A Framework for Orchestration and Federation of 5G Services in a Multi-Domain Scenario

L. Valcarenghi[1], B. Martini[1,2]
[1]Scuola Superiore Sant'Anna, Pisa, Italy
[2]CNIT, Pisa, Italy

J. Mangues-Bafalluy, R. Martinez, J. Baranda, I. Pascual
Centre Tecnològic de Telecomunicacions de Catalunya CTTC/CERCA, Spain

K. Antevski, C. J. Bernardos
Universidad Carlos III de Madrid, Spain

A. Ksentini
EURECOM, France

Xi Li
NEC Laboratories Europe, Germany
Xi.Li@neclab.eu

G. Landi, M. Capitani
Nextworks, Italy

C. F. Chiasserini, F. Malandrino
Politecnico di Torino, Italy

D. Andrushko
KhNURE, Ukraine

Konstantin Tomakh
Mirantis, Ukraine

ABSTRACT

This paper presents the design of the 5GT Service Orchestrator (SO), which is one of the key components of the 5G-TRANSFORMER (5GT) system for the deployment of vertical services. Depending on the requests from verticals, the 5GT-SO offers service or resource orchestration and federation. These functions include all tasks related to coordinating and providing the vertical with an integrated view of services and resources from multiple administrative domains. In particular, service orchestration entails managing end-to-end services that are split into various domains based on requirements and availability. Federation entails managing administrative relations at the interface between the SOs belonging to different domains and handling abstraction of services. The SO key functionalities, architecture, interfaces, as well as two sample use cases for service federation and service and resource orchestration are presented. Results for the latter use case show that a vertical service is deployed in the order of minutes.

CCS CONCEPTS

Networks, Network services, Programmable networks

KEYWORDS

5G, mobile transport, NFV, end-to-end service orchestration, federation

1 Introduction

5G networks are envisioned to expand the scope of current mobile networks services to support various vertical services, hence enriching the telecom network ecosystem. A wide range of vertical industries, such as eHealth, automotive, media, or cloud robotics, act as drivers to construct this ecosystem. The support of the diverse service requirements of different vertical industries is not only a question of providing broadband capacity, but also a matter of "ultra-reliable low-latency communications" and "massive density connections".

To enable such vision, EU H2020 5G-PPP phase 2 5G-TRANSFORMER (5GT) project [1] proposes a flexible and adaptable SDN/NFV-based design of the next generation Mobile Transport Networks. In this design, Network Function Virtualization (NFV), Network Slicing, Multi-access Edge Computing (MEC), and network federation are considered as key concepts to enable such mobile network transformation.

The 5GT solution consists of three novel building blocks, namely:

1) Vertical Slicer (VS) as the common entry point for all verticals into the system. The VS dynamically creates and maps the services onto network slices according to their requirements, managing their lifecycle. It also translates the vertical and slicing requests into NFV network services (NFV-NS) sent towards the SO. In this sense, a slice will be deployed as a NFV-NS instance.

2) Service Orchestrator (SO). It offers service or resource orchestration and federation, depending on the request coming from the VS. Orchestration entails managing end-to-end services or resources that were split into multiple administrative domains based on requirements and availability. Federation entails managing administrative relations at the interface between SOs belonging to different domains and handling abstraction of services and resources.

3) Mobile Transport and Computing Platform (MTP) as the underlying unified transport stratum, responsible for providing the resources required by the NFV-NS orchestrated by the SO. This includes their instantiation over the underlying physical transport network, computing, and storage infrastructure. It also may (de)abstract the MTP resources offered to the SO.

This paper focuses on the architecture, functionalities and interfaces of the SO of the 5GT system with a particular attention to resource orchestration and federation. Current software implementing Network Function Virtualisation Orchestrators (NFVOs) (e.g., Cloudify, OpenBaton, Open Source MANO) often

EM-5G'18 '18, December 4, 2018, Heraklion, Greece

ACM ISBN 978-1-4503-6083-8/18/12…$15.00
https://doi.org/10.1145/3286680.3286684

simply takes in input Network Service Descriptors (NSDs) where the placement of the Virtual Network Functions (VNFs), their interconnections, and the utilized resources are already specified by the vertical/user.

The SO developed within 5GT goes beyond the current state of the art by autonomously making some of the aforementioned decisions. As proposed for other frameworks [2][3], the SO takes decisions based on the slice requirements imposed by the different verticals' service requests and the network context (e.g., topology, available resources, etc.). When reaching the SO, vertical requests from the VS have been translated into NFV-NS requests. The SO takes decisions on end-to-end NFV-NS (de)composition along with the allocation of virtual resources from multiple administrative domains so that NFV-NS requirements are met. This decision implies not only the allocation of underlying network, computing and storage resources, and placement of virtual network functions, but also the interaction (federation) with the SOs of other administrative domains when, for instance, requirements cannot be met with services and resources of a single domain. In this way, the virtual resources and the services offered by multiple providers can be aggregated by federating them through their respective SOs.

2 State of the Art Solutions for the Service Orchestration

Service orchestration and automated lifecycle management are mandatory components of the 5G network softwarisation vision. Multiple software implementations of the management and orchestration (MANO) platform are already available both as open source and proprietary solutions.

Open Network Automation Platform (ONAP) was founded in 2017 by merging AT&T driven OpenECOMP project and another NFV orchestration project called Open-O [4]. This project is hosted by Linux foundation and has one of the largest Telco and vendor community among all existing NFV orchestration projects. The first platform release, called Amsterdam, was delivered in November 2017. It was mostly focused on the integration of the software artifacts from the OpenECOMP and Open-O. Thus, just two use-cases are supported in the first release and one of them utilizes proprietary Virtual Network Functions (VNFs). The second platform release is about to emerge, however still at this moment ONAP cannot be considered as a generic orchestration platform suitable for arbitrary NFV use cases.

Open Source MANO (OSM) [5] is an ETSI-hosted project focused on the development of the software stack aligned with ETSI NFV specifications. At present the platform release 4 (OSMv4) is available. The platform has a flexible architecture and each component is both replaceable and pluggable. Overall OSMv4 was targeting production readiness of the OSM platform and multiple improvements were made in this direction. Thus, this platform might be used in pre-production scenarios.

Cloudify [6] is another quite mature open source solution that is already widely implemented in production environments. It has powerfull workflow engine and flexible plugin mechanism and service templates are declared in a form of the TOSCA dialect. Despite of the open source nature of this product, Cloudify is a vendor solution and developed by a single company. Another project, OpenBaton [7] has only few developers behind and quite tiny community. OpenStack Tacker [8] project, even if it claimed to be an NFV Orchestrator, focuses on the OpenStack only.

In addition to the industry-driven communities, there are a number of R&D projects funded under the 5G-PPP/H2020 programs, which aim to deliver NFV orchestration platforms. For example, 5G-Crosshaul project [9] delivered an NFVO software prototype able to support the instantiation and termination of Network Services composed of VNFs over a Crosshaul transport network. Other 5G-PPP/H2020 projects like SONATA [10], 5GEx [11] focused on a various operation and use case specific scenarios, like VNF SDK development and multi-domain operation.

Upon a detailed analysis of the existing open source orchestration platforms (like OSM, ONAP and Cloudify), they do not support slicing, federation and MEC, which are the key technology enablers for 5G. 5GEx and 5G-Crosshaul projects explored network slicing and multi-domain aspects, but only limited support was provided, e.g., 5G-Crosshaul mainly focused on data plane solution to support network slicing, 5GEx was focused on resource federation part. Therefore, in this work the SO advances the aforementioned solutions by providing: i) support for network slicing (mapped to NFV network services) by offering isolated network service instance management along with their lifecycle management and design of novel algorithms for resource allocation and function placement per slice; ii) support for MEC services by extending orchestration for MEC applications; iii) support for federation by developing new functional components for resource and service federation including network service decomposition, service and resource advertisement entity, and interfaces to interact with other administrative domains: iv) support of integrated orchestration of heterogeneous cloud and WAN technologies by developing and integrating a set of new plugins for specific cloud and network technologies. The new features brought by the SO will pave the way to an autonomic service provisioning to support various vertical services in 5G. The SO implementation is based on Cloudify, while extensive extensions will be made to support the above new features. The selection of Cloudify was made due to its platform maturity, extendable TOSCA dialect to declare service templates and resource dependencies, powerful workflow engine and flexible plugin architecture.

3 SO Architecture and Key Functionalities

This section details the main functions and interworking of the SO subsystems and the related interfaces.

3.1 SO functional system architecture

Figure 1 presents the SO system architecture with a high level overview of the main functional modules and the interactions that need to be developed to achieve the SO operation. The SO system architecture is in line with ETSI NFV guidelines [12] and is inspired by 5GEx [11] and 5G-Crosshaul [9] design.

Northbound Interface (NBI): it offers Application Program Interfaces (APIs) to support requests for service on-boarding, service creation, service instantiation, service modification, service termination.

NS/VNF Catalogue DB and Manager: it is the repository of all usable Network Service Descriptors (NSDs) and VNF Descriptors (VNFDs) that can be accessed through the Catalogue Manager. The NSD is a deployment template for a Network Service the SO can provide (either on its own or by leveraging neighboring SOs). VNFD is a deployment template which describes a VNF in terms of its deployment and operational behavior requirements. The NSD/VNFD is used by the SO in the process of NS-VNF instantiation and their lifecycle management to obtain relevant information, e.g., configuration rules, out-scaling rules. The Catalogue Manager also takes care of advertising NSD to other domains for federation purposes.

NFV Orchestrator (NFVO): it is responsible for orchestration of virtual resources across multiple domains, fulfilling the Resource Orchestration (NFVO-RO) functions, as well as of coordinating the deployment of Network Services (NS) along with their lifecycle management, thereby fulfilling the Network Service Orchestration (NFVO-NSO) functions.

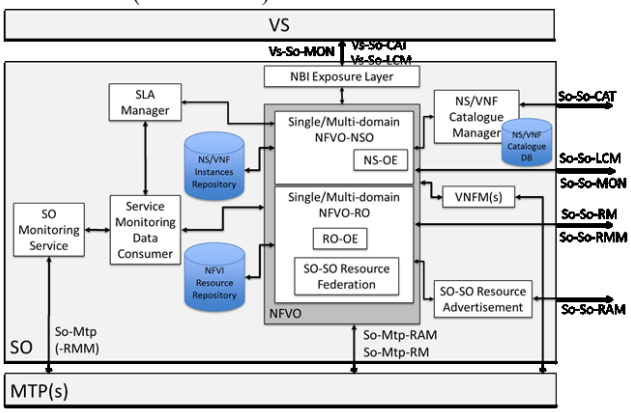

Figure 1: SO functional level architecture

More specifically, the NFVO-NSO coordinates all NS deployment operations including AAA and formal checks of service requests based on attributes retrieved from NSDs and VNFDs. Through the NS-Orchestration Engine (NS-OE), it decomposes the NSDs into several segments and decides to implement them either in the local administrative domain or by federating with the neighbor SOs. Finally, NFVO-NSO requests the NFVO-RO of the local domain or the NFVO-NSO of a neighbor domain to deploy the Network Service segment. The NFVO-RO maps the service segment into a set of virtual resources through the RO Orchestration Engine (RO-OE) by deciding where, in the virtual infrastructure exposed by the MTP, each VNF shall be placed based on specified computational, storage and networking (e.g., bandwidth, latency) requirements. Then, the NFVO-RO takes care of resource provisioning in the MTP domains directly by using So-Mtp/Southbound Interface (SBI) or indirectly by using So-So/east-westbound interface (EBI/WBI), respectively. In the latter case, the sharing of abstract views and the required coordination of correlated actions is carried out by the SO-SO Resource Federation element that executes/forwards requests to the SO NFVO-RO of other domains to request allocations.

VNF Manager (VNFM): it is in charge of the VNF deployment by using either local or remote resources (or a combination of thereof) and of the lifecycle management of the deployed VNFs. It receives relevant VNF lifecycle events from the local NFVO and provides reconfiguration according to specified counteractions decided by the NFVO through VNFDs/NSDs (e.g., auto-scaling).

NFVI Resource Repository: it stores consolidated abstract resource views received from the underlying MTP or from the SO-SO Resource Federation block as for the abstract resource views coming from other domains.

SO-SO Resource Advertisement: it is in charge of exchanging abstract resource views (e.g., abstract topologies, computing and storage capabilities) with other domains while feeding the SO-SO Resource Federation entity that consolidates inputs and stores federated resources into the NFVI Resource Repository.

NS/VNF Instance Repository: it stores the instances of VNFs and NSs that have been previously instantiated.

SO Monitoring Service: it provides the quality measurement reports for the SO to support SO monitoring management including performance monitoring and fault management, based on the collected monitoring data provided by the MTP.

Service Monitoring Data Consumer: it supports the lifecycle management of instantiated VNFs/NSs by collecting quality measurement reports from the SO Monitoring Service and reporting data to the NFVO, e.g., to trigger auto-scaling actions based on scaling rules in the NSD, or to the SLA Manager, e.g., to enable SLA on-line verification. Performance reports can be also used to trigger healing actions to recover from failures or service degradations. The aim is to adapt deployed services or provisioned resources while preventing service degradations due to the concurrent usage of resources from different services.

SLA Manager: it elaborates performance reports from the Service Monitoring Data Consumer during the service lifecycle and assures that the agreed SLAs are continuously met through on-line SLA verification. If a request's SLA parameters are not met, the SLA Manager may trigger scaling actions to prevent or recover from SLA violations.

3.2 Interfaces

The **VS-SO** is the SO northbound interface (NBI) that enables the interaction between the VS and the SO. The implementation of the VS-SO is mostly based on the ETSI NFV IFA 013 [13], which defines the NBI of an NFVO. However, the VS-SO implements further methods to allow the SO to handle Network Services extended to include MEC Applications.

The **SO-MTP** is the SO southbound interface (SBI) that addresses the interworking between the SO and the MTP building blocks of the 5GT architecture. It is worth mentioning that the SO and MTP may follow a 1:N relationship. That is, a single SO may interact via multiple SBI instances with N MTPs which handle the configuration and programmability of a number of domains including heterogeneous virtualized resources for compute, storage and networking. The SO-MTP is based on ETSI NFV IFA 005 and 006 [15][16].

The **SO-SO** is the SO eastbound/westbound Interface (EBI/WBI) that provides service and resource federation. The SO EBI/WBI enables interaction between the local SO and the SOs of other

administrative domains. This SO-SO interface enables an SO to request/offer Network Service as a Service (NSaaS) and NFVI as a Service (NFVIaaS). The SO EBI/WBI implementation is based on the ETSI NFV IFA 013 and ETSI NFV IFA 005. NSaaS provides NFV network services through the reference points of the SO NFVO-NSO mainly by implementing and extending interfaces of ETSI NFV IFA 013 [13]. NFVIaaS provides resources or resource abstractions through the reference points of SO NFVO-RO and SO-SO Resource Advertising block, generally by implementing interfaces of ETSI NFV IFA 005 [15].

4 SO Use Cases

This section presents two specific use cases involving the SO: service federation and multicloud content delivery service.

4.1 Service Federation in 5GT

In 5GT orchestration decisions are triggered whenever the VS requests the creation of a NFV-NS or changes to an existing NFV-NS. In either case, the SO NFVO must make the following decisions: i) how to decompose a complex NFV-NS (i.e., NSDs) into several segments for efficiency; ii) consequently, where to implement each segment, whether in the local domain or by leveraging neighbor SOs; iii) how to map an NFV-NS segment into a set of virtual resources, i.e., where to run the component VNFs in the virtual infrastructure based on specified resource demand.

This use case focuses on the second decision. Service federation is a mechanism for integrating multiple administrative domains unifying them into an open platform for sharing services with a certain degree of trust between each other. Different administrative entities create a peer-to-peer network with pre-established business level terms translated into Service Level Agreements (SLAs). An administrative domain that requests federation procedure is a consumer domain whereas an administrative domain that provides federated resources and/or services is a provider domain.

In 5GT, the paradigm of Network Service as a Services (NSaaS) is used for service federation. All federation procedures are executed at the SO level through the EBI/WBI. NSaaS is a federation procedure where NFV-NS are provided to a consumer domain by a provider domain. The provider domain is responsible for instantiation and life-cycle management of a federated network service. The consumer is only a user of a federated network service. The NSaaS is established in two phases: pre-phase and instantiation phase.

The pre-phase of the NSaaS consists of business agreement between administrative domains, established “offline” on business meetings. Each domain can offer to other administrative domains a set of agreed and offered NFV-NS referred to as catalogue of services. Once the pre-phase is finished, peering catalogues are shared and stored into the NS Catalogue Database of each administrative domain.

Once the instantiation phase of NSaaS is triggered, SO NFVO-NSO checks the NS Catalogue for a desired NFV-NS. A consumer SO decides how to decompose the NFV-NS (i.e., NSDs) into several segments, for example based on service or resource availability, and for the most suitable provider domain. A request for NFV-NS creation is delivered over the EBI/WBI interface to the chosen provider’s SO NFVO-NSO. The chosen provider’s SO NFVO-NSO initiates service creation and instantiation procedure to its constituent MTP. Upon creation of a federated NFV-NS, positive feedback is sent to the consumer SO NFVO-NSO. Then the consumer SO starts using the federated NFV-NS or adds it to a set of nested services.

The provider SO is responsible for the Lifecycle Management (LCM) of the federated NFV-NS. The consumer SO only keeps track of the quality of the consumed federated NFV-NS and reacts accordingly in case there are violations of the SLA or there is a change in nesting of services.

An example of service federation is included in the workflow depicted in Figure 2. The workflow describes the federation of

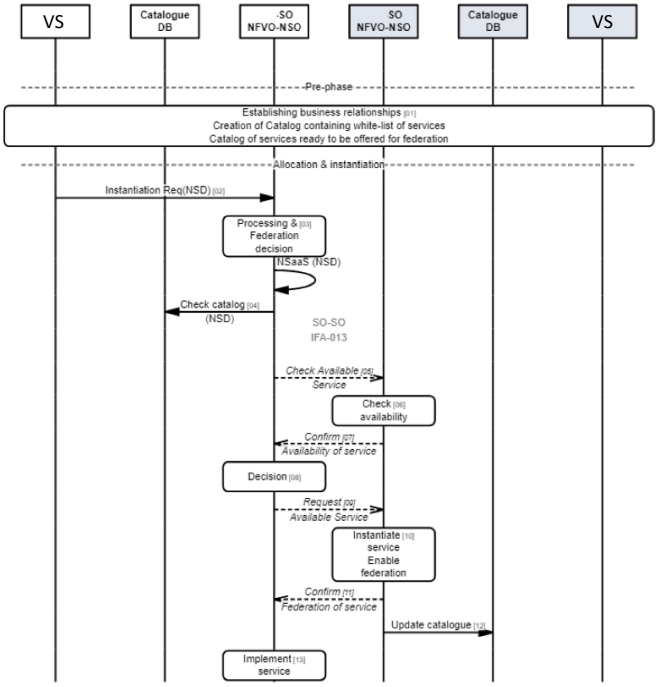

Figure 2: SO Service Federation

NFV-NSs (NSaaS) between different administrative domains. It is assumed that the consumer SO receives a request for instantiating a certain NFV-NS (with the NSD) and that it is not capable of satisfying it with its own resources or services. Thus, it exploits service federation to accomodate the requested service. The service federation steps are described here below.

1. The administrative domains (white and grey in Figure 2) create a white-list of agreed and offered NFV-NSs which is referred to as catalogue of services. The pre-phase is concluded when the catalogues from each peering administrative domain is stored into the Catalogue DB.
2. The NFV-NS instantiation request is received from the VS. The SO NFVO-NSO performs decomposition of the requested NFV-NS. In this workflow, a decision is made to consume one of the set of decomposed nested

NFV-NSs through federation of NFV-NS or by consuming NFV-NSaaS. As mentioned, the decomposition process of NFV-NS is for further study.

3. NFVO-NSO checks the NFV-NS Catalogue to find out if the desired decomposed NFV-NS can be consumed from another administrative domain. The selection of the potential peering SOs is based on either the parameters of the NSD or best-matching capabilities to enable the requested NFV-NS instance. The SO takes the role of consumer SO (white on Figure 2).
4. Consumer SO NFVO-NSO sends the "check for availability" requests to multiple peering SOs (or their NFVO-NSOs) that potentially could enable the NFV-NS.
5. The peering SO NFVO-NSOs check their availability of the requested service and make decision if it can provide a service instance to the consumer SO NFVO-NSO. The peering SO NFVO-NSOs send feedback to the consumer SO NFVO-NSO.
6. Depending on the received results, the consumer SO NFVO-NSO selects the provider SO NFVO-NSO or the procedure is terminated if all received responses are negative. The consumer SO NFVO-NSO decides for the best-matching provider SO NFVO-NSO (e.g., it provides lowest latency for a low-latency NFV-NS and/or at minimal cost). The consumer SO NFVO-NSO sends a request for NFV-NS instantiation to the selected provider SO NFVO-NSO. The selected provider SO NFVO-NSO initiates the service creation and consequently the instantiation procedure involving its constituent MTP.
7. Once the requested federated NFV-NS is instantiated, a positive feedback is sent back to the SO NFVO-NSO which becomes the consumer of the federated NFV-NS instance.
8. The selected provider SO NFVO-NSO updates its constituent Catalogue DB (grey) . The consumer SO NFVO-NSO consumes the federated NFV-NS by adding it to the set of nested services.

4.2 Multicloud Content Delivery Service

In this use case the implementation of a multicloud content delivery service is considered. The architecture of the implemented service is shown in Figure 3. Content delivery service is deployed in both Amazon Web Service (AWS) public cloud (i.e., SPR.1) and in a private Telco/NFV edge cloud (i.e., SPR.2). The SO, based on extensions of Cloudify, orchestrates the deployment of the service and resources both in the AWS public cloud and in the Telco/NFV edge cloud, including setting an IPsec tunnel between them with a predefined deployment. In this setup, a local cache is deployed in the private cloud at the edge next to the end user in order to ensure low latency content delivery to the end user.

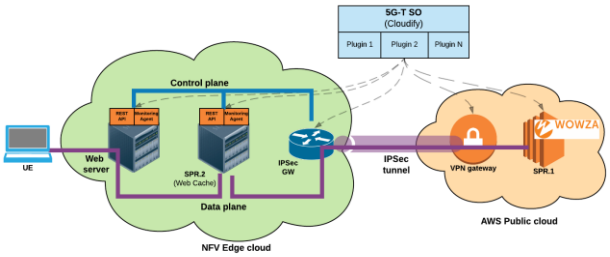

Figure 3: Multicloud content delivery service

The first set of performed experiments measures the *service deployment time*, defined as the time elapsing between the service deployment request and the successful service deployment. The phases followed by the SO to deploy the service and the time each phase is taking are summarized in Table 1. Results show that the overall service deployment time is about 280s (i.e., less than 5 minutes).

Table 1 Multicloud content delivery service phases

Order	Node	New or Exists	Start creating	End creating, end configure, node started	Duration of sum(creating, configuring, node starting) [s]
1	AWS Customer Gateway	new	2018-06-15 10:44:17	2018-06-15 10:44:26	10.01388882
2	Open Stack VM VPN Gateway	new	2018-06-15 10:44:37	2018-06-15 10:45:17	46.15972211
3	AWS VPN Connection	new	2018-06-15 10:44:37	2018-06-15 10:48:02	237.8518519
4	AWS VPN Gateway	new	2018-06-15 10:44:27	2018-06-15 10:44:36	10.26736136
5	AWSVPN Connection	new	2018-06-15 10:48:03	2018-06-15 10:48:06	3.1574069
6	AWS exists_vpc	check if exists	2018-06-15 10:44:16	2018-06-15 10:44:25	10.38078699
7	Open Stack external_network	check if exists	2018-06-15 10:44:17	2018-06-15 10:44:25	8.811342559
8	Open Stack ole_mgmt_network	check if exists	2018-06-15 10:44:17	2018-06-15 10:44:25	8.811342559
9	Open Stack	check if	2018-06-15 10:44:17	2018-06-15 10:44:24	7.628472667

	security_group	exists			
10	AWS exists_route_table	check if exists	2018-06-15 10:44:18	2018-06-15 10:44:29	12.9351858
11	Open Stack ole_data_network	check if exists	2018-06-15 10:44:18	2018-06-15 10:44:25	7.587963046
12	Open Stack external_network1_port	check if exists	2018-06-15 10:44:26	2018-06-15 10:44:32	7.756944979
13	Open Stack ole_mgmt_network1_port	check if exists	2018-06-15 10:44:27	2018-06-15 10:44:32	6.203704106
14	Open Stack ole_data_network1_port	check if exists	2018-06-15 10:44:28	2018-06-15 10:44:32	4.789352533
15	AWS route_vpc	check if exists	2018-06-15 10:44:37	2018-06-15 10:44:42	5.940972187
16	Open Stack vyos_vm_baseline_config	new	2018-06-15 10:48:15	2018-06-15 10:48:22	8.751157293
	Whole Deployment takes		2018-06-15 10:44:17	2018-06-15 10:48:22	283.6921296

Once the service is delivered, two performance parameters are measured: the data rate (both between the local cache and the end-user and the local cache and the global repository) and the round trip time (RTT) (in the two aforementioned scenarios). The traffic between the end-user and the local cache shows a bursty behavior with peaks of 10Mbps (the video content used has an average bitrate of 2Mbps). When the video is not in the local cache the traffic between this virtual machine and the global repository exhibits the same traffic pattern. On average the measured RTT between the end-user and the local cache is about 40µs (notice that the end user is directly attached to the local cache) while the one between local cache and global repository is about 40ms.

5 Conclusions

In this paper, we presented the design of the 5G-TRANSFORMER Service Orchestrator (SO). The SO is a novel component responsible for end-to-end orchestration of services and resources across multiple administrative domains. In the paper, we reviewed state of the art solutions for service orchestration already available both as open source and as proprietary solutions, and under research of several EU H2020 5G-PPP projects. We presented in detail the SO functional system architecture, including the main function modules inside the SO, the required interfaces to other components of the 5G-TRANSFORMER system, namely the VS and the MTP, as well as the interfaces to the SOs of other administrative domains enabling multi-domain service and resource federations. We finally presented two use cases where the SO is utilized: service federation and deployment of a multicloud content delivery service involving resource orchestration. For the latter use case results, collected from an experimental testbed involving private and public cloud resources, a service deployment time in the order of minutes is achieved.

ACKNOWLEDGMENTS

This work has been partially funded by the EC H2020 5G-TRANSFORMER Project (grant no. 761536).
The authors would like to thank Juan Brenes Baranzano from ATOS, Spain for his contribution to the paper.

REFERENCES

[1] K. Antevski, *et al.,* "Resource Orchestration of 5G Transport Networks for Vertical Industries", IEEE PIMRC 2018, Workshop 5G Cell-Less Nets

[2] F. Slim, *et al.*, "Towards a dynamic adaptive placement of virtual network functions under ONAP," *2017 IEEE NFV-SDN*, Berlin, 2017, pp. 210-215.

[3] V. Sciancalepore, *et al.* , "z-TORCH: An Automated NFV Orchestration and Monitoring Solution," in *IEEE Transactions on Network and Service Management*.

[4] ONAP project - https://www.onap.org/

[5] OSM project - https://osm.etsi.org/

[6] Cloudify orchestration platform - https://cloudify.co/

[7] OpenBaton project - https://openbaton.github.io/

[8] OpenStack Tacker - https://wiki.openstack.org/wiki/Tacker

[9] 5G-Crosshaul project - http://5g-crosshaul.eu/

[10] SONATA NFV project http://www.sonata-nfv.eu/

[11] 5G-Ex project http://www.5gex.eu/

[12] ETSI GS NFV 002 v1.2.1, "Network Functions Virtualisation (NFV); Architectural Framework", Dec. 2014

[13] ETSI GS NFV IFA 13, "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Os-Ma-Nfvo reference point – Interface and Information Model Specification" v2.3.1, August 2017

[14] ETSI GS MEC 010-2, "Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management", v1.1.1, July 2017

[15] ETSI GS NFV-IFA 005, "Network Function Virtualisation (NFV); Management and Orchestration; Or-Vi reference point – Interface and Information Model Specification", v2.1.1, Apr. 2016

[16] ETSI GS NFV-IFA 006, "Network Functions Virtualisation (NFV); Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification", V2.1.1, Apr. 2016