NetOr: An Inter-domain Vertical Service Orchestrator for 5G Networks

João Alegria, Rafael Direito, Daniel Gomes, Daniel Corujo, Diogo Gomes

Instituto de Telecomunicações and Universidade de Aveiro, Portugal
Email: {*joao.p, rdireito, dagomes, dcorujo, dgomes*}*@av.it.pt;*

***Abstract*—Current 5G vertical service orchestration solutions suffer from several limitations, mainly: (i) lack of standardization, (ii) monolithic architectures, and (iii) low support for inter-domain scenarios, for instance. This paper proposes the development of a new 5G vertical service orchestration platform that solves these problems and supports intricate use cases. This new system, named NetOr, will inherit many of the abstractions and functionalities already employed while tackling the mentioned issues while providing increased maintainability, flexibility, and scalability, since a Service-Oriented Architecture was followed.**

***Index Terms*—5G, Orchestration, NFV, Vertical, Slicing, Multi-Domain, Inter-domain, Micro-Services**

I. Introduction

Besides being the new generation of radio networks, Fifth Generation (5G) technologies also consider some core concepts such as Service-based Architectures, Software-Defined Networks (SDN), Network Functions Virtualization (NFV), and End-to-End (E2E) Network Slicing [1]. These technologies favor the virtualization of specialized network hardware, decreasing the infrastructure's cost while increasing the number of scenarios supported without changing the infrastructure. With the evolution in NFV and the efforts to support and integrate such technologies, scenarios and services with higher complexity appear.

Some of the more intricate and discussed are the inter-domain scenarios, where the E2E service spans across multiple administrative domains, increasing the service's coverage area. Solutions and systems supporting these scenarios already exist, but their support is limited and still in an embryonic phase. Thus, a new inter-domain approach is required to deploy a fully-fledged E2E inter-domain vertical service across different administrative domains with minimal prior knowledge of each other. This mechanism heavily relies on the network slicing capabilities of segmenting the entire network, ranging from Radio Access Network (RAN), Transport Networks (TNs), and Core Network (CN). In this scenario, the network slice is the entity that composes and manages different network services or other network slices, thus having a key role in establishing the inter-domain vertical services.

To provide distributed inter-domain vertical services, Vertical Service Orchestrators can be put to use, since they easily handle the instantiation and management of these services. Although, several known problems (discussed in Section II) may jeopardize the correct establishment and management of E2E inter-domain services.

With this paper, we aim to tackle the deficiencies of the most modern Vertical Service Orchestrators by presenting a Proof-of-Concept (PoC) for a new E2E Slicing Platform developed through a service oriented architecture, ultimately enabling increased maintainability, flexibility, and scalability. Regarding the paper's structure, it is organized as follows. Section II addresses and compares the State of the Art (SoA) in regards to modern Vertical Service Orchestrators. Section III introduces NetOr, a Network Orchestrator that aims to fix the problems encountered and described in Section II, hoping to advance the SoA regarding E2E Network Slicing. The mechanisms used to provide multi-domain E2E slices are described in Section IV, and their results are then presented in Section V. Lastly, Section VI presents some concluding remarks.

II. Vertical Service Orchestrators

This Section presents an analysis of several Vertical Service Orchestrators, detailing how they orchestrate E2E Vertical Services and presenting their advantages and drawbacks.

A. *Openslice*

Created as a 5GinFire [1] spinoff, Openslice takes advantage of the most relevant components for an open-source Operations Support System (OSS)/Bussiness Support System (BSS) platform. It also supports onboarding Virtual Network Functions (VNFs), Containerized Network Functions (CNFs) and Network Services (NSs), instantiating NSs, and has TMForum[2] Open APIs support regarding Service Catalog Management, Ordering, and more.

The Openslice system defined some abstractions over the VNFs, CNFs and NSs complexities, allowing verticals to request a given service in the most seamless way possible, focusing only on the problem's logic. The first was the Service concept, which consists of the abstraction of all the underlying complexities and enables providing a single, coherent, and correct service, ready to be instantiated and deployed. Other concepts are the Resource Facing Services (RFSs) that encapsulate all services directly connected to the infrastructure (such as the Network Service Descriptors (NSDs)) and Customer Facing Services (CFSs), which encapsulate all services and specifications that directly interact with the user. Both RFSs and CFSs are fundamental components of the Service concept.

[1] https://5ginfire.eu/
[2] https://www.tmforum.org/

To create a Service, a CFS Specification needs to be defined, possibly having Service Specification Relationships attached to interconnect the RFSs to CFSs [2]. Additionally, it has 2 specialized web portals.

As advantages, this platform has the high-level Service abstraction, the separation of customer/resource facing services, and the specialized distinct Web Portals. On the other hand, the lack of network slicing capabilities, the minimal multi-domain support, and the monolithic architecture are the disadvantages of this system.

B. ONAP

Open Network Automation Platform (ONAP) is an open-source platform that provides real-time and policy-driven orchestration, management, and automation of network and edge computing services, enabling rapid management of services and their lifecycles [3]. It supports E2E 5G Network Slicing, defining an E2E Slice as a network service composed of RAN, TN, and CN slice sub-nets. To provide Network Slicing, ONAP relies on the three layers of slice management functions defined in 3GPP TR 28.801 [4], providing an internal Communication Service Management Function (CSMF), a Network Service Management Function (NSMF), and a Network Slice Subnet Management Function (NSSMF). Although, ONAP also allows the usage of external NSSMFs, which is made possible through an adaptor specifically designed to support this functionality. ONAP also offers transparent monitoring of the Network Slice Instances (NSIs), through the slice manager interfaces that expose a vast collection of Key Performance Indicatorss (KPIs), enabling the continuous and exhaustive monitoring of these slices. On top of these KPIs, it is possible to configure several SLAs and policies, which ONAP will then manage in order to prevent the degradation of the NSIs [5].

To achieve all this, the creation, management, and operation of network slices are delegated throughout several ONAP internal components: (i) the Service Orchestrator (SO), which triggers the creation, update, and termination of services, (ii) the ONAP Optimization Framework (OOF), which is supposed to provide a policy-driven placement of VNFs across multi-domain infrastructures, although it has not been fully exploited, the (iii) SDN-C which provisions and manages network resources, (iv) the APP-C, which manages the lifecycle of VNFs, and lastly the Active and Active and Available Inventory (AAI), which observes the resources and services, along with their relations, in real-time.

Although ONAP has made quite a few advancements in E2E Network Slicing, it still deals with the inherent problems of defining a network slice as a network service, where all its internal VNFs have to be instantiated in the same tenant, which is not adequate to scenarios where performance is of utmost importance. These scenarios require VNFs to be placed where their performance and cost are most efficient, which, so far, is a feature that ONAP does not provide.

C. Vertical Slicer - 5G-Transformer

The 5G-Transformer is a 5G Infrastructure Public Private Partnership (5G-PPP) project aimed at improving the mobile transportation network and transforming it into an SDN/NFV-based network, integrating network slicing [6]. The project designed and implementend a 5G platform with three main components: the Vertical Slicer, the Service Orchestrator, and the Mobile Transport and Computing Platform.

The 5G-Transformer (5GT)-Vertical Slicer (VS) is the verticals entry point for the system and an OSS/BSS component of the 5GT administrative domain. This entity is responsible for coordinating and arbitrating vertical services, accessible to verticals through a high-level interface focused on their logic and needs [7]. A vertical must establish a Service Level Agreement (SLA) with the 5GT platform, a series of Service Level Objectives (SLOs) specified by verticals and based on their service needs (for example, the maximum E2E latency of 20 ms). Any SLA degradation may cause severe problems to verticals by compromising the technical behavior of its services. In some cases, such degradations can impact the reputation and business leadership of the vertical. For the high-level abstraction needed by verticals, the 5GT platform uses Vertical Service Blueprint (VSB) to define a network service composition, creating a scaffold ready to use when needing that topology. As a VSB extension, a vertical must create a Vertical Service Descriptor (VSD) by completing it with Quality of Service (QoS) parameters, resulting in a ready-to-use deployment recipe that will instantiate the defined topology and follow the given QoS values. The platform maps each VSD onto network slices, which in the 5GT system are an extension of the European Telecommunications Standards Institute (ETSI) NFV NSDs [7].

The advantages of the 5GT-VS are the VSBs and VSDs abstractions over the underlying complexities, the initial support of network slicing with extended NSDs, the multi-domain support, and the SLA management. As for disadvantages, the network slicing support is not according to current standards, the multi-domain support is very limited, and the architecture is monolithic.

D. Vertical Slicer - 5Growth

5Growth (5GR) is another 5G-PPP project trying to push the 5G vision of 5G empowered vertical industries closer to deployment. Its objective is the validation of 5G technologies from the verticals point of view, both technical and business-wise. For that reason, the project decided to take advantage of the achievements and developments in network slicing, virtualization, and multi-domain solutions of the 5G-PPP phase 2 projects, such as 5G-Transformer and 5G-Monarch [3]. Furthermore, the project chose two ICT-17-2018 [4] 5G E2E platforms to test their developments, namely 5G-EVE and 5G-VINNI [8]. 5Growth set out to support industry verticals processes by providing four main features: a vertical portal

[3]https://5g-monarch.eu/
[4]https://cordis.europa.eu/programme/id/H2020_ICT-17-2018

for bridging the gap between verticals and the 5G facilities; closed-loop automation; SLA control for the services lifecycle; and finally, an Artificial Intelligence (AI)-driven E2E network solution to optimize Access, Transport, Core and Cloud, Edge and Fog resources, across multiple technologies and domains [8].

The 5GR project used the 5G-Transformer platform as the starting point where extensions/enhancements over its composing blocks were added (5GT-VS, 5GT-SO and 5GT-Mobile Transport and Computing Platform (MTP)). The improvements follow both functional and service requirements of the use cases devised for this project. The platform work plan consists of different innovations, each selected to fill a given gap found in the base platform, resulting in twelve improvements. Being an extension of the 5GT project, it inherits many of its advantages and disadvantages. In addition to VSBs and VSDs abstractions over the underlying complexities and the SLA management, the 5GR project adds network slicing according to standards and better multi-domain support (both in the Service Orchestrator and the Vertical Slicer). On the other hand, the architecture remains monolithic, and since the implemented innovations were add-ons, the final platform's quality may not be optimal.

E. Comparison of the Vertical Service Orchestrators

Having briefly presented several Vertical Service Orchestrators, we can now present their summary in Table I, making it easier for the reader to compare them.

III. Network Orchestrator(NetOr)

Considering the SoA vertical service orchestration solutions, the main problems this article's PoC platform should solve are: (i) monolithic architectures, (ii) non-standardized network slicing, and (iii) inadequate multi-domain support. Hoping to advance the SoA, we developed Network Orchestrator (NetOr), a system that consists of an OSS/BSS system that operates over the operator's 5G infrastructures and services. We do not define NetOr as novel platform for network slicing, but rather a platform that provides an improvement of the modern Vertical Service Orchestrators, aiming to mend some of their drawbacks. Briefly explaining why these aspects raise several problems: concerning the architecture, although modular, all SoA solutions are a unified monolithic system composed of several Java Springboot[5] applications. Monolithic applications are not the best solution for the maintainability, flexibility, and scalability needed for this kind of orchestration system. Regarding the non-standardized network slicing support, only the 5GR project (through an add-on) and ONAP support network slicing according to current standards, which demanded workarounds and may have impacted the final system's quality. This issue is quite relevant since by not supporting the current standards, the interoperability and integrability decrease significantly. Finally, the inadequate multi-domain support problem is similar to the last one, where only the 5GR project has a more mature and complete solution, but is achieved again with an add-on. Once again, this issue is relevant, since by not correctly supporting inter-domain scenarios, vertical use cases with higher complexity that demand such functionality cannot be supported.

[5]https://spring.io/projects/spring-boot

A. System Architecture

NetOr follows a micro-service and event-driven architecture, developed with a strong focus on low-coupling, flexibility, modularity, and scalability. Platforms of this style allow abstracting both the intricate actions needed to deploy a network service, and the infrastructure and network complexities. With those abstractions in place, the end-user (vertical industry) can concentrate its efforts in developing and analyzing the services and functions needed for the organization's objectives.

By default, a micro-service oriented system favors asynchronous communications. In comparison with a sequential approach, asynchronous communications allow parallel processing, which may in some cases improve the performance and efficiency of a given process. Given its high scalability, NetOr may be able to improve the performance over an extensive execution of services management operations compared to other existing similar orchestration systems. On the other hand, this type of architecture has its own drawbacks, being one of them the need to exchange considerably more messages through the network compared to monolithic systems since it constantly exchanges information between the micro-services. One can say that, although NetOr may not provide increased performance when dealing with low system loads, when faced with a higher system load, it will outperform the Vertical Service Orchestrators presented in Section II , mainly due to its fully distributed service oriented architecture.

Finally, another requisite for the NetOr system was to facilitate its interaction with verticals. For that reason, a web portal was developed to help the vertical user to interact with the NetOr system. Through a minimal graphical interface, the portal presents the available actions in the most intuitive way possible and abstracts as much as possible the underlying complexities.

The final NetOr architecture and its main components are presented in Fig. 1.

B. APIs and Data Models

There are two distinct areas where the NetOr system should consider and support standards and norms, which are the interfaces interconnecting the NetOr with external components, namely its Northbound Interface (NBI) and Southbound Interface (SBI). The NBI should follow well-known community accepted standards to enable the system to be easily adoption by third-party platforms, to allow the seamless substitution with equivalent platforms, and to guarantee that all required parameters for the underlying systems are present in the data models. Similarly, it is also crucial that the NetOr's SBI, the interface interacting with underlying orchestrators, such as Network Function Virtualization Orchestrators (NFVOs),

TABLE I: Comparison of the presented Vertical Service Orchestrators

Vertical Service Orchestrator	VSBs and VSDs Abstraction	CFS and RFS Separation	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		✓ [9]	✓ [9]	✓		✓	✓ [9]	✓
Vertical Slicer - 5G Transformer	✓		✓				✓	✓
Vertical Slicer - 5GROWTH	✓		✓	✓	✓		✓	

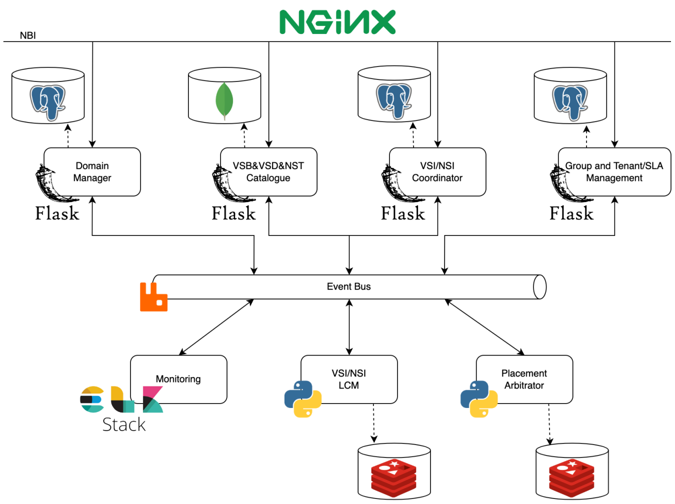

Fig. 1: NetOr's architecture

follows well-defined and accepted standards for various reasons, such as allowing the support of different orchestrator technologies, such as distinct NFVOs. Another reason for supporting such norms is that the parameters needed for the resources and features of those NFVOs are, by default, correctly detailed in the data models.

In terms of the NBI, after analyzing the most relevant standards of the area, those more related to the data model exchanged in Create, Read, Update and Delete (CRUD) operations related to Vertical Service Instances (VSIs) and auxiliary resources were selected. Those standards are TS 28.541 [10], SOL005 [11], and SOL006 [12], all mainly focused on information models. In addition, the data structures adopted in NetOr were also based on the models adopted by the most mature SoA vertical service orchestration system, which is the 5GR-VS. The VSB needed to be extended to contemplate the actions and parameters supported by a given VSI. The VSI instantiation data model was also extended to support dynamic instantiation configurations and domain deployment selection.

Concerning the SBI, it was also necessary to analyze the most relevant standards of the area and select the best-suited ones, focusing on data models and operations. The chosen standards are TS 28.530 [13], TS 28.531 [14], TS 28.541 [10], TR 28.801 [4], SOL005 [11], and SOL006 [12]. Although the standards are quite detailed and complete, they needed to be extended to support all NetOr operations. In particular, it was necessary to add management functions to allow the execution of runtime operations over instantiated network services and network slices. Additionally, the models and actions adopted for the SBI of the most mature SoA vertical service orchestration system, which is the 5GR-VS, also influenced NetOr's operations and data structures.

IV. Multidomain Automatic Mechanism

Concerning mobile service and network coverage, different operators manage distinct geographic zones, limiting the areas where each operator can provide their services directly. If a client of a given operator enters a region covered by another operator, there are mechanisms to enable the redirection of its traffic to the correct operator. Currently, the approach when instantiating Vertical Services, Network Services, and Network Slices assumes that a unique domain will provide them, redirecting to it the data and requests of end-users if they enter areas covered by other operators. It is easy to imagine many scenarios where these redirecting mechanisms could impact the final E2E service, such as automotive or medical scenarios that rely on very low latency. In those scenarios, the ideal solution is to have the service in question instantiated across the various domains with which the end-user may interact. With this in mind, the inter-domain mechanism appears to enable the on-demand connection of independent regions. This mechanism is a recent innovation in the NFV and service orchestration world. It is a powerful functionality that disrupts the current *modus operandi* of network and service providers. Currently, those providers follow guidelines and restrictions that force their clients to be only served by them, meaning that all the service and network functions are hosted and instantiated in their unique domain.

Several standards and scientific works are starting to study and propose best practices and architecture options when dealing with this inter-domain scenario. An example is the standard TR 28.801 [4], which suggested three different solutions for multiple operator coordination management. The first approach suggests the customer owning the CSMF and communicating with the various operators, which provide the NSMFs. A second option suggests the customer communicating with a main operator hosting the CSMF, which will interact with its and others operator's NSMFs. Finally, the third option suggests the customer communicating with a main operator hosting the CSMF, delegating the actions to its NSMF, which leverage the multiple operators NSSMFs.

Similarly, [15] is a scientific article proposing a 5G vertical service orchestration framework focused on network slicing on an inter-domain scenario. It contextualizes the inter-domain environment, presenting Verticals/Digital Service Consumers (DSCs) that use the services provided by the Digital Service Providers (DSPs), which in turn are dependent on the capabilities and resources provided by the Network Service Providers (NSPs). The DSP manages service lifecycles and exposes them to Verticals. The DSC only consumes the services exposed by the DSP. Finally it presents the proposed inter-domain orchestration framework, composed by a three levels of orchestration logic, the Service Orchestrator, the Slice Orchestrator, and the Resource Orchestrator.

The chosen architecture for NetOr follows the one proposed in the first multi-operator coordination solution presented in TR 28.801 [4] and the architecture proposed in [15]. Both architectures propose a centralized service orchestration agent(CSMF/Service Orchestrators) that communicates with lower-level orchestrators (NSMFs/Slice Orchestrator). Additionally, this PoC implements an improvement over said architectures, since none of them studied the solution to connect the independent domains. The proposal is to create and configure a Virtual Private Network (VPN) tunnel, creating a secure communication channel. For this PoC and use cases, Wireguard[6] was the technology used for the tunnel, a straightforward and minimalist framework that allows the rapid instantiation and configuration of VPNs. Additionally, the PoC itself is an improvement over the mentioned works, since none of them validated their architectures.

Concerning this mechanism, the main objective in this PoC is to instantiate an E2E Service across multiple domains without prior negotiations. That is technically possible by using an E2E Network Slice, ideally with its subnets composed of Network Slices, as depicted in Figure 2. To successfully achieve the inter-domain, ideally a centralized service orchestrating agent should exist independent from all domains, allowing the connection of the tunnel peers. That agent will receive and process the dynamic tunnel endpoints' information and exchange it with the remaining peers, effectively completing the tunnel configuration. NetOr, in addition to all the other features and functionalities, also serves as that centralized orchestrator.

[6]https://www.wireguard.com/

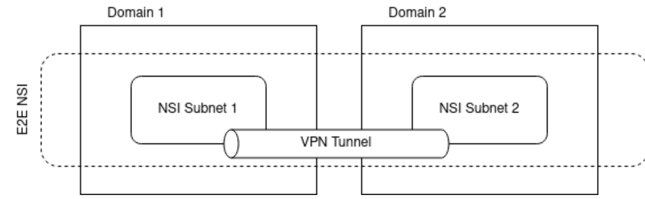

Fig. 2: NSI composed E2E NSI architecture

V. Results

As mentioned earlier, the main goal of NetOr is to fix some of the known flaws of the current Vertical Service Orchestrators. Section III already elucidated to the reader the architecture of our solution and the standards it follows. Based on what was previously described, the reader already knows that (i) NetOr was built according to a service oriented architecture, (ii) NetOr follows all relevant standards in regard to Network Slicing, (iii) NetOr fully supports inter-domain scenarios, and (iv) that NetOr can perform runtime operations over the Vertical Services.

Although, it is still missing to evaluate the performance of our solution when compared to other SoA Vertical Service Orchestrators. Despite fixing several known issues, if NetOr's performance is considerably lower than the one of the other Vertical Service Orchestrators, there will be resistance to using our solution. To validate this, in this Section, we compare NetOr's and 5GR-VS's performance, currently the most mature vertical service orchestration solution.

A. Testing Environment

Since the inter-domain environment needs at least two domains, two independent Open Source MANOs (OSMs) were deployed, one for each domain, and each one deployed in a Virtual Machine (VM), running *Ubuntu 18.04*, with 12 Gb of RAM, 4 VCPUs, 150 Gb of storage. The first OSM integrates with a fully-fledged OpenStack instance managed by IT-Aveiro, where the project defined had the limitations of 30 VM instances, 60 VCPUs, 70 Gb of RAM, 1 Tb of storage, and 10 networks. The second OSM integrates with a DevStack, a version of OpenStack focused on allowing the quick creation of development infrastructures, managed by colleagues in IT-Aveiro. The project assigned had the limitations of 10 VM instances, 20 VCPUs, 50 Gb of RAM, 1 Tb of storage, and 100 networks. Additionally, I used a dedicated VM running *Ubuntu 18.04* with 8 Gb of RAM, 4 VCPUs, and 32 Gb of storage to deploy the NetOr system.

B. Instantiation and Termination Performance Values

In terms of performance tests, we tested the delays introduced by each Vertical Service Orchestrator during the orchestration phase and also conducted E2E tests to evaluate the overall delays introduced by each solution. Regarding the first tests, each was repeated 30 times, and we present the average, maximum and minimum values obtained and also the standard deviation. Concerning the E2E tests, the same methodology was used to gather NetOr's performance metrics, and the results from [16] were used to evaluate 5GR-VS's performance.

System	Max	Min	Avg	StDev
5GR-VS	234 ms	81 ms	113,86 ms	29,20 ms
NetOr	345 ms	135 ms	190.93 ms	24.82 ms

TABLE II: 5GR-VS and NetOr System's service instantiation delay

1) Orchestrator System Delays: The first comparison test focused on the delay added by the NetOr and 5GR-VS systems to the vertical service instantiation process. This delay was obtained by calculating the time delta between the exact moment before making the instantiation request and the precise moment before each system forwarding it to the respective underlying NFVO.

By analyzing Table II, it is clear that the NetOr system generates higher delays than the 5GR-VS, but the difference between them is not that significant, being on average lower than 80 milliseconds. Nevertheless, both systems' delays are in the same order of magnitude.

System	Max	Min	Avg	StDev
5GR-VS	108 ms	19 ms	45,90 ms	17,20 ms
NetOr	187 ms	59 ms	101,24 ms	20,90 ms

TABLE III: 5GR-VS and NetOr System's service termination delay

The second comparison test was very similar to the first one, but it compared the delay of the system when terminating a vertical service. The time delays were obtained in the same manner as in the first test. The NetOr system generates higher delays than the 5GR-VS, differing on average less than 40 milliseconds, following the same behavior shown in the first comparison test.

Note that the delay values obtained from this termination operation are considerably lower than those obtained from the instantiation operation. This behavior was expected since the instantiation process is much more complex than the termination of a vertical service.

2) E2E Management Delays: Concerning the E2E tests performed with the NetOr system, they focused on validating and gathering the time needed to instantiate and terminate a fully-fledged inter-domain service. The final service was composed of an NSI composed by two distinct Network Slice Subnet Instances (NSSIs), one for each domain. Each of those NSSIs contained an NSI with the same basic architecture.

When analyzing Table IV, as expected, the instantiation intervals are considerably higher than the termination intervals. Another relevant conclusion is that the time intervals obtained for these E2E operations are dependent on the infrastructure used to support the creation of those services and their VMs. As presented in Subsection V-B1, it is safe to assume that the delay added by the NetOr system to the instantiation process is approximately 200 milliseconds. With that in mind, most of the time spent instantiating the network service is the NFVO and the Virtual Infrastructure Manager (VIM) creating the necessary resources. For that reason, the time delays of approximately 5,8 minutes for the instantiation and around 1,2 minutes for the termination operations can be lower if the infrastructure allows.

System	Max	Min	Avg	StDev
Inst.	409346 ms	279582 ms	348858,9 ms	29334,1 ms
Term.	109319 ms	38687 ms	74971,9 ms	16588,7 ms

TABLE IV: E2E interdomain service instantiation and termination time

Taking in consideration the work done in [16], which conducted the same tests in the same infrastructure as those conducted in this article, it was possible to compare the E2E management delays obtained in the NetOr and the 5GR-VS orchestrators. According to [16], the simple inter-domain connectivity service when orchestrated by 5GR-VS took on average 9 minutes to instantiate and 46 seconds to terminate.

C. Results Analysis

To facilitate the comparison between the performance of NetOr and 5GR-VS, Table V is presented.

	NetOr	5GR-VS
Avg. System's Introduced Instantiation Delay	$\sim 191\,ms$	$\sim 114\,ms$
Avg. System's Introduced Termination Delay	$\sim 101\,ms$	$\sim 46\,ms$
Avg. E2E Interdomain Instantiation Time	$\sim 5.8\,min$	$\sim 9\,min$
Avg. E2E Interdomain Termination Time	$\sim 74\,sec$	$\sim 46\,sec$

TABLE V: Comparison of the delays introduced by each Vertical Service Orchestration solution and the time needed to instantiate and terminate a Vertical Service

When compared to the NetOr values, the instantiation time in the 5GR-VS is considerably higher, differing in more than 3 minutes. The termination time resulting from the 5GR-VS is lower than the NetOr ones, differing close to 30 seconds. The difference between the instantiation time can be easily justified by the systems' implementation approaches. The 5GR-VS is a monolithic platform that sequentially manages its services, meaning that at any given time, the system can only process one operation of a given service component. On the other hand, the NetOr system, aided by its micro-service-oriented architecture, can manage various service components and various services at the same time. This parallelization enables the NetOr platform to save precious minutes by processing the independent service subnets at the same time. The reason for the 5GR-VS generating lower E2E termination times may be related to the fact that the overall process is quite simpler, so performance variations in the underlying infrastructure can affect the E2E times of this phase.

As a final analysis, when comparing the NetOr's performance values to those obtained from 5GR-VS, there are positive and negative aspects. Although NetOr generates higher delays in small and quick operations, on the other hand, comparing the performance of more complex processes that can take a substantial time, NetOr performs better than the 5GR-VS.

VI. Conclusion

As seen in the results, NetOr's performance is very similar to the 5GR-VS's one (currently, the most mature Vertical Service Orchestrator). In truth, if we consider only the instantiation time, NetOr significantly decreases the time needed to deploy a Vertical Service, by taking advantage of its parallelization capabilities.

Additionally, the NetOr platform already provides options and mechanisms that most Vertical Service Orchestrators lack, which was the primary motivation behind the work presented in this paper. We must stress that, while having similar performance to the 5GR-VS, (i) NetOr follows all relevant standards regarding Network Slicing, (ii) it was built according to an SoA, (iii) it fully supports multi-domain scenarios, and (iv) that NetOr can perform runtime operations over the Vertical Services. Besides this, similarly to the 5G-Transformer and 5Growth VSs, NetOr also provides a layer of abstraction through VSBs and VSDs.

Regarding future work, we intend to enhance NetOr by providing SLA management, propelling our solution's adoption in Vertical Services Orchestration use cases.

Acknowledgements

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

References

[1] 5G_PPP, "View on 5G Architecture," *Version 3.0, February 2020*, no. February, pp. 21–470, 2020. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2019/07/5G-PPP-5G-Architecture-White-Paper_v3.0_PublicConsultation.pdf

[2] "Introduction - Openslice." [Online]. Available: https://openslice.readthedocs.io/en/stable/

[3] "ONAP - E2E Nework Slicing Use Case." [Online]. Available: https://docs.onap.org/projects/onap-integration/en/latest/docs_E2E_network_slicing.html

[4] 3GPP, "TR 28.801-v15.1.0-Study on management and orchestration of network slicing for next generation network," Tech. Rep., 2018. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3091

[5] V. Q. Rodriguez, F. Guillemin, and A. Boubendir, "5g e2e network slicing management with onap," in *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2020, pp. 87–94.

[6] "5G-Transformer - 5G-PPP." [Online]. Available: https://5g-ppp.eu/5g-Transformer/

[7] X. Li *et al.*, "5G-TRANSFORMER: Initial System Design," Tech. Rep. [Online]. Available: http://5g-transformer.eu/wp-content/uploads/2019/11/D1.2_5G-TRANSFORMER_Initial_System_Design.pdf

[8] "5Growth - 5G-PPP." [Online]. Available: https://5g-ppp.eu/5growth/

[9] "ONAP - Project's Documentation." [Online]. Available: https://docs.onap.org/

[10] 3GPP, "Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3 (3GPP TS 28.541 version 15.0.1 Release 15)," Tech. Rep., 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_TS/128500_128599/128541/15.00.01_60/ts_128541v150001p.pdf

[11] ETSI, "GS NFV-SOL 005 - V2.6.1 - Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point," Tech. Rep., 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/02.06.01_60/gs_nfv-sol005v020601p.pdf

[12] ——, "GS NFV-SOL 006 - V3.3.1 - Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on YANG Specification," Tech. Rep., 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/006/03.03.01_60/gs_NFV-SOL006v030301p.pdf

[13] 3GPP, "Management and orchestration; Concepts, use cases and requirements (3GPP TS 28.530 version 15.0.0 Release 15)," Tech. Rep., 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/128500_128599/128530/15.00.00_60/ts_128530v150000p.pdf

[14] ——, "Management and orchestration; Provisioning (3GPP TS 28.531 version 15.0.0 Release 15)," Tech. Rep., 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/128500_128599/128531/15.00.00_60/ts_128531v150000p.pdf

[15] G. Bernini, P. G. Giardina *et al.*, "Multi-domain orchestration of 5G vertical services and network slices," in *IEEE ICC Workshops 2020 - Proceedings*. IEEE Inc., jun 2020.

[16] J. Fonseca, J. Alegria *et al.*, "Dynamic Inter-domain Network Slicing for Verticals in the 5Growth Project," *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks, SDN-NFV 20201*, 2021.