NSFaaS: Network Slice Federation as a Service in Cloud-native 5G and beyond Mobile Networks

Michail Dalgitsis*‡, Nicola Cadenelli*, Maria A. Serrano*,
Nikolaos Bartzoudis†, Luis Alonso‡, and Angelos Antonopoulos*
*Nearby Computing, Barcelona, Spain
†Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Barcelona, Spain
‡Department of Signal Theory and Communications, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

***Abstract*—Network slicing has emerged as a revolutionary solution to fifth generation (5G) network design and operation. However, the inherent mobility of the end users introduces important new and unexplored challenges with regard to the network slice continuity across different administrative domains (i.e., networks controlled by different operators). In this paper, we introduce Network Slice Federation as a Service (NSFaaS), 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 existing standards on network slicing and operator federations. In addition, we have designed and deployed a cloud-native federated 5G experimental platform to demonstrate the feasibility of the proposed framework and assess its performance in terms of "post-federation" slice creation.**

***Index Terms*—Slicing, Edge Federation, NFV, Cloud-native.**

I. Introduction

As we embrace the era of the fifth generation (5G) mobile communication technology, the demand for unprecedented connectivity and diverse service requirements has soared to new heights. To meet these evolving needs, network slicing has emerged as a promising solution, revolutionizing the conventional approach to network design and operation. By enabling the creation of multiple virtual networks tailored to specific applications or user groups within a shared physical infrastructure, network slicing unlocks a new level of flexibility, scalability, and efficiency in 5G and beyond networks [1].

On the other hand, the inherent mobility of users, especially in cross-border scenarios or across different administrative domains (e.g., in case of national roaming), introduces new challenges to network slicing. As users traverse different geographic regions and network domains, the exchange of network slicing information among different network operators becomes crucial for seamless service continuity. Ensuring efficient information exchange poses significant challenges due to varying network infrastructures, divergent management systems, and differing policy enforcement mechanisms among operators. Addressing these challenges is vital to unlock the full potential of network slicing in facilitating seamless and uninterrupted services for mobile users [2].

To tackle the service continuity challenges in mobile scenarios, slice orchestration in cloud-native networks has been lately studied in the literature. The authors in [3] study slice isolation and core-edge collaboration to achieve availability and quality of service (QoS) optimization, while the authors in [4] focus on the performance of a RAN slice subnet by improving the virtual resource utilization. However, slice mobility has not been considered in their approach. The problem of re-programming and re-provisioning slices, due to mobility events from moving end-users, is addressed in [5]–[7]. Although these works study the dynamic nature of slice management in the presence of mobile users, they do not consider multiple administrative domains, while they mainly focus on the algorithmic aspects and logic of network slice creation. Moving towards cloud-native 5G and beyond network topologies, the practical aspects towards slice creation become of utmost importance.

To that end, GSMA has recently introduced the Operator Platform (OP) concept, a key enabler for Edge Federation [8]. The OP framework encompasses established Application Programming Interfaces (APIs) and protocols for tasks such as edge computing resource discovery, resource management, and service deployment. Embracing the OP guidelines allows network operators and service providers to efficiently develop and implement novel edge services, ensuring seamless interoperability and adherence to industry standards. However, when it comes to network slicing federation, the progress of OP is currently in its emerging phase with no defined slice mechanisms or API endpoints established yet.

In this paper, motivated by the aforementioned challenges, we introduce a novel *Network Slice Federation as a Service* (NSFaaS) 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 existing APIs for the exchange of the slice resource template among operators to ensure the end users' seamless mobility. In addition, we design and deploy a cloud-native 5G testbed to emulate realistic federated environments and assess the "post-federation" network performance in terms of the slice creation under various scenarios.

The structure of the paper is as follows. Section II presents the system architecture of our work, while Section III introduces the proposed framework for network slice federation. Section IV provides the details for the deployed testbed and the evaluation of our novel framework. Finally, Section V concludes our work and provides some ideas for future research.

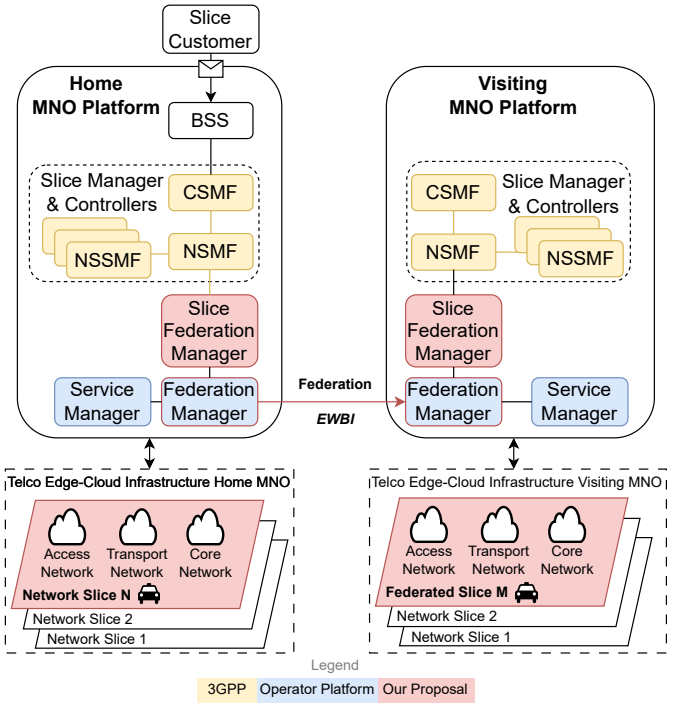

Fig. 1: System architecture

II. Network Model

Our system follows a multi-stakeholder architecture, as illustrated in Figure 1, comprising three distinct entities: i) the Mobile Network Operator (MNO)[1], ii) the slice customer, and iii) the end-user. The MNO owns the telco infrastructure, where cloud-native applications and mobile network functions (NFs) are running, along with a cloud orchestration platform based on the OP concept with extensions, consisting of a *Slice Manager*, a *Service Manager*, a *Federation Manager*, and a *Slice Federation Manager*. The slice customer requests a communication service in the form of a network slice to meet the needs of its end-users' specific characteristics and requirements. The end-users represent the final consumers of both the applications and network services and we assume that 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). It is also worth noting that the core network slicing is facilitated by the 5G service-based architecture, where the control plane functions, i.e., 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), Unified Data Repository (UDR), as well as the User Plane Function (UPF), are deployed as virtual functions. 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) have been defined by 3GPP, ensuring efficient resource allocation based on the specific requirements of each slice [9].

The *Service Manager* is defined in the OP concept and 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. Moreover, the *Federation Manager* facilitates application mobility among MNOs through the East-West Bound Interface (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.

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).

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

III. NSFaaS: Network Slice Federation as a Service Mechanism

In this section, we introduce the technical details for the implementation of the *Network Slice Federation as a Service* (NSFaaS) mechanism, along with the slice federation phases.

A. NSFaaS technical implementation

To leverage the existing Edge Federation approach by OP and initiate slice federation resource requests, we propose the encapsulation of the introduced API requests in an *NSFaaS* 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 network functions, 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[2], Kubernetes[3] and Helm Charts[4]. 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.

Furthermore, during the federation setup, the Visiting MNO shares a Kubernetes cluster at an edge site to accommodate the *SliceFedRequest*. In addition to the edge site, the Visiting MNO must also share clusters hosting the core and RAN NFs for the slice.

It is also worth noting that the *SliceFedRequest* has been designed based on the Kubernetes container lifecycle hooks[5] and liveness probes[6]. 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.

The purpose of a liveness probe in the NSFaaS mechanism is to ensure that the Visiting *Slice Federation Manager* is running correctly, to read the status of the federated slice, and to take action if it becomes unresponsive or enters a faulty state. If the liveness probe fails (i.e., the *Slice Federation Manager* does not respond within a specified timeframe or returns error status), Kubernetes takes action based on the configured probe settings. The type of the liveness probe used is the *Exec Probe*, which executes a status command to Visiting *Slice Federation Manager* and saves the command's response. The response then is read by the Home *Slice Federation Manager* to keep the synchronization with the Visiting MNO.

By using these container lifecycle hooks and liveness probes, we customize the behavior of the *SliceFedRequest* application and ensure proper slice federation initialization, cleanup, and availability processes as part of their lifecycle management.

B. Slice federation phases

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

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 Figure 2 (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 Figure 2 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

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

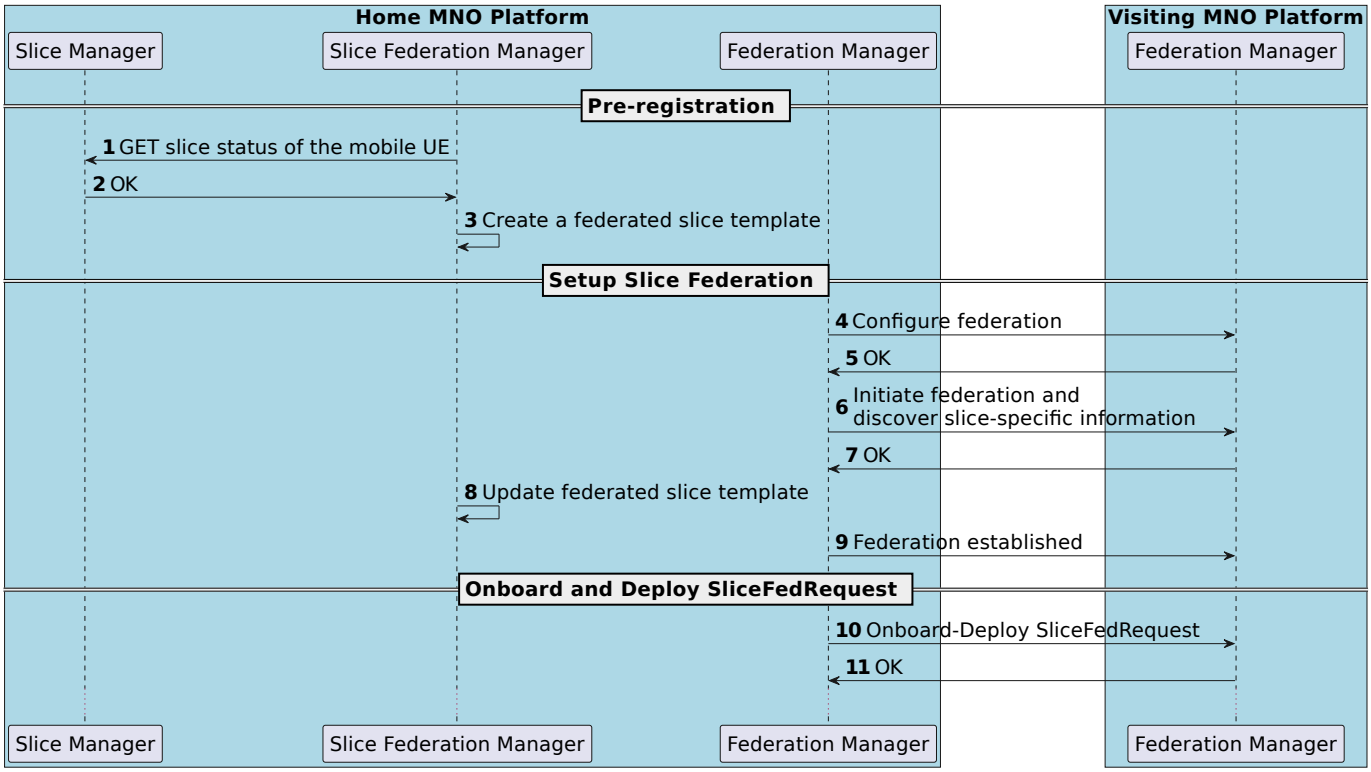

Fig. 2: Slice federation flowchart between two MNOs

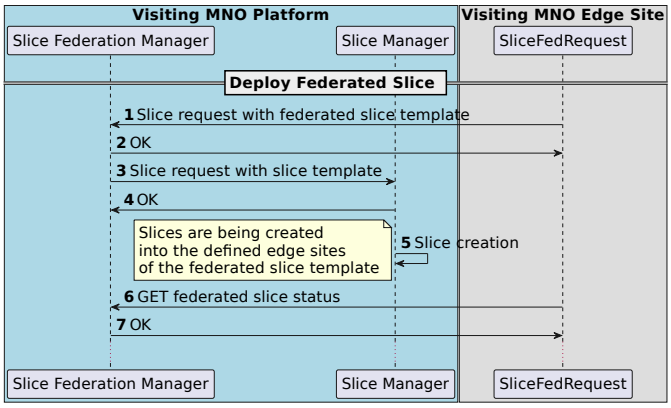

Fig. 3: Slice federated deployment flowchart in Visiting MNO

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 first uploaded into a public/private registry, then the artefacts referencing these files (e.g., helm charts with their respective Kubernetes manifests like *Deployments* and *ConfigfMaps*) and, finally, onboard the application that will reference the artefacts. With the application onboarded, the application can be installed in an edge site (e.g., Kubernetes cluster), as depicted by steps 10-11 of Figure 2, and it will start to run, triggering the federated slice request.
4) **Phase 4: Federated Slice Deployment**
After the deployment of the *SliceFedRequest*, a federated slice request is sent to the Visiting MNO's *Slice Federation Manager* (Fig. 3). The FST is translated back to ST, and the slice deployment begins through the respective domain NSSMF on the infrastructure of the Visiting MNO. The *SliceFedRequest* periodically gets the status of the federated slice request and, if it is accepted, the Home *Slice Federation Manager* updates its content for the federated slice instance. Finally, the *Network Slice Manager* of the Visiting MNO has all the pieces to deploy the federated slice to serve the user.
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. Finally, when all resources related to slice federation have been cleaned-up, the federation session is released.

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

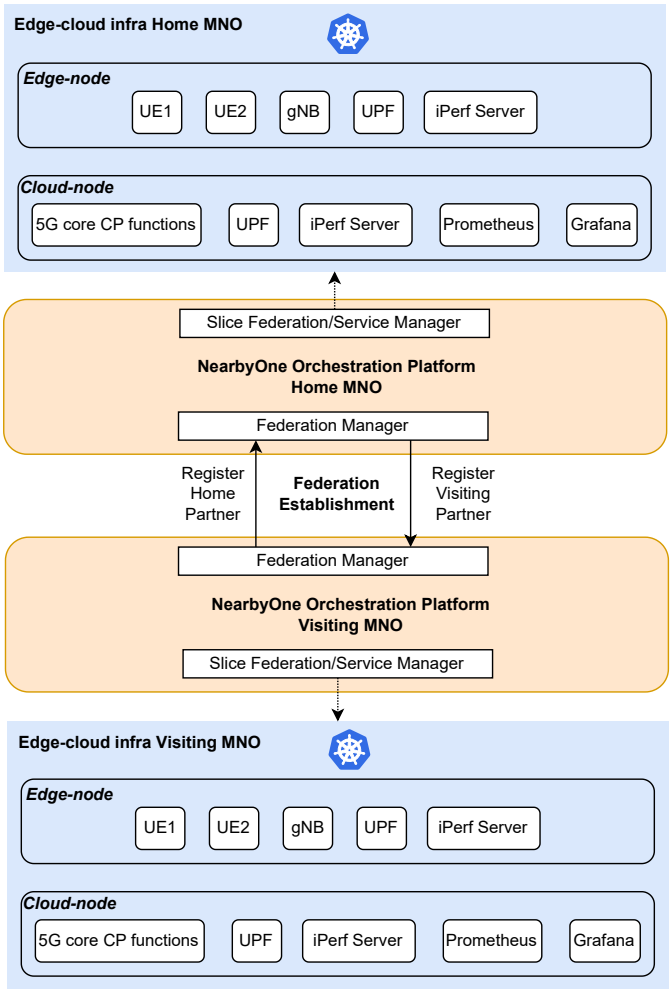

Fig. 4: Cloud-native 5G experimental platform

IV. Performance Evaluation

In this section, we first describe the setup of the environment employed in our experiments and, then, we evaluate the "post-federation" network performance.

A. Experimental setup

We have designed and deployed a cloud-native 5G experimental platform to evaluate the federation and assess the impact of post-federation slice deployments on the Visiting MNO's infrastructure and end-user performance. The architecture of the platform is depicted in Figure 4, including two MNOs, i.e., the Home MNO and the Visiting MNO, each with an orchestration platform and an edge-cloud infrastructure for slice federation and slice deployments, respectively. The NearbyOne Controller [10] serves as the orchestration platform, facilitating the federation between the operators by establishing EWBI for seamless communication between the two operators.

Within the compute infrastructure of each operator, 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[7], and Grafana[8] for 5G core connectivity, monitoring, and visualization purposes, respectively. On the other hand, the edge-node is responsible for running the UPF, an emulated gNB and a UE from the UERANSIM open-source project, as well as an iPerf server[9]. This distributed setup allows for realistic network condition simulation, RAN emulation with UERANSIM, comprehensive monitoring and analysis of testbed performance with Prometheus and Grafana, and throughput measurement and analysis with iPerf-server and UE as iPerf-client, respectively. The testbed parameters are summarized in Table I.

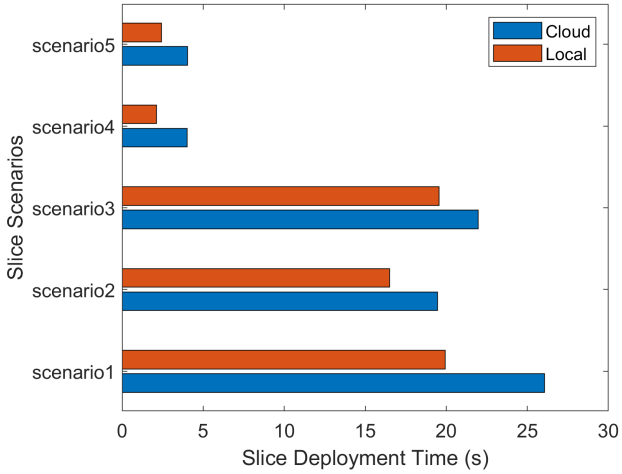

Fig. 5: Slice deployment time

TABLE I: 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

5G's modular architecture allows network slicing to serve distinct services for UEs. In this study, five slice scenarios (Table II), each varying in NF sharing and slice-specific features, were explored using the Open5Gs setup.

B. Experimental Results

Figure 5 shows the slice deployment time for the five different scenarios, for two distinct cases for the location of the core NF images: Cloud and Local. It is worth noting that the deployment duration includes tasks such as NF installations, re-configurations, and establishment of connections. Starting from the Cloud case, in Scenario 1, where all NFs are exclusive to each slice and hence they need to be deployed from scratch,

[7]https://github.com/prometheus-community/helm-charts
[8]https://github.com/grafana/helm-charts
[9]https://iperf.fr/

TABLE II: 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

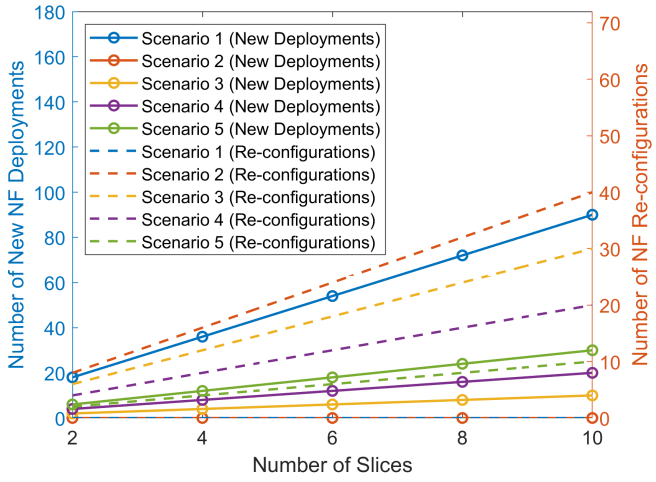

Fig. 6: Number of network functions in the slice scenarios

the slice takes approximately 27 s to be deployed. Scenarios 2 (all NF shared) and 3 (control NF shared and slice-specific UPF) record average deployment times of 19.4 s and 21.9 s, respectively. Scenarios 4 and 5 adopt mixed strategies, deploying in about 4 s each. For the Local case, the times are lower, however the same behavior among the different scenarios is observed. The results in Figure 5 highlight also the trade-off between slice deployment time and the strategy of NF sharing and isolation. Moreove, local hosting of the container images can accelerate the slice deployment, but they come a the cost of increased storage demands.

Figure 6 illustrates the relationship between the number of new NFs (left y-axis/solid lines) and re-configurations (right y-axis/dashed lines) for different number of slices across the five scenarios. As it can be observed, as the number of slices increases, the total number of deployments and re-configurations increases, but with different ratio (deployments vs. re-configurations) per scenario. This outcome is quite important, since fresh deployments may offer isolation and easy termination, however they require extra resources. On the contrary, NF re-configurations minimize resource demands but may risk service disruptions and configuration errors.

V. 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 proposed framework could guide the network operators for the exchange of slice information and the deployment of network slices in scenarios where different administrative domains (e.g., cross-border) are involved. As a next step, we are planning to extend our work considering up-to-date RAN architectures.

Acknowledgment

This work was partially funded by the 5GMED (951947) project from EC-H2020 programme, as well as the FREE-6G (TSI-063000-2021-144) and SUCCESS-6G (TSI-063000-2021-39/40/41) projects from the UNICO5G-RPTR programme.

References

[1] T. K. Rodrigues and N. Kato, "Network slicing with centralized and distributed reinforcement learning for combined satellite/ground networks in a 6G environment," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 104–110, 2022.

[2] J. Nasreddine *et al.*, "5GMED Architecture for Advanced Automotive and Railway Communication Services in Cross-Border Scenarios," in *IEEE FNWF*, 2022.

[3] Y. Wang, N. Li, P. Yu, W. Li, X. Qiu, S. Wang, and M. Cheriet, "Intelligent and collaborative orchestration of network slices," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1239–1253, 2023.

[4] M. A. Habibi, F. Z. Yousaf, and H. D. Schotten, "Mapping the VNFs and VLs of a RAN slice onto intelligent PoPs in beyond 5G mobile networks," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 670–704, 2022.

[5] S. H. Aljbour *et al.*, "An Inter/Intra Slice Handover Scheme for Mobility Management in 5G Network," in *13th International Conference on ICICS*, 2022.

[6] R. A. Addad *et al.*, "AI-Based Network-Aware Service Function Chain Migration in 5G and Beyond Networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 472–484, 2022.

[7] S. S. Shinde *et al.*, "A network operator-biased approach for multi-service network function placement in a 5G network slicing architecture," *Computer Networks*, vol. 201, p. 108598, 2021.

[8] GSMA, "Operator Platform Concept Phase1: Edge Cloud Computing," White Paper, 2020.

[9] M. Chahbar *et al.*, "A comprehensive survey on the E2E 5G network slicing model," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 49–62, 2021.

[10] Nearby Computing, "NearbyOne Edge Orchestrator," Product description, 2021.