Collaborative orchestration of multi-domain edges from a Connected, Cooperative and Automated Mobility (CCAM) perspective

Nina Slamnik-Kriještorac*, Girma M. Yilma†, Marco Liebsch†, F. Zarrar Yousaf†, and Johann M. Marquez-Barja*
* University of Antwerp - imec, IDLab - Faculty of Applied Engineering, Belgium.
† NEC Laboratories Europe GmbH, Germany.
E-mail: {Nina.SlamnikKrijestorac, Johann.Marquez-Barja}@uantwerpen.be
{Girma.Yilma, Marco.Liebsch, Zarrar.Yousaf}@neclab.eu

***Abstract*—The 5G ecosystem is comprised of the cellular 5G System, as well as a managed and orchestrated infrastructure providing virtualized network and service functions. The automotive industry with its stringent requirements for connected vehicles is a promising and yet challenging consumer of such 5G ecosystem. Deployment of service instances at distributed cloud resources of cellular network infrastructure edges enables localized low-latency access to these services from moving vehicles but comes along with challenges, such as the need for fast reconfiguration of the distributed deployment according to mobility pattern and associated service and resource demand. In this paper, we investigate a solution for the collaborative orchestration of services for Connected, Cooperative and Automated Mobility (CCAM) within such 5G ecosystem. A key objective is the service continuity for a highly dynamic automotive scenario, achieved by the associated management and orchestration of these services in distributed edge clouds. The proposed solution leverages a multi-tier orchestration system as well as localized management and protocol operations for collaborative edge resources. By means of both analytical and experimental evaluations, the paper draws conclusions on the gain in accelerating orchestration decisions and enforcements, while balancing associated protocol and computational load over the highly distributed and multi-layered orchestration system.**

***Index Terms*—collaborative multi-tier orchestration, 5G ecosystem, CCAM, distributed service deployment, orchestrated network edges**

I. Introduction

The 5^{th} generation of the cellular mobile communication system (5G) is being deployed stepwise in the mobile operators' infrastructures, thereby promising low-latency and high bandwidth communication services to not only mobile devices but also to vertical industries with diverse service requirements in a resource and energy efficient manner. The Network Function Virtualization (NFV), being one of the main technology enablers of 5G, affords the 5G core network architecture to follow a clear control-/data plane separation. This separation enables automated and agile deployment and Life-cycle Management (LCM) of the associated Virtualized Network Functions (VNFs), constituting to deliver customized network services catering to a variety of use cases over the same 5G network infrastructure. Furthermore, Multi-Access Edge Computing (MEC) systems are being widely deployed in the edge networks to deliver a low-latency and localized access to virtualized services by deploying them in close proximity to the users. However, due to the fact that 5G Core, NFV and MEC technologies are being developed by different standardization bodies, the deployment, integration and interplay of these solutions in support of the expected features and end-to-end performance figures of such 5G ecosystem is not coordinated.

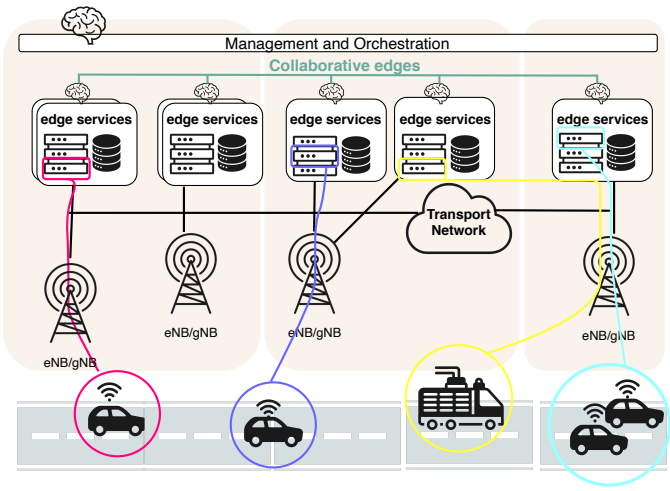

Fig. 1: High-level overview of collaboration between orchestrated network edges that host edge services for vehicles.

The challenge is thus to develop an integrated framework for the automated deployment and orchestration of an end-to-end network in support of the expected service quality. Such framework should span i) the provisioning of virtualized service instances in a centralized cloud, ii) the configuration of a transport network, which connects the service cloud with the cellular network of a mobile network operator, and iii) the configuration of the mobile radio access. The resulting architecture enables full control of the network in between centrally deployed services, and mobile devices, which connect to these services through the cellular 5G network. The automotive industry represents a promising yet challenging consumer of such 5G ecosystem that has the potential of enabling novel and performance critical use cases that were not possible with the previous generations of mobile network systems. This is especially true in the domain of assisted and autonomous driving that primarily relies on real-time and enhanced situation awareness involving high-density, low-latency, and complex processing, of the vehicular sensor data. This entails for consistent quality and low-latency communication with infrastructure service functions. As MEC systems enable low-latency due to exposing resources to the network edge, decentralization and distribution of the virtualized ser-

vice functions towards the cellular network edge help to deploy services topologically closer to vehicles (as depicted in Fig. 1). Such deployment enables the collection, processing, and provisioning, of data locally where they are generated and needed, at the same time shortening the communication path and contributing to a reduced latency as well as to core network traffic offload. However, in such highly agile automotive environment, service continuity in low latency communication with distributed services at the cellular network edge requires real-time monitoring and seamless reconfiguration as well as relocation of the connection to a service instance closer to the vehicle. To enable service continuity and promised Key Performance Indicators (KPIs) (e.g., high reliability, low latency, and high throughput), management and orchestration systems need to be effective to provide distributed service deployment, and seamless service reconfiguration and relocation in such highly mobile and resource constrained ecosystem. Reactive approaches for service continuity, which adjust a configuration after an event happened, such as a vehicle's move to a location which can be served by a closer network edge, are more and more complemented or even replaced by proactive solutions, which leverage data analytics, machine learning, and artificial intelligence for the anticipation of such event and the in-advance preparation of the network.

In this paper we propose and investigate in detail an architecture of a multi-tier orchestration platform for Connected, Cooperative and Automated Mobility (CCAM), and associated operations in support of orchestrated distributed mobile edge networks, in order to enable service continuity for vehicles, which connect to distributed mobile edge services (see Fig. 1). The presented solution extends prior work [1,2] on the end-to-end orchestration in the autonomous operation of orchestration tasks at mobile edge network as well as the connectivity between edge orchestration functions of geographically and topologically adjacent mobile edge networks, aiming at optimized edge-to-edge service continuity by enabling collaboration between mobile edge networks. The proposed orchestration platform aligns with specifications of relevant standardization bodies (i.e., 3^{rd} Generation Partnership Project (3GPP), European Telecommunications Standards Institute (ETSI) MEC, and ETSI NFV) and builds on top of the 5G System specification, which, when compared to previous generations of the mobile communication system, provides various advantages at architectural, protocol, and operational levels. This includes i) the support of a decentralized data plane and edge computing by means of the already mentioned clean control/data plane separation, and ii) the adoption of service-based communication principles and the use of web communication protocols (such as REpresentational State Transfer (REST), and Google Remote Procedure Calls (gRPC)) at the 5G control plane, which eases the integration of and inter-working with control and management functions of accompanying systems, such as edge computing systems and orchestration systems. Specifying the 5G architecture as a set of service producer and service consumer functions, which apply service-based communication, matches a cloud-native design and suits a deployment on top of an NFV infrastructure with automated management and orchestration, as described in this article, with the focus on distributed edge clouds. The promised benefits of 5G system in terms of e.g., the ultra-low latency and high bandwidth depend on the efficiency of the management and orchestration of resources and service, as if there is no collaboration between distributed edge clouds established by orchestration layers, service performance and service continuity will be affected, thereby leading to service performance degradation. The flexible deployment and use of the 5G System's data plane functions and the specified support for Service and Session Continuity (SSC) [3], which permits changes and adjustments in the data plane configuration without disrupting the mobile data session, enables local breakout of mobile data plane traffic and maintains access to edge computing resources and hosted edge services (Fig. 1). The presented solution provides new extensions for MEC-5G System coupling, management and orchestration reference points between mobile edge network orchestration functions, as well as for automated local orchestration at and between edge networks per customized policy for autonomous orchestration tasks, denoted as Management Level Agreements (MLAs) [4].

The analytical and experimental evaluation of the performance of collaborative orchestration is presented to substantiate the design choices that are made to tackle highly mobile use cases with intrinsically distributed service deployments. The evaluation is based on the KPIs associated with a deployment per the proposed architecture. These KPIs are i) the average response time needed for performing orchestration operations, ii) the load of the orchestration entities that needs to be balanced across distributed and multi-layered orchestration systems, and iii) the average power consumption of the performed orchestration requests. In the context of the aforementioned KPIs, it is of utmost importance to assess the load that any orchestration entity is exposed to, making sure that these entities can handle all the orchestration requests in a required response time frame. In particular, with the analytical and experimental evaluation presented in Sections IV and V, respectively, we aim to achieve the following goals:

- To determine how the number of available instances of reference points in the orchestration platform impacts the communication delay in the average response time of an orchestration request, as well as the amount of resources available for performing this request.
- To assess how the number of available instances of reference points between the distributed orchestration components affects the load of the orchestrators at different hierarchical tiers, and how this load further impacts the average response time.
- To showcase the benefits of direct interfaces between orchestrators.
- To determine the impact of orchestration operations on the overall power consumption in the system.
- To showcase how the orchestration operations affect service continuity.

The rest of the paper is organized as follows: in the next section we present the related work and background, where we investigate the existing management and orchestration solutions first, and then provide an overview of today's deployments for connected vehicles, thereby pointing out how the orchestration solutions can support CCAM service deployments. In Section III, we present the collaborative orchestration platform for CCAM, providing its functional overview, followed by the key design features, operational aspects, and software design principles. In Sections IV and V, the analytical model, and experimental evaluation of the orchestration platform for collaborative edges, are presented respectively. In Section VI, we discuss the main findings. Finally, we conclude the paper in Section VII.

II. Background and Related Work

A. Existing Management and Orchestration solutions

As the focus of our paper is on the management and orchestration of collaborative edges in the 5G ecosystem with distributed service deployments, this Section provides an insight in the existing research efforts within related projects, reflecting on the features of existing NFV Management and Orchestration (MANO) solutions that need to be considered in order to properly design an orchestration platform.

According to the overviews provided by Taleb et al. [5] and de Sousa et al. [6], some of the open source orchestration tools that attracted significant attention in past few years are Open Network Automation Platform (ONAP), Open Source MANO (OSM), Open Baton, Sonata (5GTango), Tacker, Cloudify, X-MANO, TeNoR, and Escape. The thorough analysis of NFV MANO solutions, which are either developed or utilized in most of the related projects, is presented in [7–9]. Such analysis is notably important for the development of our orchestration platform for CCAM, and associated network service and resource orchestration operations, because it provides a summarized information on the orchestration platforms, such as ONAP, OSM, Cloudify, among others, which are widely recognized in both industry and academia, serving as guidelines for future extensions of existing orchestrators.

Tackling virtualization environment, the cloud-native deployment of services in the orchestration platform for CCAM is following the principles of containerization, which is gaining momentum due to the opportunities to deploy services and applications in a lightweight manner that is particularly important for resource-constrained MEC platforms [7]. The support for containerization makes NFV MANO solutions (such as Open Baton, Sonata from 5GTango, latest version of OSM, Tacker, Cloudify, and Escape [7]), the valid candidates for orchestration and management of the latency constrained applications. Due to the support for various monitoring tools to be integrated in the orchestration platform for CCAM, and to provide orchestration entities with real-time information on the running edge services, it is possible to provide the VNF self-healing capabilities, decreasing the delay in communication between external monitoring tools and orchestration entities within platform. For example, similar feature is available in ONAP, as well as Sonata from 5GTango.

Since ETSI is the leader in standardizing NFV and MEC, a corresponding NFV MANO tool should be designed and developed with reference to the ETSI NFV MANO framework. This in particular means that, although designed and developed by different vendors/operators/developers, different MEC platforms and applications can cooperate if they are following the standards. Therefore, our orchestration platform is carefully designed with respect to ETSI NFV MEC framework [10], aiming at extending current standards by defining reference points between mobile edge network orchestration functions.

The multi-domain capabilities represent a strong contributing factor to filter the orchestration solutions, being able to establish a connection with MEC platforms from the other edge domains using technologies such as OpenVPN and REST, and to enable communication among different orchestration entities in multiple domains. This feature is of particular importance for our orchestration platform, as it provides distributed service deployment and collaboration between edges in 5G ecosystem. For instance, X-MANO solution [11] introduces the federation over multiple domains through the following core components: 1. Federation Agent (FA), associated to a particular domain in which it interacts with the domain orchestrators, and other modules which are in charge of the life-cycle management within a domain, 2. Federation Manager (FM), which is interfaced with one or more FAs, and 3. OpenVPN as a cross-domain link. Another solution that also considers the federation aspects is ONAP, as its modular and layered nature improves interoperability and simplifies integration, allowing it to support multiple VNF environments by integrating with multiple Virtualized Infrastructure Managers (VIMs), VNF Managers (VNFMs), SDN Controllers, etc. In particular, ONAP's service orchestrator performs orchestration at a high level, with an end-to-end view of the infrastructure, network, and applications. Moreover, ONAP's multi-site state coordination module enables scaling to multi-site environments to support global scale infrastructure requirements. Certain process specifications and policies are geographically distributed to optimize performance and maximize autonomous behavior in federated cloud environments. Furthermore, Escapev2 Orchestrator [12] provides multi-domain NFV orchestration by: i) performing recursive orchestration via north and south Unify interfaces, supporting different legacy technologies and migration between them, and ii) supporting Unify domains directly, and several technological domains via adapters. Finally, TeNoR [13] defines VNF orchestration as a multi-domain problem, considering several Points of Presence (PoPs) in the NFV infrastructure. The TeNoR orchestrator, as a product of FP7 T-NOVA project, is responsible for network services and VNFs' lifecycle management operations over distributed and virtualized network/IT infrastructures.

Although the aforementioned management and orchestration solutions are mature and robust, tackling an end-to-end perspective in virtualized network infrastructure, they are still lacking the support for automated edge-to-edge service deployment that anticipates highly challenging mobile scenarios, thereby enabling fast orchestration operations across different network edges. Another missing link is coupling with 3GPP systems, such as 5G, and design of platform and its operations in accordance with the overall 5G ecosystem. Thus, to enable service continuity in such challenging 5G ecosystem, the orchestration platform that we present in this paper responds to the aforementioned challenges by enabling collaboration between i) orchestrated network edges themselves, and ii) between edges and the 5G System, while taking into account high mobility, and resource and service demand. In such platform, all orchestration tiers collaborate in their orchestration operations for intrinsically distributed service deployments, via fast and dynamic set-up of the management and orchestration reference points between mobile edge network orchestration functions, and by providing an automated orchestration at and between edge networks within the same or different administrative domains (i.e., Mobile Network Operator (MNO)'s domain, country, etc.).

B. Connected vehicles in distributed network edge environments

To extract the potential of providing the localized access to virtualized network resources and services in the 5G ecosystem, i.e., the ultra-reliable and low-latency service deployments, challenges such as resource constraints in network edges, and high user mobility, need to be properly addressed. These challenges become even more severe when considered in highly mobile environments with connected vehicles, since

TABLE I: Overview of today's deployments for connected vehicles.

Technology	DSRC	Cellular			
Bene ts	scalability	large range (i.e., service coverage increased), high capacity, and technological maturity			
Challenges	narrow service coverage, increased communication load, inef cient congestion control, insuf cient reliability	additional delay due to the centralized control			
Type		LTE V2X PC5 sidelink/NR V2X PC5 sidelink		V2N - Uu based/ + support from collaborative orchestration	
		mode 3/mode 1	mode 4/mode 2		
Characteristics		radio resources allocated via cellular network	radio resources allocated simultaneously	vehicles allocating radio resources via cellular network, communication performed via Uu interface	
Suitable use cases		road context information sharing in a close proximity		safety, non-safety, and infotainment V2X use cases that span multiple edge domains	
Challenges		service coverage includes strongly limited number of vehicles		V2N-Uu based only	V2N-Uu + orchestration
				dynamic provision of V2X services due to the high mobility	multi-edge service deployment
		maintaining connectivity between vehicles due to the high mobility		maintaining service continuity	edge-to-edge service continuity
		burden for computing capabilities of a single vehicle due to the broadcast mode of CV2X messages		achieving application portability and immutability	cloud-native service deployment
		insuf cient information for network (re)selection			

they require continuous monitoring of network and computing resources, fast recon guration of service deployments in distributed edges, and following the user mobility patterns, as well as their associated service and resource demand, which all fall under the umbrella of network resource and service management and orchestration tasks.

Since connected vehicles are a valid representative of highly mobile users, in this Section we focus on the automotive class of use cases, as a 5G ecosystem vertical, and investigate the challenges that need to be properly tackled by collaborative service management and orchestration platforms to enable service continuity. As illustrated in Fig. 2, virtualized network services are deployed on top of the distributed edge clouds, and there are different communication technologies that enable connectivity between vehicles and services, and between services themselves. Thus, in Table I we provide an overview of these network technologies, focusing on the bene ts of cellular networks and their coupling with orchestrated edge service deployments, and identifying the bottlenecks that can be mitigated by collaborative orchestration (as shown in the top right column of the Table I).

To alleviate issues on the roads imposed by insuf cient cooperation between vehicles, a signi cant effort is being invested by automotive industry, MNOs, and research institutions, toward enabling vehicles and surrounding infrastructure with the communication capabilities. If equipped with communication engines, vehicles can share information about different events not only with surrounding vehicles (i.e., via Dedicated Short Range Communication (DSRC) and PC5 sidelink), but also with those in a larger vicinity, thanks to the cellular networks and distributed service deployments (Fig. 2).

As presented in Table I, the DSRC based on IEEE 802.11p technology, as well as cellular PC5-based communication, impose constraints to realistic Vehicle-to-Everything (V2X) scenarios in comparison to Vehicle-to-Network (V2N) Uu-based communication. Due to the short range that is covered by internet gateways in case of DSRC [14], and communication only with the vehicles in close proximity in case of

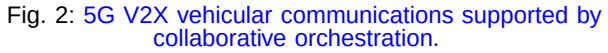

Fig. 2: 5G V2X vehicular communications supported by collaborative orchestration.

PC5 [14–17], it is challenging to cope with high mobility of vehicles on the highways, and to handle the use cases that require extended awareness that spans multiple administrative domains (e.g., countries). Such gaps can be ef ciently bridged by utilizing cellular network infrastructure [14,18–20]. The cellular infrastructure provides suf cient information for: i) central controllers to ef ciently decide on the handover timing [14], and ii) service orchestrators to perform proactive service deployment and service migration from one edge to another. This is not possible in the case of DSRC and PC5 where the local information that each vehicle contains does not involve a broad view of the overall network, thereby leading to inef cient network (re)selection, and reactive service instantiation and migration, which lead to disruptions in service performance. Despite the improved KPIs promised by 5G (i.e., ultra-low latency, high bandwidth, etc.), if management and orchestration of resources and services are not present in the cellular systems, service continuity will not be ensured due to the lack of collaboration between network edges. Thus, Fig. 2

illustrates the multi-edge deployment of cloud-native services that can be efficiently migrated from one edge to another, as a result of management and orchestration operations that take into account the resource constraints, as well as high user mobility. In the second column of Table I, we emphasize how collaborative orchestration can further improve V2N Uu based communication and to support connected vehicles by mitigating the challenges of i) dynamic provision of V2X services due to high mobility by performing multi-edge service deployment (as described in Section III-C), ii) maintaining service continuity by enabling edge-to-edge service continuity (as described in Section III-C), and iii) achieving application portability and immutability by applying cloud-native service deployment (as described in Section III-D), which further facilitates service relocation. The aforementioned benefits for 5G V2X systems are the outcome of collaborative orchestration, thus, it needs to be efficient and robust, as any delay or interruption in performing an orchestration task (e.g., service instantiation, scaling, and termination) can significantly impact the deployment and operation of services used by vehicles, thereby leading to e.g., uncoordinated manoeuvre recommendations, or outdated instructions. Thus, it is essential to carefully study the orchestration concepts, and to build efficient orchestration solutions, to be able to make use of multi-edge deployments and edge-to-edge relocation of cloud-native services, performed in a timely manner.

III. Orchestrated and Collaborative Edges as enabler of Secure and Federated CCAM

Despite the low latency benefits for CCAM services enabled by deploying services close to the vehicles, MEC deployments pose acute challenges in terms of the management and orchestration of virtualized services in a resource constrained and highly distributed environment, which if not properly managed can have adverse impact on the end-to-end service latency and service reliability. This is because of the distributed nature of the multi-domain MEC environment, where even a single domain (e.g., PoP) may have multiple geographically dispersed MEC sites. Each MEC site offers an NFV Infrastructure (NFVI) with limited compute/network/storage resource footprint (i.e., MEC host), managed by a local platform manager/orchestrator. To manage the distributed service deployments across MEC sites i) a coordination between the respective platform managers/orchestrators is required, with an additional coordination in case the service deployment encompasses MEC sites belonging to different administrative domains (e.g., countries), and ii) as service deployments may belong to different tenants, strict isolation between service instances need to be ensured without compromising Quality of Service (QoS).

The aforementioned challenges can be mitigated by enabling collaboration between orchestrated edges via the hierarchical distribution of orchestration tasks, which provides proactive multi-domain service deployments with support for service continuity. Thus, in this section we present the functional overview of the orchestration platform for CCAM among collaborative edges, its design features, operational aspects, and the software design principles. A platform prototype is being deployed in an automotive-related pilot of the 5G-CARMEN project[1], on top of the MNOs' NFV and wireless network infrastructure.

[1]5G-CARMEN: https://5gcarmen.eu/

A. Functional Overview

In Fig. 3, we illustrate the high-level functional architecture of the orchestration platform for CCAM in a federated configuration, indicating the main components that enable secure and federated cross-domain management and orchestration of 5G collaborative edges.

The orchestration platform for CCAM is designed following the cloud native principles while being aligned with the standardization framework provided by ETSI MEC [10], ETSI NFV [21], and 3GPP [3,22]. This design enables collaboration between 5G edges, thereby extending the range of the services/applications running on top of these edges, and allowing them to collaborate with peering service/application instances in different domains in order to enable service continuity.

As illustrated in Fig. 3, the MANO tasks, such as service on-boarding, instantiation, scaling, migration, and termination (more details provided in Section III-C), are performed by hierarchically organized orchestration platform elements that are distributed in two following tiers [23]: i) top-level service orchestration, and ii) edge-level service orchestration. Such functional split enables offloading, or delegating, the orchestration tasks from top-level orchestrator to the edge-level orchestrators in order to decrease the processing load at the top-level orchestrator while enabling low-latency MANO operations directly at the network edges. The top-level orchestrator, characterized by the NFV Service Orchestrators (NFV-SOs), is a centralized service orchestrator that represents larger network domains on the MNO level. On the other hand, the distributed edge-level orchestrators, characterized by a combination of NFV Local Orchestrator (NFV-LO), MEC Application Orchestrator (MEAO), and Edge Controller, are in charge of particular edge domains, within a larger MNO domain, in which the virtualized functions/applications are running. There is a 1:N relationship between the NFV-SO and NFV-LO/MEAO, while there is further a 1:M relationship between the NFV-LO/MEAO and Edge Controller ($1:(M \geq N)$).

The orchestrators interface with each other, and federate with their peer orchestrators in another MNO domain over well-defined reference points. Following are the three main reference points: i) the Or-Or reference point, which is based on the ETSI NFV standard [21], and is responsible for federating between the NFV-SOs in different administrative domains, ii) the Lo-Lo reference point, which is derived from the Or-Or reference point, and enables the coordination between the NFV-LOs for supporting state migration, service continuity, and low-latency service orchestration requirements, and iii) the Or-Lo reference point for coordinating the orchestration tasks between NFV-SO and NFV-LO. The interfaces on these reference points inherit from the standard ETSI NFV/MEC reference point interfaces with relevant extensions, such as Lo-Lo and Or-Lo as described above.

Within a single edge domain, the NFV-LO and MEAO coordinate the LCM of virtualized applications related to low-latency and mission critical services that are deployed in MEC platforms at same or different MEC-sites within an MNO domain. These applications consume MEC Value-added Services (VASs) (e.g., geolocation services, and Radio Network Information Service (RNIS)) to enhance their operation. Each MEC platform, which offers an NFVI, is managed by an Edge Controller which, according to ETSI MEC [10], is in charge of MEC Platform Management, and enforces orchestration and LCM operations as per the directives of the orchestration tiers (i.e., NFV-LO/MEAO). The Edge Controller also supports

Fig. 3: High-level functional architecture of the orchestrated platform for CCAM in a federated con guration.

coupling with the 5G mobile network infrastructure for alignment of connectivity to edge services with device mobility.

B. Key Design Features

Aiming at orchestrated mobile edge networks within a 5G ecosystem, we de ne and comply with the following key design features:

a) Coupling of 5G and MEC/NFV: In the view of an intrinsically sound 5G ecosystem, the so far separately treated speci cations for a 5G System, MEC, and NFV MANO, need to interface and interact for complete end-to-end system management and control. This is to ensure alignment of policies and con gurations associated with a mobile subscriber and its data plane on the one hand side, but to keep a certain level of independence between the two systems for the decision and enforcement of local policies. For this purpose, an Application Function (AF) per the 3GPP architecture speci cation [22] is co-located with the Edge Controller to connect to the 5G System's Control Plane through service-based communication per the 5G architecture's Naf reference point. This reference point enables the retrieval of a mobile subscriber's data plane con guration and to subscribe to events in the 5G Control Plane for receiving event noti cations, e.g., from the 5G Session Management Function (SMF) after a change in a mobile subscriber's User Plane Function (UPF) per SSC mode 3 operations during mobility. The Edge Controller holds the control function of a programmable data plane to enforce traf c treatment rules in alignment with the 5G data plane and to enable, for example, metering and traf c steering within the MEC System's network domain, e.g., for load balancing, failover handling or traf c forwarding towards a different MEC Platform or MEC System.

b) End-to-end mobile data plane control: Complementary to the previously described design feature, this feature leverages the MEC System's awareness of a mobile subscriber's data plane policy and con gurations to enforce aligned traf c treatment rules in between the UPF and the MEC service. This feature builds on top of the 5G System SSC mode 3, which enables mid-session relocation of a mobile subscriber's UPF without breaking the Packet Data Network (PDN) session by a MEC System that is able to follow a relocated UPF of a mobile subscriber connected to a MEC service. Meeting this design feature enables the maintenance of an optimized routing path between a mobile subscriber and its device, i.e., the vehicle, and the mobile network edge service to which it connects. A resulting continuity in a service with short communication paths contributes to the raised low-latency requirement.

c) Delegation of MANO operations in a federated environment: In order to optimize the performance of the MANO operations, one of the design features is the introduction of the concept of MLA [4], which allows for the delegation of MANO tasks/operations between the top-level and edge-level orchestration systems, and also between the peering edge platforms in same and/or different domains. The MLA is negotiated over the Or-Lo reference point between the two tiers within the MNO domain. The MLA also governs the coordination between the peering NFV-LOs over the Lo-Lo reference point. MLA enables the of oading of LCM operations from the top-level to the edge-level orchestrators. Such negotiated agreement determines the operations and functions that the edge-level orchestration entities are allowed to perform within their edge boundaries, thereby executing LCM operation on the relevant service applications and their respective resources [23]. Moreover, the prerequisite for establishing cross-domain federation interface, such as Or-Or and Lo-Lo, is an MLA negotiated between administrative domains, i.e., relevant NFV-SOs. Developing federation over Lo-Lo enables the interworking of edge/MEC and associated edge/MEC platforms, in order to provide a cross-edge on-demand management and orchestration in a collaborative manner, while enabling and maintaining low-latency edge-to-edge CCAM service/session continuity and seamless state migration of users.

(a) Proactive deployment of peering services (Phase 1).

(b) Maintaining service continuity (Phase 2).

Fig. 4: Message sequence chart of orchestration operations in the orchestrated multi-domain MEC system.

```
Steps 1-4: This operation assumes that NFV-SOs a
priori advertise their respective NFV-LOs and
establish MLA via Or-Or interface on a set of
orchestration operations to be delegated to the
respective NFV-LOs to collaborate via Lo-Lo
interface.
Step 5: On-board network service packages, i.e.,
VNFDs and NSDs in participating MEC domains
Steps 6-8: Deploy first instance in domain 1 and
perform LCM operations as required
Steps 9-12: While user is about to move to the next
domain deploy instance in the new domain
Step 13: Perform data sharing between two peering
application instances
Steps 14-17: Migrate important user state
information to next instance to take over the
service, and seamlessly relocate the service
endpoint of the user to the new instance
Step 18: Terminate instances that are not in use
any longer
Step 19: Notify respective NFV-SOs about
termination
Step 20: Repeat steps 9 to 19 as required
Step 21: End
```

Listing 1: Proactive deployment of peering services, and maintaining service continuity in a multi-domain MEC system.

d) Application-specific support for orchestration operations: The edge-level orchestrators constantly monitor the deployed edge services, i.e., edge applications, and allow these application instances to send notifications, as well as triggers for certain orchestration operations. To facilitate and enhance the orchestration operations (e.g., proactive service instantiation, and service migration), the application itself can proactively send notifications to orchestration entities. These notifications may reflect some application-specific data, e.g., retrieved from the data plane packets from users, which are not known by orchestrators. The orchestration entities receive such notifications (e.g., by subscribing to the notification topics with pub/sub, or by receiving them on-demand with request/response), and map them to the policies and necessary orchestration operations. For vehicular applications, such a notification might signal that a vehicle is moving out of the range of a specific MEC host, and that proactive deployment of another application instance, including service migration, will be needed. Thus, this feature is significantly important for our platform as it can leverage the applications for receiving additional information and event notifications in support of orchestration tasks, i.e., to trigger suitable orchestration operations that will enhance the support for service continuity.

C. Operational Aspects of the Orchestrated Platform

As outlined in Section III-B, our orchestration platform supports cross-edge/cross-domain management and orchestration, and thus, NFV/MANO operations that are standardized by ETSI [10,21] need to be optimized to support multi-domain/cross-edge operation. The baseline set of NFV/MANO operations, which our orchestration platform for CCAM supports, consists of: i) application on-boarding, ii) application instantiation, iii) application scaling, iv) application state migration, and v) application termination. Our platform extends beyond these baseline operations to additionally support and enable a) multi-edge service deployment, and to maintain b) edge-to-edge service continuity, the process of which is summarized below with reference to Fig. 4. Listing 1 describes high-level steps of multi-edge service deployment operation, and maintaining edge-to-edge service continuity, presented in Fig. 4 (steps 1-19).

1) Multi-edge service deployment: The operation is depicted in the Phase 1 of Fig. 4. It starts with the top-level orchestrators (i.e., NFV-SO) selecting the edge-level orchestrators (i.e., NFV-LO) that is most appropriate to the service needs (see step 1 in Fig. 4). Note that the NFV-LO selection process is out of scope of this paper. An MLA is negotiated between the NFV-SOs and the selected NFV-LOs within the respective domains in order to grant management autonomy to the NFV-LOs (see step 2 in Fig. 4). For inter-domain operation, a federation is established between the two domains characterized by the establishment of the Or-Or and Lo-Lo reference points between the NFV-SOs and the NFV-LOs respectively [24]. Moreover, the MLAs are also negotiated between the federating NFV-SOs over the Or-Or reference point in order to inform, determine, and harmonize, the scope of management autonomy required between the peering NFV-LOs in order to directly exercise granted LCM operations on the multi-domain deployed application instance over the Lo-Lo reference point (see steps 3 and 4 in Fig. 4).

Prior to the application instantiation, the orchestration platform performs application package (i.e., VNF Descriptors (VNFDs) and Network Service Descriptors (NSDs)) on-boarding as per ETSI NFV rules (step 5 in Fig. 4). In a multi-domain service deployment scenario, the application package can also be proactively on-boarded in the selected peering domain if an MLA exists between these selected platform domains. Afterwards, the NFV-SO will send a service instantiation request, triggered by an authorized external client (e.g., traffic management authority), to the NFV-LO and the service is instantiated (steps 6-8 in Fig. 4). Based on the change in user's location, which is being tracked by the application instance, the NFV-LO will receive notification from the application about the need of a peering application instance in the target domain (step 9). This will prompt the NFV-LO 1 to trigger NFV-LO 2 over the Lo-Lo reference point to instantiate the peering application instance in its domain (see steps 10-12 in Fig. 4) while the vehicle is still in domain 1. Thus, such proactive instantiation of service in the target domain by direct interaction between the peering NFV-LOs and bypassing the NFV-SOs decreases latency in orchestration operation execution.

2) Edge-to-edge service continuity: In view of the stringent QoS requirements imposed by 5G in terms of ultra-low latency (of 1ms-10ms), high capacity (above 100Mbps per user), and reliability (99.999% availability) [25], it becomes imperative for the network service management systems to follow the user mobility, and to place network services always at the most suitable MEC platforms (e.g., the closest one) [26,27], while maintaining edge-to-edge service continuity. In this context, having in place efficient means for service migration and data plane steering is a challenging proposition where service/application instances or users' session states of ongoing services are relocated from one edge to another as the user moves. Since network edges are usually resource constrained (both network and computing), migrating the application or a user's state needs to be network and resource aware and thoroughly orchestrated. To enforce a smooth service relocation strategy, our orchestration platform enables meta-data and state-data sharing between the multi-edge deployed service instances. This enables application instances to share meta-data (step 13), and to transfer application state (e.g., security token) in case of stateful applications (see steps 14-17 in Fig. 4), before a user/vehicle reconnects from source to target instance. The shared meta-data can include the information about the general context of the mobile user/vehicle (i.e. parameters of users' context/session state), such as user's location [28], or Generic Public Subscription Identifier (GPSI) as an identifier in 3GPP, which can further share this data with the target instance, thereby enabling a smooth re-connection of user from one application instance to another. The communication between service instances themselves, and between service instances and vehicles, is accomplished by two types of communication principles, i.e., i) through service based communication leveraging service communication proxies, e.g., to transfer users' session state information to peering instances in adjacent edges, and ii) through fast data I/O interfaces and a programmable data plane to steer and forward data plane traffic to a new location for seamless service continuity (Fig. 3).

D. Software design principles of the orchestration platform

As mentioned above, the design of the orchestration platform for CCAM follows the cloud native principles, which means that all functional elements are implemented as container-based pieces of software rendering a highly modular design. The modularity enables a mix and match of different open source software solutions. For instance, the NFV-SO is based on existing OSM, for the reasons discussed in Section II-B [8]. The interfaces between orchestration components (i.e., Or-Or, Lo-Lo, Or-Lo, Mv1, and NFV-LO-Edge Controller, as presented in Fig. 3) are implemented following the service based architecture. These interfaces use REST based communication.

For the purpose of developing architecture elements, we use the Kubernetes (k8s)[2] platform. As depicted in Figure 3, the MEAO/NFV-LO are implemented as separate containers within a k8s Pod[3], thereby managing the MEC applications and services via a message broker. Similarly, the MEC applications and services are implemented as container applications in different k8s Pods within each MEC host. The on-boarding procedure, described in Section III-C, practically entails the preparation of Docker images for the MEC applications and services on all required edges. Each Pod with an instance of a CCAM service application can be equipped with one or multiple customized network interfaces, such as for service based communication and data sharing with other application instances, or for fast data plane I/O and associated low-latency communication with other application instances or service clients, as described in Section III-C2. These MEC applications and services are grouped in different namespaces to ensure isolation for performance reasons. Moreover, a monitoring service comprising Prometheus and Grafana are configured in a separate monitoring namespace for collecting real-time metrics and usage statistics for all MEC hosts belonging to the edge domain and to be consumed by the orchestration entities. For the management and orchestration of the MEC applications/service an Edge Controller is configured a separate namespace running as k8s Pod.

IV. Analytical Model of Resource Management and Orchestration Operations

In this section we provide the analytical model of resource management in multi-tier hierarchical orchestration platforms that are designed for the 5G ecosystems. We first present the resource assignment problem for the distributed service deployments across network edges, and then provide the latency performance analysis for the orchestration tasks performed by orchestration entities in different tiers. Such analytical approach followed by experimental assessment in Section V can be applied to different orchestrated edge solutions, and here we substantiate our design choices, defined for highly mobile use cases with distributed service deployments.

In particular, the impact of the number of available instances of reference points in a collaborative orchestration platform on latency of an orchestration operation, and on a number of hops for an orchestration request, is further studied and presented in Section V, by analyzing the response time and the load of different orchestration tiers. Thus, in Section V we analyze KPIs in a greater detail, while in Section VI we discuss both the analytical model and the results obtained in experimental assessment. As introduced in Section I, the main evaluation goals that we target to achieve with the analytical evaluation

[2]Kubernetes: https://kubernetes.io/

[3]Kubernetes Pod is the smallest deployable unit of computing that can be created and managed in k8s.

TABLE II: Parameters in the resource management model.

Parameter	Description
Resource assignment problem	
s	top-level service orchestrator (NFV-SO)
l	edge-level orchestrator (NFV-LO)
i	application implementation
n	MEC host/node
$N_L(s)$	number of NFV-LOs in the domain of NFV-SO s
r	resource
k	type of resource
c_{nk}	amount of resources of type k that are available on the n-th MEC host
c_i	cost vector for application implementation i
c_{ik}	cost of resources of type k needed for application implementation i
$d(l)$	administrative domain of l-th NFV-LO
x_{sl}	indicates the relation between s-th NFV-SO and l-th NFV-LO
x_{slin}	decision variable that indicates the ability of l-th NFV-LO to perform orchestration operations on the application implementation i, which is hosted on the n-th node in s-th NFV-SO's domain
Latency performance	
a_{orch}	request for orchestration operation
N_{orch}	number of different orchestration operations
$f(a_{orch})$	traffic generated by orchestration operation request a_{orch}
t	unit time-slot for transmission of an orchestration request via network link
$l_1;l_2$	overall transport network latency
$t_{l_1;l_2}$	tranmission delay
$p_{l_1;l_2}$	propagation delay
$c_{l_1;l_2}$	computing delay
$q_{l_1;l_2}$	queueing delay
$;$	weighting factors that balance network characteristics
$l_{i;j}^{(l_1;l_2)}$	length of the link segment $(i;j)$ that is chained to form the overall link between local orchestrators l_1 and l_2
$B_{i;j}^{(l_1;l_2)}$	bandwidth of the link segment $(i;j)$
s	speed of light

TABLE III: Sets of elements in the resource management model.

Parameter	Description
N_S	number of NFV-SOs, $N_S \in \mathbb{N}$
N_L	number of NFV-LOs, $N_L \in \mathbb{N}$
N_I	number of implementations, $N_I \in \mathbb{N}$
N_H	number of MEC hosts/nodes, $N_H \in \mathbb{N}$
N_K	number of resource types, $N_K \in \mathbb{N}$
S	set of NFV-SOs ($s \in S, S = \{1;\dots;N_S\}$)
L	set of NFV-LOs ($l \in L, L = \{1;\dots;N_L\}$)
I	set of implementations ($i \in I, I = \{1;\dots;N_I\}$)
H	set of MEC hosts ($n \in H, H = \{1;\dots;N_H\}$)
R	set of resource types ($k \in R, R = \{1;\dots;N_K\}$)

in this, and experimental evaluation in the next section, are summarized as follows:

- To determine the impact of the number of available instances of reference point in the orchestration platform on the average response time of an orchestration request, as well as on the amount of resources available for performing this request.
- To assess how the number of available instances of reference points affects the load of the top-level orchestrator and how this load further impacts the average response time.
- To show the benefits of direct links between edge-level orchestrators.
- To test the power efficiency of the orchestration platform.
- To study how the orchestration operations affect service continuity.

TABLE IV: Scenarios for calculating the total number of reference points.

Scenario	Number of NFV-SOs	Number of NFV-LOs in NFV-SO 1 domain	Number of NFV-LOs in NFV-SO 2 domain
I	2	2	1
II	2	1	1
III	2	3	2

A. Resource assignment problem

The analytical model of our collaborative orchestration platform defines the resource assignment problem as an integer program. In the Table II, we present the parameters that are utilized in the analytical model. The resource assignment problem refers to the resources that can be assigned to edge-level/local orchestrators, i.e., NFV-LOs, in order to perform orchestration operations for the requested MEC applications.

In this analytical model, we consider the orchestration platform for CCAM as a hierarchical NFV management and orchestration architecture that consists of the top-level, and the edge-level orchestrators, i.e., NFV-SOs and NFV-LOs, respectively, and as described in Section III, we consider three types of reference points that connect them, i.e., Or-Or, Lo-Lo, and Or-Lo. The sets of elements used in our analytical model are shown in Table III.

Depending on the MLAs that are agreed between NFV-SOs and NFV-LOs in all edge and administrative domains, there is a different number of interfaces that are established on-demand between different orchestration entities. Therefore, the equation (1) represents the total number of interfaces that are established on-demand between: i) all existing NFV-LOs and NFV-SOs (Or-Lo), enabled by MLA type m_1, ii) all existing NFV-SOs between themselves (Or-Or), enabled by MLA type m_2, and iii) all existing NFV-LOs between themselves (Lo-Lo), enabled by MLA type m_3.

The value calculated in (1) is smaller or equal than the maximum number of interfaces that can be established (e.g., all NFV-LOs from all edge and administrative domains are connected directly to each other, being at the same time connected to their respective NFV-SOs). In particular, the MLAs that enable the establishment of particular reference points in the orchestration platform can be considered as a triplet, i.e., $(m_1;m_2;m_3)$. Such a triplet refers to a permutation of the three types of MLAs, i.e., m_1, m_2, m_3, which enable establishment of Or-Lo, Or-Or, and Lo-Lo, reference points, respectively. In Fig. 5, we illustrate the three examples of arrangement of the architectural elements (i.e., NFV-LOs and NFV-SOs), and pair them with the corresponding triplet. Each triplet in practice means that certain permutation of agreements (i.e., MLAs) has been achieved between the top-level and edge-level orchestrators from different edge and administrative domains, thereby allowing edge-level orchestrators to consume resources from different domains to perform orchestration operations. In particular, the simplest scenario is shown in Fig. 5 a), which depicts the case when there is only one administrative domain, e.g., no collaboration between MNOs from different countries is present, and edge-level orchestrators are allowed to orchestrate only those resources that belong to their edge domains.

Both b) and c) in Fig. 5 depict the collaboration between the top-level orchestrators, but these two scenarios differ in terms of agreements between the edge-level and the top-level orchestrators, and between the edge-level orchestrators themselves.

Fig. 5: The example of hierarchical NFV management and orchestration in the orchestration platform for CCAM (Scenario III from Table IV)

However, some of the triplets/permutations are not possible, such as $(m_1; m_2; m_3) = (1; 0; 1)$, because it is required to first establish federation between the top-level orchestrators, i.e., Or-Or reference points, in order to enable the direct Lo-Lo links between the edge-level orchestrators. This means that two NFV-LOs cannot cooperate via Lo-Lo link unless the federation between different administrative domains has been established. Hence, the complete list of MLA triplets for our orchestration platform is given as follows: $(m_1; m_2; m_3) = \{(0; 0; 0); (0; 1; 0); (1; 0; 0); (1; 1; 0); (1; 1; 1)\}$, where the most complete case, i.e. $(m_1; m_2; m_3) = (1; 1; 1)$, means that all instances of reference points are established between components in the architecture of our orchestration platform, as illustrated in example shown in Fig. 5 c).

Objective 1: To maximize utility function $U_1(i)$ that determines the number of available instances of reference points in the orchestration platform.

$$U_1(i) = \sum_{s_1=1}^{N_S} \sum_{s_2=1; s_1 \neq s_2}^{N_S} x_{s_1 s_2} \, m_{2 s_1 s_2} + \sum_{s=1}^{N_S} \sum_{l=1}^{N_S(s)} x_{sl} \, m_{1sl} + \sum_{l_1=1}^{N_L} \sum_{l_2=1; l_1 \neq l_2}^{N_L} x_{l_1 l_2} \, m_{3 l_1 l_2};$$

$$U_1(i) \quad \frac{1}{2} \left(N_S(N_S - 1) + N_L(N_L - 1) + 2N_L \right);$$

$$x_{s_1 s_2} = \frac{1}{2}; \quad x_{l_1 l_2} = \frac{1}{2};$$

$$d(l) = d(s) \; ! \; x_{sl} = 1;$$

$$d(l) \neq d(s) \; ! \; x_{sl} = 0: \tag{1}$$

To exemplify the number of instances of reference points that can be established in the orchestration platform, we consider three different scenarios that are defined in Table IV. Applying the equation (1) on these three scenarios, Fig. 6a shows the total number of instances of reference points that can be established for different MLA triplets, with the assumption that if $m_1 = 1$, then $m_1 = 1, \forall l \in L \wedge \forall s \in S$. With reference to MLAs that are previously described in the form of triplets, we can draw an important conclusion about the number of hops that a request for a certain orchestration operation needs to pass to reach the final destination. For example, if NFV-LO from one administrative domain needs to extend the scope of application implementation that is running under its scope (i.e., the edge domain), it will send a request for application instantiation in other edge domain, either in the same or in other administrative domain. Thus, the level of agreement between the administrative domains, as well as the edge domains within their scope, defines the number of hops, i.e., n_h (equation (2)), which needs to be minimized in order to ensure lower latency while maintaining the service continuity.

$$n_h = \begin{cases} 2; & m_1 = 1 \wedge m_3 \neq 1 \wedge d(l_1) = d(l_2) \\ 1; & m_1 = 1 \wedge m_3 = 1 \wedge d(l_1) = d(l_2) \\ 3; & m_1 = 1 \wedge m_2 = 1 \wedge m_3 \neq 1 \wedge d(l_1) \neq d(l_2) \\ 1; & m_1 = 1 \wedge m_2 = 1 \wedge m_3 = 1 \wedge d(l_1) \neq d(l_2) \end{cases} \tag{2}$$

Whether both edge-level orchestrators belong to the same administrative domain $d(l_1) = d(l_2)$, or to different administrative domains $d(l_1) \neq d(l_2)$, Fig. 6b shows the number of hops for an orchestration request from an arbitrarily defined edge-level orchestrator NFV-LO, which needs to reach another NFV-LO.

If we consider now the resources that can be assigned to a particular NFV-LO to perform orchestration operations, the variable x_{slin} represents a decision variable that is equal to one, if an instance of application implementation i has been assigned to l-th NFV-LO. Thus, the l-th NFV-LO can consume resources from n-th MEC host in s-th NFV-SO domain (i.e., MEC hosts that are available in NFV-SO domain). Otherwise, if the aforementioned combination is not allowed by MLA, the value of decision variable x_{slin} is equal to zero.

The amount of resources of type k that are available on n-th MEC host that is orchestrated by l-th NFV-LO, in s-th NFV-SO domain, is defined as a_{slnk}. Hence, if the federation and MLAs are agreed (either only Or-Or, or both Or-Or, and Lo-Lo, are established), the scope of resources, which NFV-LO orchestrator is allowed to consume in order to perform orchestration operations, is extended to multiple domains.

In inequation (3), the overall amount of resources of type k, which are given at disposal for performing orchestration operations on the i-th application implementation that is deployed on the n-th MEC host, cannot exceed the maximum amount of available k-type resources on this node. Let us assume that the system consists of two NFV-SOs, as described in Scenario III from the Table IV, and illustrated in Fig. 5 c), thereby spanning three, and two edge domains, respectively. Each of these edge domains is orchestrated by one of the N_L NFV-LO orchestrators. If n-th MEC host is located in the domain of a particular top level orchestrator, i.e., s-th NFV-SO, then the non-negative integer m_{1sl} determines whether l-th NFV-LO is allowed to consume k-type resources of n-th node located in the domain of s-th NFV-SO. Therefore, in the system that we previously described and illustrated in Fig. 5 c), if n-th MEC host is located in the domain of s_1 (e.g., MEC host 1, i.e., $n = 1$), then $m_{1s_1l_1}$ determines whether NFV-LO l_1 can consume k-type resources from this node or not. For example, the sum member for the combination $(s_2; l_1)$ is equal to zero in that case, because the resource will be already given to l_1 by s_1, as the selected MEC host is in the domain of s_1. Furthermore, for the combination $(s_1; l_4)$, the sum will be non-zero in case there is at least Or-Or interface established

(a) Number of instances of reference points (scenarios I, II, and III).

(b) Number of hops for orchestration requests in case $d(l_1) = d(l_2)$, and $d(l_1) \neq d(l_2)$.

(c) Latency (transmission and propagation) depending on the network bandwidth.

Fig. 6: Number of instances of reference points and number of hops for orchestration requests depending on different combinations of (m_1, m_2, m_3), and latency of transmission and propagation.

between s_1 and s_2. Thus, all NFV-SOs allow any NFV-LO to consume resources from their domains, but if $m_1 = 1$ for the l_i-th NFV-LO that is not in the domain of s_j-th NFV-SO (i.e., $d(l_i) \neq d(s_j)$), this means that $m_{2s_j s_j} = 1$, i.e., the federation between the top-level orchestrators is established.

Objective 2: To maximize utility function $U_2(i)$ that determines the amount of resources, which are distributed in the hierarchical orchestration platform for CCAM, and given at disposal for performing orchestration operations on the application implementation i.

$$U_2(i) = \sum_{s=1}^{N_S} \sum_{l=1}^{N_L} \sum_{k=1}^{N_K} r_{slink}(m_{1sl}) \Big/ \sum_{s=1}^{N_S} \sum_{l=1}^{N_L} r_{slink}(m_{1sl}) \; n_{ik} \tag{3}$$

$$\sum_{s=1}^{N_S} \sum_{i=1}^{N_I} \sum_{n=1}^{N_H} x_{slin}(m_{1sl}) \; c_{ik} \; n_{nk}, \; \forall i \in I, k \in R \tag{4}$$

The left side of inequation (4) expresses the amount of resources of type k, which are available in all MEC hosts (from all NFV-SO domains) and given at disposal to the l-th NFV-LO to perform orchestration operations. As it can be seen from (3) and (4), the resource availability is bounded by agreed level of MLA (i.e., m_{1sl}). The utility function illustrated by equation (5) models the overall utility $U_{slin}(x_{sl})$ that the system gains by assigning c_i resources to l-th NFV-LO, allowing it to deploy i-th instance of application implementation to its assignment vector x_{sl}. The assignment vector $x_{sl} \in \{0,1\}^{N_H \; N_I}$ refers to the combination of l-th NFV-LO and s-th NFV-SO, which has a task to deploy the instances of application implementations on top of the MEC hosts, and to perform orchestration operations on these instances.

Objective 3: To maximize utility function, which depends on: i) the MLAs that allow NFV-LOs to operate in a resource extended manner, which means that NFV-LOs can rely on the resources from other edge networks/domains to perform their orchestration operations, ii) the selected NFV-LO in particular NFV-SO domain to perform the orchestration operations, iii) the chosen implementation for the application instance, and iv) the selected MEC host for the deployment.

$$U = \sum_{s=1}^{N_S} \sum_{l=1}^{N_L} \sum_{i=1}^{N_I} \sum_{n=1}^{N_H} U_{slin}(x_{sl}) \; x_{slin}(m_{1sl}) \tag{5}$$

Therefore, aiming to achieve the Objective 3 that refers to the overall orchestration platform for CCAM, there is a need to achieve Objective 1 and Objective 2, for all NFV-LOs in the orchestration platform.

B. Latency performance

Here we tackle the system model for describing latency performance over the Lo-Lo, i.e., the direct link between edge-level orchestrators (i.e., NFV-LOs). This direct link is used to transfer a request for any orchestration operation that is allowed to be requested or recommended from one local orchestrator to another, as described in Section III. Henceforth, the overall transport network latency for such request can be defined as a cost function (equation (6)) that consists of the transmission delay, the propagation delay, the computational delay, and the queuing delay [29,30]. If we define the request for orchestration operation as a_{orch}, where $a_{orch} \in O$, $O = \{1, \ldots, N_{orch}\}$, then the traffic that is generated by this request can be denoted as $f(a_{orch})$. Therefore, the transmission delay is defined as a time needed for processing an orchestration request on the transmitter side (NFV-LO in domain 1), and it can be expressed as a fraction of the traffic that this request generates and the bandwidth $B^{(l_1;l_2)}_{i;j}$ of the processing link between two local orchestrators (i.e., l_1 and l_2-th). As the fraction of the traffic and the capacity determines only the number of time-slots that are required by processing link to start transferring the request, it needs to be multiplied by a unit duration of a time-slot, i.e., t, in order to calculate the overall transmission delay. In the overall transport network latency $_{l_1;l_2}$, parameters and are weighting factors that balance the networking characteristics [30]. The propagation delay depends on the length of the link between two orchestrators, and the overall propagation speed over the wireless link.

$$\begin{aligned} {}_{l_1;l_2} &= t_{l_1;l_2} + p_{l_1;l_2} + c_{l_1;l_2} + q_{l_1;l_2} \\ {}_{l_1;l_2} &= \sum_{i;j \; 2 L_{l_1;l_2}} \frac{f(a_{orch})}{B^{(l_1;l_2)}_{i;j}} t + \\ &\sum_{i;j \; 2 L_{l_1;l_2}} \frac{l^{(l_1;l_2)}_{i;j}(m_{sl_1 l_2})}{s} + c_{l_1;l_2} + q_{l_1;l_2} \end{aligned} \tag{6}$$

Let us consider that the link, which is used for transmission of the request for orchestration operation, is consisted of multiple segments, i.e., $(i;j)$ with the length $l_{i;j}$, where $i;j \; 2 L_{l_1;l_2}$. In particular, $L_{l_1;l_2}$ is the set of all link segments that can be chained to form the link between the local orchestrators l_1 and l_2, i.e., $(l_1;l_2)$. The length of the link between two

TABLE V: System characteristics

System information		
Type	Virtual Wall node	CityLab node
Reference	Node 1	Node 2
CPU (GHz)	2.252	1
System memory	48GB RAM	4GB DDR3-1333 MHz
Processor	2x 8core Intel E5-2650v2 (2.6GHz) CPU	AMD GX-412TC 1GHz Quad-core CPU
Storage	250GB	240 GB
Disk	250GB HGST HTS725025A7	240GB Samsung SSD 850

local orchestrator depends on the $m_{sl_1l_2}$ parameter, which determines whether there is a direct link between orchestrators or this link consists of the link segments that also include top-level orchestrators in the chain. Thus, taking into account the de nition of the parameter $m_{sl_1l_2}$ ($m_{1l_il_j}$, $m_{2l_il_j}$, or $m_{3l_il_j}$), it is intuitive to conclude that the overall length of the link between NFV-LOs, i.e., $l_{l_1;l_2}$ will be larger if the request from one local orchestrator needs to be passed to the top-level orchestrators rst, and not directly via the Lo-Lo link. With regards to the aforementioned, the main contributing factors to the overall transport network latency are the network bandwidth ($B_{l_1;l_2}$), and the distance that orchestration operation request needs to propagate to reach NFV-LO 2 from NFV-LO 1, or vice versa. In Fig. 6c, we show the latency that consists of transmission and propagation delay de ned in equation (6), which is calculated for a simple orchestration request (simple request carrying 13KB of data, as described in Section V) depending on the bandwidth of the network links, and the distance between respective orchestrators. Hence, the latency will be higher in case of the lower network bandwidth, but also in the case of larger distances between local orchestrators (i.e., links NFV-LO 1 - NFV-SO 1 - NFV-SO 2 - NFV-LO 2, or NFV-LO 1 - NFV-LO 2). For example, Maheshwari et al. [29] show that the average response time of servers in cloud and edge also increases with an increase in system load, which is of course affected by computation, i.e., processing of the request on the orchestrator side in our case. Thus, in Section V, we assess the overall latency, i.e., the average response time for orchestration requests generated by edge-level orchestrators.

V. Experimental assessment of the orchestration platform

In this section, we present the experimental assessment of the orchestration platform for collaborative edges, thereby demonstrating the relevance of the design choices that we made for the architecture itself, but also for the operational aspects of such orchestration platform.

A. Experimental setup

In order to conduct experiments that assess their performance, as well as the capacity to perform the orchestration operations, we de ned and performed a set of tests for both top-level and edge-level orchestrators in the testbed environment.

1) Testbed environment:The system characteristics of computing machines that we used in experimentation are listed in Table V. Taking into account the characteristics of the top-level orchestrator (presented in Section III), it is expected to run on top of the resourceful computing machines (such as Node 1 in Table V), as it serves all underlying edge domains while covering the whole administrative domain (e.g., one country). Thus, in our experiments, we leveraged on the computing capabilities of the Virtual Wall testbed [31] (Fig. 7) for the purpose of testing the response to orchestration requests of the top-level orchestrator, as well as to evaluate its average load. The Virtual Wall testbed, located in Gent, Belgium, consists of more than 550 powerful bare metal and GPU servers, which are software and hardware con gurable, i.e., con gurable in terms of software installation (e.g., operating systems, and drivers), and networking via con guring the physical interconnection between network interfaces. All these machines forge a generic environment for advanced networking, distributed software, cloud, big data, and scalability research and testing.[4]

On the other hand, the edge-level orchestrator is designed to cover smaller areas, i.e., edge domains, while performing management and orchestration operations of the deployed edge services, but also responding to the requests that are coming from adjacent or other edge domains. Thus, for testing the capabilities of an edge-level orchestrator, we utilized the CityLab[5] testbed [32] (Fig. 7), i.e., the resource constrained Node 2 presented in Table V. In particular, CityLab is a smart city large-scale wireless networking testbed, which is located in Antwerp, Belgium, whereas the experimentation nodes are attached to buildings and streetlamps providing the opportunities for experimentation at a city neighborhood level in the unlicensed spectrum. We have used public internet to establish the connectivity between different orchestration entities in this testbed environment, i.e., Lo-Lo reference point between edge-level orchestrators, and Or-Lo between top-level and edge-level orchestrators.

2) Types of tests:For both the top-level and edge-level orchestrators, we performed different types of tests, as described in Table VI. The local tests refer to the tests in which server (i.e., the orchestrator) and client (i.e., load testing tool) are deployed on top of the two separate bare-metal machines that are connected by wire. Accordingly, in the remote test, server and client are dislocated, and there is an additional contributor to the overall latency, which is imposed by sending orchestration requests via public Internet (Fig. 7).

For the local tests, we conducted experiments with different test variants, which differ in complexity of the orchestration request. With reference to the software design of our orchestration platform presented in Section III, each orchestration request is generated, received, and processed, as a REST Application Programming Interface (API) request. Therefore, we differentiate the complexity of different requests by performing: i) only simple GET requests, containing relatively small body (i.e., average content size), ii) only GET requests that involve certain transactions and checkups in database, and iii) a combination of GET and PUT requests, where PUT requests usually refer to those requests that require changes in the service deployments, re ected by applying changes in database as well. We designed the combined test in a way it generates three times more GET requests than PUT requests, as there are usually more query types of orchestration requests, where different orchestration entities ask other orchestrators about the state of a deployed service, and some of its particular parameters, than those requests that involve actions on application/service as an outcome of the orchestration algorithms (e.g., scaling up/down/in/out).

3) Load tester Locust:To generate orchestration requests and test performance of the orchestrators, we used a python-

[4]Virtual Wall testbed: https://www.ugent.be/ea/idlab/en/research/research-infrastructure/virtual-wall.htm
[5]CityLab testbed: https://doc.lab.cityofthings.eu/wiki/Main_Page

Fig. 7: The nodes used for local and remote tests on top of the Virtual Wall and CityLab testbeds.

TABLE VI: Description of the tests.

Type of machine	Platform component	Test	Type of request	Average content size (B)	
Node 1	Top-level orchestrator	Local	simple - only GET	13	
			only GET	400	
			GET & PUT	GET	140
				PUT	54
		Remote	simple - only GET	13	
Node 2	Local orchestrator	Local	simple - only GET	13	
			only GET	400	
			GET & PUT	GET	140
				PUT	54

based performance testing tool Locust[6]. As Locust is widely used for performing stress and load tests on web servers, it is suitable for our experiments as both top-level and edge-level orchestrators are here deployed and tested as web-based servers, using python web framework Flask.

Locust enables de ning the behavior of users in a regular Python code, running each of the users inside its own greenlet, i.e., a lightweight process, without the need for using call-backs. In the case of top-level orchestrator, users that generate requests are its underlying local, i.e., edge-level orchestrators, which are sending the orchestration requests. Similarly, in the case of edge-level orchestrators, users are other (adjacent or not) edge-level orchestrators that are directly connected to each other via low-latency Lo-Lo link.

4) Metrics: In all the tests that are executed, a several important KPIs are measured, which are relevant because they re ect the capability of an orchestrator to perform orchestration operations ef ciently, as well as the amount of resources that it consumes for its work. These KPIs are: i) average response time per orchestration request, ii) average Central Processing Unit (CPU) load, iii) average Random-Access Memory (RAM) load, and iv) average power consumption, and they are described as follows. For instance, the average response time of both top-level and edge-level orchestrators is the overall latency of performing a particular orchestration request, from the moment when the request is generated in the edge-level or top-level orchestrator, to the moment when this request is processed. With reference to our analytical model presented in Section IV, the latency performance model includes an orchestration request a_{orch} that generates a certain amount of traf c f (a_{orch}). In the case of this evaluation, the response time in remote tests also includes the propagation and transmission latency as a result of sending an orchestration requesta_{orch} through the communication link between two orchestrators, as well as the aforementioned time to process the upcoming traf cf (a_{orch}).

We need to make sure that both the top-level and edge-level orchestrators can handle the load of orchestration requests. Hence, the CPU and RAM load refer to the load that orchestrator can expect and experience when certain number of orchestration requests are received, which is a direct implication of the MLAs that determine a number of established interfaces between orchestration entities as described in Section IV-A (Objective 1). The goal of measuring these KPIs is to assess the average behavior of both resourceful, and resource-constrained machines, which can host top-level and edge-level orchestrators, respectively. As we presented in Section IV, in case no direct link between edge-level orchestrators is established by MLA, it ultimately results in an increase in number of orchestration requests towards the top-level orchestrator. That is why in our tests we aim to assess the impact of such increase in number of requests, on the performance of the top-level orchestrator, and to evaluate the burden it imposes to the operations in the lop-level orchestrator.

In the experimental evaluation we also measure the average power consumption of the top-level and the edge-level orchestrators, while they are performing orchestration requests. Since energy ef ciency is considered as one of the ultimate goals of 5G ecosystem [33], the applications and processes that are executed on the edge and cloud computing devices need to be energy ef cient. According to the European Commission's nal study report on energy ef cient cloud computing technologies [34], the design of any application has a high impact on its energy consumption. This becomes even more evident when similar applications may require different consumption of CPU load, and memory load, and ultimately different energy consumption. Therefore, in [34] it is stated that software is a major factor for energy-ef ciency when the energy consumption is measured for a cloud computing product. Thus, it is important to measure the impact of orchestration operations on the energy and power consumption, thereby designing orchestration solutions to be low energy consuming techniques.

The experiments that we described in this section enabled us to evaluate a relative average response time, and CPU/RAM

[6]Locust: https://docs.locust.io/en/stable/

TABLE VII: Average RAM load.

Orchestrator	Type of test		Waiting time (s)		
			1	0.5	0.1
			Average RAM (%)		
Top-level orchestrator	local	simple - only GET	0.2	0.2	0.2
	local	only GET	0.5	0.5	0.5
	local	GET & PUT	0.2	0.51	0.68
	remote	simple - only GET	0.2	0.2	0.2
Edge-level orchestrator	local	simple - only GET	0.7	0.7	0.7
	local	only GET	0.8	0.8	0.8
	local	GET & PUT	0.8	0.8	0.8

load, and average power consumption, for orchestration requests that originate at the edge-level orchestrator for example and terminate on another edge-level orchestrator in case there is a direct link between these two edge-level orchestrators, and in case the orchestration requests need to be forwarded via top-level orchestrators. All the results that we present in the following section reflect the relative behavior of orchestration entities within our orchestration platform, because this behavior depends on the type of machine that hosts the orchestrator, the type of the orchestrator, and the complexity of orchestration operations that this orchestrator performs.

B. Results

Let us consider a scenario with one top-level orchestrator per whole administrative domain (e.g., country), and multiple edge-level orchestrators, with no direct Lo-Lo link established between them as per definition in equation (1) (Section IV-A). In such case, all the traffic that edge-level orchestrators generate in their domains by sending orchestration operation requests towards other edge domains, first reaches their responsible top-level orchestrator. In Figures 8a and 9a, and Table VIII, we can clearly see the increasing trend in CPU load and average response time, respectively, for the top-level orchestrator with the number of edge-level orchestrators that are simultaneously sending orchestration requests towards it. The same trend applies to the edge-level orchestrators (Figures 8b and 9b, and Table IX) with the increase in total number of direct Lo-Lo connections.

For each total number of edge-level orchestrators, and direct Lo-Lo connections, shown on the x-axis of all graphs shown in Figures 8, and 9, and Tables VIII, and IX, we run tests for different waiting time between successive requests that are coming from a single edge-level orchestrator. It means that in case of waiting time equal to 1s, each edge-level orchestrator is generating one orchestration request per second. Accordingly, each of them is generating 10 orchestration requests per second in case of waiting time equal to 0.1s. Therefore, in case there are 100 edge-level orchestrators distributed across a single administrative domain, the top-level orchestrator needs 68ms in average to process a simple orchestration request (e.g., response to a query about resource availability in a certain edge domain). In practice, that means that edge-level orchestrator will wait 68ms only for the first top-level orchestrator to process its request, which will then include also an additional latency that propagation and transmission of this request, as per equation (6) in Section IV, take from i) local to the top-level orchestrator, ii) from the top-level orchestrator in domain 1 to the top-level orchestrator in domain 2, and iii) from the top-level orchestrator in domain 2 to the target edge-level orchestrator in domain 2. On the other hand, if a direct Lo-Lo link is established from originating to the target edge-level orchestrator, Fig. 9b shows that one edge-level orchestrator (although resource constrained) will take only 19ms to process the same orchestration request. Taking into account the propagation latency in equation (6), we can assume that the overall latency via link Lo-Lo will be lower than in case when request is sent through the top-level orchestrators (Section IV-B), which in total results in a at least three times lower latency in processing orchestration request in case of having Lo-Lo link.

If we now reflect on the remote test for the top-level orchestrator, which is depicted in Fig. 7, the increase in average response time per orchestration request can be seen in Fig. 9c in comparison to Fig. 9a. For example, in the case 10 edge-level orchestrators are simultaneously sending two requests per second towards the top-level orchestrator, we can see that in remote test the average response time is 527ms while being only 20ms in the local test. Such an increase in average response time is expected due to delay in sending orchestration requests via public internet, as well as queuing in the gateways, highly depending on the number of the network links between orchestrators, their length and of course bandwidth. As such result might severely disrupt the performance of vehicular applications, especially the latency constrained ones, due to the increase in orchestration execution, we emphasize the importance of the direct low-latency Lo-Lo links that should significantly decrease the overall delay. A further reduction of the latency, caused by congested network nodes, can be achieved also by dedicating more processing power, or more network adapters, to a particular orchestrator.

Tackling the load that the top-level orchestrators need to handle in case the MLA do not allow edge-level orchestrators to directly collaborate via low-latency links, we assess the average CPU load (Fig. 8, and Tables VIII, and IX), as well as the average RAM load (Table VII). The average RAM load remains stable in all tests, being slightly increased with complexity of orchestration requests, whereas the average CPU load is highly affected by the amount of orchestration tasks to process. In particular, Figures 8a and 8b, and Tables VIII and IX show that for both top-level and edge-level orchestrators, the average CPU load increases with the number of edge-level orchestrators generating requests, and with the number of requests per second. One specific case when this load decreases is the GET & PUT test, in which the average CPU load for 100 and 300 edge-level orchestrators (Table VIII) is smaller than in case of less complex tests (Fig. 8a). This decrease happens due to request queuing that significantly increases average response time (Table VIII), which also results in failed requests, i.e., with rate of 2.27% for GET, and 7.35% for PUT requests, in case of 100 edge-level orchestrators, and 8.27% for GET, and 35.52% for PUT requests, in case of 300 edge-level orchestrators. To measure the average power consumption of different orchestration components while performing orchestration operations, the same set of experiments has been executed for local tests as for measuring the average response time and CPU/memory load. We have utilized the Linux-based command-line program PowerTOP[7], which provides an estimate of the total power consumption of the overall system, but also individual power consumption for individual processes, devices, kernel workers, etc. The obtained results are shown in Fig. 10, and we can see an increasing trend in average power consumption of both top-level and edge-level orchestrators with the increasing number of orchestration requests in the simple - only GET test. However, when number of edge-level orchestrators that

[7] Managing Power Consumption with PowerTOP: https://red.ht/2T9ZF3z

(a) Top-level orchestrator, local test. (b) Edge-level orchestrator, local test. (c) Top-level orchestrator, remote test.

Fig. 8: Average CPU load in Simple - only GET test.

(a) Top-level orchestrator, local test. (b) Edge-level orchestrator, local test. (c) Top-level orchestrator, remote test.

Fig. 9: Average response time per orchestration request in Simple - only GET test.

(a) Top-level orchestrator, local test. (b) Edge-level orchestrator, local test. (c) Top-level orchestrator, local test.

Fig. 10: Average power consumption in Simple - only GET test (a and b), and only GET test (c).

simultaneously send orchestration requests towards the top-level orchestrator increases above 100, we can see that average power consumption drops. The same happens also in the case of more complex orchestration operations, as Fig. 10c shows. As described for CPU and memory load, average power consumption also decreases due to the request queuing that signi cantly increases average response time.

TABLE VIII: Results for the top-level orchestrator in local tests.

Top-level orchestrator	only GET test waiting time			GET & PUT test waiting time		
	Average CPU load (%)					
Total number of local orchestrators	1	0.5	0.1	1	0.5	0.1
1	0.4	1	3.6	0.4	0.8	2.9
2	0.8	1	7.9	0.8	1.3	6.4
10	3.8	8.6	29.7	3.9	23.9	24
100	31.9	52.9	123.8	19.9	24.2	26.1
300	71.7	114.4	127.9	34.1	35.8	39.7
	Average response time (ms)					
1	7	7	7	17	18	18
2	7	10	10	20	20	22
10	7	10	10	23	27	70
100	10	17	68	370	64	1139
300	35	69	408	2050	2238	2410

VI. Discussion

With reference to analytical evaluation of the collaborative orchestration platform in Section IV, and its experimental assessment in Section V, here we brie y pinpoint a few main aspects to consider for an orchestration platform that reinforces the orchestrated mobile edge networks.

1) Number of instances of reference points impacts the communication delay and resource availability: The number of available instances of reference points (equation (1), Fig. 6a) in the orchestration platform reduces the overall number

TABLE IX: Results for the local orchestrator in local tests.

Local orchestrator	only GET test waiting time			GET & PUT test waiting time		
	Average CPU load (%)					
Total number of direct Lo-Los	1	0.5	0.1	1	0.5	0.1
1	0.9	1.7	7.7	1.02	1.96	8.83
2	1.7	3.6	17.3	2.67	3.9	18.23
3	3.4	5.4	26.3	2.99	7.21	23.3
4	3.6	8.5	33.9	4.04	9.01	34.74
5	4.3	10.2	42	5.68	11.35	42.71
10	10.4	21	89.7	11.29	22.05	79.9
Average response time (ms)						
1	22	22	23	25	25	25
2	22	23	26	25	25	30
3	22	23	27	26	27	30
4	22	24	32	26	27	35
5	23	24	32	26	32	43
10	23	30	56	27	32	61

of hops (equation (2), Fig. 6b) that certain orchestration request, originating from an edge-level orchestrator, needs to pass in order to reach target edge-level orchestrator. Thus, such number of instances of reference points needs to be increased, as it not only reduces the communication delay in orchestration requests but it also increases the amount of available resources, given to each edge-level orchestrator at disposal to ef ciently perform orchestration operations (equation (3), with the maximum amount of available resource in all domains expressed by inequation (4)). Considering diversity in resource availability on the edges from the same or different administrative domains, it is important for orchestrators to have more resources at disposal for deploying CCAM services. On the contrary, if orchestrators running on different edges do not establish agreements for collaboration, CCAM services might suffer from performance degradation due to the limited amount of resources at the available edges. Due to the lack of resources in its own domain, an orchestrator might not be able to deploy e.g., a relevant safety CCAM service (e.g., change the lane warning, brake warning, slow down warning) that needs to support emergency situations on the road.

2) Number of instances of reference points impacts the orchestration load:The negotiated MLAs increase the number of used instances of reference points, thereby signi cantly reducing the load of the top-level orchestrator, as the upcoming requests from the edge-level orchestrators do not need to be transferred via the top-level orchestrator to other edges. Otherwise, the increase in number of requests increases the CPU load (Fig. 8a), which then causes a signi cant increase in average response time per orchestration request (Fig. 9a). Such an increase in average response time might signi cantly delay e.g., the instantiation of a CCAM service, or any runtime operation such as scaling up/out. Let us consider that vehicle is driving on the highway with the speed of 80km/h, thereby consuming the CCAM service that sends noti cations about the conditions on the road. If CCAM service is unavailable due to the scale-up operation, which is triggered to improve service resource utilization and decrease service latency, and if scaling-in lasts for approximately 500ms, it will result in at least 500ms delay in road information update. Such an increased response time will imply an outdated or delayed noti cation sent to the vehicle that needs to change its maneuvre, i.e., vehicle will already pass the additional 11,1m which can prevent it from changing the lane in time.

3) Direct Lo-Lo links impact the average response time: Given the aforementioned importance of the average response time of orchestration for the CCAM services, the design choices might include more direct links between edge level orchestrators to decrease the response time, i.e., to fasten the runtime orchestration operations such as scaling and service relocation (Objective 3, i.e., equation (5)). Although deployed on resource-constrained edge clouds, if low-latency links are established, and used as per MLA, the edge-level orchestrators process the orchestration requests with a reduced average response time comparing to the top level orchestrators (Fig. 9), due to i) the decreased load, and ii) the decreased propagation and transmission latency over the direct link (Fig. 6c). With respect to results presented for the average response time, and CPU load, one reasonable design choice for the orchestration can enforce using direct Lo-Lo links for those orchestration operations that directly affect the runtime of the service (e.g., scaling from the previous example, or service migration), while other operations such as instantiation/termination can be performed via top-level orchestrators to balance the load properly.

4) Orchestration operations impact the overall power consumption on the edges:Albeit neither the top-level orchestrators, nor the edge-level orchestrators, are intended to run on low-energy Internet of Things (IoT) devices, their power usage is still relevant for the overall energy consumption plan in the 5G ecosystem, especially due to the evident increase in consumption with the increase and complexity of orchestration requests. As shown in Fig. 10a, average consumption increases for more than 100mW in case number of edge-level orchestrators increases from two to 10, with two requests per second from each. Thus, balancing the orchestration load across multiple edge-level orchestrators is essential, as it also balances the energy consumption across edges, making the resource and service orchestration an energy-aware technique for 5G ecosystem.

5) Orchestrators' response time affects the service continuity: We learnt that it is important to carefully consider the number of hops presented in Section V, as it signi cantly impacts the average response time per orchestration request, which is also seen in the results presented in Section V. The load on the orchestrators needs to be balanced in order to keep their response time low. As we presented in Section III-C, and illustrated in Fig. 3, achieving edge-to-edge service continuity is possible if orchestration entities i) deploy a peering service instance in the target domain towards which the vehicle is driving, ii) relocate the application state from the source to the target domain, and iii) relocate the user endpoint to the target application instance. All these operations are performed by the orchestration entities, thus, their response time is critical for achieving timely relocation of the service, and maintaining service continuity when vehicle is moving from one domain to another.

VII. Conclusion

The 5G ecosystem is comprised of the cellular 5G system along with a properly managed and orchestrated deployment of virtualized network and service functions in distributed cloud resources. Such ecosystem enables customized deployment and operation of services for different sectors of the vertical industry, and the automotive industry is a promising consumer due to the high mobility and service demand with stringent QoS requirements. In this paper, we propose a solution for the orchestration of CCAM services within such 5G ecosystem to meet the stringent requirements of moving users, which connect to services in the network infrastructure. A key objective is the availability and continuity of low-latency services at the

network infrastructure edges for a highly dynamic automotive scenario and the associated management and orchestration of these services in distributed edge clouds. Our proposed solution leverages a multi-tier orchestration system as well as localized management- and protocol operations for connected and collaborative edge resources. With the analytical and experimental evaluation, we draw conclusions on the gain in accelerating orchestration operations while balancing associated protocol and computational load over the distributed and multi-layered orchestration platforms. Considering the results, we signify the importance of the overall number of instances of reference points in the orchestration platform, which are established on-demand, and used as per MLA, because it reduces the number of hops for an orchestration request, thereby facilitating the access of edge-level orchestration entities to required resources for performing orchestration operations. Also, our results show that the more instances of reference points are set up and authorized between edge-level orchestrators, the lower load is offloaded to the top-level orchestrators, which ultimately results in the decrease in their overall response time. A prototypical implementation of the described solution is currently being deployed in an automotive pilot of the H2020 5G-CARMEN project, on top of the MNOs' NFV and wireless network infrastructure, and it will be also leveraged in the 5G-Blueprint project.

VIII. Acknowledgement

This work has been performed in the framework of the European Union's Horizon 2020 project 5G-CARMEN co-funded by the EU under grant agreement No. 825012. The work has been also supported by the Horizon 2020 Fed4FIRE+ project, Grant Agreement No. 723638, and the Horizon 2020 5G-Blueprint project, under Agreement No. 952189. The views expressed are those of the authors and do not necessarily represent the project.

IX. Annex

Acronyms

3GPP 3rd Generation Partnership Project
AF Application Function
API Application Programming Interface
CCAM Connected, Cooperative and Automated Mobility
CPU Central Processing Unit
DSRC Dedicated Short Range Communication
ETSI European Telecommunications Standards Institute
FA Federation Agent
FM Federation Manager
GPSI Generic Public Subscription Identifier
gRPC Google Remote Procedure Calls
IoT Internet of Things
k8s Kubernetes
KPI Key Performance Indicator
LCM Life-cycle Management
MANO Management and Orchestration
MEAO MEC Application Orchestrator
MEC Multi-Access Edge Computing
MLA Management Level Agreement
MNO Mobile Network Operator
NFV Network Function Virtualization
NFV-LO NFV Local Orchestrator
NFV-SO NFV Service Orchestrator
NFVI NFV Infrastructure
NSD Network Service Descriptor
ONAP Open Network Automation Platform
OSM Open Source MANO
PDN Packet Data Network
PoPs Points of Presence
QoS Quality of Service
RAM Random-Access Memory
REST REpresentational State Transfer
RNIS Radio Network Information Service
SSC Service and Session Continuity
UPF User Plane Function
V2N Vehicle-to-Network
V2X Vehicle-to-Everything
VAS Value-added Service
VIM Virtualized Infrastructure Manager
VNF Virtualized Network Function
VNFD VNF Descriptor
VNFM VNF Manager

References

[1] A. de la Oliva, X. Li, X. Costa-Perez, C. J. Bernardos, P. Bertin, P. Iovanna, T. Deiss, J. Mangues, A. Mourad, C. Casetti, J. E. Gonzalez, and A. Azcorra, "5G-TRANSFORMER: Slicing and Orchestrating Transport Networks for Industry Verticals," IEEE Communications Magazine vol. 56, no. 8, pp. 78–84, 2018. doi: http://dx.doi.org/10.1109/MCOM.2018.1700990.

[2] J. Baranda Hortiguela, J. Mangues-Bafalluy, R. Martinez, L. Vettori, K. Antevski, C. J. Bernardos, and X. Li, "Realizing the Network Service Federation Vision: Enabling Automated Multi-domain Orchestration of Network Services," IEEE Vehicular Technology Magazine vol. 15, no. 2, pp. 48–57, 2020. doi: http://dx.doi.org/10.1109/MVT.2020.2979558.

[3] 3GPP, "Technical Specification Group Services and System Aspects; Procedures for the 5G System (5GS) Stage 2," 3GPP TS 23.502 V16.6.0 2020. Online [Available]: https://www.3gpp.org/ftp//Specs/archive/23series/23.502/.

[4] F. Z. Yousaf, V. Sciancalepore, M. Liebsch, and X. Costa-Perez, "MANOaaS: A Multi-Tenant NFV MANO for 5G Network Slices," IEEE Communications Magazine vol. 57, no. 5, pp. 103–109, 2019. doi: https://doi.org/10.1109/MCOM.2019.1800898.

[5] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," IEEE Communications Surveys Tutorials vol. 19, no. 3, pp. 1657–1681, 2017. doi: https://doi.org/10.1109/COMST.2017.2705720.

[6] N. F. Saraiva de Sousa, D. A. Lachos Perez, R. V. Rosa, M. A. Santos, and C. Esteve Rothenberg, "Network Service Orchestration: A Survey," Computer Communications vol. 142-143, p. 69–94, Jun 2019. doi: http://dx.doi.org/10.1016/j.comcom.2019.04.008.

[7] N. Slamnik-Krijestorac, E. de Britto e Silva, E. Municio, H. Carvalho de Resende, S. Hadiwardoyo, and J. Marquez-Barja, "Network Service and Resource Orchestration: A Feature and Performance Analysis within the MEC-Enhanced Vehicular Network Context," Sensors 2020 vol. 20, 2020. doi: https://doi.org/10.3390/s20143852.

[8] G. M. Yilma, Z. F. Yousaf, V. Sciancalepore, and X. Costa-Perez, "Benchmarking open source NFV MANO systems: OSM and ONAP," Computer Communications vol. 161, pp. 86 – 98, 2020. doi: https://doi.org/10.1016/j.comcom.2020.07.013.

[9] P. Trakadas, P. Karkazis, H. C. Leligou, T. Zahariadis, F. Vicens, A. Zurita, P. Alemany, T. Soenen, C. Parada, J. Bonnet, E. Fotopoulou, A. Zafeiropoulos, E. Kapassa, M. Touloupou, and D. Kyriazis, "Comparison of Management and Orchestration Solutions for the 5G Era," Journal of Sensor and Actuator Networks vol. 9, no. 1, 2020. doi: http://dx.doi.org/10.3390/jsan9010004.

[10] ETSI, "Multi-Access Edge Computing (MEC); Framework and Reference Architecture," ETSI ISG MEC, ETSI GS MEC 003 V2.1.1 2019. Online [Available]: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.01.01_60/gs_MEC003v020101p.pdf.

[11] G. Baggio, A. Francescon, and R. Fedrizzi, "Multi-domain service orchestration with X-MANO," in 2017 IEEE Conference on Network Softwarization (NetSoft) pp. 1–2, 2017. doi: http://dx.doi.org/10.1109/NETSOFT.2017.8004259.

[12] B. Sonkoly, J. Czentye, R. Szabó, D. Jocha, J. Elek, S. Sahhaf, W. Tavernier, and F. Risso, "Multi-Domain Service Orchestration Over Networks and Clouds: A Unified Approach," Computer Communication Review vol. 45, pp. 377–378, 2015. doi: http://dx.doi.org/10.1145/2829988.2790041.