Meneses, F., Fernandes, M., Corujo, D. y Aguilar R.L. (2020). SliMANO: An Expandable Framework for the Management and Orchestration of End-to-end Network Slices. In *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*.

DOI: https://doi.org/10.1109/CloudNet47604.2019.9064072

SliMANO: An Expandable Framework for the Management and Orchestration of End-to-end Network Slices

Flávio Meneses[1], Manuel Fernandes[2], Daniel Corujo[3], Rui L. Aguiar[4]
Instituto de Telecomunicações and Universidade de Aveiro, Portugal
Email: {*flaviomeneses[1], manuelfernandes[2], dcorujo[3]*}*@av.it.pt; ruilaa@ua.pt[4];*

***Abstract*—This paper proposes a slice management and orchestration framework for abstracting the instantiation of end-to-end network slices, which are composed by a chain of both physical and virtual network functions. In this line, the proposed SliMANO framework is a plug-in based system that requests network resources and coordinates the interaction among network orchestration entities for its instantiation and chaining in order to perform an end-to-end slice. These entities could range from management and orchestration (MANO), Software Defined Networking (SDN) controllers and Radio Access Network (RAN) controllers. A proof-of-concept prototype was implemented and experimentally evaluated, with results showcasing its feasibility. The results revealed a increase in the delay, associated with instantiation and deletion operations, when compared with the recently introduced network slicing feature (NetSlice) of the Open-source Management and Orchestration (OSM). Results showed that the delay is mostly associated to SliMANO being an entity external to the orchestrator itself, which comes as a trade-off for its added inter-operation capabilities. Moreover, SliMANO goes beyond the MANO domain and actually allows the interaction with SDN and RAN controllers.**

***Keywords*—Network Slicing, SDN; NFV; MANO.**

I. Introduction

The world is rapidly moving towards the fifth generation (5G) [1] of telecommunication networks. 5G aims to bring a greater degree of flexibility to the mobile network, by allowing a highly customizable, scalable and adaptable network capable of addressing the user specific requirements (e.g., Service Level Agreements (SLAs)). In this context, the Next Generation Mobile Networks (NGMN) has defined a set of service types, in order to group SLAs in a well defined set of services - namely, Enhanced Mobile Broadband (eMBB), massive Internet of Things (mIoT) and ultra-reliable low latency communications (URLLC) - each of them with different network requirements [2]. At the time of this writing, despite initial real-world deployments in cities around the globe, 5G development is still an ongoing effort due to the complexity associated to its new targeted deployment scenarios.

To address these service requirements, 5G networks explore techniques and paradigms, such as Network Functions Virtualization (NFV) and Software Defined Networking (SDN). On one hand, NFV allows to decouple network functionalities from the hardware and move them to data-centers as Virtual Network Functions (VNFs). On the other, SDN provides the flexibility to dynamically reconfigure datapaths via standardised APIs. In this context, network services are defined as a chaining of VNFs, that via SDN mechanisms are highly reconfigurable, allowing to accomplish the service types expected to take part of the 5G networks in a holistic architecture.

Nevertheless, the ETSI Industry Specification Group for NFV [3] has evidenced the need for on-demand deployment of such network virtualization and softwarization capabilities, in order to enable the different service types. In this line, the Network Functions Virtualization Orchestrator (NFVO) aims to orchestrate VNFs by deploying and configuring VNFs in a pre-defined set of Virtual Infrastructure Managers (VIMs), allowing the creation of a network to support a certain service type. This concept of having a dedicated and isolated network deployed on top of another network substrate, for supporting specific service types, is often presented as network slicing.

The NFVO is one key component of Management and Orchestration (MANO) architectures, and it is in charge of the deployment and configuration of VNFs which are part of a network slice, leaving the interconnection of these VNFs out of its scope. However, this aspect still needs to be addressed, for enabling a truly automated slice orchestration. In this context, 3GPP proposes in their study for network slicing MANO [4] the following terms and definitions: (i) Network Slice Template (NST), which has all the slice definitions used as a skeleton to build-up network slices; (ii) Network Slice Instance (NSI), which uses a NST as base and a set of custom parameters to build-up a network slice. In addition, the (iii) Network Slice Subnet Template (NSST) represents a slice template that is part of a higher level slice template, and the Network Slice Subnet Instance (NSSI) that is a NSI that is part of a higher level NSI.

This papers proposes a framework that follows the 3GPP specification for the management and orchestration of end-to-end network slices. The proposed framework, named Slice Management and Orchestration (SliMANO), was implemented and experimentally evaluated in an in-house data-center using OpenStack and Open Source MANO (OSM) as VIM and MANO, respectively. Results showcased the feasibility of the framework, which shows a slight increase in delay when compared with the slice component of the OSM, due to it being an external entity in regards to the orchestrator. This comes as a trafe-off for the added flexibility and interoperability of

SliMANO.

The paper is organised as follows. Section II presents the related work. The framework is introduced in section III, while section IV presents the high-level signalling for a network service deployment use case. Implementation details and its deployment are presented in section V. Section VI evaluates and discusses the framework proposal. Finally, the paper concludes in section VII.

II. Related Work

In a Network Service, NFVOs are responsible for deploying and interconnecting virtual appliances. However, the NFVO is limited to the realization of such interconnection inside a singular VIM domain, being unable to perform a logical service with virtual appliances distributed among multiple VIMs. In this line, in [5] the authors propose a NFVO framework architecture which aims to bring MANO to multiple VIM domains, allowing it to orchestrate appliances on multiple VIM and interconnect them using WAN Infrastructure Managers (WIMs). This work, besides implementing a NFVO framework, has a slice management framework, capable of orchestrating slices using the internal NFVO framework with the correspondent monitoring mechanisms in order to fulfill slice mandatory SLA/QoS requirements. Nevertheless, this slice management framework only works with the internally developed NFVO and cannot deploy appliances in NFVOs located outside the framework's domain. This evidences a limitation when considering heterogeneous network environments, that can have different types of NFVOs deployed. Similarly, NESMO [6] develops a slice management framework in a network operator's point of view, taking into account the development of mechanisms to automate the design, deployment and management of network slices. However, it does not take into consideration the 3GPP slice management specification [4], and it does not have a proof-of-concept to consolidate its viability.

In this context, ETSI standardised the functionalities of network components, with solutions for slicing orchestration from both academia and open-source organizations (such as ETSI itself and Linux Foundation) following these specifications. In fact, ETSI proposes the Opensource MANO (OSM)[1] which was primarily built to be a NFVO framework based in ETSI's NFV specification [3]. Recently, slice management and orchestration capabilities, compliant with ETSI's specification, were added to OSM. Alternatively, ONAP[2] is a Linux Foundation project, whose main objective is to tackle the orchestration of virtual appliances in a policy-driven and automated way. However, despite its broad set of features, ONAP currently poses a more stringent installation resource footprint. There are other important opensource NFVOs like OpenBaton[3] and Cloudify[4]. Cloudify was not built based on ETSI NFV MANO like OSM and OpenBaton, however a plugin was built in order to support ETSI NFV MANO specification.

It is worth to mention the work done in Maestro [7]. In this work, authors propose a NFV MANO framework with a focus on Radio Access Networks (RAN). The main objective was to split a VNF-based RAN in more atomic elements, with the authors proposing to split the RAN's VNFs in more granular VNF elements. The framework allows to choose which is the best VNF for each specific situation, based on the current operator network requirements and RAN state. The results showed that a reduction of network delay is possible with a VNF split by a factor of 3 with a slight increase in processing time.

The work in [8] proposes an event-driven slice management framework capable of orchestrating network slices composed by VNFs, PNFs and Virtualized Network Applications (VNAs). The proposed framework relies on plugins for supporting additional network resources (e.g. FlexRAN[5] for RAN slice management). However, currently it is limited to internal NFVOs and VNF Managers (VNFM), thus using JUJU[6] as VNFM and a self-developed solution for managing the lifecycle of VNFs.

Moreover, existing solutions, such as OSM and FlexRAN, are limited to a specific scope: while some are more focused on RAN slice management, others are focused on slices based on network functions chaining within a data-center leaving Physical Network Functions (PNFs) out-of-scope. Notwithstanding, end-to-end network slices are composed by multiple slices (e.g., a radio slice chained with an infrastructure slice), which are orchestrated by different entities. As such, there is a need for a network entity able to interact and coordinate the overall network slicing life cycle management. In this context, in [9] and [10] authors propose such interactions with the different entities to be performed by an SDN application along with the Ryu SDN controller. However, the diversity of SDN controllers and MANO frameworks creates the urge to develop a third-party entity for supporting such diversity.

This is where SliMANO aims to contribute. SliMANO is an ETSI-compliant end-to-end network slice manager and orchestrator that abstracts network slicing actions (e.g., instantiation, decommission and reconfiguration) from the responsible network orchestration entities (i.e., MANO, SDN controller and RAN controller). SliMANO is also responsible for doing the monitoring and optimization of those slices and act in case of failures or lack of performance. Additionally, SliMANO was built using the more recent network slicing and NFV standards, which eases the integration with some components already present in network operators.

III. Framework Overview

The highly automated deployment of end-to-end slices and the multi-domain of its service orchestration requires a new level of abstraction of the framework. As such, SliMANO is

[1]OSM: https://osm.etsi.org/

[2]ONAP: https://www.onap.org/

[3]OpenBaton: https://openbaton.github.io/

[4]Cloudify: https://wp.cloudify.co/

[5]FlexRAN: http://mosaic-5g.io/flexran/

[6]JUJU: https://jaas.ai/

presented as a plug-in system of network modules that enables the slice orchestration to be agnostic of both MANO and VIM frameworks and, ultimately, of SDN controllers. Figure 1 illustrates the motivational scenario, where for deploying end-to-end network slices over the physical network, different networks need to operate and coordinate actions. For example, while the NFVO instantiates network services in datacenters and the RAN controller manages 3GPP radio slices, the SDN controller re-configures the datapath interconnecting PNFs and VNFs. Fig. 2 depicts the SliMANO architecture divided in three main building blocks, namely the SliMANO Core, SliMANO Plug-in Framework and SliMANO Agents Framework.

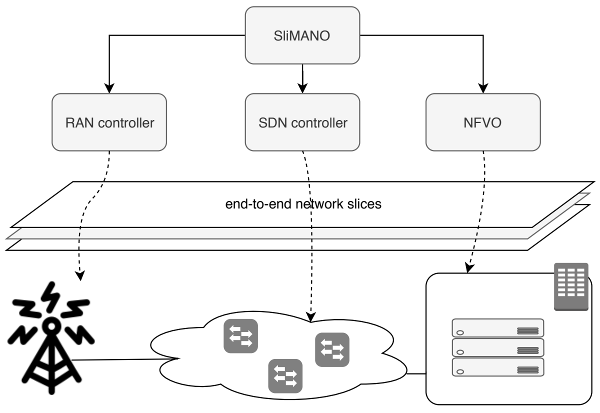

Fig. 1: Motivational scenario.

A. SliMANO Core

The *core* building block is responsible for coordinating SliMANO's functionalities. In this line, currently it implements three main components, described as follows.

Slice lifecycle management: It handles operations related to network slice resources, such as their update (to modify the resource's behaviour) and deletion.

Slice monitoring: This module is responsible for monitoring the resources allocated to each network slice instance. In case of resource failure, it notifies the slice lifecycle management, which in turn verifies how to recover from the failure (if possible). In addition, the slice monitoring module manages the slice resources and its capability to meet the imposed Quality of Service (QoS). This results in a highly scalable service assurance system and effective closed-loop lifecycle management.

Slice orchestration: It verifies the availability of resources for the instantiation of a network slice. Also, it is responsible for requesting the necessary resources to the entitled entities (e.g., MANO, SDN controller, FlexRAN controller). Resources can be a NFVO Network Service (NS), a SDN application or a network slice in a Radio Access Network (RAN).

B. SliMANO Plug-in Framework

This block allows SliMANO to be a generic framework, by building an API between the *core* (via plugins) and *agents*, facilitating the continuous development of new plugins and respective agents. Note that each plugin has its respective *agent* for external communication.

NFVO plug-in: It builds a contract to communicate with NFVOs. For example, in a scenario where OSM is the NFVO, the plugin uses the OSM agent for performing the necessary operations involving OSM (e.g., instantiate and delete NSs).

Network controller plug-in: Similarly, it uses the respective agent to request operations to the network controllers (e.g., reconfigure the datapath and establish QoS), such as SDN controllers.

RAN plug-in: It performs operations (via agents) on 3GPP-based network slices (e.g., re-dimension RAN slices).

C. SliMANO Agents Framework

As mentioned above, the agents framework is coupled with the plug-in framework. Thus, it performs the actions requested by plugins to the respective external network entities.

NFVO agents: The NFVO agents perform the actions on the respective NFVO external entity. As such, it is required to develop an agent for each supported NFVO (e.g., OSM, ONAP, Cloudify, etc.). Usually, actions to the NFVOs are performed via REST APIs.

Network controller agents: Similarly, a network controller agent for each supported controller (e.g., OpenDayLight, ONOS, Ryu), in order to request network operations, such as the reconfiguration of the datapath for interconnecting VNFs.

RAN agents: The RAN agent requests actions to the 3GPP network. For example, the FlexRAN exposes a REST API for radio slice instantiation, reconfiguration and removal.

Northbound Interface: Finally, the NBI allows a client (e.g., a network entity, network operator or network administrator) to request the instantiation of a network slice. This request is made via REST, with the SliMANO's NBI translating the request to the necessary network operations.

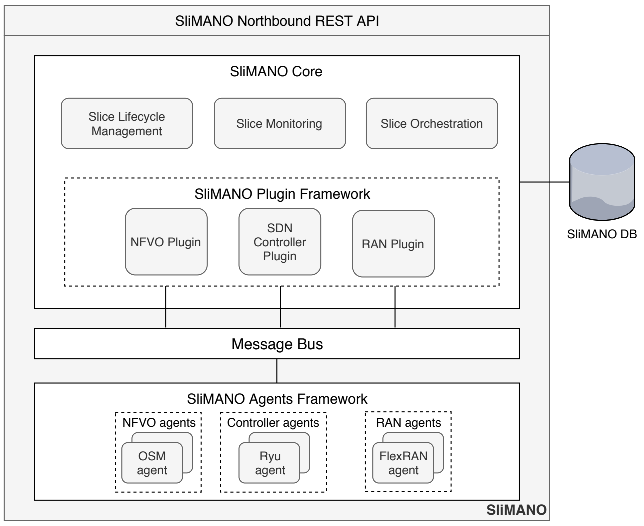

Fig. 2: SliMANO architecture.

IV. Use case and Signalling

This section presents a proof-of-concept scenario where the SliMANO instantiates a network slice in the data-center via a supported NFVO. Here, a network slice is defined as an end-to-end chain of NSIs. Fig. 3 depicts the high-level signalling of such procedure.

A. Network slice instantiation

As explained in the previous section, the NBI translates the request to network operations. The instantiation of a NSI is divided into two main procedures, namely deployment and configuration.

First, on the engine side, SliMANO verifies the dependencies of the requested NSI with other NSIs, deploying such dependencies if necessary. As such, SliMANO verifies the availability of a plugin for deploying the required NFV via a supported NFVO, and a plugin for a support network controller to apply the required network actions. Fulfilling the dependencies, SliMANO internally employs a set of workers for the deployment of the necessary resources into the correspondent external entities (such as, the NFVO). Here, each resource deployment is managed by an engine worker, which in turn is also responsible for ensuring the necessary operations conformity. After each operation completion, the respective worker informs the result (success or fail) to the core engine. Note that, usually, each resource operation execution is done only by one worker, which communicates with a plug-in for applying actions to external entities (e.g., the NFVO). The resources request to the external network entities may involve its configuration. Alternatively, it is possible to apply further configuration through actions defined in the payload of the NSI request.

Finally, the network slice's configuration is stored in the database, and feedback is retrieved to the network slice requester (i.e., client) using the user provided callback.

B. Network slice deletion

Similarly to the instantiation procedure, for deleting a NSI the client requests it through SliMANO's NBI via REST. The NBI validates the payload and proceeds with a request to the core's engine. The engine gets the NSI's information from the database, and starts a Lifecycle Management (LCM) instance for the delete action. In turn, the LCM instantiates a delete task for each resource of the NSI. These tasks contact the corresponding agent (via its core plugin) to perform the delete action via on the respective resource manager (i.e., the NFVO)[7].

V. Implementation and deployment

This section presents the implementation details of the SliMANO architecture, followed by the proof-of-concept deployment of the scenario described in section IV and illustrated in Fig. 3.

[7]In the current version, the delete action callback was not implemented. Thus, the client needs to verify the correct operation by requesting the available NSIs via SliMANO's NBI.

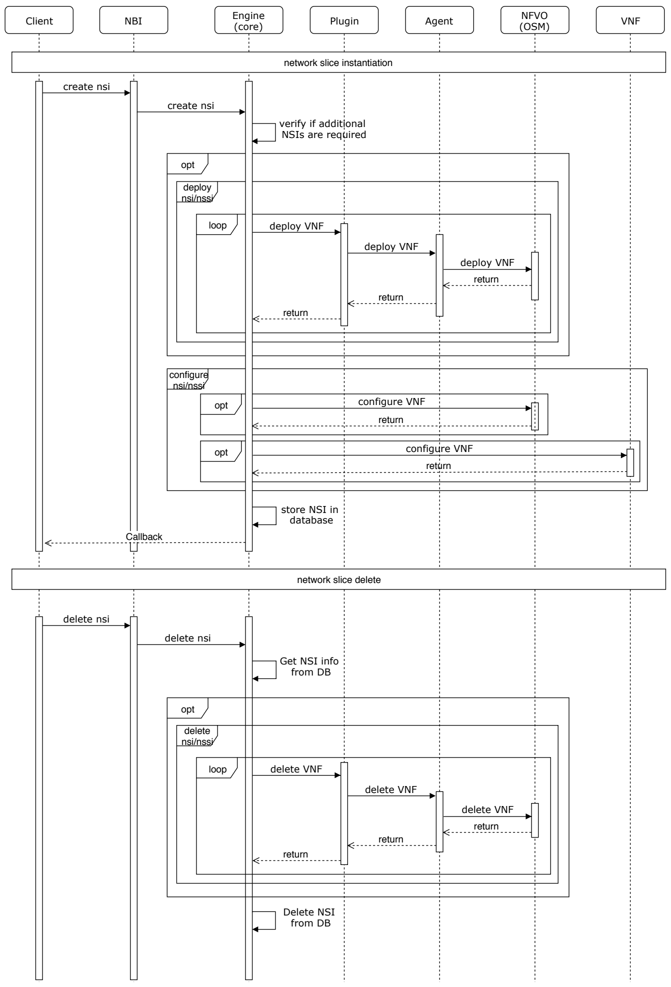

Fig. 3: High-level signalling for NSI instantation and deletion.

A. SliMANO implementation

SliMANO was developed in Python and implemented following a microservices architecture. Here, the Nameko[8] framework was used, offering remote procedure calls (RPC) services over Advanced Message Queuing Protocol (AMQP). In this line, SliMANO's *core* and *agent* components were implemented in different Docker containers, and using the RabbitMQ[9] message broker for asynchronous messaging. Finally, for the database, MariaDB[10] was used. The microservices architecture in conjunction with Docker containers increases deployment flexibility, allowing each SliMANO's component to be deployed in a single host or dispersed among hosts.

B. Proof-of-concept deployment

Our proof-of-concept scenario was deployed in an in-house data-center running OpenStack Queens as VIM and OSM release 5 as MANO[11]. Regarding computational resources,

[8]Nameko: https://www.nameko.io
[9]RabbitMQ: https://www.rabbitmq.com
[10]MariaDB: https://mariadb.org
[11]ONAP was not tested due to system requirements restrictions

OSM was deployed in a VM with 4 vCPUs and 8GB of RAM, and SliMANO was deployed in a VM with 2 vCPUs and 4GB of RAM. For Docker containers orchestration, we used Docker Compose with the containers being deployed in a single host. Finally, in order to compare SliMANO's performance in terms of overall delay, the scenario was also deployed using the NetSlice feature of OSM introduced in release 5 [11], which allows the instantiation of network slices.

C. *SliMANO vs OSM NetSlice: Functional comparison*

SliMANO aims to be a generic and modular solution and in order achieve that, SliMANO base architecture was built upon microservices. Due to that fact, it provides a high degree of flexibility in component development. Hence, the possibility of deploying new plugins/agents to support new kinds of network entities requires low effort.

The previous mentioned characteristic is a key comparison point between SliMANO and OSM's NetSlice module. OSM NetSlice is an OSM internal module that only has an interface with OSM's internal components and, therefore, suffers from limited expandability. The module processes network slice templates, which are composed by a set of NSs and their interconnection definition. This interconnection can be made internally in the OSM target if all the NSs described in the template are located in that OSM instance. The interconnection could be made between different OSM instances if there are NSs that need to be deployed on different OSM instances. For that case, OSM has a component called WIM (WAN Infrastructure Manager) that is responsible for configuring a multi-site network to interconnect those NSs. At the time of this writing, in the tested OSM version, this component is in development stage and still cannot be used.

Our solution can achieve the same OSM WIM functionality by describing the interconnection in the slice template, with SliMANO using the configured plugins/agents to configure it. Nevertheless, like the OSM WIM, this functionality is under development. The main advantage of SliMANO over OSM's NetSlice module is the fact that, as mentioned above, SliMANO is not dependent of any internal framework interface, component or framework. And as a consequence of that, SliMANO is more expandable than OSM's NetSlice module, as it has the capability to support different kinds of frameworks and network premises with a relatively low development effort.

Regarding to standards fulfillment, both solutions follow 3GPP slice management specification [4]. Thus, both of them are on par in that regard.

VI. Evaluation

This section experimentally evaluates SliMANO framework and compares the results with the NetSlice feature of OSM release 5. The proof-of-concept scenario deploys multiple OSM Network Services (NSs) with 1 VNF, which in turn is composed by 2 Virtual Deployment Units (VDUs) deployed on Openstack VIM as Virtual Machines (VMs). For evaluation purposes, the NS is a Kubernetes [12] cluster that contains a VNF with a master and a worker node as VDUs. The experiments were run 50 times, with results presenting their average with a confidence interval of 95%.

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

A. *Network slice deployment delay*

Table I compares the SliMANO's overall instantiation and delete delays of a network slice with the NetSlice feature of OSM release 5.

As can be seen, for both instantiation and delete actions, and independently of the number of the NSs (and, consequently, VNFs with their VDUs) per slice, the SliMANO and OSM presented similar results. Nevertheless, despite SliMANO presented slightly faster deploys when more than 2 NSs are considered, it showed slightly longer values for a single NS deployment, and for network slice deletion. However, such delay increments were all under 3 seconds, which are negligible when compared against the overall procedure delay.

TABLE I: Overall OSM and SliMANO delay for instantiation and deletion

	Instantiation[seconds]		Deletion[seconds]	
NS	OSM	SliMANO	OSM	SliMANO
1	58.00±1.45	60.55±1.16	20.16±0.01	22.70±0.06
2	107.29±1.75	108.74±1.69	28.39±1.11	29.99±1.47
5	258.64±2.96	254.87±5.77	51.49±0.94	53.71±0.42
10	499.53±2.33	498.07±5.32	95.09±1.44	94.13±2.91

B. *Internal components delays*

In order to evaluate the delay imposed by SliMANO in the overall network slice instantiation procedure, we measured the time delay of SliMANO's internal components for different network slice dimensions. Thus, Table II presents the instantiation and delete delays imposed by SliMANO's NBI and core components for network slices of 1, 2, 5 and 10 NSs. As mentioned above, each NS is composed by 1 VNF with 2 VDUs. Thus, for example, in the 5 NS scenario, SliMANO instantiates 5 VNFs and 10 VDUs.

TABLE II: SliMANO's components delay for instantiation and deletion of a network slice

	Instantiation[ms]		Deletion[ms]	
NS	NBI	Core	NBI	Core
1	169.02±3.66	274.48±5.06	175.12±2.36	294.74±5.14
2	165.14±3.16	291.74±4.81	178.38±3.11	326.44±6,31
5	168.38±3.07	463.54±7.63	174.96±2.90	388.90±10.26
10	166.36±2.76	719.58±25.42	176.3±3.50	524.28±12.46

Comparing the values of Table II, we verify that SliMANO imposed similar delays for both instantiation and deletion actions. This results from the fact that both actions have similar procedures (as illustrated in Fig. IV). Additionally, the overall delay imposed by SliMANO is composed by the NBI and core engine. The delay of the SliMANO's NBI remained constant for the evaluated slice dimensions. Contrarily, the SliMANO's core increased its delay as the number of NSs (and consequently VNFs and VDUs) increased. This was mainly due to the fact that as the number of deployed NSs increases, the operations related with thread creation for each NS at the beginning and the ones related to the persistence at the end of deploying are taking more processing time to accomplish. Moreover, Nameko RPC framework adds some significant amount of delay, mainly because it sets up a new connection to the message broker for each remote service (in this case a service corresponds to one agent), which impacts efficiency.

C. Final remarks

OSM and ONAP are two well known opensource MANO architectures that recently added the slice management capabilities. However, such capabilities are restricted to their internal functionalities, such as its own NFVO. Contrarily, our framework proposal, namely SliMANO, offers an external solution agnostic of both the physical and virtual environment, allowing the integration of different NFVOs, SDN controllers and RAN controllers. Nevertheless, being an external solution, SliMANO adds a degree of delay, mainly due to the communications among network entities[13]. However, this cost can be seen as negligible when compared with the overall instantiation delay of a network slices (Table I). In contrast, SliMANO offers a lightweight and independent solution for slice management and orchestration that abstracts the operator from the NFVOs and other network resource managers.

VII. Conclusions and Future Work

This paper proposed a framework for network slice management and orchestration (akin, SliMANO) following the ETSI specifications. SliMANO is implemented as plug-in based system, able to operate with multiple MANO and VIM entities, providing a new level of abstraction to the deployment of network slices. An initial proof-of-concept implementation was evaluated and compared with OSM release 5 in terms of overall delay. Results showcased that SliMANO imposes a delay increase in slices with lower number of NSs. However, as the number of NSs increase the delay decreases, this behaviour shows an efficiency gain of SliMANO over OSM as the number of NSs increases.

Furthermore, results also shown that delays result from the NBI and are associated to SliMANO operating as an entity that is external to the orchestrator. Nonetheless, such delays are negligible in regards to the total duration of an orchestration operation, and are a result from the added flexibility the framework provides in comparison with other solutions, enabling the development of new slice mechanisms agnostic to the underlying network, VIM and MANO entities and procedures.

As future work, we will continue the development of remaining CRUD operations and develop more plugins/agents, in order to integrate other MANO frameworks, SDN controllers and RAN controllers, to address multiple types of network deployment scenarios. In addition, in terms of performance, different RPC framework will be considered to reduce the communication delay between services, and the use of a *no-SQL* database will be explored to assess its potential performance impact in persistence operations on the Core module.

[13]As shown in results, the major delay was imposed by HTTP REST interfaces of both the NBI and SBI to communicate with the other components (e.g. NFVO, SDN controller).

Acknowledgment

This work is funded by FCT/MEC through national funds under the project PTDC/EEI-TEL/30685/2017 and by the European Commission through the H2020 project 5GROWTH (Project ID 856709).

References

[1] "Technical specification group services and system aspects: System architecture for the 5g system," 3GPP, 06 2019, spec.: 23.501 version: 16.1.0.

[2] "NGMN 5G White Paper," NGMN, February 2015, version: 1.0. [Online]. Available: https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2015/NGMN_5G_White_Paper_V1_0.pdf

[3] N. ETSI, "Gs nfv-man 001 v1. 1.1 network function virtualisation (nfv); management and orchestration," 2014.

[4] "Study on management and orchestration of network slicing for next generation network," 3GPP, 01 2018, spec.: 28.801 version: 15.1.0.

[5] C. Parada, J. Bonnet, E. Fotopoulou, A. Zafeiropoulos, E. Kapassa, M. Touloupou, D. Kyriazis, R. Vilalta, R. Muñoz, R. Casellas *et al.*, "5gtango: A beyond-mano service platform," in *2018 European Conference on Networks and Communications (EuCNC)*. IEEE, 2018, pp. 26–30.

[6] A. Devlic, A. Hamidian, D. Liang, M. Eriksson, A. Consoli, and J. Lundstedt, "Nesmo: Network slicing management and orchestration framework," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2017, pp. 1202–1208.

[7] A. G. Dalla-Costa, L. Bondan, J. A. Wickboldt, C. B. Both, and L. Z. Granville, "Maestro: An nfv orchestrator for wireless environments aware of vnf internal compositions," in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2017, pp. 484–491.

[8] K. Katsalis, N. Nikaein, and A. Huang, "Jox: An event-driven orchestrator for 5g network slicing," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–9.

[9] F. Meneses, R. Silva, D. Santos, D. Corujo, and R. L. Aguiar, "An integration of slicing, nfv, and sdn for mobility management in corporate environments," *Transactions on Emerging Telecommunications Technologies*, p. e3615, e3615 ett.3615. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3615

[10] F. Meneses, R. Silva, D. Corujo, A. Neto, and R. L. Aguiar, "Dynamic network slice resources reconfiguration in heterogeneous mobility environments," *Internet Technology Letters*, p. e107. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/itl2.107

[11] "Osm release five technical overview," ETSI, January 2019, version: 1.0. [Online]. Available: https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseFIVE-FINAL.pdf