Towards a Real Deployment of Network Services Orchestration and Configuration Convergence Framework for 5G Network Slices

Ibrahim Afolabi, Miloud Bagaa, Walid Boumezer and Tarik Taleb

***Abstract*—A seamless interworking between network function virtualization (NFV) and software defined networking (SDN) to orchestrate network services for the 5G systems is very fundamental for network slice creation. The orchestration of large scale network slices across multiple administrative as well as technological domains with heterogeneous resources and a distributed form of slice management can benefit from harnessing existing NFV orchestration (NFVO) solutions. In this regard, this paper presents a network service orchestration and configuration convergence framework that is capable of providing large scale network slicing solution for 5G network operators. Using this framework, 5G network operators can orchestrate and configure network slices directly from their infrastructure and that of credible registered slice providers who have resources for the orchestration of only a subset of the overall network slice. The framework is equipped with mechanisms that allow a distributed form of slice configuration and management.**

***Index Terms*—5G, network slicing, end-to-end slices, NFV, SDN, network softwarization, Network service, and MANO.**

I. Introduction

THE 5G communication system aims to expand both the concept and scope of network connectivity services to cover wide varieties of use cases. These use cases will push the limits of communication technologies and bring onboard more stakeholders than the case of the 4G technology. By supporting vertical applications with varying and sometime opposing service requirements [1], the 5G system will provide a plethora of network services that will bring about interactions between 5G vertical application developers/industry, the 5G communication system providers and the 5G telecommunication infrastructure providers [2]. These vertical applications will be orchestrated and deployed over 5G network slices [3] as 5G-ready applications.

Network slicing, enabled through network function softwarization [5], is no longer new in the mobile telecommunications world. The network slicing concept which was first introduced by the NGMN alliance [9] has witnessed a widespread exploration by well-meaning standardization bodies such as the 3rd Generation Partnership Project (3GPP) [8], the International Telecommunication Union Telecommunication standard sector (ITU-T) [10] and the European Telecommunications Standards Institute (ETSI) [11]. Fundamentally driven by advancements in both the ETSI NFV Management and Orchestration (MANO) framework and the Software Defined Networking (SDN) technology, network slicing is today realizable over programmable shared virtualized resources running over commodity hardware server infrastructure[5].

Hitherto, network slices are orchestrated and deployed as connected virtual network functions (VNFs) running as services over shared abstracted resources used mainly for computation, storage and networking while ensuring certain level of isolation. These VNFs are hosted mostly on virtual machines or nowadays on containers with certain amount of dedicated virtual compute, storage and network links. For example, a mobile network slice will be composed of sub-slices of the radio access network, the core network and perhaps the transport network [4]. Therefore, such a slice should consist of PNF (physical network function)/VNF of the next generation Node B (gNB) and VNFs of the core network e.g., the Access and Mobility Management Function (AMF), Session Management Function (SMF), User Plane Function (UPF), and Unified Data Repository (UDR) [7], in order to deliver a complete mobile network functionality while satisfying certain predefined requirements [1].

The orchestration of mobile network slices could already be realized thanks to existing NFVO frameworks such as OSM, OpenBaton, 5G-Norma, and 5G-EX. Virtually all of them are implemented following well known standards such as those of ETSI and 3GPP. Moreover, these NFVO platforms have varying levels of maturity based on their supported slicing enabling technologies and functions. While each of the existing NFVO studies has different identified weaknesses as well as strengths as discussed in [13] in terms of supported functions, their collective strengths could be leveraged to complement for their weaknesses. One of the identified weaknesses across most, if not all, of the existing frameworks is the lack of an abstraction support for converging the orchestration and configuration of network services in a seamless manner. This makes them rigid for a specified domain or technology and may have a negative impact on the network elasticity. In contrast to these solutions, we have suggested a service abstraction that is recognized as an essential component in any orchestration framework that offers a multi-domain slice orchestration support, similar to the *service conductor concept described* in [13]. In this work, we have implemented an orchestration and configuration convergence (OCC) framework that provides the functionalities needed to decompose, coordinate and carefully interpret multi-

Ibrahim Afolabi, Miloud Bagaa, Walid Boumezer and Tarik Taleb are with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, Finland (e-mails: {firstname.lastname}@aalto.fi). Tarik Taleb is also with the Faculty of Information Technology and Electrical Engineering, Oulu University, and with the Department of Computer and Information Security, Sejong University, Seoul 05006, South Korea.

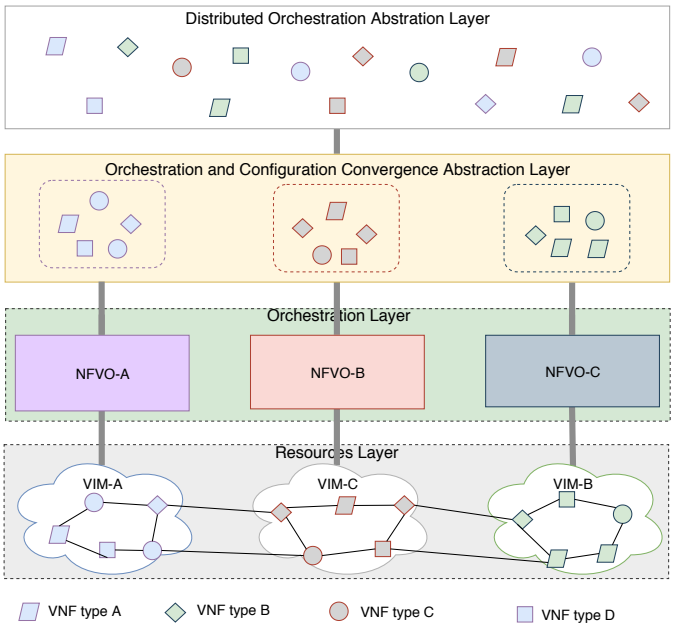


Fig. 1. High-level Architecture of the Framework.

domain slice requests into domain-specific units that will be orchestrated and managed by an administrative domain, and the corresponding resources composition across the different identified domains.

In this light, this paper presents a network service orchestration and configuration convergence framework that provides an end-to-end network slicing solution for 5G network operators. The solution leverages existing network service orchestrators to provision multi-domain network slices by enabling an orchestration convergence platform in which network services could be orchestrated across variant heterogeneous network components including diverse VIM and edge resources and RAN controllers. As much as the framework provides mechanisms and APIs to independently interact with different instances of the supported orchestrators in a distributed manner, it also provides RESTful API modules to configure and manage the various VNF instances separately through third party solutions.

The rest of this paper is organized as follows. Section II discusses the related work explaining the background of this work. Section III presents an overview of the network services orchestration convergence and configuration framework. In Section IV, we present the interactions between the building blocks of the orchestration convergence and configuration framework. Section V reveals the different back-end server processes that make the system components work in cohesion and as an integrated platform. Also, this section presents a multi-domain slice orchestration and dynamic configuration use case. In Section VI, we present some performance evaluation of the proposed framework. Finally, we draw concluding remarks in Section VII.

II. Related work and background

With arrays of promises for vertical industries, the network slicing concept has witnessed a wide spread and tremendous exploration in different levels of studies among the next generation systems market players. Leveraging both the NFV and SDN technologies, a wide range of suggestions is available on how to enable network slice orchestration and management by considering different types of architectures including most notably the ETSI MANO architecture and also benefit from the recommendations of the 3GPP standards. For example, the 3GPP SA5 working group in [16] introduces the concept of network slice as a service (NSaaS) between different players i.e., a communication service provider (CSP) and a communication service customer (CSC), and also presents service requirements that should be taken into account when providing such services, considering different technological domains, e.g., 5G RAN and 5G CN respectively, while maintaining a high-level approach in defining different aspects of network slice management, namely, Preparation, Commissioning, Operation and Decommissioning.

Moreover, the work in [17] provides definitions and details of management services and functions for network slices, wherein the 3GPP recognizes the NFV-MANO orchestrator for enabling the deployment of network slice subnet instances (NSSI) and providing standard management interfaces to support its utilization. Although, the ETSI MANO paradigm was conceived as an orchestration architecture for orchestrating network services using VNFs, it was however, not specially designed for enabling end-to-end network slice orchestration [6]. In addition, the application of the architecture is mostly suitable for orchestrating single domain network slices. However, the need to support multi-domain slicing has been identified and significant progress has been recorded. In fact, the work in [15] provides detailed information on the different use cases for the need to practically enable multi site slicing which are mainly motivated by customer needs and innovative business ideas.

Nonetheless, with the monumental amount of research and implementation efforts that have been dedicated towards developing the ETSI MANO concept from both the academia and open source communities, the development of the Open Source MANO (OSM[1]), does not come as a surprise. OSM is perhaps one of the most notable open source network services orchestration solutions available today. Another prominent open source orchestration platform that provides an extensive implementation of the ETSI MANO concept is the Openbaton project. In addition, other open source network service orchestration and automation platforms which are not expressly in compliance with the ETSI NFV reference architecture are the ONAP[2] (formerly known as OPEN-O) and Central Office Re-architected as a Datacenter (CORD[3]) projects. While the ONAP project is not entirely compliant with ETSI, its main MANO components such as the VNFM and VIM can be mapped to the ETSI NFV reference model.

[1]https://osm.etsi.org/
[2]https://www.onap.org/
[3]https://opencord.org/

The CORD architecture, on the other hand, is very SDN-centric in its implementation, very much following the approach of OpenFlow by adopting the ONOS framework. With a dedicated NFVO component, called XOS, most of the architectural elements of the CORD could be technically aligned with the ETSI-defined reference model. Perhaps, this is only by coincidence, the objectives of the CORD projects extend beyond the NFV network service provisioning. Similar to the CORD project, the ONAP project also readily provides fundamental support for SDN-based networking.

III. Overview of the Network Services Orchestration Convergence and Configuration Framework

A. Overview of the High-level Architecture

The envisioned network service deployment convergence and configuration framework is aimed at providing two essential functionalities. As the name suggests, it aims to 1) serve as a convergence platform for the orchestration of network services and 2) to provide a robust approach for the configuration of these services. The framework is designed in such a way to support multiple orchestration platforms by providing the necessary mechanisms to manage and interact with them seamlessly. As a result, the described framework herein is very suitable for the orchestration of network slices with varying level of service characteristics and sizes, especially, those that span across multiple domains or sites as the case maybe. Taking advantage of an orchestration and configuration convergence framework such as this, different slicing orchestration scenarios can be actualized from a single unifying platform considering the complexities that may be involved in instantiating the different subnet/sub-slices of a network slice. This framework is designed primarily as a generic platform for enabling the orchestration and configuration of network services including multi-domain 5G network slices, and the actual composition of the slices is already discussed in details in [4].

To this end, our proposed architecture takes into consideration four main layers as shown in Fig. 1: *Distributed Orchestration Abstraction Layer (DOAL)*, *Orchestration and Configuration Convergence Abstraction Layer (OCCAL)*, *Orchestration Layer (OL)* and *Resources Layer (RL)*.

- **DOAL:** is the layer responsible for processing multi-cloud domain service orchestration request. It provides an abstraction interface for the submission, preprocessing, sorting, decomposition, recomposition and aggregation of VNFs that constitute sub-network service, which is intended to be orchestrated from either one or a set of NFVOs. Basically, a sunny day scenario is one in which the DOAL receives an orchestration request of a network service that spans across multiple administrative as well as technological domains and regions in a standard known format (e.g., TOSCA, YAML, and JSON). The DOAL validates and authenticates the received request, and then parses and aggregates all of the essential information contained in the request. The network service request would include both the instantiation and configuration of a combination of different types of VNFs that are available from the different registered NFVOs. The DOAL handles the received request and then processes it by sorting and mapping the different sub-tasks of the requests to the corresponding NFVOs and powers them using resources from the VIMs attached to each of the NFVOs. The DOAL tightly coordinates with the OCCAL in enabling the orchestration of the received request while reflecting the complex characterization of the service request, traffic steering profile and convoluted links between the respective VNFs after simplifying and decomposing the complex request. After the variant VNFs have been sorted and recomposed in the DOAL, the VNFs belonging to the same NFVOs are then passed down to the OCCAL for fulfilling the orchestration request.
- **OCCAL:** is directly responsible for registering and managing multiple NFVOs. This layer provides the necessary APIs needed to effectively communicate with the registered NFVOs and their respective VIM(s) on its south-bound interfaces. Similarly, on its north-bound interface, it offers RESTful interface through which orchestration as well as configuration instructions are handed down from the DOAL layer siting above it. This layer directly interacts with the DOAL in ensuring effective orchestration of the services that constitute the requested slice creation. After the DOAL layer preprocesses an orchestration request, the sorted, decomposed and aggregated requests are mapped directly to the variant registered NFVOs that are responsible for instantiating and configuring the different VNFs forwarded from the OCCAL as a preprocessed orchestration and configuration instruction. At this stage in the OCCAL, the convoluted combination of network slice request that was received by the DOAL is now simplified, decomposed and recomposed into multiple sub-network service orchestration and configuration requests, whereby, each sub-network request is then mapped to a clearly identified NFVO. Each of the intended NFVOs is guaranteed to have within its management the specified VNF and NS descriptors that are indicated in the received orchestration instruction based on timely updates that are published to the DOAL layer. In this way, the DOAL is always aware of any changes that might have taken place in the orchestration layer in terms of supported services or available resources through the OCCAL, which further allows the OCCAL to support service recomposition and suggestion. The service recomposition and suggestion may be necessary in a situation whereby certain resources become unavailable shortly after a request has just been accepted based on the previous resource update.
- **OL:** is directly involved with the instantiation and configuration of the individual sub-network service that is sorted and forwarded from the OCCAL layer siting above it. This layer directly communicates in parallel with each of the identified registered NFVOs. In this way, the system is able to provide fine management and adequate control of the individual sub-network service instances. As presented in Fig 1, the OL primarily provides RESTful API for processing orchestration and configuration requests that are received from OCCAL and also in a transparent

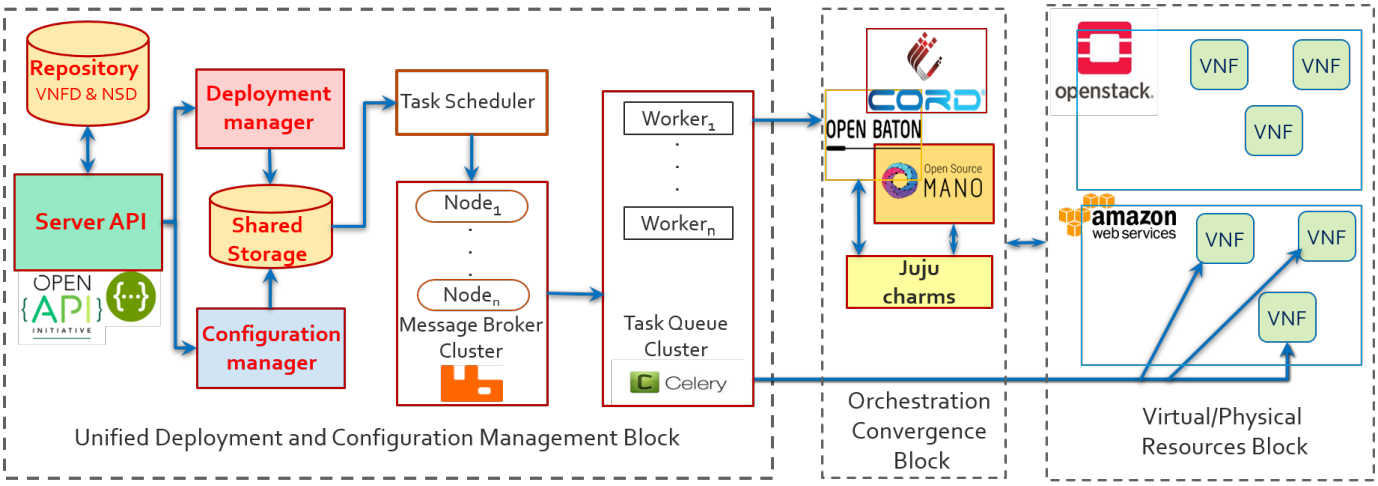


Fig. 2. Implementation architecture of the framework.

way, for controlling the resources of the VIMs that are registered to each of the NFVOs. It should be noted that the links between the NFVOs and the VIMs are meant to reflect a one-to-many relationship, which implies that one NFVO could control resources from multiple VIMs but has been left so for the sake of simplicity in the presentation of Fig. 1. This layer manages and effectively controls the instantiation and resources provisioning for the network services, which are requested from OCCAL.

- **RL:** is an abstraction of all the resources that are under the control of each of the VIMs that is connected to it. The resources are either virtual or physical resources such as the compute, storage and networking resources that are used in the orchestration of VNFs.

IV. Implementation Architecture Leveraging Existing Enabling Technologies

This section presents the details of the implementation and deployment of the unified platform based on the high-level architecture presented in Fig. 1 and described in Section III. In Subsection IV-A, we shall discuss the details of our implementation, which follows the approach introduced in the high-level architecture as described above. Here, we will present our deployment architecture, describing each of its interacting components and classifying them into enabling blocks in relation to each layer of the high-level architecture as shown in Fig. 1.

A. Interactions between the building blocks of the framework

In this sub-section, the details of the implementation of this framework are presented and the interactions between its building blocks are discussed. As shown in Fig. 2, the architecture is composed of three essential and integral parts, the Unified Deployment and Configuration Management Block (UDCMB), Orchestration Convergence Block (OCB) and Virtual/Physical Resources Block (V/PRB). Each of these building blocks can be mapped directly to each of the layers described in Fig. 1 in the following way: UDCMB to DOAL + OCCAL, OCB to OL and finally, V/PRB to RL. The main features and components of each of the building blocks of this implementation framework, which is a major contribution of this paper, are described in the remaining part of this subsection.

1) Unified Deployment and Configuration Management Block (UDCMB): is a major building block of the implementation architecture shown in Fig. 2. This important building block is an encapsulation of the essential software modules, namely, the **Server API**, **VNFD & NSD Repository**, **Deployment manager**, **Shared Storage**, **Configuration manager**, **Task Scheduler**, **Message Broker Cluster**, and **Task Queue Cluster** that are developed in order to enable a unification platform for the orchestration and management of network slices above and beyond a single administrative domain. Using these important software modules, the UDCMB provides an integrated unified front for the orchestration and configuration of variant network services with support for a number of well-known existing MANO-compliant orchestration platforms.

Server API: The Server API module is implemented following the OpenAPI specification, originally known as Swagger API[4] Specification. It is a RESTful API compliant software tool that is used for producing machine-readable interface files that can be utilized to describe, produce, consume and visualize web services, especially from the front-end. The OpenAPI is used in order for the Server API to follow industry standard practice for RESTful design. Since this modules follows the RESTful implementation, it then implies that any RESTful compliant client can interact with the system seamlessly through the different API endpoints that are exposed to the end users. Using these API endpoints, different users can request and consume services from the UDCMB. Thanks to the use of Swagger, the developed framework is orthogonal with any existing system, and allows interoperability with other components developed to extend its functionalities.

VNFD & NSD Repository: This software module serves as a storage repository for the uploading and retrieving of

[4]https://swagger.io/specification/

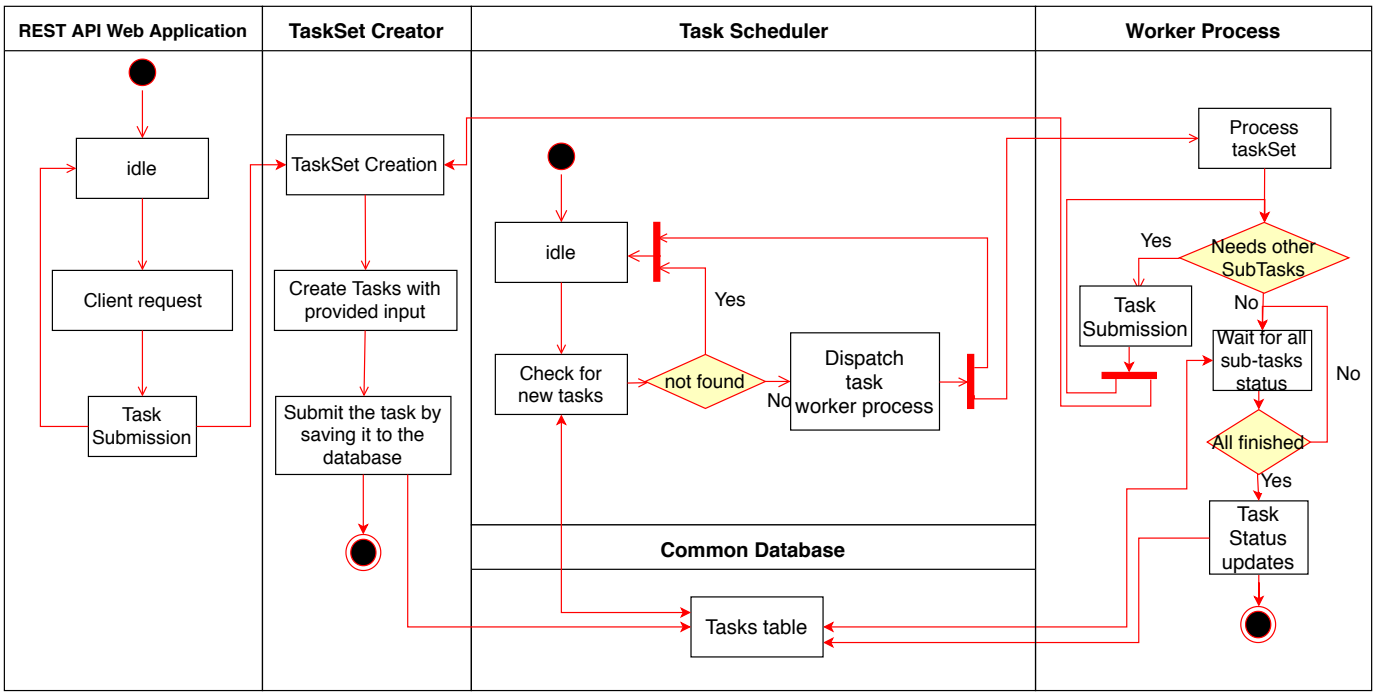


Fig. 3. Back-end task submission and execution architecture.

Virtual Network Function Descriptors (VNFD) and Network Service Descriptors (NSD) that are used in the deployment and orchestration of VNFs and network services. Using this repository, the UDCMB is able to easily retrieve the corresponding VNFDs and NSDs needed to orchestrate a VNF or NS on the appropriate indicated orchestration platform if such VNFD/NSD is not already available on the orchestrator.

Deployment manager: The deployment manager is an essential software module that handles the main tasks of VNF/NS orchestration requests and reply from and to the respective supported orchestrators. This manager effectively unifies the underpinning orchestration platforms by presenting them all as one. It interacts directly with the OCB (Orchestration Convergence Block). The deployment manager interprets requests from network service users and resolves any form of conflicts that might arise from the NS's orchestration requests, especially, those involving the orchestration of VNFs from multiple cloud platforms that are managed by different orchestrators. Consider a VNF deployment scenario where VNFs X and Y are orchestrated from clouds A and B, respectively. While cloud A is managed by orchestrator α, cloud B is managed by another orchestrator β, for reasons best known to the network operators. The deployment manager is able to account for the independent orchestration of these VNFs between the two orchestrators.

Shared Storage: The shared storage is centrally position between the Deployment and Configuration managers in order to provide a common storage functionality, which allows software modules to submit both deployment and configuration task requests received from the Server API. In this way, their implementation can be kept simple, modular and focused on one functionality as can be deduced from their respective names. In addition, this database allows the system to easily keep track of task execution state.

Configuration manager: The configuration manager simply manages the configuration of the instances of network functions that together make up a network service. The VNF instances that belong to a network service are configured to exhibit certain expected behaviour using this software module. This software module acts as a generic VNF configuration tool that is capable of carrying out day1 and day2 configurations, similar to the VNF Configuration and Abstraction (VCA) in the OSM orchestrator's configuration component, Juju charm[5]. In our implementation, we have deployed the Ansible[6] configuration automation software tool. Moreover, VNF configuration can be also accomplished using Juju charms. However, it should be noted that the communication between OSM and its Juju component is not always established properly.

Task scheduler: The task scheduler entity is needed in order to simplify the functionality of the deployment and configuration managers by decoupling the actual managerial roles from the deployment and configuration tasks and subtasks. In this way, the task scheduler sees the list of all submitted tasks in the configuration and deployment tables found in the shared storage, simply as tasks that have to be executed and it does so through an identified message broker node.

Message Broker Cluster: The message broker plays a vital role in the entire system. It is in fact the *integration fabric* as per [14] occupying a central and essential role in enabling the seamless interworking and communication of messages between other building blocks of the framework.

[5]https://jaas.ai/
[6]https://www.ansible.com/

Implemented using the advanced message queuing protocol, the message broker is deployed as the central gel gluing the entire modules together and enables them to work coherently. In our implementation, we have deployed the lightweight RabbitMQ[7] message-broker to manage the task queues emitted by the Task Queue module.

Task Queue Cluster: The task queue is responsible for managing the execution of different set of tasks and coordinating among them. It is a module implemented using the Celery[8] distributed task queue framework that controls the asynchronous tasks execution in real-time. It creates tasks execution units and executes them concurrently on working servers using the multiprocessing approach. Using this approach, the UDCMB is able to manage deployment and configuration of network services smoothly and coherently in a distributed fashion. More details about the task queue module will be presented in subsequent sections.

2) Orchestration Convergence Block (OCB): is the abstraction module responsible for registering and directly interacting with the different supported NFV orchestration systems. This software module implements all the necessary RESTful API endpoints needed to seamlessly connect to the registered orchestrators. They are also aware of the respective cloud domains they are associated with. This block enables our framework to operate as a genuine multi-domain network service orchestrator with the capability to orchestrate network services across multiple administrative as well as technological domains with plugins for concurrently operating variant existing MANO orchestrators [13], while also coordinating and managing their respective resources.

Supported MANO orchestrators: Thanks to the use of the industrial standard OpenAPI framework, we are able to create RESTful API endpoints that allow for the registration and operation of multiple MANO orchestrators, yet with a clear distinction of their individual orchestration operations. At the moment, our framework supports the deployment of open source solutions such as the OSM and OpenBaton[9] orchestrators. We are presently widening the scope to provide support for a more mobile network operator tailored solution such as the CORD project.

3) Virtual/Physical Resources Block (V/PRB): is a layer/block that represents the cloud infrastructure and its corresponding resources. Precisely, they are the virtualized Infrastructure Managers (VIMs) that are directly associated with the orchestration convergence module. One or more VIMs could be associated with one or more orchestrators and as such give the orchestrators the possibility to instantiate VNFs on multiple VIMs independently. This block exposes the resources of the orchestrators under the control of the OCB to the framework. In this way, the framework is aware and able to adequately provision the necessary resources needed to power a network slice already at the point of making the requests. This singular opportunity makes it viable for an orchestrator to orchestrate VNFs across multiple cloud domains.

[7]https://www.rabbitmq.com/
[8]http://www.celeryproject.org/
[9]https://openbaton.github.io/index.html

V. Back-end server processes and use case scenario

In this section, we introduce the intricacies of the back-end processes that form the foundation and supporting structure for the implementation of our orchestration framework. Each of the discussed software modules is cohesively connected and consistently functioning as a whole, thanks to the sophisticated Django RESTful framework that we have deployed in the back-end. Together with the adoption of a task and queue management paradigm, the framework is able to perform asynchronous tasks execution procedure flawlessly. It is a common knowledge that the orchestration of a multi-domain network slice usually involves the execution of a set of tasks (instantiation and configuration of VNFs) carried out both in series and in parallel depending on the slice requirements. Monitoring the status of the task execution process is of high importance, considering the fact that success for one may imply success for others and vice versa. Therefore, while the instantiation of the VNFs may be independent, their configuration may be dependent on one another.

A. Intricacies of the Asynchronous Tasks Execution in the Server Back-end

To maximize the utilization of the system, it was essential to incorporate a task management abstraction mechanism in the UDCMB block as depicted in Fig. 2. The UDCMB has been designed in a way that allows for:

- Separation of concern: The UDCMB is composed of loosely coupled sub-blocks that allow, for instance, the introduction of high availability and replication for most sub-components; a feature which improves the overall resiliency and fault tolerance. Each sub-block handles a subset of the overall functionalities; for instance, the API servers solely handle interactions with the platform's users and then relate their actions to the right sub-blocks, namely the Deployment and Configuration Managers. Whereas the managers are responsible for the interpretation of the said actions to schedulable tasks
- Parallel execution: The schedulable tasks are executed in parallel by a pool of workers and when required, the message broker component is also used as a semaphore for concurrent access to certain resources.
- Better feedback: The task execution mechanism allows for better feedback about the status of user actions, for instance, how long they take, whether they are successfully executed or not, and detailed logs for the whole process can be provided.
- Fault tolerance: The task execution process is able to reschedule recoverable failed tasks and failures related to the workers themselves.

The basic unit of scheduling is called a task, which could be for instance deployment or configuration of an NSI, configuration of a single VNFI, deletions, updates and so on. A grouping of one or more related tasks is called a task-set which is created by the Deployment and Configuration managers by interpreting user actions. Certain resource-consuming tasks/task-sets can generate other sub-tasks. The basic workflow is depicted in Fig. 3 and follows these steps:

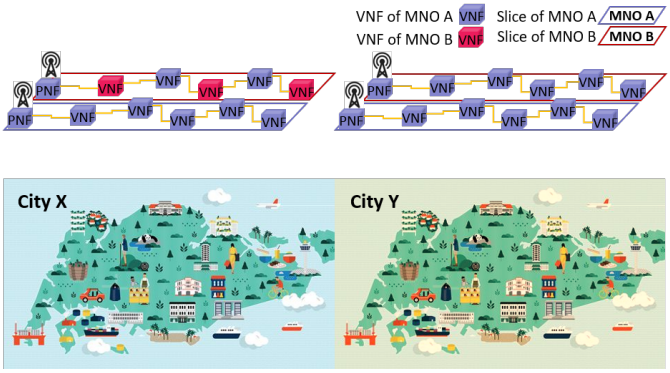


Fig. 4. Multi-domain mobile network slice orchestration use case.

- The platform subscriber consumes a server API (REST API Web Application in Fig. 3) endpoint such as: uploading an NSD or VNFD, instantiating an NS, configuring an NSI, or configuring a specific VNFI
- Depending on the type of the user-triggered action, either the Deployment or Configuration manager (TaskSet Creator in Fig. 3) would create the necessary tasks definition (task-set) and populate a shared storage with the necessary metadata for the execution of the tasks.
- A task scheduler that polls the shared storage detects new tasks and submits them for execution on an appropriate worker based on some predetermined criteria such as worker load and task-set dependencies.
- A schedulable task is submitted for execution through a message broker
- Task execution is finally handled by worker. Depending on the type of the task, execution could include interaction with the OCB and/or V/PRB.

B. Use case1: Multi-domain mobile network slice orchestration and dynamic configuration

The orchestration of network slices ordinarily involves the instantiation and configuration of multiple VNFs (i.e., consisting of mainly core network functions) as well as the reservation of network resources of PNFs, such as the functions of the RAN and its configuration within a single administrative domain with a defined set of characteristics. Already such an orchestration operation would span across multiple technological domains. When the functionality is extended further to cover a range of different cloud administrative domains involving multiple orchestration systems, the term multi-domain orchestration is introduced. In other words, multi-domain mobile network slice orchestration involves the process of instantiating and configuring VNFs/PNFs of the mobile network across multiple technological as well as administrative domains [13].

Let's consider two mobile network operators (MNO)s, namely MNO A and B, who are in a mutual network service provisioning agreement for their respective customers in cities A and B, as depicted in Fig. 4. The envisioned scenario is such that MNO A has a better coverage in cities A and B than MNO B. However, due to the planned hosting of a major event, such as a soccer tournament in cities A and B, MNO B is aware of the fact that a potential number of its subscribers will be visiting cities A and B from other cities and will need a better coverage in both cities during the period of the event. MNO B can serve its customers in city X with its network capacity, but has a poor coverage in city Y. As a result, there are a handful of options available to MNO B to provision coverage for its subscribers during the soccer event. To temporarily cater for the needs of the additional subscribers of MNO A that will be also visiting cities A and B from other cities during the event, MNO A intends to deploy network slices. So, the options are: (1) for MNO A to orchestrate network slices for both its subscribers and MNO B's subscribers during the event, whereby, all the resources (VNFs and PNFs) of the slices belong to MNO A, (2) for MNO A to orchestrate network slices for both its subscribers and MNO B's subscribers during the event, whereby, only sub-slices are orchestrated from MNO A's infrastructure.

With the above scenario, it turns out that both options could be deployed separately in the two cities for MNO B. Since MNO B has extra capacity to serve more customers in city X, then, MNO B's slice for city X could be orchestrated partly from MNO B's and MNO A's VNFs and PNFs respectively, possibly from multiple cloud domains. This is while MNO A will orchestrate the slices for the event in city Y entirely from its own network infrastructure. Hence, at least three orchestrators should be managed on the platform: two from MNO A (for both cities A and B) and one from MNO B (only for city X). Therefore, MNO A could deploy our framework to manage and coordinate the orchestration activities of each of their individual orchestrators. Basically, MNO A begins by registering the two orchestrators on our orchestration convergence platform. Then, it initiates the orchestration of slices for both cities A and B to serve its own subscribers. Afterwards, MNO A simultaneously orchestrates the slice for MNO B in city X that will be composed of resources from both MNO B and A. Finally, MNO A orchestrates another slice for MNO B in city Y entirely from its own resources based on their agreement. All of these orchestration procedures could be done in parallel considering the fact that the event would often start and end at the same time.

C. Use case2: Dynamic orchestration, configuration and auto-scaling of a streaming service

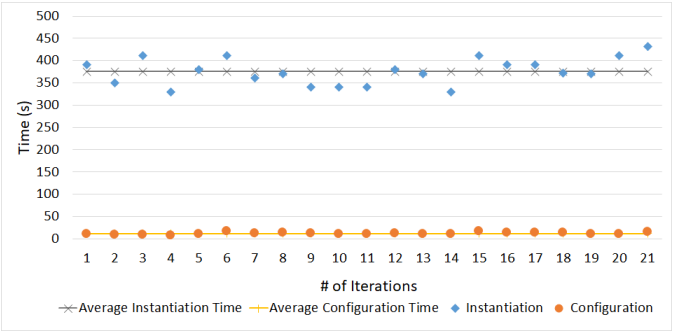
Streaming services are usually a part of content delivery networks (CDN), which are primarily responsible for delivering video contents to end users or viewers. Being able to orchestrate video streaming servers across multiple administrative domains and then provide suitable and customized mobile network slices that will enable the viewers to connect to them and watch the videos in a seamless and glitch-free manner is not a straighforward solution. Such type of a network slice, known as enhanced mobile broadband (eMBB) slice, is not only characterized by large size of bandwidth but also reduced amount of delay to prevent lag in the streaming.

It turns out that not only could our platform orchestrate this kind of use case scenario, but could be also used to

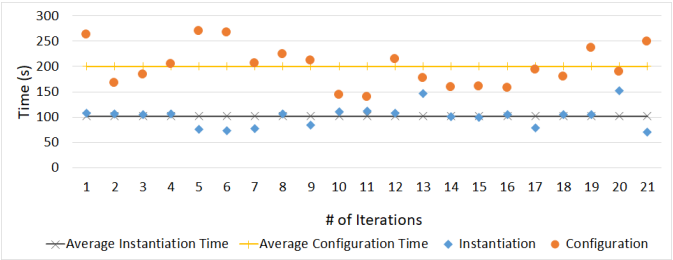
dynamically configure each of the VNFs that constitute both the streaming service as well as the underlying mobile network slice needed to view the video streams. In addition, it provides functionalities for the scaling of the streaming service. This implies that in the event that the number of views of the video streams increases to an extent whereby the number of streaming servers can no longer provide a smooth streaming experience for the viewers, the orchestration convergence framework can dynamically scale the streaming service, thereby, increasing the number of streaming servers with a load balancer. The scaling is achieved in such a seamless fashion that the viewers of the video that is being streamed would not notice any changes, any service disruption.

Deploying our framework would afford the network provider the opportunity to orchestrate both the mobile network service and streaming service slices in parallel and from dedicated independent NFVOs. This possibility would not only increase the speed at which the services can be deployed but would also enable an efficient use of the underpinning resources. In addition, this concept could serve as a real life technology deployment towards enabling the orchestration of a real user plane function (UPF) of the 5G core network. This scenario could be used to drive the enhancement of the orchestration of services closer to the network edge, in which case, a data plane function such as an application function (AF, e.g., a video streamer) could be orchestrated and placed within the network. This network enhancement would go along way in improving the quality of experience of the end users.

VI. Deployment and Evaluation



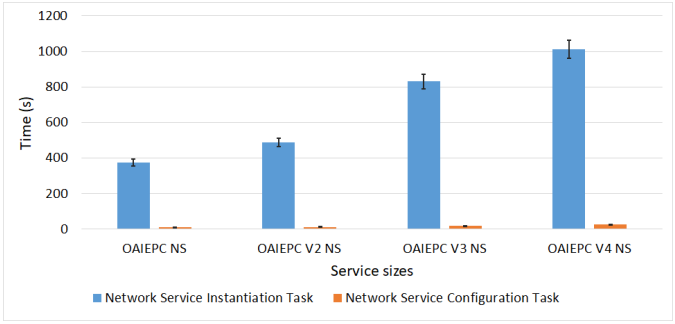
(a) Ave. time (sec) of instantiation and configuration of a complete OAI EPC.



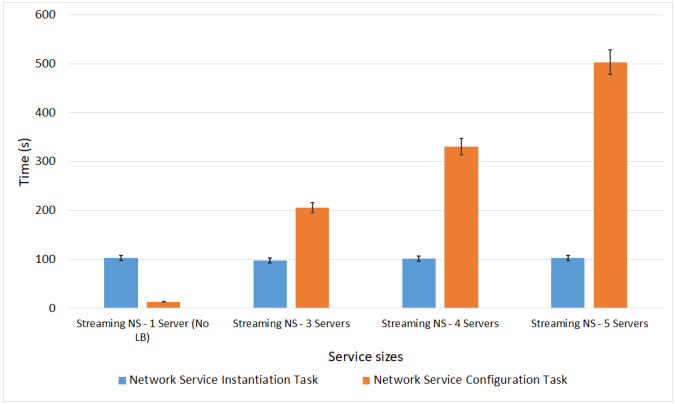
(b) Ave. time (sec) of instantiation and configuration of a video streaming service including a load balancer.

Fig. 5. Experiment results in time series.

In this section, we present the end-to-end performance evaluation of the framework. In this end-to-end evaluation of the system, our tests have been based mainly on the use case scenarios we have discussed in the previous section. Basically, we have carried out the test with a major focus on how the system performs in terms of the instantiation and configuration of network services. With that in mind, we orchestrated different sizes of mobile network slices and configured them. Each mobile network slice size was orchestrated multiple times and the average time taken to instantiate and configure each was calculated. The sizes of the network slices are determined by the different combinations of the constituent VNFs of the network slice. During our system evaluation experiments, we have considered the instantiation and configuration of two different services, namely the mobile network service and the video streaming service, respectively.



(a) Ave. time (sec) of instantiation and configuration of different compositions of a complete OAI EPC network service.



(b) Ave. time (sec) of instantiation and configuration of different compositions of a video streaming service including a load balancer.

Fig. 6. Experiment results of each step.

For the network service orchestration, we have considered the Open Air Interface mobile network 4G solution, which consists mainly of the Mobility Management Entity (MME), Serving-Packet Data Network Gateway (S-PGW) and the Home Subscriber Server (HSS). The streaming service on the other hand is made up of Nginx software solution, configured both as a streaming server and a load balancer. The sizes of both services considered are determined by the number of each of the VNFs that constitute each service that is orchestrated. For both services, we have made comparison of four different sizes of the same service. So, for example, in the case of the streaming service, we have orchestrated increasing number of streaming servers with a load balancer and compared how long

it took to complete both the instantiation and configuration of each service size.

As presented in Fig. 5(a), it took an average of about 375sec to instantiate a complete OAI 4G mobile network service and an approximate average of 12sec to configure the service. Both the instantiation and configuration procedures were repeated a total of 21 times. This experiment reveals that the time taken in the service instantiation phase for the orchestration of a mobile network service is longer than the time taken in the configuration phase. Conversely, in the case of the streaming service orchestration as shown in Fig. 5(b), the average time taken during the instantiation phase is an approximate 100sec, while the configuration phase took an average of almost double the time for a streaming service consisting of 3 servers (a load balancer and 2 streamers). Obviously, more time is lost in the configuration phase for the orchestration of a streaming service, which is totally different from a mobile network service.

Similarly, as presented in Fig. 6(a), for mobile network service orchestration, the instantiation phase clearly takes significantly more time than the configuration time with respect to an increase in their different sizes. For example, for orchestrating the OAIEPC V4 NS, which consists of seven VNFs instances, it takes a little above 1000sec to instantiate them. Whereas, to configure all seven VNFs, it takes as little as 27sec. Moreover, as presented in Fig. 6(b), for orchestrating the streaming NS-5 Servers, it takes around 100sec to instantiate them but about 500sec to configure all of them.

These test results have revealed that while it is generally expected that the instantiation of network services will consume more time than its configuration, especially, for those orchestrated from uncached VNF images [4], this expectation can not be made general across all types of VNFs and their resulting network services. This implies that while the instantiation of some types of network services may take longer time, others may not necessarily exhibit the same behaviour depending on their VNF composition and image types. Similarly, in the case of network service configuration, while some services may take longer duration to be fully configured, others may take less depending on size and number of VNFs involved in the configuration phase. The configuration duration is often determined by the number of I/O writing that has to be effected and the number of VNFs whose files have to be updated during the configuration phase. So, in the case of the streaming service, the configuration time varied quite noticeably than in the case of the OAI EPC network service, since the number of files and parameters that are updated are much more compared to that of the OAI EPC network service.

VII. Conclusion

In this work, we have presented our implementation of a convergence platform that is capable of harnessing the resources and functionalities of different existing orchestration systems, especially those which are ETSI MANO compliant. Using this framework, potential network operators could benefit from the comprehensiveness of its functionalities in orchestrating multi-domain network slices from a mixture of network resources belonging to multiple administrative domains. This framework is robust and implemented in such a way that the configuration and instantiation tasks are executed using different dedicated processes and the status of each task could be tracked for a fine-grained monitoring and control. Finally, we present an evaluation of the system in terms of the average time taken to complete both network service instantiation and configuration. The evaluation results have provided meaningful insights that could help us understand better the instantiation and configuration behaviour of different services during their respective orchestration. Based on the results, it is understood that network services should be treated individually in terms of the time it takes to instantiate and configure each service.

Acknowledgment

This work is partially supported by the European Union's Horizon 2020 research and innovation program under the MATILDA project with grant agreement No. 761898. The work is also funded by the Academy of Finland Project CSN - under Grant Agreement 311654 and the 6Genesis project under Grant No. 318927, respectively.

References

[1] *5G; Service requirements for next generation new services and markets*, (3GPP TS 22.261 version 15.5.0 Release 15), July 2018.

[2] *MATILDA; Deliverable D1.1, MATILDA Framework and reference architecture*, Dec. 2017.

[3] I. Afolabi, A. Ksentini, M. Bagaa, T. Taleb, M. Corici, and A. Nakao, Towards 5G Network Slicing over Multiple-Domains, in IEICE Trans. on Communications, Vol. E100.B, No. 11, Nov. 2017.

[4] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa and A. Ksentini, "Network Slicing-Based Customization of 5G Mobile Services," in IEEE Network, vol. 33, no. 5, pp. 134-141, Sept.-Oct. 2019.

[5] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, Network Slicing Softwarization: A Survey on Principles, Enabling Technologies & Solutions, in IEEE Communications Surveys Tutorials, 3rd Quarter, Vol. 20, No. 3, Mar. 2018, pp. 2429-2453.

[6] *Network Functions Virtualization (NFV); Management and Orchestration*, (ETSI GS NFV-MAN version 1.1.1), Dec. 2014.

[7] *5G; System Architecture for the 5G System*, (3GPP TS 23.501 version 15.2.0 Release 15), June, 2018.

[8] *Study on management and orchestration of network slicing for next generation network*, 3GPP Technical Specification TR 28.801, Sep. 2017.

[9] NGMN Alliance: "Description of Network Slicing Concept", January 2016.

[10] Network slice ITU-T Y.3100

[11] Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework, Dec., 2017.

[12] D. Luong, H. Thieu, A. Outtagarts and B. Mongazon-Cazavet, "Telecom microservices orchestration," 2017 IEEE Conference on Network Softwarization (NetSoft), Bologna, 2017, pp. 1-2.

[13] T. Taleb, I. Afolabi, K. Samdanis and F. Z. Yousaf, "On Multi-Domain Network Slicing Orchestration Architecture and Federated Resource Control," in IEEE Network, vol. 33, no. 5, pp. 242-252, Sept.-Oct. 2019.

[14] *Zero-touch network and Service Management(ZSM); Reference Architecture*, (ETSI GS ZSM 002 version 1.1.1 Release 15), Aug, 2019.

[15] *Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on Management and Connectivity for Multi-Site Services*, (ETSI GR NFV-IFA 022 version 3.1.1 Release 15), April, 2018.

[16] *Management and orchestration; Concepts, use cases and requirements*, (3GPP TS 28.530 version 16.1.0 Release 16), Dec, 2019.

[17] *Management and orchestration; Architecture framework*, (3GPP TS 28.533 version 15.4.0 Release 15), Mar, 2020.

Ibrahim Afolabi obtained his Bachelors degree from VAMK University of Applied Sciences, Vaasa, Finland, in 2013 and his Masters degree from the School of Electrical Engineering, Aalto University, Finland in 2017. He is presently pursuing his doctoral degree at the same university where he obtained his Masters degree from and his research interests include Network Slicing, cloud computing, , machine learning, MEC, network softwerization, NFV, SDN, and dynamic network resource allocation.

Miloud Bagaa received the bachelors, masters, and Ph.D. degrees from the University of Science and Technology Houari Boumediene Algiers, Algeria, in 2005, 2008, and 2014, respectively. He is currently a Senior Researcher with the Communications and Networking Department, Aalto University. His research interests include wireless sensor networks, the Internet of Things, 5G wireless communication, security, and networking modeling.

Walid Boumezer received the B.Sc. degree in Telecommunications and Networks Engineering in 2016 and the M.Sc. degree in Information Systems Security in 2018 from The University of Science and Technology Houari Boumediene (USTHB) in Algiers, Algeria. He is currently persuing a D.Sc. degree in Communications and Networking with the school of Electrical Engineering, Aalto University, Finland. His research interests include Network Softwarization, Content Distribution Networks, Cloud Computing, Machine learning and Network Slicing.

Tarik Taleb received the B.E. degree (with distinction) in information engineering in 2001, and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2003, and 2005, respectively. He is currently a Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. He is the founder and the Director of the MOSA!C Lab. He is the Guest Editor-in-Chief for the IEEE JSAC series on network Softwarization and enablers.