Cloud-Native Orchestration Framework for Network Slice Federation Across Administrative Domains in 5G/6G Mobile Networks

Michail Dalgitsis, Nicola Cadenelli, Maria A. Serrano, Nikolaos Bartzoudis, Luis Alonso, and Angelos Antonopoulos

***Abstract*—The recent advancements in cellular Vehicle-to-everything (C-V2X) and edge computing paradigms foster novel use cases for connected and automated vehicles that come with strict performance requirements. These guarantees can be provided by network slicing, which is currently enabled by the virtualization and cloudification of mobile networks, following the 5G/6G core service-based approach and the cloud-native Open Radio Access Network (O-RAN) principles. However, a significant challenge arises when considering the movement of end users: ensuring network slice continuity as they transition between different network operators. In this paper, we introduce a novel cloud-native orchestration framework for network slice federation that incorporates well-defined interfaces to exchange federated service and slice resource templates among operators. The proposed framework is fully compliant with i) standardized slice service models, and ii) GSMA efforts that are building the fundamental aspects for Edge Federation to allow the sharing of network resources across mobile operators. To validate our approach, we have designed and deployed a cloud-native federated 5G experimental platform. An extensive series of experiments have been carried out, revealing that the federation implementation directly influences federation time, and the operator's slice deployment strategy significantly impacts both infrastructure and end user performance.**

***Index Terms*—5G SBA, C-V2X, cloud-native, edge federation, multiple administrative domains, network slicing, NFV, O-RAN, OPG, orchestration, UPF.**

I. Introduction

THE vehicle-to-everything (V2X) paradigm enables vehicles to interact with their environment, including other vehicles, infrastructure, and the network. By 2027, connected vehicles are projected to number 367 million globally [1]. Meanwhile, the fifth generation (5G) of mobile communications, especially with its edge computing and network slicing capabilities, is seen as the key enabler for V2X, meeting the necessary connectivity, capacity, and security demands [2], [3].

For network slicing and edge computing to fully realize their potential, virtualization and cloudification of mobile networks are vital [4], [5]. With this evolution, 5G core networks adopt cloud-native methods, aligning with the 3 rd Generation Partnership Project's (3GPP) service-based architecture (SBA) [6]. In the Radio Access Network (RAN) under the Open RAN Alliance (O-RAN) [7], the gNodeB (gNB) split into O-Centralized Unit (O-CU), O-Distributed Unit (O-DU), and O-Radio Unit (O-RU). This transition, emphasizing open standards, promotes vendor flexibility and RAN growth [8]. O-RAN units like O-CU and O-DU are adopting cloud-native design, indicating a shift to cloud RAN practices [9].

The aforementioned trends have fueled various end-to-end (e2e) service and slice orchestration initiatives. Regarding the former, Management and Orchestration (MANO) and Multi-Access Edge Computing (MEC) by European Telecommunications Standards Institute (ETSI), propose frameworks to manage and orchestrate network and edge resources, respectively. In the radio access domain, O-RAN proposes the Service Management and Orchestration (SMO) framework for managing and orchestrating network functions (NFs) [10]. Regarding the latter, standardization bodies further define specifications and frameworks for network slicing management [11], [12], [13].

Nonetheless, mobility still brings significant challenges in 5G cloud-native networks where users may move across different administrative domains, i.e., domains controlled by different operators. For instance, end users may need to handover to a different operator in rural/remote areas where the connectivity of their currently serving operator is low, or the communication resources are scarce. A similar situation could be also noticed in cross-border scenarios, where service continuity needs to be guaranteed in each region (e.g., teleoperated driving or healthcare scenarios where the ambulances can operate in both sides of the border) [14]. Further, in V2X scenarios like Vehicle-to-Network-to-Vehicles (V2N2V) communications, application servers at the network's edge could be managed by different

Manuscript received 30 June 2023; revised 12 October 2023 and 22 January 2024; accepted 29 January 2024. Date of publication 8 February 2024; date of current version 16 July 2024. This work was supported in part by 5GMED through EC-H2020 Programme under Grant 951947, in part by the FREE-6G under Grant TSI-063000-2021-144, in part by SUCCESS-6G under Grant TSI-063000-2021-39/40/41, and in part by 6G-OASIS through the UNICO5G-RPTR Programme under Grant TSI-063000-2021-24. The review of this article was coordinated by the Guest Editors of the Special Section on Open Radio Access Networks: Architecture, Challenges, Opportunities, and Use Cases in Vehicular Networks. *(Corresponding author: Angelos Antonopoulos.)*

Michail Dalgitsis, Nicola Cadenelli, Maria A. Serrano, and Angelos Antonopoulos are with Nearby Computing, 08006 Barcelona, Spain (e-mail: michail.dalgitsis@nearbycomputing.com; ncadenelli@nearbycomputing.com; maria.@nearbycomputing.com; aantonopoulos@nearbycomputing.com).

Nikolaos Bartzoudis is with the Telecommunications Technological Centre of Catalonia, 08860 Castelldefels, Spain (e-mail: nikolaos.bartzoudis@cttc.es).

Luis Alonso is with the Polytechnic University of Catalonia, 08034 Barcelona, Spain (e-mail: luis.g.alonso@upc.edu).

Digital Object Identifier 10.1109/TVT.2024.3362583

operators [15]. If a vehicle exits its primary operator's area, its data might reroute through the home network, potentially slowing V2X performance. Moreover, events like stadium gatherings highlight how operators, using third-party resources, can offer tailored services, such as live broadcasts or extensive user-generated content uploads. To tackle this kind of situations, the concept of *Edge Federation* has been recently introduced, which enables the interconnection of multiple administrative network domains (multi-ADs) and allows the sharing of network resources and services across different operators [16], [17].

A key enabler for Edge Federation is the Operator Platform (OP) concept proposed by GSMA [18]. OP includes standard Application Programming Interfaces (APIs) and protocols for edge computing resource discovery, resource management, and service deployment. By adopting the OP guidelines, network operators and service providers can create and deploy new edge services quickly and easily, while ensuring interoperability and compliance with industry standards. However, in terms of network slicing federation, OP is at a very early stage, as there are only some simple references, while no slice mechanisms and API endpoints have been defined. To ensure service continuity, as the users move from one administrative area of one operator to another, the service configuration files and slice resources should be also transferred to the destination operator in real-time. The OP concept in its current release defines the role of the Federation Manager, which is responsible for implementing the east-west bound interface (EWBI) and managing the federation among multiple OPs instances. At its core, EWBI defines a set of resources that can be used to define and deploy cloud-native applications, while no slice-related resources have been identified.

Since the slice federation concept is new, a number of unresolved challenges may arise. Beyond general concerns like slice security, privacy, and inter-operator billing, deeper issues arise: i) when accessing network slicing, operators decide whether to use the visited network's slice features or their home network design. This emphasizes the challenge of sharing slice template requirements with visiting operators; ii) adapting to an operator's unique cloud orchestrator can spark conflicts, highlighting the need for standardized protocols. Incorporating O-RAN adds complexity: i) efficient management of both intra and inter-slice resources is vital, stressing the role of O-RAN applications like xApps [19], [20]; ii) there is a need for thorough e2e O-RAN testing and strategies to integrate O-RAN with open-source projects [21].

Recent studies emphasize service and slice orchestration in cloud-native networks for mobile scenarios. Works like [22], [23], [24] focus on the dynamic nature of slice management in the presence of mobile users, but within the infrastructure of a single operator. Transitioning from single to multi-AD contexts, standardization efforts like [25], [26], EU-funded projects such as the [27], [28], and research works such as the [14], [29], [30], [31], [32], [33] have paved the way by defining essential architectures and interfaces for service and slice federation. However, a significant limitation is the reliance of many works on simulation-based results or non-integrated testbeds, revealing a gap in real-world evaluations. A survey [34] and a recent work by the same authors [35] on operator slice mobility showcased a cloud-native proof of concept, though without detailing specific orchestration schematics or interfaces.

In this paper, motivated by the above challenges and the gaps in the literature, we introduce a cloud-native orchestration framework for network slice federation in different administrative domains. In particular, our contribution is threefold:

- We propose a novel *Slice Federation as a Service* (SFaaS) framework that extends the OP with network slice federation capabilities. The proposed framework is 3GPP-compliant (i.e., it follows the 3GPP slice management system) and leverages the EWBI to federate a specific application called *SliceFedRequest*, for the exchange of the slice resource template among operators to ensure the end users' seamless mobility.
- We design and deploy an e2e 5G testbed with cloud-native 5G core and O-RAN features to demonstrate the proposed framework. Our testbed is implemented using a combination of open-source technologies (e.g., Open5GS[1] and UERANSIM[2]) and state-of-the-art market solutions (i.e., NearbyOne edge orchestrator [36]) to emulate realistic federated environments.
- We perform a thorough experimental performance evaluation to assess: i) the impact of the federation process in such scenarios, and ii) the post-federation network performance in terms of the slice creation under various scenarios, as well as the quality of service for the "federated" users.

The structure of the paper is as follows. Section II reviews related work in the field. Section III introduces the system architecture of our work. In Section IV, we present the proposed framework for network slice federation. Section V provides details of the deployed testbed and evaluates our novel framework. Section VI discusses lessons learned, offers broader context based on our findings, and suggests directions for future research. Finally, Section VII concludes our work.

II. Related Work

As 5G networks evolve, network slicing is crucial for diverse service requirements, driving the need for effective slice federation in single and multi-AD contexts.

In the context of single administrative domains, many studies have focused on service and slice orchestration in cloud-native networks. However, most are limited to the infrastructure of one operator [22], [23], [24]. Moving to multi-AD orchestration, standard efforts like [25], [26] along with EU-funded projects like [27], [28], have expanded this domain. Still, many rely on simulations or non-integrated testbeds, highlighting a gap in real-world evaluations. Additionally, many of these approaches have not incorporated the OP concept by GSMA, for service federation.

For multi-AD frameworks, works such as [29] and [30] have emphasized the significance of such frameworks. They have emphasized the relevance of centralized placement and service function chains in multi-AD 5G architectures, and service mobility in MEC-enabled networks, respectively. However, these

[1]https://github.com/open5gs/open5gs
[2]https://github.com/aligungr/UERANSIM

TABLE I
COMPARISON OF SLICE FEDERATION IN CLOUD-NATIVE MANAGEMENT AND ORCHESTRATION SYSTEMS

Study	Multi-ADs	Slice Federation	Cloud-native Approach	Testbed	GSMA-compliant Federation	O-RAN Features	5G Network Implementation
[29]	✓	x	x	✓	x	x	x
[30]	✓	x	✓	x	x	x	x
[14]	✓	✓	x	x	x	x	x
[31]	✓	✓	x	x	x	x	x
[32]	✓	✓	x	x	x	x	x
[33]	✓	x	x	✓	x	x	x
[34]	✓	✓	✓	✓	x	x	x
[35]	✓	✓	✓	x	x	x	x
Our work	✓	✓	✓	✓	✓	✓	✓

works either overlook 5G implementations or rely on simulations, leaving gaps in practical deployments

Moreover, a significant portion of research has leaned toward service and slice federation. Notable contributions include [14] which has enhanced a 5G network slicing management model, and [31] which has emphasized multiple domain network slicing orchestration. Despite their importance, they have not fully integrated cloud-native methods, leaving a research gap. This is further highlighted by a study like [32], which has prioritized information exchange but sidelined the cloud-native approach.

Among the relevant research, [33] has notably advanced network service federation by enabling automated multi-AD orchestration of network services. The work demonstrated this advancement through an open-source orchestration implementation. However, a recurring theme is the limited integration of real 5G solutions or the absence of a cloud-native approach in their strategies. It is also worth noting that none have considered O-RAN features, which our framework integrates.

In the realm of cloud-native network slicing, a survey like [34] has highlighted Kubernetes benefits. The same authors, as seen in [35], proposed a novel microservice and SDN-based MEC network slicing architecture, aligning with ETSI requirements. While making significant contributions to cloud-native slice federation, [35] lacks consideration for: i) integration with Kubernetes for effective MEC slice resource management; ii) adapting a cloud-native 5G core, specifically developing NFs and MEC services as microservices.

Distinct from the mentioned works, we offer a framework that combines orchestration in multi-ADs, integrates service and slice orchestration, and presents a real-world testbed. Rooted in cloud-native principles and combined with practical 5G solutions, our approach sets a new standard in next-gen mobile networks. For a detailed comparison, please refer to Table II, which shows how our multi-AD architecture stands out from current research.

III. SYSTEM ARCHITECTURE

In this section, we provide a background on cloud-native network slicing implementations in 5G networks. We then present the system architecture for our proposed solution, accompanied by a toy example that illustrates the concept of network slice federation.

A. 5G Cloud-Native Network Slicing

In 5G, network slicing creates virtual networks across domains, optimizing resource utilization and service delivery. Control-user plane separation (CUPS) enables the allocation of control and user plane (CP and UP, respectively) resources separately for each network slice. This separation allows both the sharing of CP functions across slices, as well as the deployment of dedicated UP functions to specific slices.

In the core domain, NFs can be shared among network slices, reducing management complexity. The User Plane Function (UPF), as well as the main CP 5G core NFs, like the Access and Mobility Management Function (AMF), Session Management Function (SMF), Policy Control Function (PCF), Authentication Server Function (AUSF), Network Slice Selection Function (NSSF), Network Repository Function (NRF), Unified Data Management (UDM) and Unified Data Repository (UDR), can be dedicated or shared among slices. Different approaches include implementing dedicated core NFs for each slice or sharing certain control plane functions while assigning user plane functions exclusively [37].

In the evolving O-RAN-based access domain, the gNB adopts a functional split into O-CU, O-DU, and O-Radio Unit (O-RU), with O-CU further divided into O-CU-CP and O-CU-UP. O-RAN introduces the RAN Intelligent Controller (RIC) that separates RAN control and monitoring from the base station [8]. RIC comprises two entities: Near-Real-Time (Near-RT) RIC and Non-RT RIC, serving as platforms for xApps and rApps, which optimize the radio network. xApps run on Near-RT RIC, while rApps operate on Non-RT RIC within the management plane. Hence, for O-RAN network slicing, certain NFs might be shared across RAN slice subnets, like O-CU-CP, O-DU, O-RU, while others, such as O-CU-UP and xApps, may be slice-specific [13].

B. Network Architecture

Our system follows a multi-stakeholder architecture, as illustrated in Fig. 1, comprising three distinct entities: i) the Mobile Network Operator (MNO),[3] ii) the slice customer, and iii) the end-user. The MNO owns the telco infrastructure, where cloud-native applications and mobile NFs are running, along with a cloud orchestration platform based on the OP concept with extensions, consisting of a *Slice Manager*, a *Service Manager*,

[3]Please note that the terms MNO and operator are used interchangeably in this paper.

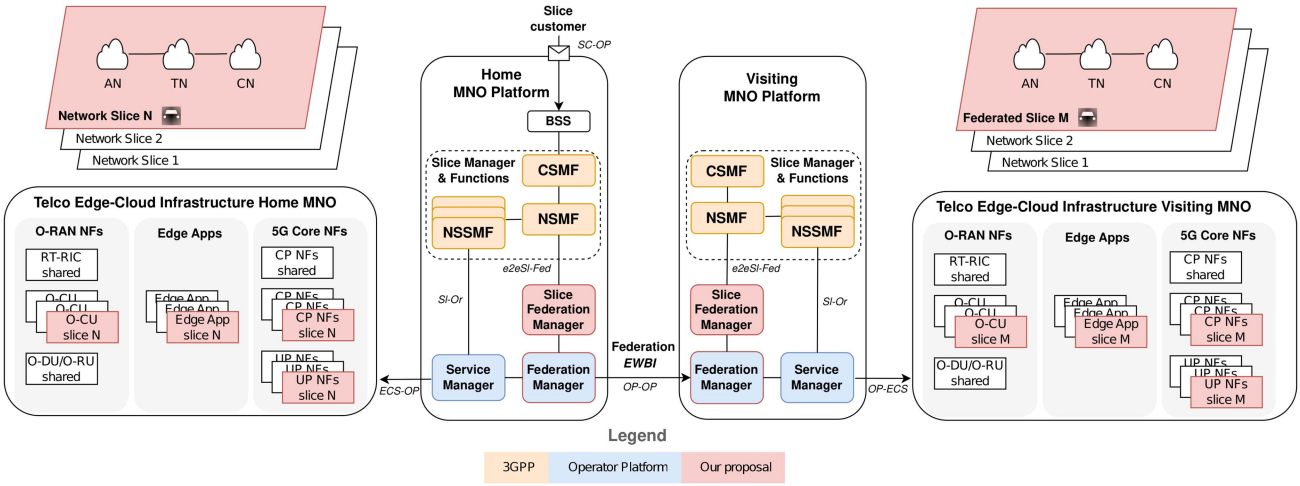

Fig. 1. System architecture.

a *Federation Manager*, and a *Slice Federation Manager*. The *Slice Manager* with its respective management and orchestration functions, such as the Communication Service Management Function (CSMF), the Network Slice Management Function (NSMF), and the Network Slice Subnet Management Function (NSSMF) has been defined by 3GPP [38], while the *Service Manager* and the *Federation Manager* roles have been presented by OP to enable Edge Federation [18]. On the other hand, the *Slice Federation Manager*, is introduced in this paper to facilitate the implementation of slice federation. Regarding the network operation, initially, an application provider offers services for the end users to be deployed within the telco infrastructure, while the slice customer requests a communication service in the form of a network slice to cater for its end users' specific characteristics and requirements. The end users represent the final consumers of both the applications and network services. As in Fig. 1, the user's home network is provided by the Home MNO, and the visited network by the Visiting MNO.

The telecommunication network of each operator is divided into network slices, with each slice customized to fulfill the specific requirements of the end users across various technical domains, including the access network (AN), transport network (TN), and core network (CN). The *Slice Manager* of each MNO, provides and manages network slices within the telco infrastructure and encompasses domain-specific slice functions. These functions handle the orchestration and management of network slices within their respective domains, ensuring efficient resource allocation based on the specific requirements of each slice.

From the OP concept, the *Service Manager* is responsible for service and resource provisioning within the telco site infrastructure. In particular, the telco infrastructure is considered a collection of multiple edge and cloud nodes across the compute continuum, capable of hosting, executing, and orchestrating cloud-native applications and NFs (5G core and O-RAN NFs). Moreover, the *Federation Manager* facilitates application mobility among MNOs through the EWBI endpoints. At its core, EWBI defines a set of resources that can be used to define and deploy cloud-native applications. Each of these resources plays a crucial role in specifying and managing cloud-native applications across multiple sites.

Finally, each orchestration platform includes the newly introduced role of this proposal, the *Slice Federation Manager*. The *Slice Federation Manager* in Home MNO translates the slice template (ST), which has been generated by the slice customer request into a federated slice template (FST). The ST can be represented as a JSON file that includes various details about the slice, such as the slice type, slice tenant, slice ID, slice requirements (e.g., guaranteed bit rate, latency, maximum number of users, etc.), and slice domains (e.g., RAN, core). On the other hand, the FST serves as a structured representation of the ST extended with a federation field while referring to edge sites and resources of the Visiting MNO. In Fig. 1, the interfaces among the primary components are also illustrated. A detailed discussion on these interfaces can be found in Section IV.

C. Slice Federation: Toy Example

To further illustrate the concept of slice federation in mobile environments, let us consider a scenario involving two MNOs and the seamless transition of a V2X slice as a user moves between their coverage areas (Fig. 2).

To begin with, inside the Home MNO's network reside two types of slices, a V2X and an eMBB (Enhanced Mobile Broadband) slice enabling V2X and video streaming applications, respectively, while the Visiting MNO has deployed a URLLC (Ultra-reliable and low-latency communications) slice. The V2X and the eMBB slice share certain components in both core and access domains. The core network shares some of the CP NFs, while in the access domain, the RT-RIC and the O-DU/O-RU are common between the two slices, ensuring efficient resource utilization and coordination. Nevertheless, a dedicated SMF and UPF, as well as a specific xApp and a dedicated O-CU are running in the V2X slice, providing functionalities on the V2X application.

As the user of the V2X slice moves from an area covered by Home MNO to Visiting MNO's coverage area, the Visiting

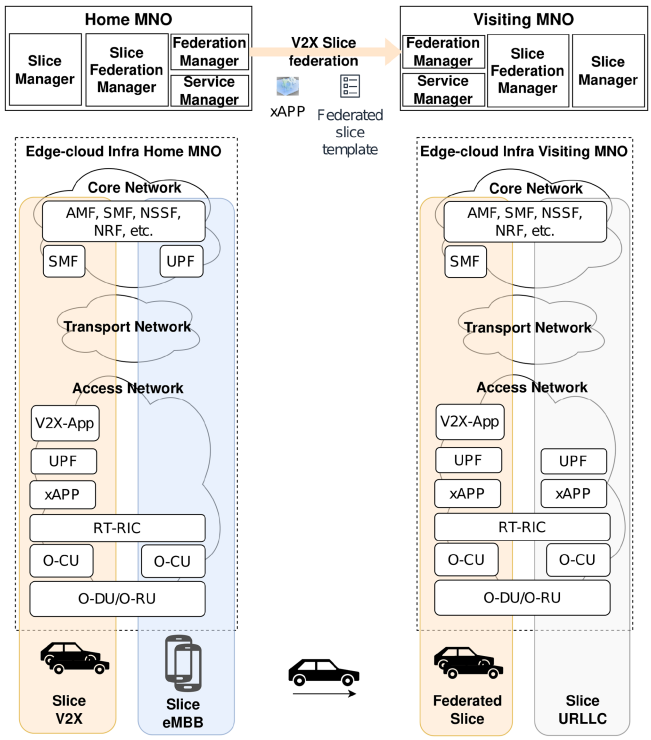

Fig. 2. Slice federation example.

MNO needs to extend its service to accommodate the user during its transition ensuring seamless connectivity and real-time communication between the user's vehicle and the surrounding infrastructure. During the federation procedure, the Home *Slice Federation Manager* and the Home *Federation Manager* query the Home *Slice Manager* and the Visiting *Federation Manager* respectively, to allocate resources and manage the seamless handover of services (V2X slice configuration and xApps) for the V2X slice between the two operators. The *Federation Managers* facilitate the east-west communication between the two operators, ensuring that the V2X slice resources are seamlessly constructed into Visiting MNO's network as the user moves.

IV. Slice Federation Mechanism: Integrating 3GPP Slice Management System and Operator Platform Concept

In this section, we extend the architectural concept of the OP towards the 3GPP network slice management system to address the needs for slice federation in 5G/6G mobile cloud-native communication networks. First, we describe the missing interfaces for slice federation within the *Federation Manager* role. Next, we delve into the protocols and connections within the framework, outlining how different entities communicate within our 5G system. Following that, we introduce the *Slice Federation as a Service* (SFaaS) mechanism, along with an xApp federation description in O-RAN architecture. Finally, we detail the slice federation phases.

A. 3GPP-OP Integration for Slice Federation

3GPP has defined the CSMF, NSMF, and NSSMF model to provide a hierarchical approach to network slicing in one operator, enabling the translation of service requirements into network slice characteristics and their implementation across the network infrastructure. In multi-ADs though, there is a need for an additional control layer to be added to the single-administrative domain architecture. To that end, we leverage the OP to map the slice requirements to the capability of the infrastructure domain by identifying the edge sites with the required resources. The OP offers interfaces to other parties (application providers, end users, other operators) using common definitions based on the requirements of [39].

Within the context of OP, we have proposed the addition of the *Slice Federation Manager* module, which is responsible for the management and monitoring of the resources related to the federated slice, while the existing *Federation Manager* role ensures secure and reliable connectivity between two MNOs. However, since no slice endpoints have been identified as of now, new fields related to slicing must be included in the EWBI. The following HTTP endpoints can enrich the EWBI of OP under a new API resource called Slice Federation:

- POST request to create a slice federation session: `/slice/session`
- GET request to read the status of the slice session: `/slice/session/{sessionID}/status`
- DELETE request to delete the slice session: `/slice/session/{sessionID}`
- POST request to create a slice instance: `/slice/session/{sessionID}/nsi`
- GET request to read the status of the slice instance: `/slice/session/{sessionID}/nsi/{nsiID}`
- PUT request to update the content of the slice instance: `/slice/session/{sessionID}/nsi/{nsiID}`
- DELETE request to delete the slice instance: `/slice/session/{sessionID}/nsi/{nsiID}`

These endpoints can serve to initiate customer requests for slice creations to another MNO. Based on the Service Level Agreements (SLAs), this request is forwarded to the Visiting *Slice Federation Manager*, which uses the FST to translate it back to an ST and then forward it again to the Visiting *Slice Controller* to create a network slice instance (NSI).

B. Protocols and Interfaces Within the Framework

In our 5G framework, seamless communication between various entities is essential. This communication is facilitated by a range of protocols and interfaces, each tailored to the specific needs of individual connection types. This section explores these connections in detail, highlighting the interactions between different system components and the protocols that enable them. Interfaces are illustrated in Fig. 1.

1) Slice Customer-to-Operator (SC - OP): A customer portal as a part of the slice Business Support System (BSS) provided by the Operator Platform, has allowed the enterprise-slice customers to order network slices.

2) Slice-to-Orchestration (Sl - Or): Orchestration and management of network slices utilize a client-server API mechanism, bridging the slice management and orchestration layers. Notably, domain-specific slice functions, such as NSSMF, send instructions from the management layer to the orchestration controllers. The *Service Manager*, acting as the RAN and core controller, deploys the NFs into edge-cloud sites. Consequently, it provides an API, allowing clients like core-NSSMF to initiate a network slice.

3) e2e Slice-to-Federation (e2eSl - Fed): NSMF offers a set of northbound interface (NBI) endpoints, providing a key gateway through which the detailed status and characteristics of each slice can be accessed. This information, serving both individual customers and served users, is critically important. The *Slice Federation Manager* uses this mechanism by accessing the NSMF's HTTP endpoints, operating within an HTTP client model. This interaction promotes efficient, real-time monitoring of slices across various network domains and simplifies the process of slice federation by ensuring smooth information retrieval and manipulation.

4) Operator-to-Edge Cloud Sites (OP - ECS): For the actual deployment of the NFs into the edge-cloud sites, we have harnessed the capabilities of the Kubernetes API, since we consider all the telco infrastructure of each operator to be a Kubernetes-based cloud platform. The *Service Manager* is the service orchestrator within the platform.

5) Operator-to-Operator (OP - OP): To interconnect two operator platforms, we have implemented the set of APIs from EWBI, as suggested by the OP concept. The components involved in this interconnection are the *Federation Managers* of each MNO platform.

C. Slice Federation as a Service (SFaaS) Mechanism

To leverage the existing Edge Federation approach by OP and initiate slice federation resource requests, we propose encapsulating these requests in a *Slice Federation as a Service* (SFaaS) mechanism. This involves the federation of a novel application, named *SliceFedRequest*, which carries the FST and specifies the slice customer's requirements. The FST encompasses the slice type, the QoS parameters, the NFs, the resource allocation, and session details.

Our framework adopts a cloud-native approach, deploying the application as a containerized microservice on a cloud-native environment, utilizing technologies and tools like Docker,[4] Kubernetes[5] and Helm Charts.[6] First, a Docker image is captured from the application. A Docker image serves as a self-contained package that encapsulates all the necessary components and dependencies of an application. Then, a container is an instance of a container image running in isolation, providing a lightweight and consistent runtime environment. Kubernetes acts as an orchestration platform, automating the deployment, scaling, and management of containers across a cluster of machines. It ensures high availability, scalability, and fault tolerance for applications. Helm Charts, on the other hand, serve as a packaging format for Kubernetes resources and their configurations, simplifying the management and deployment of complex applications and services.

During federation setup, the Visiting MNO provides a Kubernetes cluster at an edge site for the *SliceFedRequest*. This MNO also shares clusters for core and O-RAN NFs necessary for the slice.

It is also worth noting that the *SliceFedRequest* has been designed based on the Kubernetes container lifecycle hooks[7] and liveness probes.[8] Container lifecycle hooks in Kubernetes provide a way to run specific commands or scripts at various stages of a container's lifecycle. These hooks allow us to perform certain actions before or after important events in the container's lifecycle, such as starting or stopping the container. We leverage the following two hooks:

1) *PostStart:* This hook is executed immediately after a container is started. It enables performing actions or configurations that should happen after the container has started, such as requesting a slice federation session and the federated slice instance. The data of this request contain the FST.
2) *PreStop:* This hook is executed immediately before a container is stopped. It provides an opportunity to gracefully terminate processes, close connections, or perform any necessary cleanup tasks before the container is terminated, such as deleting the federated slice instance and the slice federation session.

Liveness probes is a mechanism to determine the health status of a container running within a pod (pod is the smallest and most basic unit of deployment in Kubernetes). A liveness probe periodically checks the container's health by sending a request to a specified endpoint and analyzing the response.

In the SFaaS mechanism, a liveness probe ensures the Visiting *Slice Federation Manager* operates correctly, reads the federated slice status, and acts if issues arise. If this probe fails (e.g., due to non-responsiveness or error status), Kubernetes reacts based on probe settings. We use the *Exec Probe* type, which sends a status command to the Visiting *Slice Federation Manager* and captures its response. This response is then accessed by the Home *Slice Federation Manager* to synchronize with the Visiting MNO.

Moreover, it's through the strategic integration of these container lifecycle hooks and the aforementioned liveness probes that we are able to refine and customize the operation patterns of our SliceFedRequest application. This ensures seamless slice federation initialization, efficient cleanup, and consistent availability throughout its lifecycle management.

Federating xApps in O-RAN-based architectures: The SFaaS mechanism not only federates the *SliceFedRequest* application but also extends its federation to include access network applications, such as the xApps within the O-RAN-based architecture. In the context of RAN slicing, O-RAN incorporates slicing capabilities by assigning dedicated O-CU-UP instances

[4]https://docs.docker.com/
[5]https://kubernetes.io/docs/
[6]https://helm.sh/docs/
[7]https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/
[8]https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/

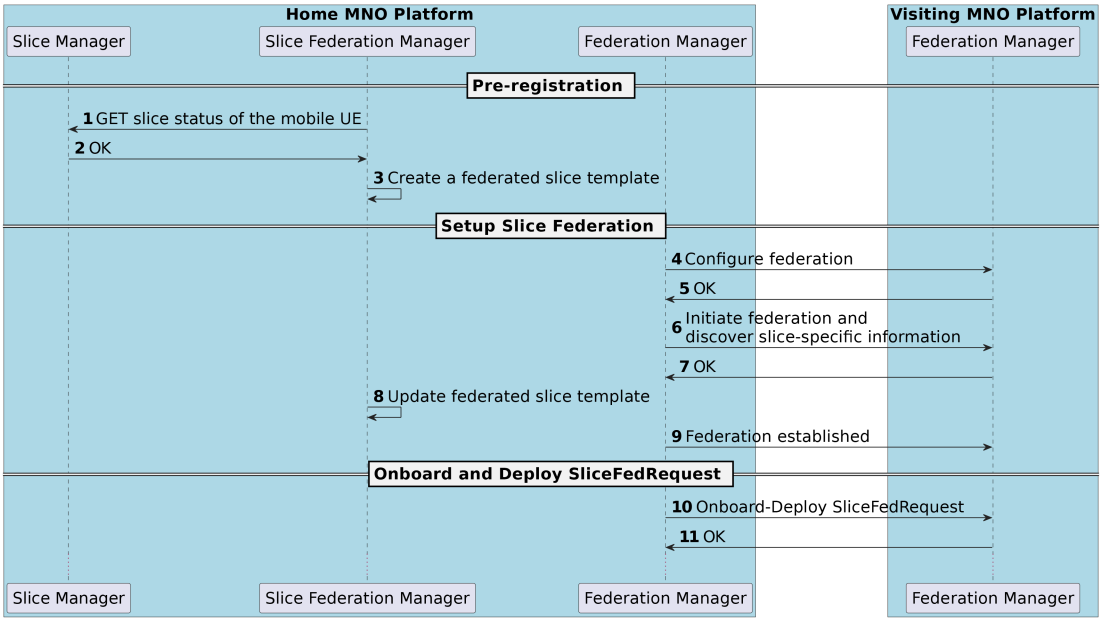

Fig. 3. Slice federation sequence diagram between two MNOs.

to specific slices, while the shared O-CU-CP and Near-RT RIC handle inter-slice operations and optimizations.

Moreover, O-RAN leverages the capabilities of the Non-RT and Near-RT RICs. These components play a vital role in ensuring resource isolation, optimization, and scalability. They dynamically respond to traffic demands and leverage dedicated or aware xApps and machine learning models specific to each slice. This allows them to anticipate and respond to expected fluctuations in traffic volume and location. For instance, a slice-dedicated xApp like traffic steering can employ a simple scheme for static IoT/mMTC scenarios, while more complex schemes like dynamic channel/mode selection may be required for demanding use cases such as V2X [40]. Additionally, slice-aware xApps, such as the NexRAN inter-slice management xApp within the O-DU scheduler, can prioritize the allocation of physical resource blocks to ensure SLA compliance across different slices [41].

Consequently, these xApps need to be deployed on the Near-RT RIC of the Visiting MNO as federated services, following cloud-native practices. It is important to note that these xApps enter the federation process only upon acceptance of the *SliceFedRequest*.

D. Slice Federation Phases

When all the building blocks from OP, 3GPP management slice functions, and the proposed innovations come into play, the network slice federation process involves five phases, as described below:

1) *Phase 1: Slice Federation Pre-registration*
   It is assumed that a slice for a user or a group of users has been already created in the Home MNO. CSMF and NSMF have been involved in translating customer slice requests into slice requirements. Updates of the NFs and applications involved, and the status of the slice are always reported back to Slice Manager from the subslice domain Controllers. The *Slice Federation Manager* constantly gets and updates its state by retrieving the slice instance and translating it into a federated slice template. It leverages a *Slice Manager* interface to request the status of the user's slice. The status includes information on subslice domains, slice requirements, and specific NFs associated with the slice. The flow of these actions is depicted in Fig. 3 (steps 1-3).
2) *Phase 2: Slice Federation Setup*
   The federation establishment shall be performed to setup the federation relationship between the two MNO orchestration platforms. Steps 4-9 in Fig. 3 show the actions needed to establish the federation and exchange infrastructure and network slice-related information between the MNOs, such as the edge site that will host the *SliceFedRequest* and the endpoints of the Visiting *Slice Federation Manager*, respectively.
3) *Phase 3: SliceFedRequest* Application Onboarding and Deployment
   To onboard the *SliceFedRequest*, files (e.g., Docker images) must be uploaded into a public/private registry. Then, the artefacts referencing these files (e.g., helm charts with Kubernetes manifests like *Deployments* and *ConfigMaps*) are added, followed by the application that references the artefacts. Once onboarded, the application can be installed at an edge site (e.g., Kubernetes cluster), as shown in steps 10—Fig. 3, initiating the federated slice request.
4) *Phase 4: Federated Slice Deployment*
   After deploying the *SliceFedRequest*, a federated slice request is sent to the Visiting MNO's *Slice Federation Manager* (Fig. 4, step 1). The FST translates back to

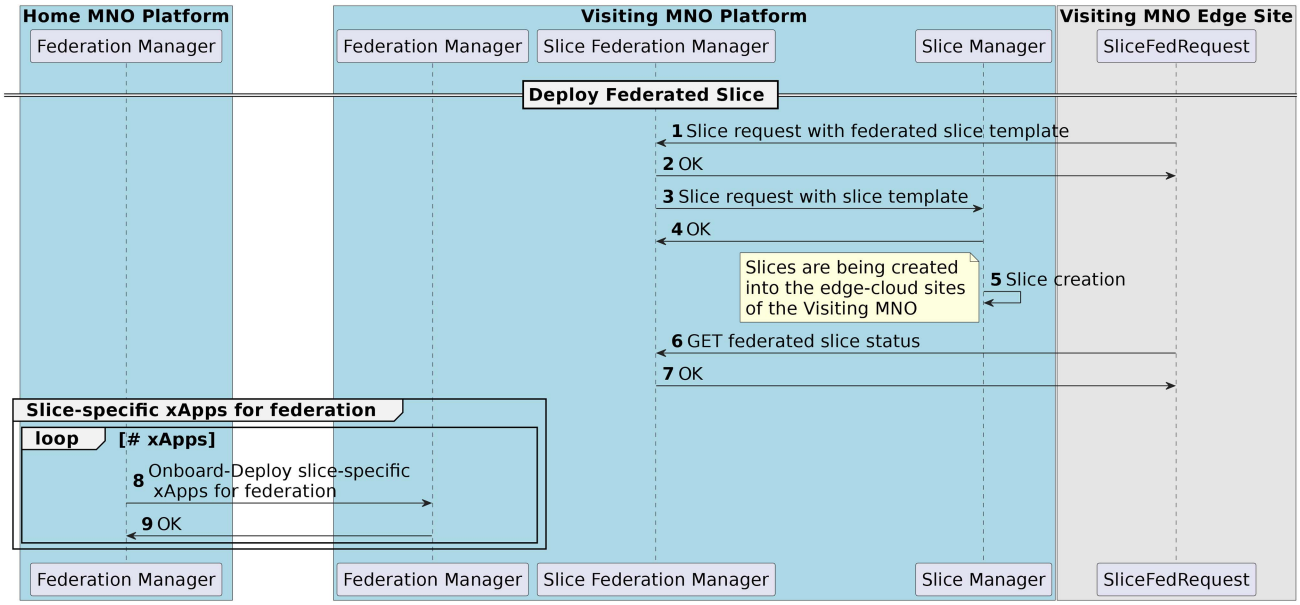

Fig. 4. Slice federated deployment sequence diagram in Visiting MNO.

ST, and slice deployment begins on the Visiting MNO's infrastructure. The *SliceFedRequest* periodically checks the status of the federated slice request and, if accepted, the Home *Slice Federation Manager* updates its content for the federated slice instance and informs the slice customer about the installation, connectivity, and management points. It also notifies the slice customer of any SLA breaches or major slice failures. If more services, like xApps in the access domain, are needed for the user's slice, the federation continues with their onboarding and deployment (Fig. 4, steps 8-9).

5) *Phase 5: Slice Federation Termination*
When the federated slice is no longer needed, the Home MNO through the EWBI uninstalls the *SliceFedRequest*. This action triggers a DELETE request to undeploy and destroy the federated slices. In addition, the Home MNO uninstalls any other federated services, e.g. xApps, from the other MNO's edge site. Finally, when all resources related to slice federation have been cleaned-up, the federation session is released.

The steps above detail the network slice federation process, highlighting the role of the *Slice Federation Manager* and the information flow between various components.

To provide a clearer representation of the aforementioned slice federation phases, Fig. 5 offers a visual depiction of the entire slice federated deployment sequence between the two operators. The flowchart begins with the initiation of a slice customer request. The system first evaluates if federation is required. If it is not, the process reaches its conclusion. However, if federation is deemed necessary, a federation session is enabled. Subsequently, a federated slice template is prepared and the *SliceFedRequest* is sent. Upon sending the *SliceFedRequest*, the system then awaits a response from the Visiting Operator. If the federated slice request is declined, the initiating customer is immediately informed of this outcome. On the other hand, if the

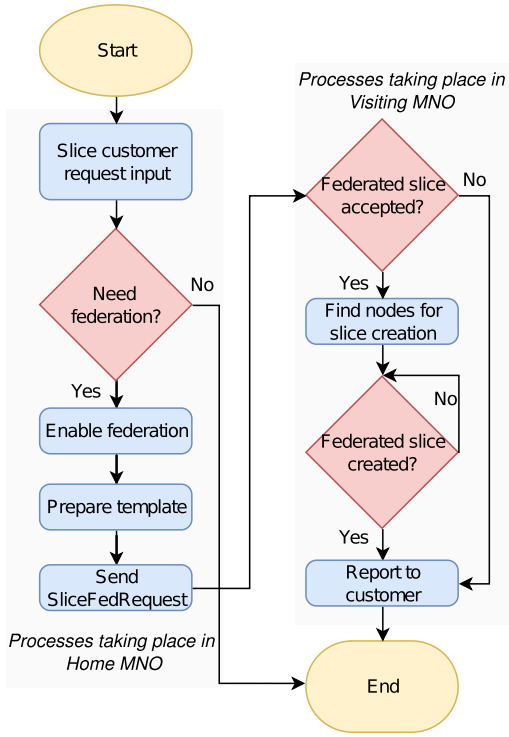

Fig. 5. Slice federation instantiation flowchart.

request is accepted, the *Slice Manager* of the Visiting Operator is tasked with determining suitable sites for deploying the NFs constituting the network slice. Once these slices are successfully created, the initiating customer is notified that their slice has been federated and is now operational within an alternative administrative domain. If the slices are not immediately established, the system periodically checks their status and keeps the customer updated until their slice is fully federated.

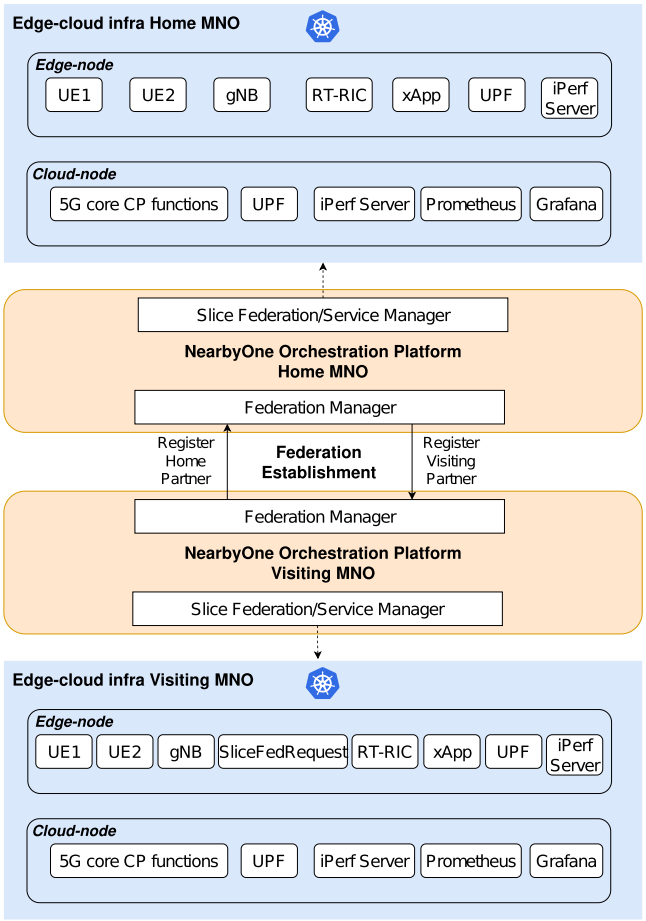

Fig. 6. Cloud-native 5G experimental platform.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed slice federation approach. First, we describe the setup of the environment employed in our experiments. Subsequently, we define the performance metrics employed to assess the effectiveness of our strategy and analyze the results, discussing their implications for the telco edge-cloud infrastructure and the end users. Our testbed implementation follows cloud-native paradigms, contemplating both the infrastructure and the applications. Within this context, we evaluate two aspects: i) the federation mechanisms between operators and ii) the deployment of slices in the post-federation phase.

A. Experimental Setup

We have designed and deployed a cloud-native 5G experimental platform to evaluate federation and assess the impact of post-federation slice deployments on the Visiting MNO's infrastructure and user performance. The platform's architecture is shown in Fig. 6, with two MNOs: the Home and Visiting MNOs, each having an orchestration platform and an edge-cloud infrastructure for slice federation and deployments. The NearbyOne Controller [36] serves as the orchestration platform, facilitating federation between operators by establishing EWBI for seamless communication between them.

TABLE II
Testbed Parameters

Parameter	Value
Kubernetes Cluster Type	RKE
CPU (Cores)	4
RAM (GB)	8
Number of Nodes	2
Number of Slices	2
Number of UEs	2
Available Bandwidth (Mb/s)	550
Added Network Latency (ms)	{33,38,45}

Within each operator's compute infrastructure, there is a Kubernetes cluster with two nodes: the edge-node and the cloud-node. The cloud-node hosts the Open5Gs control functions (i.e., AMF, SMF, PCF, AUSF, NRF, NSSF, UDM, and UDR), a Prometheus server,[9] and Grafana,[10] for 5G core connectivity, monitoring, and visualization, respectively. On the other hand, the edge-node is responsible for running the UPF, an emulated gNB and a user equipment (UE) from the UERANSIM open-source project, as well as an iPerf server.[11] Additionally, the edge node runs the O-RAN Software Community (O-RAN-SC) Near-RT RIC (release E) [42] and the NexRAN xApp, which enables policy-driven closed-loop control of RAN slices by reading the current state of RAN elements.

The specifics of our deployment (edge-cloud infra per operator), including computational resources and network parameters, are detailed in Table II. For deploying our Kubernetes cluster, we have utilized the Rancher Kubernetes Engine[12] (RKE), chosen for its robustness and reliability. Given the intended workload, the nodes have been provisioned with 4 CPU cores and 8 GB of RAM. These specifications, while surpassing the minimum requirements, have been carefully chosen to provide ample resources for both the 5G core control plane functions on the master and the RAN stack on the worker node. This two-node setup gives us also the flexibility to experiment with the distributed UPF scenario. The deployment also facilitates two individual slices, with each user registered to a distinct slice, capturing a typical scenario where users are distributed based on unique network demands. In addition, the bandwidth limit of 550 Mb/s has been set as a representative value for high-speed connectivity in 5G networks. Finally, to perform the edge-cloud experiments of Section V-D, the Connection Health Check AWS[13] tool has been leveraged to measure the simulated cloud-based network latency from a user perspective in Spain, targeting services hosted in cloud regions of London and Ireland. From our tests targeting the London region, we have recorded a minimum average latency of 33 ms and a maximum average of 38 ms after multiple iterations. These distinct values have been retained to depict the minor variations in latency, particularly for a scenario where both the UPF and the user's service are hosted on the cloud. For tests targeting the Ireland region, a mean latency value of 45 ms has been observed.

[9] https://github.com/prometheus-community/helm-charts
[10] https://github.com/grafana/helm-charts
[11] https://iPerf.fr/
[12] https://www.rancher.com/products/rke
[13] https://clients.amazonworkspaces.com/Health.htmls

TABLE III
APPLICATIONS FOR FEDERATION

	SliceFedRequest	NexRAN xApp
Image	curl	NGINX, curl, NexRAN
Image Registry	Public Cloud	Public Cloud
Helm Registry	Public Cloud	Public Cloud
Functionality	Customer slice request	Deployable xApp on RIC
Interactions	Federation Manager	RIC services

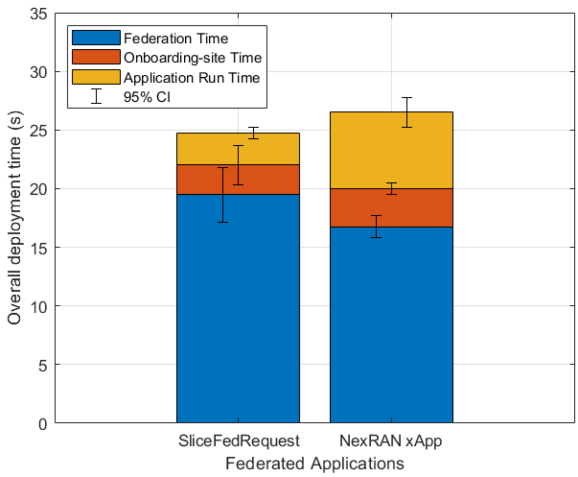

Fig. 7. Federated deployment application time.

In our experimental approach, we have leveraged the NearbyOne dashboard, a user-friendly interface for slice federation tasks. To measure the network's performance, we have used standard testing tools, primarily *iPerf* and *ping* tests. Network slice deployments have been executed as Helm chart deployments supported by an automated bash script, ensuring consistency and accuracy across the board. In analyzing our data, performance metrics have been averaged across multiple runs, further reinforced by a 95% confidence interval (CI), highlighting the dependability and consistency of our findings.

B. Evaluation Results on Slice Federation

First, we present the experimental results on the performance of the proposed slice federation using the NearbyOne Controllers. The experiments have been initiated from the Home MNO's NearbyOne dashboard, focusing on the deployment of two applications: i) the *SliceFedRequest*, which includes the FST and applies it to the endpoint of the *Slice Federation Manager* of the Visiting MNO, and ii) the NexRAN xApp, responsible for RAN slicing. More information about these applications is presented in Table III.

The overall application deployment time is divided into three parts: i) the federation time, which is the time required for the orchestrators to exchange information related to the federated applications, ii) the onboarding time to the Visiting MNO's edge site, and iii) the application runtime, which is the time required for the containerized application to be executed in the cloud-native environment and perform its action.

Fig. 7 presents the application deployment time, with markers illustrating the 95% CI for each time component, providing a form of sensitivity analysis. The larger span in the *SliceFedRequest* federation and onboarding-site time arises from potential variations in network conditions, while the significant CI in the xApp's application runtime suggests fluctuations due to the Kubernetes cluster load. As observed, federation time dominates the other two parts. When federating an application, the average federation time is in the order of 17 s. The federation times depend on the implementation of the federation approach followed by the orchestrator platforms. On the other hand, the onboarding-site time is around 2 s, representing the time required for the orchestration platform to reach the edge Kubernetes cluster and deploy the helm chart. Lastly, the application runtime depends on the Kubernetes platform, Kubernetes resources, and other application-specific factors (e.g., init-containers, readiness probes, container image size, and location). As the *SliceFedRequest* is using a lightweight *curl* image and performs only a *curl* request with some data, it takes on average 2 s to send the request. Regarding the NexRAN xApp, the application runtime is significantly large as we follow the O-RAN-SC Near-RT RIC onboarding and deploying method in the same microservice. For this, we deploy a web server to host the xApp configuration in a JSON format and then specific requests pointing to the Near-RT RIC services to onboard and install the helm chart. Specifically, xApps, being a new and evolving technology, are closely tied to the specific purposes they are created for and the particular RIC implementations provided by vendors [43]. The findings emphasize the importance of accurately determining when to kick off service and slice pre-relocation. This action ensures consistent service availability for mobile users.

C. Evaluation Results on Post-Federation Slice Deployments Impacting Operator's Infrastructure

While current 5G network slicing specifications provide the guidelines, they do not dictate the exact implementation methodology. In essence, the 3GPP primarily focuses on drafting these specifications, leaving the actual implementation details to MNOs and associated vendors. In this study, Table IV presents various approaches an operator could adopt for 5G network slice implementation, each varying in NF sharing and slice-specific features.

Fig. 8 represents the time spent in deploying slices under each scenario. Scenario 1, where all NFs are exclusive to the slice, took approximately 27 s to deploy. Scenarios 2 and 3 share NFs among slices, and employ slice-specific UPF, respectively, recording average deployment times of 19.4 s and 21.9 s. Scenarios 4 and 5 adopt mixed strategies, deploying in about 4 s each. The deployment duration includes tasks such as NF installations, re-configurations, and establishment of connections. The analysis highlights the trade-off between slice deployment time and the strategy of NF sharing and isolation. Local hosting of the container's images accelerated deployment, but increased storage demands.

Fig. 9 illustrates the relationship between the number of new NFs and re-configurations across the five scenarios. As it can be observed, although fresh deployments may offer isolation and easy termination, they require extra resources. On the contrary,

TABLE IV
SLICE DEPLOYMENT SCENARIOS

Scenario Name	Shared NFs	Slice-specific NFs	NF Deployments	NF Reconfigurations
Scenario 1	-	All 5G NFs	All 5G NFs	-
Scenario 2	All 5G NFs	-	-	AMF, NSSF, SMF, UPF
Scenario 3	5G CP NFs	UPF	UPF	AMF, NSSF, SMF
Scenario 4	5G CP NFs	UPF, SMF	UPF, SMF	AMF, NSSF
Scenario 5	5G CP NFs	UPF, SMF, AMF	UPF, SMF, AMF	NSSF

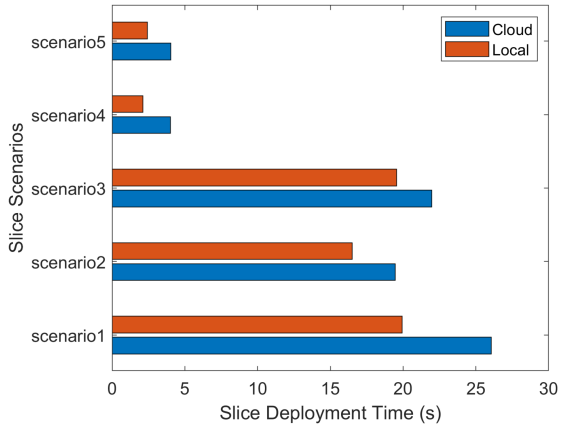

Fig. 8. Slice deployment time.

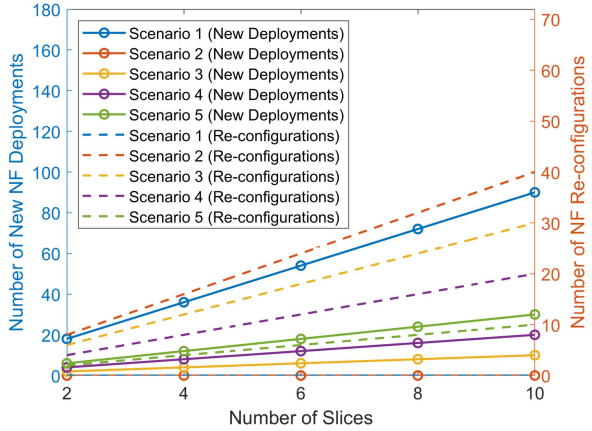

Fig. 9. Number of network functions in the slice scenarios.

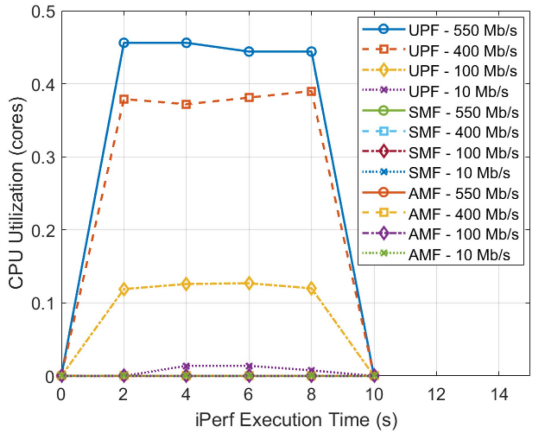

Fig. 10. CPU utilization of 5G core network functions.

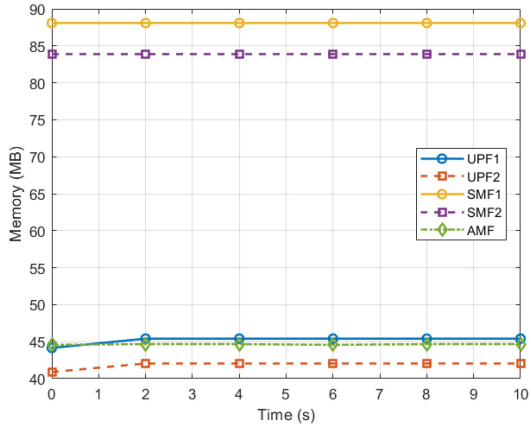

Fig. 11. Memory measurements for two slices.

NF re-configurations minimize resource demands but may risk service disruptions and configuration errors.

To assess resource utilization on the Visiting MNO's infrastructure, we have monitored CPU and memory under different UE traffic loads. CPU experiments have been conducted within a slice, with its data rate varying by user subscription. The *iPerf* tests, demonstrated increased CPU utilization, particularly in the UPF, under high UE traffic (Fig. 10). Memory measurements have been performed in two distinct slices: slice1 and slice2, based on Scenario 4 with slice-specific SMF and UPF. Slice1 (SMF1, UPF1) has been set with a maximum data rate of 500 Mb/s, while slice2 (SMF2, UPF2) with a rate of 100 Mb/s. Despite the variations in data rate and traffic conditions, memory usage has remained relatively stable across the board (Fig. 11). It is worth noting that, control plane functions, especially the SMFs, have consumed significantly more memory than the UPFs.

TABLE V
EDGE-CLOUD SITE PLACEMENTS

Scenario Name	UE/gNB	UPF	App	5G CP NFs
edge1	Edge	Edge	Edge	Cloud
edge2	Edge	Edge	Cloud	Cloud
cloud1	Edge	Cloud	Cloud	Cloud
cloud2	Edge	Cloud	Edge	Cloud

D. Evaluation Results on Post-Federation Slice Deployments Impacting End User's Performance

In addition to evaluating metrics that impact an MNO, we also analyzed end-user performance, focusing on latency and throughput crucial metrics for the end-user experience.

In the 5G-edge computing combination, low latency is crucial. By leveraging edge computing and 5G's distributed UPF model, applications can be located nearer to users, enhancing response times. Table V illustrates various application and NF placements

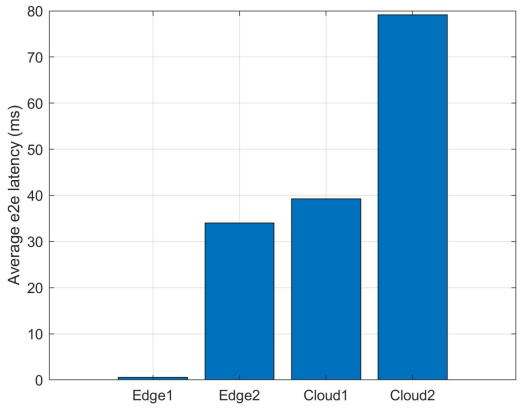

Fig. 12. Latency measurements to edge-cloud scenarios.

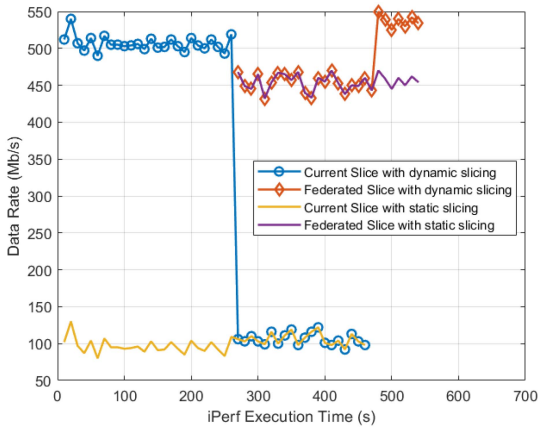

Fig. 13. Slice data rate in Visiting MNO.

between edge and cloud, each reflecting potential real-world deployments and latency variations. We then have evaluated different UPF placements, with the UE accessing the iPerf server (mimicking user service) via 5G and edge-cloud computing, to measure latency.

In the edge1 scenario, the UPF and iPerf server are co-located in the edge node, providing low latency, vital for V2X services. Conversely, edge2 allocates the iPerf server to the cloud, balancing edge resource efficiency with cloud computation power—practical for video streaming or online gaming. In both cloud scenarios (cloud1 and cloud2), the UPF is paired with control plane functions in the cloud, centralizing orchestration but increasing latency due to potential infrastructure limitations. Also, in edge2, cloud1, and cloud2, additional simulated network latency is added per service (UPF pod and iPerf server pod), using the *tc* command of the *iproute* package, representing real-world scenarios. Fig. 12 illustrates *ping* experiments from the UE to iPerf server. Edge1 scenario exhibits the lowest latency without added simulation network latency. Cloud-hosted UPF and/or iPerf server results in increased latency, highlighting the importance of strategic UPF placement for meeting slice delay requirements.

User throughput within a slice needs attention, especially in scenarios with coexisting federated and existing slices. Fig. 13 shows that static slicing allocates only the guaranteed bit rate to both slices. However, dynamic slicing adjusts resources based on each slice's needs, leading to variable data rates. Table VI summarizes the slice requirements. We have set the federated slice's maximum data rate at 550 Mb/s to emphasize its demands. The gap between maximum and guaranteed rates highlights the bandwidth flexibility of network slices. This setup illustrates the challenges faced when federated slices operate near peak capabilities. In resource-constrained situations, unassociated users might experience degraded performance due to limited resources.

TABLE VI
SLICE REQUIREMENTS IN VISITING OPERATOR

	Current Slice	Federated Slice
Maximum DR (Mb/s)	500	550
Guaranteed DR (Mb/s)	100	450

VI. LEARNINGS AND CONSIDERATIONS

In this section, we summarize the key lessons from previous sections and discuss potential applications and future directions for our novel slice federation framework.

A. Key Learnings

1) Operational Complexities: The complexity of the SFaaS framework has been closely linked with the number of services that need to be federated to the Visiting Operator, per slice request. This study has showcased a single *SliceFedRequest* and xApp, but real-world scenarios might involve multiple xApps or combinations. This lengthens the federation process and complicates communication between orchestrator platforms, behaving largely in an $O(n)$ or $O(n^2)$ manner. Moreover, the deployment strategy of the slice can modulate this complexity. Choosing entirely new network functions for each slice mirrors $O(n)$ complexity, while adjusting settings of existing functions to accommodate a new slice can range between $O(n)$ to $O(n^2)$, depending on interactions and configurations. However, it is important to note that external factors, such as network latency or hardware efficiency, can modify the practical outcomes despite the theoretical complexity profile.

2) Federation Dependencies: Creating a federated slice in the Visiting Operator depends on computational resources, federation implementation, and the orchestrator's logic. On-demand provisioning of network slices requires dynamic NF and service migrations across edge clouds. Results in Fig. 7 show that service migrations for user groups take seconds, emphasizing the need for self-adaptive networks, such as those using Deep Reinforcement Learning, and the importance of machine learning for mobility prediction, service migration, and efficient slice instantiation [44]. MLOps advancements, especially in Kubernetes, have simplified this process [45].

3) Security Implications: Security remains complex, particularly during federation sessions or slice creations. Shared network slices require a security-centric design, addressing user privacy and robust controls. Advances like [46] and [47] highlight privacy and decentralized systems in healthcare cyber-physical

systems. Our framework prioritizes security in the federation phase, ensuring authentications during orchestrator access and reinforcing post-federation security. In O-RAN architectures spanning different domains, security and trust management between vendors are crucial.

B. Potential Applications

1) Automotive: Whether human or automated, drivers benefit from "look ahead" data such as potential road hazards. As vehicles transition between networks, slice federation becomes vital for continuous service. Especially for remote driving involving drones or vehicles, it ensures consistent service quality and response, no matter the network connection.

2) Cloud Gaming: Players engage in real-time augmented reality games, and due to game fairness or enhanced experiences, some might need to access edge services of another operator. Slice federation ensures consistent gameplay and performance even when players roam across networks.

3) Privacy-Preserving Health Assistant: Wearable health devices collect vast amounts of data. An edge-based health assistant manages this data flow. Slice federation ensures seamless monitoring and feedback, even when accessing services from another operator's edge server.

4) Infrastructure Sharing and the MNO-MVNO Paradigm: MNOs and Mobile Virtual Network Operators (MVNOs) collaborate to share infrastructure. Slice federation ensures seamless operation and consistent service delivery, even when transitioning between MNO and MVNO networks.

C. Future Directions

Based on this study's findings, several promising research areas emerge. Investigating O-RAN-based base stations that support network slicing, as well as CU/DU splits, is crucial to further evaluate RAN slice federation. Additionally, there is potential in expanding our framework to incorporate machine learning-based analytics, improving the efficiency of slice instantiation. In the evolving 5G landscape, it would also be beneficial to explore orchestrating core and RAN solutions, particularly in environments where APIs are not exposed.

VII. Conclusion

In this paper, we presented a novel framework for slice federation between operators in cloud-native 5G networks, aligning with the OP concept and the 3GPP service slice model architecture. The presented results highlighted two key aspects: Firstly, we analyzed the overall federation deployment time, which indicates the duration required for an application to be executed in the Visiting MNO's edge site. We evaluated this metric across two different use-case applications. Secondly, we assessed the impact of post-federation slice deployments on the operator's infrastructure and examined the performance experienced by end users. Looking ahead, our future work aims to extend our framework with machine learning-based analytics for predicting user mobility, optimizing service migration, and ensuring efficient slice instantiation.

References

[1] N. Maynard and D. Sat, "Why operators are driving connected vehicles forward," White Paper, 2023.

[2] A. Boualouache et al., "5G vehicle-to-everything at the cross-borders: Security challenges and opportunities," *IEEE Internet Things Mag.*, vol. 6, no. 1, pp. 114–119, Mar. 2023.

[3] D. Liu, F. Sun, W. Wang, and K. Dev, "Distributed computation offloading with low latency for artificial intelligence in vehicular networking," *IEEE Commun. Standards Mag.*, vol. 7, no. 1, pp. 74–80, Mar. 2023.

[4] N. Li et al., "Transforming the 5G RAN with innovation: The confluence of cloud native and intelligence," *IEEE Access*, vol. 11, pp. 4443–4454, 2023.

[5] NGMN, "Cloud native manifesto: An operator view 1.0," NGMN Alliance, Sep. 13, 2023.

[6] Q. Zhao et al., "Customizable cloud-native infrastructure for private 5G," in *Proc. 26th Conf. Innov. Clouds Internet Netw. Workshops*, 2023, pp. 50–57.

[7] I. Chih-Lin, S. Kuklinski´, and T. Chen, "A perspective of O-RAN integration with MEC, SON, and network slicing in the 5G era," *IEEE Netw.*, vol. 34, no. 6, pp. 3–4, Nov./Dec. 2020.

[8] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 1376–1411, Second Quarter 2023.

[9] BubbleRAN, "MX-ORS: 4G/5G open RAN Studio." Accessed: Feb. 14, 2024. [Online]. Available: https://bubbleran.com/products/mx-ors/

[10] O-RAN Working Group 6, "O-RAN acceleration abstraction layer general aspects and principles 7.0," O-RAN Alliance, R003, Oct. 2023.

[11] 3GPP TS 28.530, "5G; management and orchestration; concepts, use cases and requirements," 3GPP, V17.4.0, Release 17, Apr. 2023.

[12] GSMA, "E2E network slicing architecture version 1.0," White Paper, 2021.

[13] O-RAN Working Group 1, "O-RAN slicing architecture 11.0," O-RAN Alliance, R003, Oct. 2023.

[14] A. Cárdenas, D. Fernández, C. M. Lentisco, R. F. Moyano, and L. Bellido, "Enhancing a 5G network slicing management model to improve the support of mobile virtual network operators," *IEEE Access*, vol. 9, pp. 131382–131399, 2021.

[15] B. Coll-Perales et al., "Improving the latency of 5G V2N2V communications in Multi-MNO scenarios using MEC federation," in *Proc. IEEE 95th Veh. Technol. Conf.*, 2022, pp. 1–5.

[16] X. Cao, G. Tang, D. Guo, Y. Li, and W. Zhang, "Edge federation: Towards an integrated service provisioning model," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1116–1129, Jun. 2020.

[17] M. Valdez, D. Carrera, and A. Antonopoulos, "Standardization initiatives and market approaches in edge federation," in *Proc. IEEE Conf. Standards Commun. Netw.*, 2022, pp. 78–81.

[18] GSMA, "Operator platform concept whitepaper; phase 1: Edge cloud computing," GSMA Assosiation, Jan. 2020.

[19] S. D'Oro, L. Bonati, M. Polese, and T. Melodia, "OrchestRAN: Orchestrating network intelligence in the open RAN," *IEEE Trans. Mobile Comput.*, early access, Dec. 13, 2023, doi: 10.1109/TMC.2023.3342711.

[20] S.-P. Yeh, S. Bhattacharya, R. Sharma, and H. Moustafa, "Deep learning for intelligent and automated network slicing in 5G open RAN (ORAN) deployment," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 64–70, 2024.

[21] N. Godinho et al., "OREOS: Demonstrating E2E orchestration in 5G networks with open-source components," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2023, pp. 1–2.

[22] S. H. Aljbour and A. Y. Alma'aitah, "An inter/intra slice handover scheme for mobility management in 5G network," in *Proc. 13th Int. Conf. Inf. Commun. Syst.*, 2022, pp. 87–92.

[23] R. A. Addad, D. L. C. Dutra, T. Taleb, and H. Flinck, "AI-based network-aware service function chain migration in 5G and beyond networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 19, no. 1, pp. 472–484, Mar. 2022.

[24] S. S. Shinde et al., "A network operator-biased approach for multi-service network function placement in a 5G network slicing architecture," *Comput. Netw.*, vol. 201, 2021, Art. no. 108598.

[25] ETSI GS NFV-IFA 030, "Network functions virtualisation (NFV) release 3; management and orchestration; multiple administrative domain aspect interfaces specification," NFV ETSI ISG, V3.2.1, Apr. 2019.

[26] MEF Standard 55.1, "Lifecycle service orchestration (LSO): Reference architecture and framework," MEF Forum, Jan. 2021.

[27] X. Li et al., "5Growth: An end-to-end service platform for automated deployment and management of vertical services over 5G networks," *IEEE Commun. Mag.*, vol. 59, no. 3, pp. 84–90, Mar. 2021.

[28] J. Nasreddine et al., "5GMED architecture for advanced automotive and railway communication services in cross-border scenarios," in *Proc. IEEE Future Netw. World Forum*, 2022, pp. 36–42.
[29] I. Sarrigiannis, A. Antonopoulos, K. Ramantas, M. Efthymiopoulou, L. M. Contreras, and C. Verikoukis, "Cost-aware placement and enhanced lifecycle management of service function chains in a multidomain 5G architecture," *IEEE Trans. Netw. Serv. Manage.*, vol. 19, no. 4, pp. 5006–5020, Dec. 2022.
[30] S. D. A. Shah, M. A. Gregory, S. Li, R. D. R. Fontes, and L. Hou, "SDN-Based service mobility management in MEC-Enabled 5G and beyond vehicular networks," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13425–13442, Aug. 2022.
[31] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, "On multi-domain network slicing orchestration architecture and federated resource control," *IEEE Netw.*, vol. 33, no. 5, pp. 242–252, Sep./Oct. 2019.
[32] A. Solano and L. M. Contreras, "Information exchange to support multi-domain slice service provision for 5G/NFV," in *Proc. IFIP Netw. Conf.*, 2020, pp. 773–778.
[33] B. Hortiguela et al., "Realizing the network service federation vision: Enabling automated multidomain orchestration of network services," *IEEE Veh. Technol. Mag.*, vol. 15, no. 2, pp. 48–57, Jun. 2020.
[34] S. D. A. Shah, M. A. Gregory, and S. Li, "Cloud-native network slicing using software defined networking based multi-access edge computing: A survey," *IEEE Access*, vol. 9, pp. 10903–10924, 2021.
[35] S. D. A. Shah, M. A. Gregory, and S. Li, "Toward network-slicing-enabled edge computing: A cloud-native approach for slice mobility," *IEEE Internet Things J.*, vol. 11, no. 2, pp. 2684–2700, Jan. 2024.
[36] Nearby Computing, "NearbyOne edge orchestrator; product description," Jul. 2021.
[37] A. Jalalian et al., "Network slicing in virtualized 5G core with VNF sharing," *J. Netw. Comput. Appl.*, vol. 215, 2023, Art. no. 103631.
[38] M. Chahbar, G. Diaz, A. Dandoush, C. Cérin, and K. Ghoumid, "A comprehensive survey on the E2E 5G network slicing model," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 49–62, Mar. 2021.
[39] GSMA OPG, "Operator platform telco edge requirements version 4.0," GSMA Association, Mar. 29, 2023.
[40] R. Ntassah, G. M. Dell'Aera, and F. Granelli, "xApp for traffic steering and load balancing in the O-RAN architecture," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 5259–5264.
[41] P. Upadhyaya et al., "Prototyping next-generation O-RAN research testbeds with SDRs," 2022, *arXiv:2205.13178*.
[42] O-RAN Software Community, "The O-RAN software community (SC) documentation," O-RAN Alliance, 2023. Accessed: Feb. 14, 2024. [Online]. Available: https://docs.o-ran-sc.org/en/latest/
[43] A. Kliks et al., "Towards autonomous open radio access networks," *ITU J. Future Evolving Technol.*, vol. 4, pp. 251–268, 2023.
[44] M. Zawish, N. Ashraf, R. I. Ansari, and S. Davy, "Energy-aware AI-Driven framework for edge-computing-based IoT applications," *IEEE Internet Things J.*, vol. 10, no. 6, pp. 5013–5023, Mar. 2023.
[45] G. Samaras, V. Theodorou, D. Laskaratos, N. Psaromanolakis, M. Mertiri, and A. Valantasis, "QMP: A cloud-native MLOps automation platform for zero-touch service assurance in 5G systems," in *Proc. IEEE Int. Mediterranean Conf. Commun. Netw.*, 2022, pp. 86–89.
[46] Z. Lian, W. Wang, Z. Han, and C. Su, "Blockchain-based personalized federated learning for internet of medical things," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 4, pp. 694–702, Oct.–Dec. 2023.
[47] Z. Lian et al., "DEEP-FEL: Decentralized, efficient and privacy-enhanced federated edge learning for healthcare cyber physical systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3558–3569, Sep./Oct. 2022.

Michail Dalgitsis is currently working toward the Ph.D. degree from the Polytechnic University of Catalonia, Barcelona, Spain. He is currently a Research Engineer with Nearby Computing, Barcelona, Spain. His main research interests include network slicing and orchestration of cloud-native 5G/6G networks.

Nicola Cadenelli received the M.S. degree in computer science and engineering with the Universita degli Studi di Brescia, Brescia, Italy, in 2014, and the Ph.D. degree in computer architecture with the Technical University of Catalonia, Spain, in 2019. He is currently a Senior Solution Architect with Nearby Computing, Spain. During his studies, he was affiliated and visited various intuitions, including the University of Applied Sciences of Leipzig, Leipzig, Germany, during 2012–2013, Julich Supercomputing Center, Germany in 2014, Massachusetts Institute of Technology, Cambridge, MA, USA in 2018, and Barcelona Supercomputing Center, Barcelona, Spain during 2015–2020.

Maria A. Serrano received the Ph.D. degree from the Technical University of Catalonia (UPC), Barcelona, Spain, in 2019. She is currently a Senior Researcher with Nearby Computing. Prior to that, she was a Research Engineer at GTD, and a Postdoctoral Research Scientist with Barcelona Supercomputing Center. She has experince supervising M.Sc. and Ph.D. students with UPC, organizing research events, and participating and leading activities in several research projects. Her research interests include various topics including task scheduling in critical real-time embedded systems, and design, implementation and tasks orchestration of distributed systems in edge/cloud environments.

Nikolaos Bartzoudis received the Ph.D. degree in electronic engineering from Loughborough University, Loughborough, U.K., in 2006. He is currently a Senior Researcher and the Head of the ADAPT Research Unit with CTTC (Spain), where he has acted as principal investigator in 13 projects since 2011. He gave invited lectures/talks, co-organized events, supervised students and was PhD examiner. His research interests include adaptive FPGA computing, functional splits, O-RAN, and AI-driven resource orchestration.

Luis Alonso is currently a Full Professor with the Department of Signal Theory and Communications, UPC-BarcelonaTECH. He has been the Director of the School of Telecommunications and Aerospace Engineering of this university for eight years and Fulbright Fellow with Florida Polytechnic University, Lakeland, FL, USA. He has focused his research on the optimization of wireless communications systems.

Angelos Antonopoulos received the Ph.D. degree from the Polytechnic University of Catalonia (UPC), Barcelona, Spain, in 2012. He is currently the R&I Director of Nearby Computing, leading the innovation efforts of the company in the edge/cloud orchestration domain. He has authored or coauthored more than 100 publications on various topics, including 5G/6G architectures, energy-efficient network planning, and zero-touch service management.