and EVS VNFs, (iv) the geographical area that has to be covered by each service, and (v) the estimated number of users to serve. Also, since the EVS should be combined with other collision avoidance mechanisms based on physical sensors aboard the vehicles, the maximum target latency specified by the vertical in the VSD is set to 20 ms. The VSD is then translated into the NFV NSD, and the [DF, IL] pairs are set for each VNF instance according to the output of the arbitration algorithm running at the 5GT-VS. In particular, for all VNFs composing the VS and the EVS service, except for the CD, only one instance using up to 1 vCPU is foreseen as both minimum and maximum allocation, since the processing latency of such VNFs is limited and does not significantly increase with the traffic load. For the EVS CD, instead, the [DF, IL] indicates the possibility to have from 1 up to 2 instances, each using 1 vCPU. The 5GT-SO, which is aware of the computing and network resources available as exposed by the 5GT-MTP (which, in turn, interacts with underlying VIMs and WIMs), computes the most appropriate placement for the VNFs, and instantiates them as Openstack VMs, following the NSD requirements. Service monitoring is performed by the 5GT-MON, which pulls and stores metrics from the VMs hosting the aforementioned VNFs.

With regard to the vertical service operation, the test field used for our experimental evaluation consists (unless otherwise specified) of a urban intersection with two vehicles, one of which is an automated car equipped with an Automatic Emergency Braking (AEB) system. Both vehicles are equipped with an IEEE 802.11p OBU, thus they transmit CAMs every 100 ms and can receive DENMs. The vehicle equipped with the AEB can also process DENMs and has the necessary on-board logic to translate the DENM content into a command for the AEB. The vehicles travel on perpendicular roads and approach the intersection at full urban speed (namely, 50 km/h).

Fig. 4. Map of the geographical area served by the EVS service, featuring two intersections and including an IEEE802.11p RSU: one EVS instance and corresponding covered intersections highlighted in yellow (left), two EVS instances and corresponding covered intersections highlighted in pink (right)

The field trial includes three phases, as detailed below.

- **Phase 1:** The vertical asks for the deployment of two VS instances, one *full-fledged* and the other in *reduced* configuration. This first part of the trial shows how an automotive vertical can use the 5GT platform to instantiate two different VS services, just by providing high-level service parameters and without any detailed knowledge of the underlying infrastructure.
- **Phase 2:** The vertical asks for the deployment of an EVS instance, which, being a safety service, has higher priority than VS. Due to limited resource budget available as per the vertical SLA, the instantiation of the EVS requires that service priority is properly handled by the *Arbitrator* at the 5G-VS (service-level scaling).
- **Phase 3:** The vehicle density on the area served by the EVS increases, which impacts significantly on the computing load and in turn the application latency. Thus, whenever load changes, scaling at the resource level is needed to keep up with the SLA latency requirements (resource-level scaling).

To emulate a high vehicle density, in Phase 3 we consider a urban section of the city of Torino, depicted in Fig. 4(left), including two urban intersections, and we leverage a mobility trace obtained with the SUMO simulator [22]. This trace is processed so as to generate the CAMs corresponding to the simulated vehicles; such CAMs are then injected through the same data plane connections used for real cars in the field trial. This allows us to handle such CAMs in exactly the same way as those generated by real vehicles, i.e., they are transmitted on air and, upon being received, the information they carry is stored in the CIM.

V. Field Trial Performance Results

In this section, we report the actions taken by the 5GT platform during the Phase 1 to 3 of the field trial, as well as the performance of the services that are deployed. We first report the decisions made by the 5GT-VS upon receiving the VS and EVS set-up requests, and the decisions made by the 5GT-SO as the vehicle density increases (Sec. V-A). Then we present some results obtained by profiling the service creation time components, for both VS and EVS (Sec. V-B). Finally, we

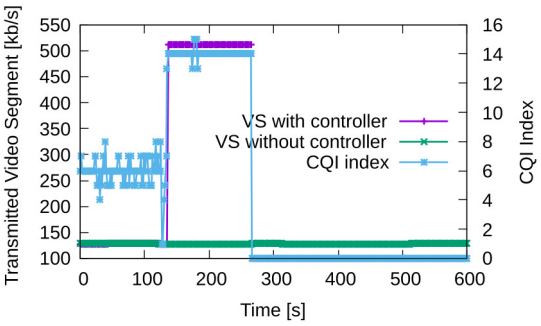

Fig. 5. Video service segment bit rate and received CQI before (0–273 s) and after arbitration (273–585 s)

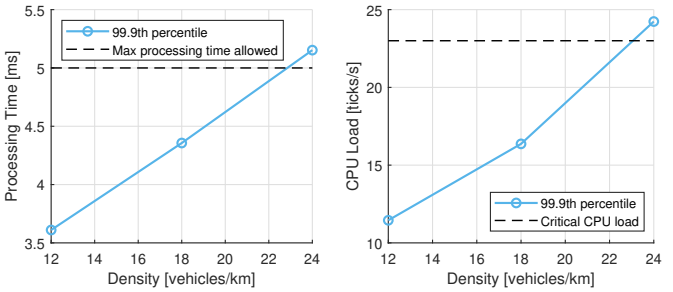

Fig. 6. VNF implementing the CD algorithm: processing time (left) and CPU load (right), as functions of the vehicular density

show results related to service-level and resource-level scaling, which take place during, respectively, Phase 2 (Sec. V-C) and Phase 3 (Sec. V-D) of the trial.

A. Services Performance – All phases

In Phase 1, upon receiving the request of the two VS services, the 5GT-VS goes through all the steps described in the previous sections and ultimately generates two NFV-NSs. One NFV-NS is for the *full-fledged* VS service with an IL corresponding to a large amount of allocated resources and enforcing that the VSC is placed in the MEC. The other NFV-NS is for the *reduced* VS version, with an IL for low resource footprint. We recall that the VSC must be deployed in the MEC, since it includes a radio manager requiring the RNIS (MEC) service for the tracking of the user channel quality. The 5GT-SO then deploys the VSS instances of the two VS NFV-NSs as VMs in the Pisa site, and one VSC instance in the Torino (MEC) site, and sets up the links to interconnect the VSC with its associated VSS in the Pisa site through the Barcelona transport infrastructure by interacting with the 5GT-MTP.

Fig. 5 (Phase 1: from 0 to 273 s) shows the bit rate of the two VSs (i.e., with and without VSC). The *full-fledged* VS (purple line) increases the video segment bit rate when the quality of the radio channel (reflected by the Channel Quality Indicator (CQI), blue line) is high, while the *reduced* VS (green line) maintains the same video segment bit rate.

In Phase 2, the 5GT-VS receives the request for an instance of EVS, characterized by a target maximum end-to-end latency of 20 ms, which implies that the corresponding NFV-NS constraints the service deployment in the MEC (Torino site). However, following the *Arbitrator* algorithm in Sec. III-B, the 5GT-VS detects that the amount of resources necessary to deploy the EVS exceeds the total resource budget specified in the vertical SLA, and that the resources previously allocated for the VS services must be revised. In particular, the algorithm indicates that the *full-fledged* VS service must be terminated so that the resources allocated to the VM implementing the VSC VNF in the MEC host are made available to the EVS service.

Fig. 5 (Phase 2: from 273 to 585 s) reports the bit rate of the video segment after arbitration, when only the *reduced* video service (the one without VSC) remains in place. Thanks to the availability of real cars, this part of the trial not only demonstrates the correct behavior of the *Arbitrator*, but also that the EVS is successfully deployed and meets the automotive safety requirements and the maximum target latency of 20 ms. In particular, the value of end-to-end latency, from the transmission of the CAM to the reception of the corresponding DENM, averaged over 10 different tests, is equal to 8.870 ms, with a standard deviation of 1.447 ms, and a maximum and a minimum value equal to 11.637 ms and 5.050 ms, respectively.

In Phase 3, the *reduced* VS service and the EVS service run simultaneously, one in the Pisa site and the other in the Torino (MEC) site. With regard to the EVS, as the vehicle density in the area highlighted in yellow in Fig. 4(left) increases, the CAM and processing load grows as well. Such an increased load of the VMs implementing the VNFs yields an increased processing time, hence an increased end-to-end latency for the EVS. In particular, we observed that the major contribution to the processing time is due to the VM implementing the CD algorithm, while the contribution of the other VNFs is negligible (e.g., 10 times smaller) and does not significantly scale up as the vehicle density grows. As a consequence, we focused on the CD VNF and measured its processing time (Fig. 6(left)) and the corresponding CPU load (Fig. 6(right)), for different values of vehicular density.

Given the latency contributions of the other VNFs and the latency of the data radio transfer, we computed a threshold for the processing time of the CD algorithm (namely, 5 ms) that cannot be exceeded. Then, leveraging the results in Fig. 6(left), we identified the critical vehicle density (e.g., 22.5 vehicles/km in the plot) below which a CD processing time of less than 5 ms is recorded for the 99.9% of the time. Finally, we used this density value in Fig. 6(right) and derived the critical CPU load (namely, 23%) as the threshold for resource scaling.

Once we determined the critical CPU load threshold, we configured the monitoring jobs and alerts in the 5GT-MON platform to generate an alert whenever the CPU consumption of the CD VM reaches such a value. The alert is handled by the *SLA Manager* module of the 5GT-SO, which generates a scale-out request, i.e., the deployment of a second CD instance, and makes the EVS service self-reconfigure to split the load between the two CD VNFs. In this way, half of the cars in the area covered by the service can be handled by the initial CD VNF and the rest by the new CD VNF. Specifically, with reference to Fig. 4(right), each CD instance processes

only the CAMs generated by the vehicles crossing one of the intersections highlighted in pink. As a consequence, the CPU load is halved, and the target latency required by the EVS service can be fulfilled. An initial functional prototype was demonstrated in [18]. Importantly, the above discussion applies to scale-in operations as well. Hence, when the density decreases and some resources can be freed, a scale-in operation is performed exploiting the same mechanism as described above.

B. Service Creation – Phase 1

We analyzed Phase 1 from the viewpoint of the 5GT platform. The focus is on profiling of the service creation and instantiation process for the requested VS and the EVS services, considering the different ILs available for each considered NFV-NS (see Table I). We remark that, in all plots presented here and in the following sections, boxplots represent the experienced maximum, minimum, average, median, 20th- and 80th-percentile of the ten repetitions performed for each experiment.

Fig. 7 shows the various components of the service creation time, when deploying the EVS service consisting of 2 CD VMs The phase taking longer is the Allocate VNFs one, whose duration is roughly 6 times higher than the longest of the remaining components. This phase accounts for the time it takes to the OSM wrapper to interact with OSM, which, in turn, deploys the VMs that implement the EVS service by interacting with the 5GT-MTP and Openstack. In this case, there are four VMs to deploy, i.e., 1 CIM VM, 1 DENM Decider VM, and 2 CD VMs. The following components in order of importance are the creation of the intra-PoP networks (6.986 s on average) and the 5GT-VS processing (6.385 s on average). The former accounts for the time it takes to the OSM wrapper to interact with the MTP, which, in turn, interacts with Openstack to create all the required intra-PoP networks of the 2 CD-EVS service. The latter one has a much larger dispersion due to the polling that the 5GT-VS does to the 5GT-SO to know if the instantiation process has finished (in addition to the internal processing, which is much lower). The configured polling period is of 20 s. Next, the 5GT-SO Resource Orchestrator Engine (ROE) processing accounts for the interaction with the 5GT-MTP to retrieve the topology and available resources (with the largest component equal to 1.964 s on average), the interaction and placement calculation in the Placement Algorithms (PA) server (522.3 ms), and other much smaller components. Finally, the processing in the 5GT-SO Service Orchestrator Engine (SOE) lasts 292.6 ms on average. This operation accounts for the time it takes to the SOE to interact and coordinate the operations at the different entities in the 5GT-SO module.

Fig. 8 presents the service creation time for the *full-fledged* VS service featuring same pattern of EVS service creation time in terms of relative importance of the components. The only remarkable differences compared to the above EVS service are the larger time for intra-PoP network creation (12.862 s on average) and the smaller time for VNF allocation (29.732 s on average). As for the former, this happens even if the service

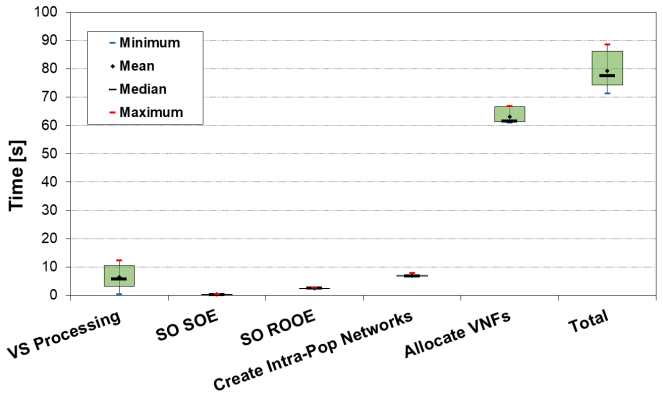

Fig. 7. Service creation time for the scaled-out EVS service (2 CD VMs)

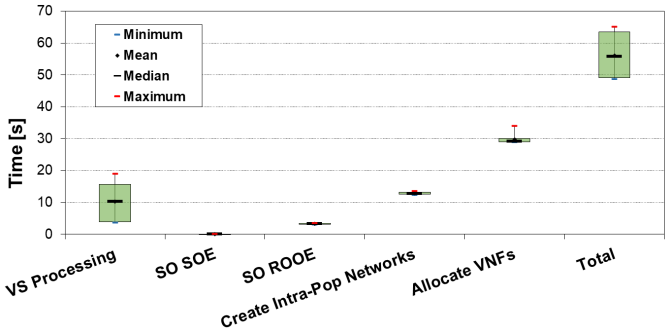

Fig. 8. Service creation time for the full-fledged VS service

to deploy is simpler (i.e., 2 VNFs instead of 4), because in the previous case all 4 VNFs were deployed within the same host (the MEC in Torino) and the same intra-PoP networks were used by all of them. Conversely, in this case each VNF is deployed in a different PoP. Therefore, two sets of intra-PoP networks must be created and the 5GT-SO must interact twice with the 5GT-MTP, which, in turn, interacts with different Openstack instances to create them. Furthermore, since these two intra-PoP networks must be stitched to allow both VNFs to interact as part of the vertical service logic, the deployment time of inter-PoP logical links is not zero (293.2 ms on average), which also makes the processing time at the ROE larger (3.271 s vs. 2.501 s on average), despite being a simpler service. This time also includes the interaction with the 5GT-MTP, which, in turn, interacts with the WAN controller to configure the required transport network connection [16]. Finally, the shorter VNF allocation time in this case is due to the fact that only two VNFs, instead of 4, have to be deployed by the respective Openstack intances at each PoP.

Fig. 9 presents the experienced service creation time for the different NFV-NSs and ILs presented in Table I, ordered by the total amount of VNFs in the NFV-NS. We can observe that similar considerations can be made for the other services involved (i.e., 1-CD EVS and *reduced* VS). The main components are the time for allocating VNFs and the time for creating intra-PoP networks, with the latter being of the same order as that obtained for the EVS service with two CD VNF instances, since there is only a single PoP involved.

TABLE I
DESCRIPTION OF MAIN SERVICE CREATION TIME COMPONENTS

Service configuration	Composition	Observation
reduced VS	1 VSS VNF	Low resource footprint IL
full-fledged VS	1 VSS VNF, 1 VSC VNF	High resource footprint IL
EVS (IL with 1 CD)	1 CIM VNF, 1 DENM VNF, 1 CD VNF	Initial IL
EVS (IL with 2 CD)	1 CIM VNF, 1 DENM VNF, 2 CD VNFs	IL for high density scenarios or after scale out operation

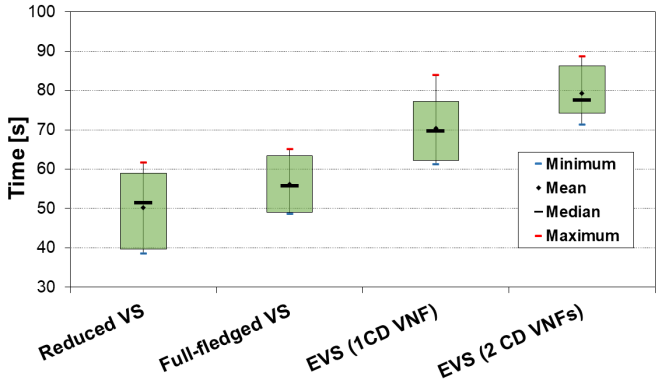

Fig. 9. Service creation time for the different considered NFV-NSs and ILs

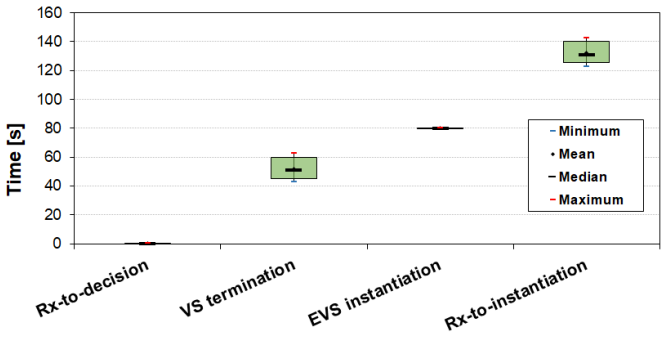

Fig. 10. Service-level scaling time

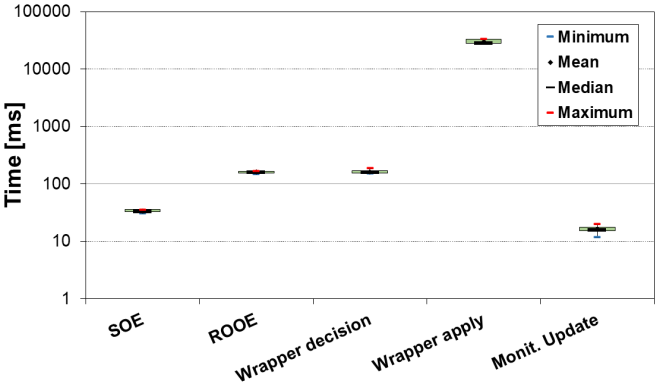

Fig. 11. Scale-out breakdown time analysis for the EVS service

C. Service-level Scaling – Phase 2

As mentioned, in Phase 2 an EVS service creation request arrives at the 5GT-VS, but this vertical is already running two VSs (i.e., *full-fledged* and *reduced* VS) that consume most of the available resource budget. As a consequence, the *Arbitrator* decides to terminate the *full-fledged* VS service to make resources available for the (higher priority) EVS service.

Fig. 10 shows the three components of the service-level scaling time. First, the Rx-to-decision time accounts for the time it takes to the *Arbitrator* to realize that the new service request does not fit in the vertical's budget and to decide to terminate the *full-fledged* VS service. This is the smallest component, since it only involves internal processing inside the 5GT-VS (165 ms on average). Out of this time, 45% (on average) is spent inside the *Arbitrator* module. Second, VS service termination accounts for all the operations (including interactions) carried out into the 5GT platform stack to remove the inter-PoP logical links, to terminate the VMs, and to terminate the intra-PoP networks created in the PoPs. These interactions are triggered by a message from the 5GT-VS to the 5GT-SO once the decision has been made at the *Arbitrator*, and it involves the SOE, the wrapper, the core MANO platform, the ROE inside the 5GT-SO and its interaction with the 5GT-MTP. The latter, in turn, interacts with the underlying network infrastructure (WAN controllers) and Openstack instances of the involved PoPs (acting as VIMs). Finally, the databases at all layers are also updated. The whole process takes on average 52.020 s. Third, once enough resources are freed for the new service, the EVS instantiation phase accounts for the deployment of the EVS service (80.009 s on average). In this case, the deployed EVS NFV-NS counts a single instance of the CD VNF. As shown in Fig. 10, the addition of these three components (Rx-to-instantiation) results in a total average of 132.194 s, from the reception of the high priority service request to its instantiation after having terminated other low priority service.

D. Resource-level Scaling – Phase 3

We also evaluated the resource-level scaling process by the 5GT-SO, according to the auto-scaling rules that are generated as a consequence of the evaluation presented in Sec. V-A.

Fig. 11 shows the components of the scale-out operation for the EVS service. Out of the total scale-out time (30.862 s on average), the largest time component corresponds to the deployment of a new instance of the CD VNF in a VM (see *Wrapper apply* component of Fig. 11). In this case, this step (30.481 s on average) accounts for the time from requesting scaling to the wrapper (sent by other building blocks of the 5GT-SO), the interaction with OSM and the 5GT-MTP, and the interaction with Openstack to deploy the new VM and to attach it to the corresponding intra-PoP network. The remaining components are much smaller (tens or hundreds of ms), as depicted in Fig. 11 (notice the applied logaritmic scale). First,

the SOE processing (33.8 ms on average) measures the time the SOE takes to handle the scaling request coming from the SLA *Manager* as a consequence of an alert being triggered at the 5GT-MON. This alert (and the associated monitoring job) was previously configured at instantiation time based on the scaling rules (and monitoring jobs) in the NSD of the EVS service. Database update operations are also included in the SOE processing time. Second, the ROE processing (161.8 ms on average) accounts for the time it takes to prepare all the information to be sent to the wrapper to switch from the initial EVS (the running service) to the scaled-out EVS (with an additional CD VNF instance to balance the vertical service load). Furthermore, the ROE is also in charge of creating the new logical links between the new CD VNF and the rest of VNFs of the service (in the same way it was done at the instantiation time for the original one) to maintain the service logic for the vehicles that are going to be served by the new VNF. Third, Wrap decision (163.9 ms on average) accounts for the time to prepare/translate the scaling request received by the wrapper to trigger the associated procedure at the core MANO platform (i.e., OSM). Finally, update monitoring (16.3 ms on average) measures the time to update the monitoring jobs to also monitor the new created CD VNF instance and to make this data accessible through the corresponding interface.

Fig. 12 shows the components of the scale-in operation for the EVS service. in Fig. 12. This is the process through which the 5GT-SO autonomously decides to downscale the resources assigned to the service by changing the scaled-out EVS to the initial EVS IL with only one CD VNF. The scale-in process is triggered by the 5GT-MON by noticing that the CPU consumption is below a certain threshold. The process is equivalent to the above one but for freeing resources (logical links and VMs) instead of creating them. Terminating services and freeing resources (scale in) takes less time (21.051 s on average) than allocating resources (30.862 s on average for scale out), as can be observed in Fig. 12. More specifically, all components described above other than those related with the core MANO platform (i.e., OSM) are very similar: (i) SOE processing (33.5 ms), (ii) ROE processing (163.7 ms), (iii) Wrap decision (157.9 ms), and (iv) Monitoring update (15.0 ms). However, the main component (Wrap apply), related to releasing the resources by interacting with the 5GT-MTP (and Openstack), is 10 s smaller (20.676 s vs. 30.481 s on average).

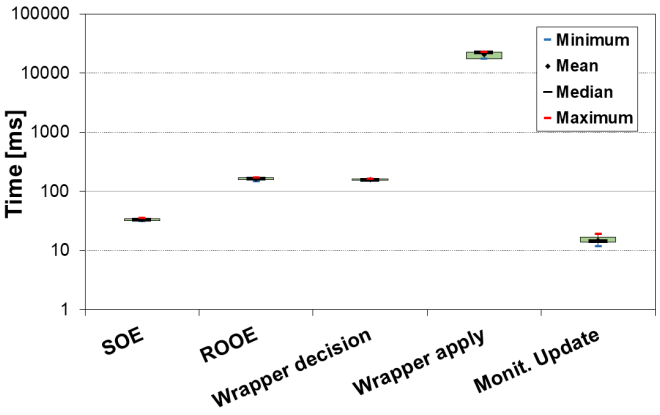

Fig. 12. Scale-in breakdown time analysis for the EVS service

VI. Related Work

Several resource and service orchestration solutions have been investigated (both within and outside NFV scope) to effectively improve user experience and SLAs [23], [24]. *SLA management* is also a quite well-investigated topic within network and service management in different areas, e.g., cloud computing [25], enterprise networks [26], web services [27], and considering multi-domain scenarios [28]. None the less, an important challenge still needs to be addressed: how to automate SLA management operations to avoid, or promptly respond, to possible agreement violations. Existing solutions to this issue often differ in the level of dynamicity of the considered context, i.e., how quickly the resource usage changes. Most of them, however, focus on making monitoring and enforcement tasks as flexible as possible [29].

In the context of network slicing and of the NFV ecosystem, SLA management has mainly focused on the automated scaling of network services, performed by NFV orchestrators to meet the target KPIs despite load surges [30]. In particular, most works have dealt with mechanisms based on virtual resource usage prediction [31], or strategies for virtual resource reallocation [32]. In these cases, policy-based rules and input data used for triggering scaling are provided manually for each network service. In [33], scaling rules are specified in the network service descriptors, which is indeed a preferable option toward more agile and less error-prone scaling solutions in automated slice management operations. In [34], an autonomic policy-based network service deployment with SLA is presented where high-level parameters (performance, availability, security) specified in the SLA are linked to low-level requirements encapsulated in the respective policies. The policy associated with a network service-level SLA is included in a specific descriptor that is created, validated, and uploaded to a catalogue and then applied when the network service is instantiated. Other scaling solutions rely on Artificial Intelligence (AI)-assisted operations to adapt slice or VNF capacity under varying user demand [35] or while minimizing the waste of resources [36].

Recently, consensus has emerged on the need for a network slicing-aware NFV orchestration where management and orchestration functionalities are extended to the slice level (i.e., managed at the telco OSS/BSS), so as to handle end-to-end operations and efficiently address SLAs negotiated with verticals [37]. The study in [38] addresses the modeling, deployment, and orchestration of an end-to-end network slice, which includes the RAN, core, and transport network. Slice management functions are performed through the Open Network Automation Platform (ONAP) [3], and the authors propose an architecture to enforce negotiated SLAs. This solution exploits monitoring information and the policy enforcement component from ONAP to realize automated closed-loop management, while scaling operations are presented at the

[3]https://www.onap.org/

conceptual level. A data-driven approach for intelligent slice management is presented in [39]. Therein the authors propose a framework for data-driven slicing resource provisioning, including the development of slice traffic predictors, resource allocation models, and constrained SLA enforcement. Both the above works, however, address SLA during slice deployment (i.e., SLA enforcement), while they do not focus on scaling operations. Additionally, some recent works have envisioned SLA solutions in the context of network slicing for 5G services, also leveraging AI-based solutions. Most of them, however, address SLA from a theoretical point of view [40], [41], [42], [43], or, even if they present operational solutions to NFV orchestration, they focus on the service deployment phase without specifically addressing scaling operations [44], [45].

To the best of our knowledge, there exist only few works on the application and benchmarking of SLA management in combination of scaling operations, triggered by management and orchestration platforms (MANO) in experimental setups. In addition to the system evaluation we present, and demonstrated in [18], our framework is designed to integrate a hierarchical SLA management in NFV orchestration, involving multiple levels at which service scaling can be handled.

In this context, relevant works to ours are [46][47]. The study in [46] presents a benchmarking analysis with respect to scaling, between the SONATA MANO platform and other open-source MANO solutions like OSM and Cloudify. In [46], however, the SLA management is performed at the NFV-NS level only, unlike in our work where this is just one of the possible levels at which we can act. [47], instead, presents an integrated SLA management framework within the SONATA MANO platform for real 5G environments, aiming at binding business requirements between network operators and verticals, with measurable attributes. The framework proposed therein is demonstrated as a web and multi-platform application that allows the management of the whole SLA lifecycle for a network service. After the network service instantiation into a network slice, the SLA framework is populated with infrastructure monitoring information, in order to assess the agreement with real-time usage data, and efficiently avoid or manage possible violations. However, scaling needs of the verticals or shared slice subnets are not considered in the scaling process.

Ultimately, most of the above previous works on SLA management focus on the network service and the resource level, with the aim to let them adapt to the time varying resource demand as well as resource availability from the underlying infrastructure. To our knowledge, none of them accounts for the vertical services and the application level, or takes into account dynamic changes in the service demand and/or in the application needs.

As far as *service arbitration* is concerned, a large body of work has addressed call and service admission control in the context of wireless networks, focusing on radio resource allocation (see, e.g., [48], or [49] for a survey on this topic). Relevant examples of works on resource allocation include [50], [51]. In particular, [50] leverages reinforcement learning to proportionally allocate budget-constrained radio resources to competing services whose properties are partially unknown at the time of decision making. [51], instead, introduces methods for computational resource arbitration among virtual networks within a node, and for migrating network functions among nodes within a virtual network. Note however that, unlike resource arbitration in 5G, to the best of our knowledge the problem of service arbitration aimed at guaranteeing the SLAs between verticals and the 5G provider has not been previously tackled. Indeed, none of the existing studies have considered the support of vertical services in a 5G network, accounting for SLAs in place between verticals and network provider. This scenario implies not only a finite resource budget for a set of services, but also that resources are properly allocated (i) using a coarse knowledge of the resource status such as that available from a business perspective, and (ii) in a way that the allocation itself can be varied over time so as to adapt to the network and services dynamics as well as to the target KPIs specified by the verticals. An initial study on such aspects was presented in our conference paper [52].

In summary, though previous works dealt with various aspects of SLA management, this has been done in a quite focused way for the specific problem at hand. When moving to complex and heterogeneous virtualized networks, the number of network components, and most importantly, of architectural objects to manage (e.g., vertical service, network slice, network service, virtual function, virtual link) substantially increases. This also has implications in the layers and entities included in the MANO stack (see Fig. 1). Therefore, when deploying a given service, typically there exist SLA constraints involving different architectural objects associated to different architecture layers and stakeholders (e.g., verticals, service providers, operators, infrastructure providers). Unlike previous studies, our work presents a global hierarchical framework for SLA management that is capable of handling SLA at various layers, it is hence concerned with various key architectural objects related with the vertical service, network slice, and network service (including associated resources). Furthermore, the understanding of the dynamics of SLA management and scaling procedures in operational deployments and the availability of representative datasets allow for an efficient integration in our proposed approach of functional AI-based solutions, as presented in [53].

VII. Conclusions

5G networks have expanded the scope of traditional mobile networks to support the digital transformation of vertical industries such as automotive, factories, media, e-Health, and robotics. It follows that nowadays telco providers need to simultaneously deploy and manage multiple vertical services over a shared mobile network infrastructure. In this paper, we presented a hierarchical service and SLA management framework, which leverages service scaling mechanisms at different levels, namely, application-, service- and resource-level, and we have implemented the proposed framework and different scaling functions over the 5GT platform. We demonstrated the performance of our solution using a proof-of-concept testbed. Our results, obtained through the real

field tests with relevant automotive vertical services, show the feasibility of the proposed solution and its ability to automatically deploy and update service instances, while fully meeting the established SLAs. Importantly, the hierarchical service and SLA management framework presented here has been adopted as a baseline solution for future 5G vertical industry technology developments, as the ones considered in the 5Growth project [54][55].

ACKNOWLEDGMENTS

This work has been partially supported by European Commission H2020 5GPPP 5G-TRANSFORMER and 5GROWTH projects (Grants No. 761536 and 856709).

REFERENCES

[1] EU H2020 5G-PPP 5G-TRANSFORMER project, "5G Mobile Transport Platform for Verticals," Available at: http://5g-transformer.eu/, accessed: 2021-02-15.

[2] A. Oliva et al., "5G-TRANSFORMER: Slicing and Orchestrating Transport Networks for Industry Verticals," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 78 – 84, 2018.

[3] ETSI, "3GPP TS 28.541, 5G Network Resource Model (NRM); Stage 2 and state 3 (Release 16), v16.4.1," 2020.

[4] O. Adamuz-Hinojosa, P. Munoz, P. Ameigeiras, and J. M. Lopez-Soler, "Sharing gNB components in RAN slicing: A perspective from 3GPP/NFV standards," *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, Oct 2019. [Online]. Available: http://dx.doi.org/10.1109/CSCN.2019.8931318

[5] R. Ferrus, O. Sallent, and J. e. a. Pérez-Romero, "On the automation of RAN slicing provisioning: solution framework and applicability examples," *EURASIP Journal on Wireless Communications and Networking volume 2019*, Jun 2019.

[6] S. E. Elayoubi, S. B. Jemaa, Z. Altman, and A. Galindo-Serrano, "5G RAN Slicing for Verticals: Enablers and Challenges," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 28–34, 2019.

[7] 3GPP, "Technical Specification Group Services and System Aspects; Management and Orchestration; Concepts, use cases and requirements (Release 17), TS 28.530, v. 17.0.0," Dec. 2020.

[8] J. Mangues-Bafalluy et al., "5G-TRANSFORMER Service Orchestrator: Design Implementation and Evaluation," in *Procs of the 28th European Conf. on Networks and Communications (EuCNC)*, June 2019, pp. 31 –36.

[9] J. Baranda et al., "Realizing the Network Service Federation Vision: Enabling Automated Multidomain Orchestration of Network Services," *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 48–57, 2020.

[10] F. Paganelli, M. Ulema, and B. Martini, "Context-aware Service Composition and Delivery in NGSONs over SDN," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 97–105, 2014.

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

[12] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A Survey on Data Center Networking (DCN): Infrastructure and Operations," *IEEE Comm. surveys & tutorials*, vol. 19, no. 1, pp. 640 – 656, 2017.

[13] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Procs of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, April 2015, pp. 1346–1354.

[14] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "VNF Placement and Resource Allocation for the Support of Vertical Services in 5G Networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 433 – 446, 2019.

[15] L. Kleinrock, *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.

[16] J. Mangues-Bafalluy et al., "Experimental framework and evaluation of the 5G-Crosshaul Control Infrastructure," *Elsevier Computer Standards and Interfaces*, vol. 64, pp. 96–105, 2019.

[17] G. Avino et al., "A MEC-based Extended Virtual Sensing for Automotive Services," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1450–1463, 2019.

[18] J.Baranda et al., "Automated deployment and scaling of automotive safety services in 5G-Transformer," in *IEEE Conf. on Network Function Virtualization and Software Defined Networking, (NFV-SDN)*, Nov. 2019, pp. 1–2.

[19] G. Avino et al., "Support of Safety Services through Vehicular Communications: The Intersection Collision Avoidance Use Case," in *Procs of the IEEE International Conf. of Electrical and Electronic Technologies for Automotive*, July 2018, pp. 1–6.

[20] M. Malinverno, G. Avino, C. Casetti, C. F. Chiasserini, F. Malandrino, and S. Scarpina, "Edge-Based Collision Avoidance for Vehicles and Vulnerable Users: An Architecture Based on MECs," *IEEE Vehicular Technology Magazine*, vol. 15, no. 1, pp. 27 –35, 2019.

[21] P. A. Frangoudis, F. Giannone, A. Ksentini, and L. Valcarenghi, "Orchestrating Heterogeneous MEC-based Applications for Connected Vehicles," *submitted to Elsevier Computer Networks*, 2020.

[22] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012. [Online]. Available: http://elib.dlr.de/80483/

[23] W. Cerroni et al., "Cross-layer Resource Orchestration for cloud service delivery: A seamless SDN approach," *Elsevier Computer Networks*, vol. 87, pp. 16 – 32, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128615001644

[24] M. Irfan et al., "SLA (Service Level Agreement) Driven Orchestration Based New Methodology for Cloud Computing Services," in *Future Optical Materials and Circuit Design*, ser. Advanced Materials Research, vol. 660. Trans Tech Publications Ltd, 4 2013, pp. 196–201.

[25] P. Patel, A. Ranabahu, and A. P. Sheth, "Service Level Agreement in Cloud Computing," in *Wright State University report*, 2009.

[26] J. R. Verma D., Beigi M., "Policy Based SLA Management in Enterprise Networks," in *Lecture Notes in Computer Science*, 1995.

[27] F. Zulkernine, P. Martin, C. Craddock and K. Wilson, "A Policy-Based Middleware for Web Services SLA Negotiation," in *IEEE International Conf. on Web Services (ICWS)*, July 2009, pp. 1043–1050.

[28] P. Bhoj, S. Singhal, and S. Chutani, "SLA Management in Federated Environments," *Elsevier Computer Networks*, vol. 35, no. 1, pp. 5 – 24, 2001, selected Topics in Network and Systems Management. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128600001493

[29] M. A. Rahim, I. U. Haq, H. Durad and E. Schikuta, "Generalized SLA Enforcement Framework Using Feedback Control System," in *Procs of the IEEE International Conf. on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET)*, Dec. 2015.

[30] O. Adamuz-Hinojosa, J. Ordonez-Lucena, P. Ameigeiras, J. J. Ramos-Munoz, D. Lopez, and J. Folgueira, "Automated Network Service Scaling in NFV: Concepts, Mechanisms and Scaling Workflow," in *IEEE Communications Magazine*, vol. 56, no. 7, 2018, pp. 162–169.

[31] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, R. Boutaba, "Topology-aware prediction of virtual network function resource requirements," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 106–120, 2017.

[32] J. G. Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518 – 532, 2016.

[33] A. Boubendir, F. Guillemin, S. Kerboeuf, B. Orlandi, F. Faucheux and J. Lafragette, "Network Slice Life-Cycle Management Towards Automation," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, April 2019, pp. 709–711.

[34] G. Xilouris et al., "Towards autonomic policy-based network service deployment with sla and monitoring," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2018, pp. 1–2.

[35] M. Bouzid, D. H. Luong, D. Kostadinov, Y. Jin, L. Maggi, A. Outtagarts, and A. Aghasaryan, "Cooperative ai-based e2e network slice scaling," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 959–960.

[36] J. Zhou, W. Zhao, and S. Chen, "Dynamic network slice scaling assisted by prediction in 5g network," *IEEE Access*, vol. 8, pp. 133 700–133 712, 2020.

[37] H. Khalili *et al.*, "Network slicing-aware NFV orchestration for 5G service platforms," in *European Conference on Networks and Communications (EuCNC)*, 2019, pp. 25–30.

[38] V. Q. Rodriguez, F. Guillemin, and A. Boubendir, "5G E2E Network Slicing Management with ONAP," in *2020 23rd IEEE Conf. on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Feb. 2020, pp. 87–94.

[39] H. Chergui and C. Verikoukis, "Big data for 5g intelligent network slicing management," *IEEE Network*, vol. 34, no. 4, pp. 56–61, 2020.
[40] M. Iannelli, M. R. Rahman, N. Choi, and L. Wang, "Applying machine learning to end-to-end slice sla decomposition," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 92–99.
[41] H. Chergui and C. Verikoukis, "Offline sla-constrained deep learning for 5g networks reliable and dynamic end-to-end slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 350–360, 2020.
[42] B. Khodapanah, A. Awada, I. Viering, D. Oehmann, M. Simsek, and G. P. Fettweis, "Fulfillment of service level agreements via slice-aware radio resource management in 5g networks," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, 2018, pp. 1–6.
[43] D. De Vleeschauwer, C. Papagianni, and A. Walid, "Decomposing slas for network slicing," *IEEE Communications Letters*, pp. 1–1, 2020.
[44] A. Papageorgiou, A. Fernández-Fernández, L. Ochoa-Aday, M. S. Peláez, and M. Shuaib Siddiqui, "Sla management procedures in 5g slicing-based systems," in *2020 European Conference on Networks and Communications (EuCNC)*, 2020, pp. 7–11.
[45] F. Fossati, S. Moretti, and S. Secci, "Multi-resource allocation for network slicing under service level agreements," in *2019 10th International Conference on Networks of the Future (NoF)*, 2019, pp. 48–53.
[46] P. Trakadas et. al, "Comparison of Management and Orchestration Solutions for the 5G Era," *MDPI, J. Sens. Actuator Networks*, vol. 9, no. 1, p. 4, 2020.
[47] M. Touloupou, E. Kapassa, C. Symvoulidis, P. Stavrianos, and D. Kyriazis, "An Integrated SLA Management Framework in a 5G Environment," in *2019 22nd IEEE Conf. on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Feb. 2019, pp. 233–235.
[48] M. Fidler and V. Sander, "A parameter based admission control for differentiated services networks," *Elsevier Computer Networks*, vol. 44, pp. 463—479, 2004.
[49] M. H. Ahmed, "Call admission control in wireless networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 7, no. 1, pp. 49–68, 2005.
[50] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, 2017.
[51] T. Miyazawa, M. Jibiki, V. P. Kafle, and H. Harai, "Autonomic resource arbitration and service-continuable network function migration along service function chains," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–9.
[52] C. Casetti et al., "Arbitration among vertical services," in *Procs. of the IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 153–157.
[53] J. Baranda et al., "On the Integration of AI/ML-based scaling operations in the 5Growth platform," in *Procs of the 6th IEEE Conference on Network Functions Virtualization and Software Defined Networking (IEEE NFV-SDN 2020), 10-12 November 2020.*
[54] X. Li et al., "5Growth: An End-to-End Service Platform for Automated Deployment and Management of Vertical Services over 5G Networks," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 84–90, March 2021.
[55] EU H2020 5G-PPP 5GROWTH project, "5G-enabled Growth in Vertical Industries," Available at: http://5growth.eu/, accessed: 2021-02-15.

Xi Li (M.Sc.' 2002, Ph.D.'2009) is a Senior Researcher on 5G and 6G R&D at NEC Labs Europe, focused on research of 5G and Beyond Networks. She is the Technical Manager of 5GPPP project 5Growth and has led work packages on the orchestration of 5G mobile networks in the 5GPPP projects 5G-Crosshaul and 5G-TRANSFORMER.

Carla Fabiana Chiasserini (M'98, SM'09, F'18) worked as a visiting researcher at UCSD and as a Visiting Professor at Monash University in 2012 and 2016. She is currently a Professor at Politecnico di Torino and EiC of Computer Communications.

Josep Mangues-Bafalluy (PhD'2003 UPC) is Senior Researcher and Head of the Communication Networks Division of the CTTC. He has participated in various roles (incl. leadership) in several public funded and industrial research projects (e.g., 5GPPP 5Growth or 5G-REFINE). He was vice-chair of IEEE WCNC 2018 (Barcelona).

Jorge Baranda (M.Sc 2008, SM'19) is a Senior Researcher in the Communication Networks Division at CTTC. He has participated in several European, national and industrial projects related with SDN/NFV based orchestration of mobile networks, efficient routing for mobile backhauling and novel wireless communication systems.

Giada Landi (M.Sc 2005) is R&D leader of architectures and network design at Nextworks. She has participated in many industrial and European research projects. She holds 10+ years experience in telecommunication networks, with focus on control plane architectures and protocols, and consultancies on PCE, SDN, and NFV topics.

Barbara Martini is Head of Research at the CNIT National Laboratory of Phononic Networks and Technologies, Pisa, Italy. Her research interests include network virtualization and service platforms for next-generation networks, resource and service orchestration in SDN/NFV/5G environments, security for multi-domain networks, network control/management architectures.

Xavier Costa-Pérez received his M.Sc. and Ph.D. in telecommunications from the Polytechnic University of Catalonia and is ICREA Professor, i2cat Director and Head of 6G R&D at NEC Labs Europe. His team contributes to products roadmap evolution as well as to European Commission R&D collaborative projects and received several awards for successful technology transfers.

Corrado Puligheddu (STM'20) received his MSc in 2019 from Politecnico di Torino, Italy, where he is currently a PhD student. His research interests are in 5G mobile networks, with a focus on ML techniques for the fulfilment of KPI requirements in radio access and edge networks.

Luca Valcarenghi is an Associate Professor at the Scuola Superiore Sant'Anna of Pisa, Italy, since 2014. He has been a Fulbright and JSPS fellowship awardee. His main research interests are optical networks design, analysis, and optimization; communication networks reliability; energy efficiency in communications networks; fixed and mobile network integration; 5G transport. He published more than 200 papers in International Journals and Conference Proceedings.