





NetOr: A Microservice Oriented Inter-Domain Vertical Service Orchestrator for 5G Networks

Rafael Direito   [Instituto de Telecomunicações and Universidade de Aveiro | rdireito@av.it.pt]

Daniel Gomes  [Instituto de Telecomunicações and Universidade de Aveiro | dagomes@av.it.pt]

João Alegria [Instituto de Telecomunicações and Universidade de Aveiro | joao.p@av.it.pt]

Daniel Corujo  [Instituto de Telecomunicações and Universidade de Aveiro | dcorujo@av.it.pt]

Diogo Gomes  [Instituto de Telecomunicações and Universidade de Aveiro | dgomes@av.it.pt]

 Instituto de Telecomunicações, Campus de Santiago, Universidade de Aveiro, Aveiro, Portugal.

Received: 01 March 2023 • Accepted: 05 June 2023 • Published: 12 September 2023

Abstract Most modern 5G Vertical Service Orchestrators present various limitations. Among these, one may highlight (i) the employment of monolithic architectures, (ii) the lack of standardized APIs and methodologies, (iii) the minimum support for inter-domain scenarios, and (iv) the impossibility of performing run-time operations over Vertical Services. To achieve a fully-fledged Vertical Service Orchestrator, these problems must be solved. This is the focus of the work presented in this article. Our work presents a new 5G Vertical Service orchestration system named NetOr, which tackles all the listed limitations and can support complex and intricate use cases. NetOr was implemented according to a microservice architecture. Thus, it has increased flexibility, scalability, and maintainability. Moreover, NetOr inherited most advantages of the modern Vertical Service Orchestrators and, therefore, can be considered an improvement of said orchestrators. Furthermore, this work also presents a Proof-of-Concept solution to achieve inter-domain communication through the orchestration of an End-To-End Network Slice that establishes several VPN tunnels between the domains encompassed by the Network Slice.

Keywords: 5G, Vertical Service, Orchestrator, Inter-Domain, Slicing, Microservices, NFV

1 Introduction

In addition to being a new generation of radio networks, Fifth Generation (5G) technologies take into account key basic concepts such as Service-based Architectures, Software-Defined Networks (SDN), Network Functions Virtualization (NFV), and End-to-End (E2E) Network Slicing (Erunkulu *et al.*, 2021). These technologies foster the virtualization of specialized network gear, lowering infrastructure costs while expanding the number of scenarios supported without modifying the infrastructure. With the expansion of NFV and the attempts to support and integrate such technologies, more complex scenarios and services emerged.

Some of these include inter-domain, in which an E2E service crosses multiple administrative domains, extending the service's coverage area. These scenarios are highly valuable for specific Vertical sectors. An example of one Vertical sector that may heavily leverage inter-domain scenarios is the transportation one. In this sector, most times, there are geographically distributed assets. Furthermore, the different geographical regions where the assets are distributed across, may be controlled by different operators (Fonseca *et al.*, 2021). Thus, this Vertical Sector requires E2E Vertical Service orchestration mechanisms, that can coordinate and manage services distributed among different domains.

To deploy a fully-fledged E2E inter-domain Vertical Service across several administrative domains with no prior knowledge of each other, a new strategy is required. This approach largely relies on network slicing capabilities for segmenting the whole network, which includes the Radio Ac-

cess Network (RAN), the Transport Network (TN), and the Core Network (CN). In this scenario, the network slice is an entity that encompasses and administers various network services or other network slices, playing an important part in the establishment of inter-domain vertical services.

Solutions and systems to handle these scenarios already exist, although they are restrictive or in their early stages. Most of the existing inter-domain solutions are offered as monoliths, and thus cannot scale when a high number of services is being orchestrated/managed, which is restrictive. Furthermore, most existing solutions suffer from insufficient inter-domain support, being very limited in the actions they can perform over the services distributed across the different domains. Lastly, one of the most troublesome drawbacks of the current solutions is the fact that they do not contemplate well-accepted network slicing standards. This makes it extremely complicated to integrate these solutions with network slicing systems, and thus reduces the impact and the adoption of such solutions.

Vertical Service orchestrators may be extended to deliver distributed inter-domain vertical services since they can handle the instantiation and administration of these services effortlessly. However, the previously mentioned issues (further described in Section 2) may risk the proper construction and maintenance of E2E inter-domain services.

With this work, we aim to address the shortcomings of today's Vertical Service orchestrators by presenting a Proof-of-Concept (PoC) for a new E2E Slicing Platform. This Platform was developed through a service-oriented architecture, resulting in better maintainability, flexibility, and scalability.

Furthermore, our solution addresses most of the problems the existing inter-domain solutions suffer from, providing (i) adequate support for inter-domain, (ii) network slicing mechanisms that obey the current slicing standards, and (iii) mechanisms to perform run-time operations on Vertical Services, for instance. We consider our solution to be novel in the way it offers a fully-fledged system that does not suffer from the various problems addressed before.

The paper is arranged as follows. Section 2 discusses and contrasts the current State of the Art (SotA) to modern Vertical Service Orchestrators. Section 3 presents NetOr; a Network Orchestrator developed to address the issues identified and discussed in Section 2. Section 4 describes the procedures for providing multi-domain E2E slices, and Section 5 presents the achieved results. Finally, Section 6 provides some concluding remarks.

2 Related Works

This Section examines multiple Vertical Service orchestrators, outlining how they orchestrate E2E Vertical Services and describing their strengths and weaknesses. Furthermore, it also comprises some solutions that, although they cannot be considered fully-fledged Vertical Service orchestrators, address network slicing and inter-domain orchestration.

The literature review methodology consisted on a Scopus search for "NFV", "5G", "MANO", "OSS" and "BSS" keywords (take into consideration that both the abbreviated and expanded form of the acronyms were searched). Our initial Scopus search resulted in 94 works, published between 2016 and 2022. Then, the abstracts of these articles were screened for the foundational principles associated with our work, which allowed us to reduce our initial list of 94 articles to 22 works. This filtered set of papers was thoroughly analyzed, selecting the ones that provided novel insights, approaches or addressed critical challenges associated with the work, leading to the final list of nine selected papers.

We now move on to further elaborate on the Vertical Service orchestration systems that resulted from our SotA research strategy.

2.1 Openslice

Openslice, a 5GinFIRE¹ spinoff, offers an open-source Operations Support System (OSS)/Business Support System (BSS) platform. It also allows users to onboard Virtual Network Functions (VNFs), Containerized Network Functions (CNFs), and Network Services (NSs), as well as instantiates NSs. Moreover, it provides TMForum² Open APIs for Service Catalog Management, Ordering, and more.

Openslice defined some abstractions over the VNFs, CNFs and NSs complexities, allowing verticals to request a given service in the most seamless way possible, focusing solely on the problem's logic. The first abstraction is the Service concept, which enables abstracting all underlying complexities and allows for the provision of a single, coherent, and precise service that is ready to be instantiated

and deployed. Other concepts include Resource Facing Services (RFSs), which encapsulate all services that are directly connected to the infrastructure (such as Network Service Descriptors (NSDs)) and Customer Facing Services (CFSs) that encapsulate all services and specifications that interact directly with the user. RFSs and CFSs are both crucial parts of the Service concept. To design a Service, a CFS Specification must be developed, which may include Service Specification Relationships to connect RFSs to CFSs (Tranoris, 2021). Furthermore, Openslice also includes two specialized web portals to simplify the client's interaction with the system.

This platform has the benefits of high-level Service abstraction, segregation of customer/resource facing services, and specialized individual Web Portals. The downsides of this system are its lack of network slicing capabilities, low multi-domain support, and monolithic design.

2.2 ONAP

Open Network Automation Platform (ONAP) is an open-source platform that enables quick administration of services and their lifecycles through real-time, policy-driven orchestration, management, and automation of network and edge computing services (Rodriguez *et al.*, 2020). It offers E2E 5G Network Slicing, in which an E2E Slice is defined as a network service consisting of RAN, TN, and CN slice subnets. ONAP relies on the three layers of slice management functions specified in 3GPP TR 28.801 (3GPP, 2018) to provide Network Slicing. These functions include an internal Communication Service Management Function (CSMF), a Network Service Management Function (NSMF), and a Network Slice Subnet Management Function (NSSMF). However, ONAP also permits the employment of external NSSMFs, which is made feasible via an adapter created particularly to handle this feature. ONAP additionally provides transparent monitoring of Network Slice Instances (NSIs) via slice management interfaces that expose a huge variety of Key Performance Indicatorss (KPIs), allowing for the continuous and comprehensive monitoring of these slices. On top of these KPIs, it is possible to specify many Service Level Agreements (SLAs) and rules, which ONAP will then use to avoid the deterioration of NSIs (Rodriguez *et al.*, 2020).

To do this, the creation, administration, and operation of network slices are delegated to a multitude of internal ONAP components: (i) the Service Orchestrator (SO), which prompts the creation, update, and termination of services; (ii) the ONAP Optimization Framework (OOF), which is meant to provide a policy-driven placement of VNFs throughout multi-domain infrastructures, but has not been fully exploited; (iii) the SDN Controller (SDN-C), which provisions and administrates network resources and (iv) the Application Controller (APP-C), which manages the lifecycle of VNFs and (v) the Active and Available Inventory (AAI), which monitors the services and resources in real-time.

2.3 Free5GMano

Another approach to the coordination of network slices' lifecycles is presented in (Chang and Lin, 2021). There, the au-

¹<https://5ginfire.eu/>

²<https://www.tmforum.org/>

thors suggest the integration of a Management and Orchestration (MANO) with an OSS/BSS system to achieve such coordination. This system relies on open-source solutions, such as free5GMANO, Openstack, Tacker, and free5GC. Free5GMANO is the solution's OSS/BSS. However, only its NSSF was fully integrated within the presented system. Regarding the solution's MANO components, the Network Function Virtualization Orchestrator (NFVO) was provided through Tacker, while the Virtual Infrastructure Manager (VIM) relied on Openstack. Finally, free5GC was employed to enable the 5G Core Network provision.

To offer all intended functionalities, the authors of (Chang and Lin, 2021) developed new protocols and interfaces. These aimed to facilitate communication between the MANO and the OSS/BSS, which resulted in a cooperative workflow. One of the authors' desired functionalities was the integration of the Element Management System (EMS) with the employed OSS. However, such functionality was not provided by free5GMANO and thus had to be implemented by them. This integration enabled the provisioning of 5G Network elements through the OSS by having this system activate the EMS and afterward send a request to MANO. Consequently, the MANO allocates the required virtualized resources and takes control of the remaining Network Services' lifecycle. This lifecycle includes (i) instantiating, (ii) supervising, and (iii) healing. Moreover, the authors' solution also offers fault tolerance mechanisms to handle VNFs failures. To this end, two types of VNF failure causes were identified: virtualized resources and VNF-specific failures. A monitoring service was then designed and integrated with the MANO's Virtual Network Function Manager (VNFM) to monitor virtualized resource failures. Besides detecting faults, this service also provides recovery mechanisms to deal with them.

The solution showcased in (Chang and Lin, 2021) heavily profits from its automation, interoperability, and improved resource utilization. However, some issues are yet to be addressed. The solution can only provision 5G local network slices rather than end-to-end ones. Additionally, it needs to provide more abstraction for the Verticals to deal with the intrinsic complexities of managing NFV artifacts and their requirements. Such abstraction could have been achieved through Vertical Service Blueprints (VSBs) and Vertical Service Descriptors (VSDs). Finally, only Tacker is supported as the solution's NFVO. This system is rather limited when compared to Open Source MANO (OSM) and ONAP. Thus, it restricts the potential of the developed solution.

2.4 Decoupled Inter-Domain Framework for NFV MANO

The authors of (Choi *et al.*, 2022) proposed a cloud-native federation and orchestration framework for managing and orchestrating inter-domain NFV environments. This framework enabled decoupling the federation processes from the orchestration one across networks and cloud and edge domains. To this end, the overall framework comprises two internal frameworks: the federation framework and the orchestration one.

The federation framework employs hierarchical brokering

and distributed binding. Hierarchical brokering is supported by a publish/subscribe paradigm to enable the exchange of abstracted information related to accesses, connectivity, catalogs, and blueprints. Furthermore, this framework minimizes overhead costs and enables a more flexible federation across domains.

The orchestration framework is responsible for resource and services provisioning, configuration, and life cycle management in a cross-domain environment. Besides this, it also ensures that the resources and services being provisioned meet the required SLAs. To achieve this, it performs admission control. This involves evaluating the availability of the resources and services necessary to meet the defined SLAs.

Several advantages result from the solution proposed in (Choi *et al.*, 2022) and are mainly related to the management and orchestration of services in inter-domain environments. The strengths of this solution are (i) its compliance with European Telecommunications Standards Institute (ETSI) NFV standards, (ii) efficient resource usage, (iii) the abstraction for Verticals through VSBs and VSDs, and (iv) the management and assurance of the Services' SLAs. Regardless of these advantages, the proposed system has several drawbacks, including the lack of support for run-time operations and not following the relevant network slicing Standards. Despite the authors' assertions that the solution complies with ETSI NFV standards, the network slicing mechanisms do not.

2.5 Vertical Slicer - 5G-Transformer

The 5G-Transformer project was a 5G Infrastructure Public Private Partnership (5G-PPP) initiative to enhance the mobile transportation network and its transition into a SDN/NFV-based network. Furthermore, it also addressed network slicing scenarios (de la Oliva *et al.*, 2018). This project developed a 5G platform comprised of three primary components: the Mobile Transport and Computing Platform, the Service Orchestrator, and the Vertical Slicer.

The 5G-Transformer (5GT)-Vertical Slicer (VS) is the system's entry point for verticals and the OSS/BSS component of the 5GT administrative domain. This entity coordinates vertical services, making them accessible to verticals via a high-level interface tailored to their logic and requirements (Mangues-Bafalluy *et al.*, 2019). A vertical must negotiate a SLA with the 5GT platform, which is comprised of a collection of Service Level Objectives (SLOs) specified by verticals and centered on their service's needs (for example, the maximum desirable E2E latency). By jeopardizing the technical behavior of the services, a degraded SLA can pose significant issues for verticals. Sometimes, such degradations may even affect the vertical's reputation and commercial leadership. The 5GT platform offers VSBs to deal with the high-level abstraction that is required by verticals. These blueprints can define a network service's composition, creating a scaffold ready to be used when that topology is needed. As a VSB extension, a vertical may design a VSD. This descriptor comprises Quality of Service (QoS) parameters, yielding a deployment procedure that instantiates the established topology and respects the specified QoS values. The platform links each VSD to network slices, which in the

5GT system are an extension of ETSI NFV NSDs (de la Oliva et al., 2018).

The strengths of 5GT-VS include (i) the abstractions of VSBs and VSDs over the intrinsic complexities, (ii) the preliminary support of network slicing through extended NSDs, (iii) the support for multi-domain, and (iv) the SLA management. In regard to drawbacks, (i) the network slicing support does not follow the current slicing standards, (ii) the multi-domain support is rather restricted, (iii) and its architecture is monolithic.

2.6 Vertical Slicer - 5Growth

5Growth (5GR) was another 5G-PPP initiative attempting to expedite the adoption of 5G empowered vertical markets. Its purpose was to validate 5G solutions from the technical and business perspectives. Consequently, this project opted to leverage the achievements and advancements in network slicing, virtualization, and multi-domain solutions of phase 2 5G-PPP projects, such as 5G-Transformer and 5G-Monarch. In addition, the project selected two ICT-17-2018 5G E2E systems, namely 5G-EVE and 5G-VINNI, to test their improvements (5G-PPP, 2019). 5Growth aimed to support industry verticals by offering four key features: (i) a vertical portal for bridging the gap between verticals and 5G facilities, (ii) closed-loop automation, (iii) SLA control for the services lifecycle, and (iv) an Artificial Intelligence (AI)-driven E2E network solution to optimize access, transport, and core networks, along with cloud, edge and fog resources, across multiple domains and technologies (5G-PPP, 2019).

The 5GR project initially leveraged the 5G-Transformer platform, to which expansions and improvements were added. These improvements focused mainly on its building blocks (5GT-VS, 5GT-SO, and 5GT-Mobile Transport and Computing Platform (MTP)). The enhancements comply with both the functional and service requirements of the project's use cases. The platform work plan comprised twelve innovations, each chosen to address a specific deficiency in the underlying platform provided by the 5G-Transformer project. As a continuation of this project, 5GR inherited many of its advantages and flaws. In addition to VSBs and VSDs abstractions over the intrinsic complexities and SLA management, the 5GR project included network slicing according to standards and improved multi-domain support (both in the Vertical Slicer and the Service Orchestrator). However, 5GR's architecture remained monolithic, and since the added innovations were implemented as add-ons, the final platform's quality may not be ideal.

2.7 Comparison of the Vertical Service Orchestrators

Having briefly introduced numerous Vertical Service orchestrators and other network slicing and inter-domain orchestration tools, we can now summarize them, along with their strengths and drawbacks, in Table 1. This summary aims to make it easier to compare them. Furthermore, the last row of Table 1 describes the solution proposed by this work - NetOr. When compared to the SotA Vertical Service orchestrators,

currently, NetOr only does not offer SLA management capabilities, even though we aim to provide such capabilities in a near future, as addressed in Section 6. Sections 3 and 4 address how NetOr provides the remaining capabilities/functionalities listed in Table 1.

3 Network Orchestrator(NetOr)

Considering the SotA vertical service orchestration solutions, we defined a collection of issues that our PoC platform should address: (i) non-standardized network slicing, (ii) insufficient multi-domain support, and (iii) monolithic architectures. In an effort to move forward the SotA, we created Network Orchestrator (NetOr), a system that provides an OSS/BSS system that operates on top of the operator's 5G infrastructures and services.

We do not consider NetOr to be a novel and revolutionary platform for network slicing. Instead, we consider it a platform that improves the existing Vertical Service orchestrators by mending some of their deficiencies.

A critical deficiency is the employment of monolithic architectures by almost all SotA solutions. Even though some vertical service orchestrators state their design is highly modular, many are comprised of various Java Springboot applications³, which must be deployed together. Thus, making these orchestrators monoliths. However, monolithic applications are not the best solution for the scalability, flexibility, and maintainability required for this kind of orchestration system. Considering network slicing, only ONAP, Free5GMano, and 5GR's VS (through an add-on) provide mechanisms that follow the current standards. This issue is highly critical since, without support for current network slicing standards, the interoperability and integrability of these orchestrators are severely diminished. Lastly, regarding multi-domain support, only the 5GR-VS and the solution presented in (Choi et al., 2022) provide adequate support. Still, in the case of the 5GR-VS, this is offered through an add-on. Again, this is a crucial problem because, without proper support for inter-domain situations, intricate vertical use cases that need such capabilities can be provided.

3.1 System Architecture

NetOr, by being an OSS/BSS system that operates on top of the operator's 5G infrastructures and services, abstracts both the intrinsic operations required to establish a network service, and the complexity of the infrastructure and network. Furthermore, by providing such abstractions, it allows for the end-users (vertical industry) to focus solely on designing and developing the services and functions required to accomplish the organization's goals.

NetOr follows a microservice and event-driven architecture. Each entity in the system is handled by a unique component, which communicates with others through a centralized message bus to exchange event messages asynchronously. Several studies have shown the relevance of asynchronous communications in microservice architectures.

³<https://spring.io/projects/spring-boot>

Table 1. Comparative analysis of the presented solutions

Vertical Service Orchestrator	VSBs and VSDs Abstraction	Web Portal	Uses the Defined Slicing Standards	Multi-Domain Support for Network Slices	Service Oriented Architecture	SLA Management	Run-time Operations on Vertical Services
Openslice		✓					✓
ONAP		✓ (ONAP, 2022)	✓		✓	✓ (ONAP, 2022)	✓
Free5GMano			✓		✓		
Decoupled Inter-Domain Framework	✓ (Choi <i>et al.</i> , 2022)			✓	✓	✓	
Vertical Slicer - 5G Transformer	✓	✓				✓	✓
Vertical Slicer - 5GROWTH	✓	✓	✓	✓		✓	
NetOr (Proposed Solution)	✓	✓	✓	✓	✓	Currently not available	✓

In (Shafabakhsh *et al.*, 2020), the authors evaluate the trade-off between synchronous and asynchronous communication in inter-process communication. The results have shown that asynchronous communication induces higher performance, efficiency, and availability with a large scale of users. Moreover, in (Karabey Aksakalli *et al.*, 2021), despite not reaching a consensus regarding the ideal communication pattern, the authors defend that synchronous communication induces more coupling between services. Consequently, this communication pattern may cause timeouts leading to the disruption of services. Such timeouts happen since the communication service must continuously be available to "finish transactions" (Karabey Aksakalli *et al.*, 2021). NetOr's architecture makes the system more scalable, flexible, modular, and efficient.

Distinct and isolated components compose the system, enabling it to scale a unique microservice if needed, easily creating more workers to manage and handle that specific set of operations and entities. That was made possible by having each microservice as a stateless component, meaning they don't need to persist any information. Whenever a microservice needs to persist any information, it will store it in an external memory cache or database, enabling a fault-tolerant mechanism that guarantees data persistence even if the microservice fails at some point. Given such a loose coupling between the services, scaling services in micro-service architectures is easier when compared to monolithic architectures

(Kalske *et al.*, 2018). Thus, NetOr will present higher performance with a large scale of Vertical Services instances compared to other orchestrators in Section 2.

Flexibility is another system characteristic achieved again by having separate and independent components. With each microservice having a robust communication interface aligned with the most recent standards, it allows a seamless replacement of any sub-component, given that the new component provides the same functionalities and follows the same interfaces and standards. With this flexibility, any microservice's internal implementation can be quickly replaced and updated without restrictions over the language or technologies used. This substitution can occur without interfering with the remaining system. Due to its decentralized nature, microservice architectures adopt a "Decentralized Governance" property. In other words, each system component can be developed using its most suitable technology, leading to higher flexibility (Shakir *et al.*, 2021). Such property is not present in monolithic architectures where all system modules must share the same underlying code basis, leading to a lower capacity for replacing or adding new modules.

Modularity, although similar to flexibility, focus on allowing an effortless addition of new components. With a centralized message bus containing all published events, a new microservice can use that information and add new functionalities to the system without impacting the remaining platform. Even if the new component interacts with others, given

that the environment is as decoupled as possible, minimal changes are needed. In addition, modularity also facilitates removing any microservice, assuring the modifications to the system are as minimal as possible. On the other hand, in monolithic architectures, providing updates to the system can be a complex process. As the code basis increases, it becomes complex to maintain, deploy and scale the systems that adopt monolithic architectures (Kalske *et al.*, 2018). The reason for such complexity is that the whole system must be tested and redeployed again, making it impractical to make small changes to the system in short periods (Kalske *et al.*, 2018).

Event-driven architectures prioritize asynchronous communication by default. In contrast to a sequential approach, asynchronous communications enable parallel processing, resulting in enhanced performance and efficiency of operations in most scenarios. Because of its high scalability, NetOr can increase the performance of complex service management operations compared to other orchestration platforms. However, service-oriented architectures also have their limitations. One of which is the necessity to exchange significantly more messages over the network compared to monolithic systems. This is due to the continuous information flow between microservices. As a result, NetOr may not outperform monolithic Vertical Service Orchestrators if dealing with minor system loads. Contrastingly, when dealing with larger system loads, NetOr is expected to outperform the Vertical Service orchestrators mentioned in Section 2, given the high scalability provided by its microservice architecture.

NetOr is composed of 8 main components:

- **VSB/VSD/Network Slice Template (NST) Catalogue:** Offers a centralized persistence layer responsible for Create, Read, Update and Delete (CRUD) operations on descriptors and templates related to Vertical Services. These descriptors and templates are the following: VSBs, VSDs, NSTs, NSDs, and Virtual Network Function Descriptors (VNFDs), highlighting the VSBs. A VSB is a template tailored for a specific vertical service defined by an operator. Thus, it defines the topology, QoS parameters, policies, and lifecycle details. Based on such blueprints, verticals can develop VSDs, which are specifications of the determined service structure and the QoS requirements their scenario requires. Moreover, this service's operations are supplied via a REST API.
- **Group/Tenant Manager:** Handles both the groups and the tenants of the system, providing an Identity Provider (IdP) to authenticate tenant and group clients. Additionally, the system users, i.e., tenant and group clients, are handled through CRUD operations offered via a REST API.
- **Domain Manager:** Manages all aspects related to domains. Besides the CRUD operations that are offered, it is also responsible for communicating with the orchestration entities accountable for handling each domain, such as NFVOs and SDN controllers. Thus, the Domain Manager deals directly with the NFV artifacts orchestration. Furthermore, this module is agnostic to technology, allowing the integration with different solutions of the orchestration services.
- **Vertical Service Instance (VSI)/NSI Coordinator:** Responsible for triggering the VSI orchestration processes. This component is in charge of keeping track of each VSI's record. Thus, the coordinator has full context and observability about every aspect of all vertical services, including their status. Similarly to the previously described services, it exposes a Representational State Transfer (REST) Application Programmable Interface (API) to interact and trigger the operations provided.
- **VSI/NSI LifeCycle Manager (LCM) Manager:** Responsible for following up on the Vertical Services after their instantiation. It manages and operates the VSIs and their sub-components, such as NSIs and NSs. To enable these mechanisms, the LCM Manager creates a new agent for each vertical service that manages all operations related to it. Thus, such an agent handles the VSI's lifecycle.
- **Placement Arbitrator:** Processes all blueprints and descriptors related to Vertical Services, defining each Vertical Service sub-components deployment location and possible restrictions. Furthermore, it considers dynamically defined parameters during instantiation and SLAs related to the tenant.
- **Metrics Repository:** Supports Vertical Service use cases that require the Verticals to persist service metric. Verticals can use these metrics to arbitrate over specific service operations. As it is an optional service according to each use case, this component is not an intricate part of NetOr's orchestration lifecycle.
- **Domain Name System (DNS) Server:** Allows for Service Discovery mechanisms to be employed by the Verticals that use NetOr. Thus, it allows Verticals to implement Service Discovery in their services. Furthermore, Vertical Service developers may lean on Internet Engineering Task Force (IETF)'s RFC 6763 (Cheshire and Krochmal, 2013) to standardize their solution's DNS-Based Service Discovery (SD).

Additionally, when designing NetOr, we also aimed to provide a straightforward interaction between the verticals and our system. Therefore, a web portal was also developed to facilitate interaction between the NetOr system and vertical users. Through a basic graphical interface, the portal displays all available activities in the most straightforward manner feasible while greatly abstracting the underlying intricacies of orchestrating vertical services.

Figure 1 presents NetOr's architecture.

We now move on to present an overview of an internal workflow of NetOr, describing all the messages exchanged between its components.

The Instantiation of Vertical Services is achieved through a complex workflow between NetOr's components. Once the Coordinator receives a request to instantiate a new VSI, a new zone in the DNS Server is created. The created zone only concerns the newly created VSI for Service Discovery purposes. Consequently, the Coordinator will notify the remaining services related to the VSI Creation via the message bus, preparing them to start performing their duties. For in-

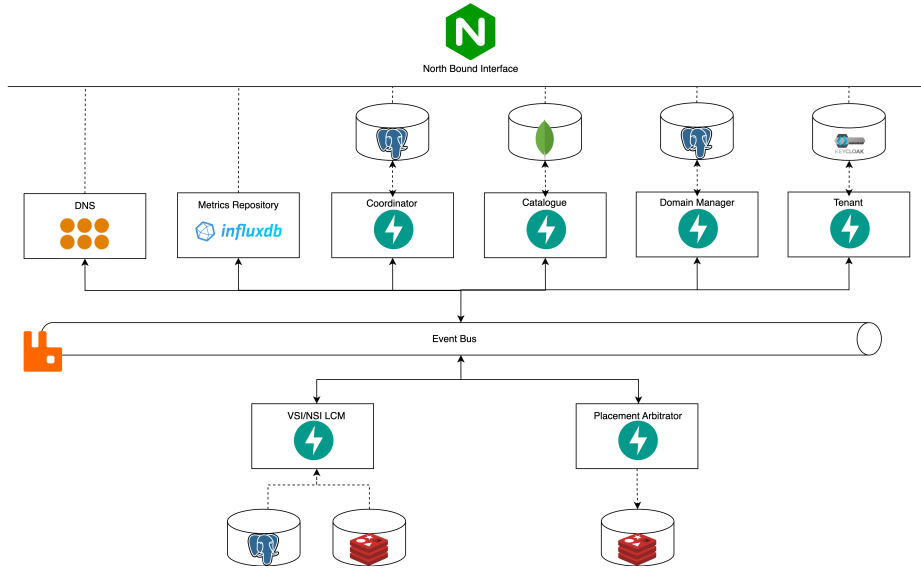


Figure 1. NetOr's Architecture

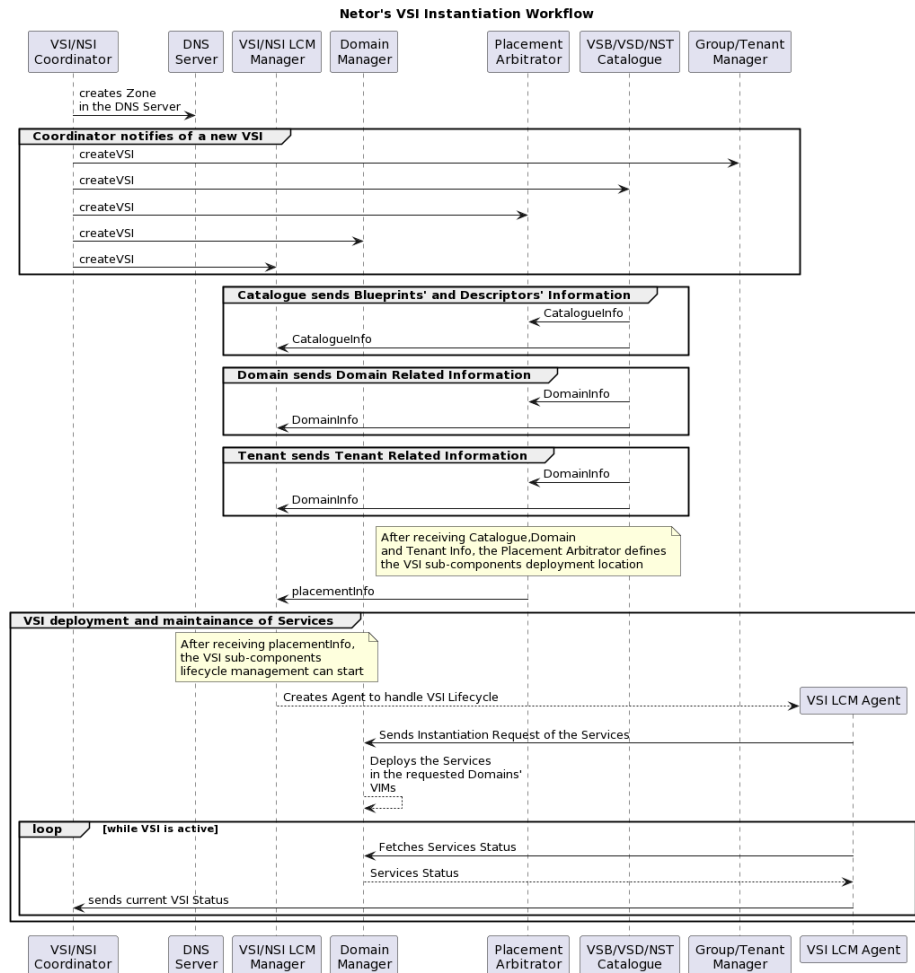


Figure 2. NetOr's VSI Instantiation Workflow

stance, the Catalogue will collect the VSI Blueprints and Descriptors information and send it to the LCM Manager and Tenant. The Domain Manager and Tenant will also gather the information related to the Domains required and Tenants involved and send it to the same components. Upon receiving

the information from (i) the Domain Manager, (ii) the Tenant, and (iii) the Catalogue, the Placement Arbitrator can define each Vertical Service sub-component deployment location. The result of such a task is then sent to the LCM Manager to start the VSI sub-components Instantiation. To do so, the

LCM Manager creates an agent, as mentioned earlier, which interacts with the Domain Manager. Subsequently, the Domain contacts each Domain's NFVO to deploy the required services at the Domain's VIM. This workflow is illustrated in Figure 2.

3.2 APIs and Data Models

There are two independent NetOr interfaces that must be standardized. These are NetOr's Northbound Interface (NBI) and Southbound Interface (SBI). The NBI had to comply with widely acknowledged NFV standards to facilitate the system's adoption by other third-party platforms. This enables NetOr to replace analogous platforms seamlessly and guarantees that the necessary parameters for the underlying systems are always available to them. NetOr accomplishes this by leveraging widely known data models. On the other hand, NetOr's SBI is the interface that interacts with underlying orchestrators, such as the NFVOs. Thus, it must also abide by well-accepted and widely employed standards. This way, NetOr may support a plethora of different NFVOs. Furthermore, adopting such standards also allows the necessary parameters for the resources of these NFVOs to be correctly detailed in NetOr's data models.

Regarding the NBI, after evaluating various significant standards, we selected the most adequate ones for the data models exchanged in CRUD operations regarding the VSIs and their additional resources. These standards consist of TS 28.541(ETSI and 3GPP, 2021), SOL005(ETSI, 2021a), and SOL006(ETSI, 2021b) and are all related to the management of the information models. Moreover, the data structures employed by NetOr were based on the models embraced by the most mature SoA Vertical Service orchestrator, the 5GR-VS. However, 5GR-VS's VSBs had to be extended in order to accommodate the actions and parameters that specific VSIs may require. Additionally, the VSI instantiation data model was also enhanced to accommodate dynamic instantiation parameters and domain deployment selection.

Regarding the SBI, a proper analysis of the most relevant standards of the area was also conducted. We concluded that the most appropriate standards for the data models and operations were TS 28.530(ETSI and 3GPP, 2020), TS 28.531(ETSI and 3GPP, 2022), TS 28.541(ETSI and 3GPP, 2021), TR 28.801(3GPP, 2018), SOL005(ETSI, 2021a), and SOL006(ETSI, 2021b). Even though these standards are fairly comprehensive and complete, their expansion was required to support all NetOr operations. Specifically, it was required to expose management functions to enable the execution of run-time actions over already instantiated network services and network slices. Moreover, the models and operations adopted by 5GR-VS's SBI were also taken into account when designing NetOr's SBI.

4 Inter-domain Automatic Mechanism

Regarding mobile service and network coverage, various operators administer different geographic areas, restricting the locations in which each operator can directly offer its services. Currently, the process for instantiating Network Services, Network Slices, and Vertical Services assumes that a single domain would supply them. When the users are in a domain controlled by a different operator, it will route end-users' data and requests to the domain where the intended services are located. However, one may conceive several scenarios in which these redirection techniques may affect the final E2E services, such as medical or automotive scenarios that need extremely low latency. In such circumstances, the best approach is to instantiate the required services across all domains with which the end user may interact. Consequently, inter-domain mechanisms were developed to address this issue. These allow the on-demand connection of separate geographical areas. Thus, the novelty of such solutions. As previously stated, nowadays, service providers abide by norms and limitations that require their clients to be serviced exclusively by them, i.e., all network and service-related functions are hosted and deployed within their domain. However, given the potentialities of inter-domain solutions, these can disrupt the current *modus operandi* of the service providers.

Several standards and scientific works have begun to address and suggest the best practical and architectural approaches for addressing the complexity of inter-domain scenarios. The TR 28.801 (3GPP, 2018) standard is an example of one of these. It proposes three distinct methods for managing the coordination of multiple operator domains. The first strategy proposes that the client that wishes to orchestrate an inter-domain solution controls the CSMF and communicates with the multiple NSMF-providing operators. A second possibility involves the consumer interacting with a primary operator hosting the CSMF, which will engage with the client's and other operators' NSMFs. Finally, the third approach proposes that the client communicate with the primary operator hosting the CSMF and delegate the operations to its own NSMF, which will then leverage the different operators' NSSMFs.

The work presented in (Bernini *et al.*, 2020) proposes a 5G vertical service orchestration system centered on network slicing in an inter-domain scenario. This work provides some contextualization on inter-domain scenarios, introducing Verticals/Digital Service Consumers (DSCs) that leverage the services supplied by Digital Service Providers (DSPs). However, these DSPs are dependant on the resources and functionalities provided by Network Service Providers (NSPs). In this scenario, the DSPs are responsible for managing the services' lifecycles and exposing them to Verticals. Ultimately, (Bernini *et al.*, 2020) describes and proposes an inter-domain orchestration framework. This framework comprises three different entities for the orchestration of inter-domain domain solutions in network slicing scenarios: (i) the Resource Orchestrator, the Slice Orchestrator, and the Service Orchestrator.

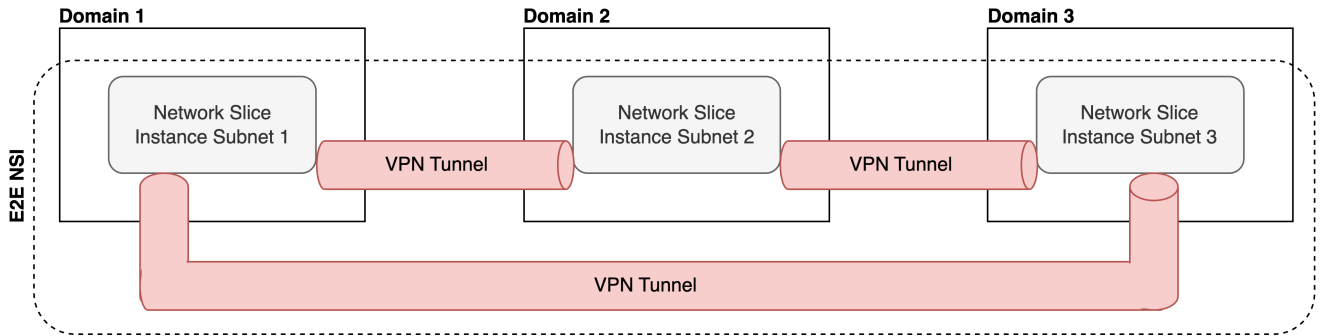


Figure 3. E2E NSI with Subnets comprised of Network Slices

The selected architecture for NetOr is based on the one suggested in the first multi-operator coordination solution given in TR 28.801 (3GPP, 2018) and the one proposed in (Bernini et al., 2020). Both approaches suggest a centralized agent for service orchestration (CSMF/Service Orchestrators) that interacts with other lower-level orchestrators (NSMFs/Slice Orchestrators).

Regarding our inter-domain mechanism, the primary goal of our PoC is the instantiation of an E2E Virtual Private Network (VPN) Service across various domains without previous negotiation. Technically, this can be accomplished by employing an E2E Network Slice, ideally with subnets comprised of Network Slices, as shown in Figure 3. Considering the VPN Service, our proposal is to design and set up a VPN tunnel, thereby establishing a secure communication channel between the domains. The technology that we relied upon to instantiate the proposed VPN tunnel was Wireguard⁴, a simple and minimalist framework that enables the swift instantiation and deployment of VPNs.

Hypothetically, inter-domain connectivity would require the existence of a centralized service agent that is independent of all domains and enables the connecting of VPN tunnels' peers. This agent would gather and process all VPN Nodes' related information and share it with all remaining Nodes of the VPN. Thus, providing the required information for establishing the VPN tunnels between the different VPN Nodes. However, NetOr also makes available a DNS Server that may be used for DNS-Based SD, according to IETF's RFC 6763 (Cheshire and Krochmal, 2013). Thus, two approaches were employed to design and implement our inter-domain service.

The first approach, *Approach A*, relies on NetOr's *VSI/NSI LCM* component to configure the VPN tunnels. Thus, it relies on a centralized management and configuration agent. Contrastingly, our second approach, *Approach B*, delegates this responsibility to the VPN Nodes themselves.

Both approaches result in an automated orchestration of a full-mesh VPN between different domains, without prior negotiation. Besides this, our approaches are also considered zero-touch since no human intervention is needed to achieve the desired scenario. Furthermore, it is possible to fully orchestrate it through NetOr since it provides all required functionalities and mechanisms. To make available Wireguard Servers in each domain we deploy a Wireguard

VNF in the NSs distributed across the different domains. The initial configuration of such VNF is achieved by employing the VNFMs provided by the NFVOs residing in the domains.

Approach A assumes that NetOr's *VSI/NSI LCM* component will constantly monitor the status of the VPN Nodes. When Wireguard is installed on the Nodes, NetOr's *VSI/NSI LCM* will become aware of this and trigger the establishment of new VPN tunnels. This assumes that the VSBs and VSDs contain information that NetOr's *VSI/NSI LCM* component may use to get acquainted with all management operations that will be required. Such information could stipulate that whenever a new VPN Node has finished installing Wireguard, the *VSI/NSI LCM* component would have to invoke an operation to retrieve the Node's information and then use this information to trigger the establishment of new VPN tunnels with the remaining VPN Nodes. To trigger the tunnels' establishment process, NetOr's *VSI/NSI LCM* invokes day-1 operations on the VPN Node VNFs.

On the other hand, *Approach B* assumes that the VPN Nodes are in charge of configuring the VPN tunnels themselves. However, to achieve this, the VPN Nodes will have to interchange information between them. This information will then be used to configure the VPN tunnel endpoints.

In *Approach B*, NetOr's DNS Server may be used for the VPN Nodes to share information among themselves. Upon a request to instantiate our solution, NetOr's Coordinator component generates a specific DNS zone that shall be used by the Vertical Service to employ SD practices. The *Coordinator* will also inject all parameters required to provide such functionality. One of these parameters is a cipher key that the Vertical Service members shall use to publish and share protected information. In our scenario, such information can be the public keys of the Wireguard Servers or even the networks that should be forwarded through the VPN tunnels. The cipher key is the same for all Vertical Service members, which enables them to share protected information by symmetrically encrypting it. After this step, the *Coordinator* will request the *Domain* component to submit the extended instantiation request to the NFVOs residing in the desired domains.

When the VPN Node VNFs are fully instantiated and have configured Wireguard, the VPN Nodes can then share the information required for the establishment of the VPN tunnels. Thus, they will exchange between one another their (i) public key, (ii) their location, and (iii) information on the networks that should be accessible to the other VPN Nodes with all nodes that shall be part of the mesh VPN. In this phase, the

⁴<https://www.wireguard.com>

Table 2. DNS Records Created by a VPN Node

DNS Records		
Name	Type	Value
<code>_wg._udp</code>	PTR	<code>_wg < id > ._wg._udp. < zone_name > .netor.</code>
<code>_wg < id > ._wg._udp</code>	SRV	<code>Priority = 0Weight = 0Port = < wg_port ></code> <code>Target = wg < id > . < zone_name > .netor.</code>
<code>wg < id ></code>	A	<code>< vpn_node_public_ip ></code>
<code>_wg < id > ._wg._udp</code>	TXT	<code>public_key = < encrypted_key ></code> <code>allowed_networks = < encrypted_allowed_network ></code> <code>local_port = < encrypted_peer_local_port ></code> <code>local_ip = < encrypted_peer_local_ip ></code>

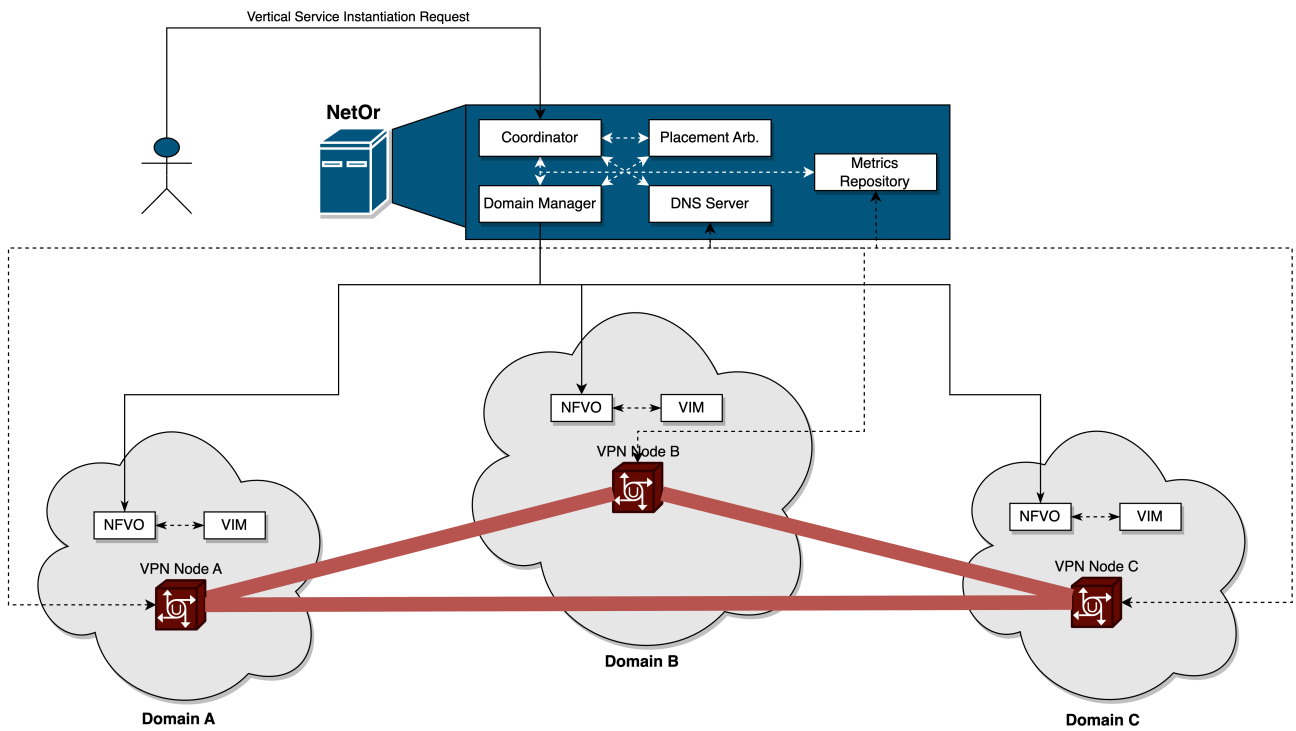


Figure 4. Architecture of the presented Full Mesh VPN Solution (Approach B)

cipher key injected by NetOr is necessary. The information shared by a VPN Node through NetOr’s DNS Server obeys IETF’s RFC 6763(Cheshire and Krochmal, 2013) standard and is presented in Table 2.

After a VPN node has published its information to the DNS Server, it will continue to probe this entity to find additional eligible VPN Nodes. As soon as a new VPN Node is discovered, the process of establishing a VPN tunnel begins. Wireguard’s configuration file is updated with the new VPN Node’s information, and a new VPN tunnel is created. The scenario in which a full-mesh VPN is established is depicted in Figure 4. The presented scenario comprises a VPN mesh composed of 3 VPN nodes, but more nodes could be added to this mesh VPN. It is worth mentioning that throughout the VPN Nodes’ lifecycle, they can leverage NetOr’s *Metrics Repository* to publish their service-level metrics. These can then be used by the Vertical Service clients to monitor their service and enforce service-level SLAs. The usage of the

Metrics Repository applies to both our design approaches.

5 Results

As previously stated, the primary objective of NetOr is to address some of the identified problems of existing Vertical Service orchestrators. Section 3 has already addressed our solution’s design and the standards it complies with. From that section, one is already aware that (i) NetOr was developed adopting a service-oriented design, (ii) NetOr adheres to all relevant Network Slicing standards, (iii) NetOr totally supports inter-domain scenarios, and that (iv) NetOr may conduct runtime operations over the Vertical Services.

However, evaluation of our solution’s performance compared to other SotA Vertical Service orchestrators is still lacking. If NetOr’s performance is significantly inferior to the one of the other Vertical Service orchestrators, there will be

Table 3. 5GR-VS and NetOr System's added delays

Measurement	System	Max	Min	Avg	StDev
Added Instantiation Delay	5GR-VS	234 ms	81 ms	113,86 ms	29,20 ms
	NetOr	345 ms	135 ms	190,93 ms	24,82 ms
Added Termination Delay	5GR-VS	108 ms	19 ms	45,90 ms	17,20 ms
	NetOr	187 ms	59 ms	101,24 ms	20,90 ms

resistance to adopting our solution, even though we have addressed numerous recognized problems. In this section, we compare the performance of NetOr with the 5Growth Vertical Slicer, one of the current most mature Vertical Service orchestrators. This comparison was made through the orchestration of the solution we presented as *Approach A*, where NetOr's *VSI/NSI LCM* component manages the VPN Nodes configuration. This approach is more impactful to the performance of the Vertical Service orchestrator, thus being employed to compare the performance of NetOr with the 5Growth Vertical Slicer's one. Besides this, *Approach A* is fully aligned with the features offered by the 5Growth Vertical Slicer. We also evaluated which approach has better performance, having NetOr's *VSI/NSI LCM* managing the VPN Nodes or relinquishing their management to themselves through the usage of the provided DNS Server. Finally, we conducted additional performance tests on NetOr to discover how it deals with several simultaneous Vertical Service instantiation requests and with the complexity of orchestrating an E2E Vertical Service across a high number of independent domains.

5.1 Testing Environment

Since the inter-domain scenario requires a minimum of two domains, two separate OSMs were deployed, one for each domain. Each OSM was deployed in a Virtual Machine (VM) with 12 GB of RAM, 4 VCPUs, and 150 GB of storage. The first OSM has as a VIM an OpenStack cluster with the limitations of 30 VM instances, 60 VCPUs, 70 Gb of RAM, 1 TB of storage, and 10 networks. On the other hand, the second OSM was integrated with a DevStack with the limitations of 10 VM instances, 20 VCPUs, 50 Gb of RAM, 1 TB of storage, and 100 networks. Moreover, during our tests, NetOr was deployed in an Ubuntu 18.04 VM with 8 GB of RAM, 4 VCPUs, and 32 GB of storage.

5.2 Comparison Between NetOr's and 5GR-VS's Performance Considering the Solution Developed Using *Approach A*

Regarding the performance tests, using the solution presented as *Approach A* (where the VPN tunnels' configuration is achieved via NetOr's *VSI/NSI LCM*), we evaluated the delays added by each orchestrator during the processing of an initial instantiation request and its forwarding to the underlying NFVOs. Moreover, we also compared both orchestration systems regarding the time needed to fully orchestrate and configure all the VPN Node VNFs. Each test was performed 30 times. Regarding the full orchestration and configuration of the VPN Nodes, we followed the same method-

ology to evaluate NetOr, whereas the results from (Fonseca *et al.*, 2021) were employed to evaluate the 5GR-VS's performance, given that (Fonseca *et al.*, 2021) presented the same tests we did in the same environment but on the 5GR-VS.

The first comparison test focused on the delay added by each orchestrator to the vertical service instantiation process. This measurement was achieved by computing the time delta between the orchestrator's receiving an instantiation request and its forwarding to the underlying NFVOs. On the other hand, the second test evaluated the delay added by each orchestrator to the vertical service termination process. Table 3 presents the obtained results.

Analyzing Table 3, we conclude that NetOr engenders higher instantiation and termination delays than the 5GR-VS. However, the difference between NetOr's and 5GR-VS's delays is minimal, differing in less than 80 milliseconds regarding the added instantiation delay and less than 40 when considering the added termination delay. These results could be anticipated when considering the architectures of both Vertical Slicers. NetOr was designed as a microservice architecture, where NetOr's components exchange asynchronous messages through a Broker. Even though this architecture provides better scalability, it also results in additional delays in the message exchange process. In a low-usage scenario, this is further highlighted. Contrastingly, 5GR-VS was built according to a monolithic architecture. This architecture involves synchronous messages between its components, and no communication Brokers are needed. Thus, systems built according to a monolithic architecture perform operations very fast in low-usage scenarios. However, when under heavy loads, the performance of monolithic applications heavily degrades. Since the tests previously described were performed in a low-usage scenario, and the operations required for processing a Vertical Service instantiation/termination request and forwarding it to the NFVOs are fairly simple, the architectures of each Vertical Slicer justify the obtained results.

However, we must still evaluate the performance of both Vertical Slicers under more real-world alike conditions. Thus, we conducted end-to-end tests to obtain and analyze the time required for NetOr to fully instantiate and configure our inter-domain service. Additionally, we also computed the time needed for NetOr to terminate the Vertical Service. The results are presented in Table 4. Then, we compared our results to the ones of the 5GR-VS, gathered from (Fonseca *et al.*, 2021).

Table 4 showcases that NetOr requires around 5,8 minutes to fully instantiate our inter-domain solution when developed according to *Approach A*. The time needed for NetOr to terminate the inter-domain service is, on average, around 1,2 minutes. Contrastingly, 5GR-VS takes around 9 minutes to

Table 4. NetOr’s E2E inter-domain service instantiation and termination time

Measurement	Max	Min	Avg	StDev
E2E Instantiation Time	409346 ms	279582 ms	348858,9 ms	29334,1 ms
E2E Termination Time	109319 ms	38687 ms	74971,9 ms	16588,7 ms

Table 5. NetOr’s E2E inter-domain service instantiation time for both design approaches

Approach	Max	Min	Avg	StDev
<i>Approach A</i>	409346 ms	279582 ms	348858,9 ms	29334,1 ms
<i>Approach B</i>	380147 ms	248024 ms	310395,1 ms	31432,3 ms

instantiate the same solution and 46 seconds to terminate it (Fonseca *et al.*, 2021). Since most Operators are more concerned with the instantiation time of their solutions rather than the termination one, we can affirm that our solution, performance-wise, is a better fit for these organizations. Furthermore, NetOr also solves several known issues of most Vertical Service orchestrators, further accentuating its advantages over them. Once again, the performance differences are justified by the different architectures employed by each orchestrator. By following a microservice-oriented design, NetOr can manage several service components and services concurrently. This parallelization enables this orchestrator to save valuable minutes by simultaneously processing the separate service subnets involved in the Vertical Service.

5.3 Comparison Between Both Approaches Followed to Design our Inter-Domain Service

We followed to evaluate the performance of both inter-domain solutions we developed. As stated before, the first solution follows what we have named *Approach A*, where the VPN tunnels’ configuration is achieved via NetOr’s *VSI/NSI LCM* component. On the other hand, our second solution (*Approach B*) delegates the VPN tunnels’ configuration to the VPN Nodes themselves. NetOr only provides a DNS Server for the VPN Nodes to publish their information, making it available to the remaining Nodes. These will consume that information to configure and establish VPN tunnels.

To evaluate both solutions, we only focused on their instantiation time since their termination time will be the same, and this measure was already obtained and presented in Table 4. The testing scenario where these tests were performed is the one described in Section 5.1, and we followed the same testing methodologies as we did for comparing the performance of NetOr with the one of 5GR-VS. The obtained results are presented in Table 5.

As shown in Table 5, the solution where the configuration of the VPN tunnels is delegated to the VPN Nodes was instantiated and configured faster. This can be justified by the fact that, in this approach, the configuration of the tunnels is not managed by a single entity, NetOr, but is distributed among the VPN Nodes. When NetOr’s *VSI/NSI LCM* component is responsible for configuring the VPN tunnels, it will have to collect the VPN Nodes information, to validate that they have already installed Wireguard and are ready to be part of a VPN tunnel. Then, NetOr will process this infor-

mation, evaluate to which VPN Nodes it must be sent, obtain their location, and forward it to them. Once this information reaches the remaining VPN Nodes, they may start establishing the tunnels. On the other hand, the solution relying on the DNS Server simplifies all the processes that take place before a VPN Node receives information regarding other Nodes, to which it will establish VPN tunnels. All VPN Nodes will publish their information to the DNS Server, sharing it with the remaining ones. Furthermore, the information published by the VPN Nodes already encompasses their location, which, in *Approach A*, had to be obtained by NetOr’s *VSI/NSI LCM*. Thus, the approach involving the DNS Server removes many operations that NetOr’s *VSI/NSI LCM* component executed, diminishing the end-to-end instantiation and configuration time of the inter-domain service.

5.4 Additional Performance Tests on NetOr

We further tested the performance of NetOr, to evaluate how it deals with several simultaneous Vertical Service instantiation requests and with an increasing number of domains where a Vertical Service spans across. To this end, we triggered until 10 simultaneous Vertical Service instantiation requests and instantiated our inter-domain solution (*Approach B*) across 3 to 10 mock domains. These domains ultimately are only mapped to the 2 domains presented in Section 5.1, but in NetOr they were configured as being fully independent domains.

The first test, where 10 simultaneous Vertical Service instantiation were requested, allows for evaluating the performance of NetOr when dealing with an high number of Vertical Service requests, while the second test, where we orchestrated our inter-domain solution (*Approach B*) across 3 to 10 mock domains, allows for evaluating how NetOr behaves when it manages complex services distributed across and high number of domains.

Each test was executed 10 times. Since the major impact of these stress tests will be on the initial processing of the instantiation requests, we evaluated the instantiation delay added by NetOr. The obtained results are showcased in Figures 5, and 6. Figure 5 showcases the results obtained when several equal Vertical Services were instantiated simultaneously, while Figure 6 showcases the results of the instantiation of our inter-domain solution across 3 to 10 domains.

Looking at Figure 5, we can affirm that NetOr can provide a good throughput when dealing with several simultaneous Vertical Services being instantiated simultaneously.

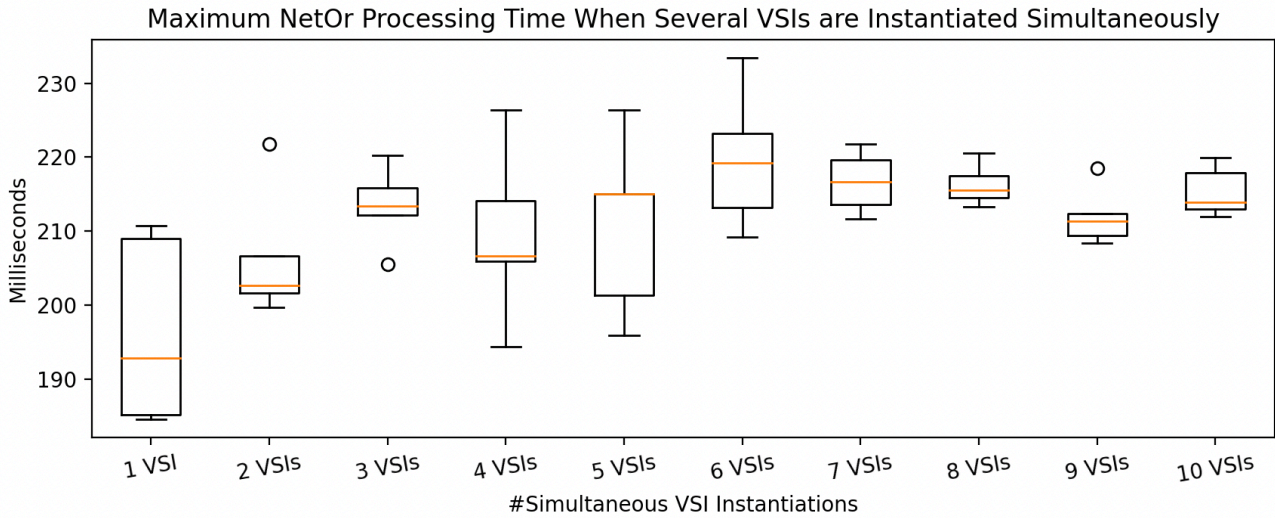


Figure 5. Performance test where several Vertical Services were instantiated simultaneously

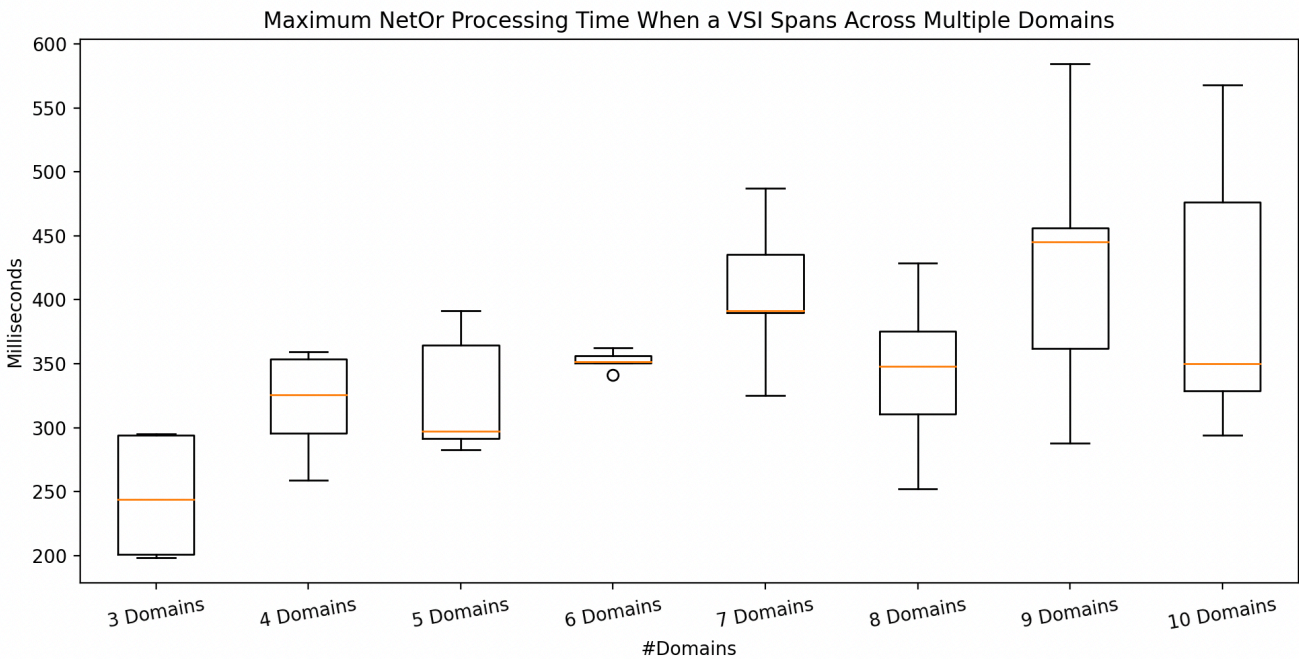


Figure 6. Performance test where the same Vertical Service was instantiated across several domains

Furthermore, we can say that NetOr’s performance will remain almost the same when instantiating 3 to 10 Vertical Services simultaneously. Since we always orchestrated the same VSI across the same domains, the information on these domains was cached by NetOr’s Placement Arbitrator, which enables it to be very rapidly accessed, thus diminishing NetOr’s added instantiation delay.

Contrastingly, Figure 6 points out that NetOr’s performance may decrease when deploying a Vertical Service across an increasing number of domains. This is because whenever a Vertical Service spans across a new domain, NetOr will have to obtain information on the location of the new domain and how to communicate with its NFVO. Thus, the increase in the NetOr’s added instantiation delay. However, this increase is almost irrelevant. For instance, orchestrating our inter-domain solution in 3 domains resulted in an aver-

age added instantiation delay of 246,37 milliseconds, while when considering 10 domains, the average added instantiation delay was of 403,49 milliseconds. Both measurements are in the same order of magnitude and differ in less than 200 milliseconds. Thus, this delay will be almost imperceptible when considering our solution’s end-to-end instantiation and configuration time.

6 Conclusion

As shown by the results presented in Section 5.2, NetOr’s performance is identical to the one of the 5GR-VS (currently one of the most mature Vertical Service orchestrators). Using its parallelization capabilities, NetOr substantially reduces the time required to instantiate a Vertical Service. The results

showcased in Section 5.4 also point out that NetOr is able to maintain its good performance when dealing with various simultaneous Vertical Service instantiation requests and when the number of domains where a Vertical Service spans increases.

In addition, NetOr already offers alternatives and procedures that the vast majority of Vertical Service orchestrators lack, which was the fundamental impetus for the work presented in this article. We must emphasize that, while having a performance comparable to the one of the 5GR-VS, NetOr complies with all applicable Network Slicing standards, (ii) it was developed in accordance with a service-oriented architecture, (iii) it completely supports multi-domain scenarios, and (iv) it can execute runtime operations over the Vertical Services. In addition, like the 5G-Transformer and 5Growth VSSs, our orchestrator also offers a layer of abstraction through VSBs and VSDs.

Furthermore, the results shown in Section 5.3 also point out that our inter-domain solution's configuration can be optimized by not having a central entity configuring all VPN tunnels, in our case NetOr's *VSI/NSI LCM* component. Instead, better performance can be achieved when this configuration is distributed among the VPN Nodes.

Regarding future developments, we want to improve NetOr by including SLA management, hence accelerating the acceptance of our solution in Vertical Services Orchestration use cases.

Funding

This work was supported by the European Horizon 2020 Programme for research, technological development and demonstration under Grant 101016448 (5GASP).

Authors' Contributions

D. Gomes contributed to the problem formalization and steered the research direction to reach the results presented in this article. Moreover, he revised the final manuscript. D. Corujo participated in the literature review of the 5G-Growth Vertical Slicer and evaluated this orchestrator. Furthermore, he contributed to the revision of the final manuscript and provided meaningful insight into the writing of the final manuscript. J. Alegria contributed to the literature review and was responsible for the initial implementation of NetOr, evaluating and implementing the different standards applied to NetOr's SBI and NBI. He was also responsible for the initial implementation of the Inter-Domain Service presented as Approach A. Finally, he participated in the experimentation of the Inter-Domain Service presented as Approach A and in its results' analysis. Da. Gomes contributed to the related work presented in this article. Besides this, he adapted NetOr to provide all functionalities needed to orchestrate the Inter-Domain Service presented in Approach B. Moreover, he also participated in the experimentation of both our Inter-Domain Service approaches. He also contributed to the writing of the final manuscript. R. Direito contributed to the related work presented in this article. He developed and implemented the Inter-Domain Service presented in Approach B and also improved the implementation presented in Approach A. He was the main writer of the final manuscript and was also responsible for analyzing the obtained

results. Alongside R. Direito, Da. Gomes also contributed to the analysis of the results.

Competing interests

The authors declare that they have no competing interests.

References

- 3GPP (2018). TR 28.801 - V15.1.0 - Study on management and orchestration of network slicing for next generation network. Technical report. Available at: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3091>.
- 5G-PPP (2019). 5Growth. Available online. Available at: <https://5g-ppp.eu/5growth/>. Accessed on December 29, 2022.
- Bernini, G., Giardina, P. G., et al. (2020). Multi-domain orchestration of 5G vertical services and network slices. In *IEEE ICC Workshops 2020 - Proceedings*. IEEE Inc.. DOI: 10.1109/ICCSWorkshops49005.2020.9145221.
- Chang, W.-C. and Lin, F. J. (2021). Coordinated management of 5g core slices by MANO and OSS/BSS. *Journal of Computer and Communications*, 09(06):52–72. DOI: 10.4236/jcc.2021.96004.
- Cheshire, S. and Krochmal, M. (2013). DNS-Based Service Discovery. (6763). DOI: 10.17487/RFC6763.
- Choi, J. S., Chun, S. J., and Lee, S. (2022). Hierarchical distributed overarching architecture of decoupled federation and orchestration frameworks for multidomain nfv manos. *IEEE Communications Magazine*, 60(9):68–74. DOI: 10.1109/MCOM.005.20949.
- de la Oliva, A., Li, X., Costa-Perez, X., Bernardos, C. J., Bertin, P., Iovanna, P., Deiss, T., Mangues, J., Mourad, A., Casetti, C., Gonzalez, J. E., and Azcorra, A. (2018). 5g-transformer: Slicing and orchestrating transport networks for industry verticals. *IEEE Communications Magazine*, 56(8):78–84. DOI: 10.1109/MCOM.2018.1700990.
- Erunkulu, O. O., Zungeru, A. M., Lebekwe, C. K., Mosalao, M., and Chuma, J. M. (2021). 5g mobile communication applications: A survey and comparison of use cases. *IEEE Access*, 9:97251–97295. DOI: 10.1109/ACCESS.2021.3093213.
- ETSI (2021a). GS NFV-SOL 005 - V3.5.1 - Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point. Etsi group specification. Available at: https://docbox.etsi.org/isg/nfv/open/Publications_pdf/Specs-Reports/NFV-SOL%20005v3.5.1%20-%20GS%20-%20Os-Ma-nfvo%20APIs%20spec.pdf.
- ETSI (2021b). GS NFV-SOL 006 - V3.5.1 -Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on YANG Specification. Etsi group specification. Available at:

- SOL%20005v3.5.1%20-%20GS%20-%20s-Ma-nfvo%20APIs%20spec.pdf.
- ETSI and 3GPP (2020). ETSI TS 128 530 - V16.2.0 - 5G; Management and orchestration; Concepts, use cases and requirements (3GPP TS 28.530 version 16.2.0 Release 16). Etsi technical specification. Available at:https://www.etsi.org/deliver/etsi_ts/128500_128599/128530/16.02.00_60/ts_128530v160200p.pdf.
- ETSI and 3GPP (2021). ETSI TS 128 541 - V16.8.0 - 5G; Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3 (3GPP TS 28.541 version 16.8.0 Release 16). Etsi technical specification. Available at:https://www.etsi.org/deliver/etsi_ts/128500_128599/128541/16.08.00_60/ts_128541v160800p.pdf.
- ETSI and 3GPP (2022). ETSI TS 128 531 - V17.3.0 - 5G; Management and orchestration; Provisioning (3GPP TS 28.531 version 17.3.0 Release 17). Etsi technical specification. Available at:https://www.etsi.org/deliver/etsi_ts/128500_128599/128531/17.03.00_60/ts_128531v170300p.pdf.
- Fonseca, J., Alegria, J., et al. (2021). Dynamic Inter-domain Network Slicing for Verticals in the 5Growth Project. *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks, SDN-NFV 2021*. DOI: 10.1109/NFV-SDN53031.2021.9665037.
- Kalske, M., Mäkitalo, N., and Mikkonen, T. (2018). Challenges when moving from monolith to microservice architecture. In Garrigós, I. and Wimmer, M., editors, *Current Trends in Web Engineering*, pages 32–47, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-74433-9₃.
- Karabey Aksakalli, I., Çelik, T., Can, A. B., and Tekinerdoğan, B. (2021). Deployment and communication patterns in microservice architectures: A systematic literature review. *Journal of Systems and Software*, 180:111014. DOI: 10.1016/j.jss.2021.111014.
- Mangues-Bafalluy, J., Baranda, J., Pascual, I., Martínez, R., Vettori, L., Landi, G., Zurita, A., Salama, D., Antevski, K., Martín-Pérez, J., Andrushko, D., Tomakh, K., Martini, B., Li, X., and Salvat, J. X. (2019). 5g-transformer service orchestrator: design, implementation, and evaluation. In *2019 European Conference on Networks and Communications (EuCNC)*, pages 31–36. DOI: 10.1109/EuCNC.2019.8802038.
- ONAP (2022). ONAP - Project's Documentation. Available online. Available at:<https://docs.onap.org/>. Accessed on October 13, 2022.
- Rodriguez, V. Q., Guillemin, F., and Boubendir, A. (2020). 5g e2e network slicing management with onap. In *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 87–94. DOI: 10.1109/ICIN48450.2020.9059507.
- Shafabakhsh, B., Lagerström, R., and Hacks, S. (2020). Evaluating the impact of inter process communication in microservice architectures. In :, volume 2767 of *CEUR Workshop Proceedings*, pages 55–63. CEUR-WS.org. QC 20210114. Available at:<http://ceur-ws.org/Vol-2767/07-QuASoQ-2020.pdf>.
- Shakir, A., Staegemann, D., Volk, M., Jamous, N., and Turowski, K. (2021). Towards a concept for building a big data architecture with microservices. *Business Information Systems*, 1:83–94. DOI: 10.52825/bis.v1i.67.
- Tranoris, C. (2021). Openslice: An opensource OSS for delivering network slice as a service. *CoRR*, abs/2102.03290. DOI: 10.48550/arXiv.2102.03290.