Mangues-Bafalluy, J., Baranda, J., Pascual, I., Martínez, R., Vettori, L., . . ., Salvat, J. X. (2019). 5G-TRANSFORMER Service Orchestrator: design, implementation, and evaluation. In *2019 European Conference on Networks and Communications (EuCNC)*.

DOI: https://doi.org/10.1109/EuCNC.2019.8802038

5G-TRANSFORMER Service Orchestrator: design, implementation, and evaluation

Josep Mangues-Bafalluy, Jorge Baranda, Iñaki Pascual,
Ricardo Martínez, Luca Vettori
Centre Tecnològic de Telecomunicacions de Catalunya,
CTTC/CERCA, Spain

Arturo Zurita, David Salama
ATOS, Spain

Dmitriy Andrushko, Konstantin Tomakh
MIRANTIS, Ukraine

Giada Landi
Nextworks, Italy

Kiril Antevski, Jorge Martín-Pérez
Universidad Carlos III de Madrid, Spain

Barbara Martini
Scuola Superiore Sant'Anna, Italy

Xi Li, Josep X. Salvat
NEC Laboratories Europe GmbH, Germany

***Abstract* — 5G networks will pose complex network management challenges due to the variety of vertical services they will need to serve and the diversity and heterogeneity of underlying infrastructure. The service orchestration functionality is fundamental to enable fulfilling the requirements of the different verticals while efficiently sharing the infrastructure resources. This paper details the 5G-TRANSFORMER service orchestrator implementation and operation. It also evaluates and profiles service creation time showing how the automation offered by the platform allows reducing it from hours to minutes. It also shows that the most time-consuming steps correspond to the deployment of the virtual network functions and post-deployment configuration, which consume one order of magnitude more time than the rest of steps (e.g., network creation, port creation).**

Keywords*— *5G, Service orchestration, End-to-End, Experimental, Vertical industry, Service creation time

I. INTRODUCTION

5G networks are targeted to expand the scope of existing 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. Towards this vision, the EU H2020 5G-PPP phase 2 5G-TRANSFORMER (5GT) project [1] proposes an open and flexible 5G transport and computing platform tailored to support diverse service requirements of various vertical industries. 5GT proposes a novel architecture [2] aligned with ETSI NFV Interface and Information Model Specifications (IFA) that consists of three main building blocks: (i) the Vertical Slicer (5GT-VS) as the vertical front-end and the common entry point for all verticals, (ii) the Service Orchestrator (5GT-SO) as the E2E service orchestration platform and (iii) the Mobile Transport and Computing Platform (5GT-MTP) as the underlying unified transport stratum for integrated fronthaul and backhaul networks. In the 5GT system, network slices requested by verticals to deploy their vertical services, are managed as NFV Network Services (NS). On the request of the 5GT-VS, the 5GT-SO [3] is in charge of provisioning these services along with managing their entire life-cycle. More specifically, it is responsible for the End-to-End (E2E) service and resource orchestration in a single domain or across multiple administrative domains (in the context of federation). A preliminary high-level functional architecture of the 5GT-SO was presented in [4], though that paper, including some rough evaluations, remained at the design and concept levels. Based on those concepts, some very basic functions were demonstrated in [5]. On the other hand, major updates at the architectural and implementation level are presented in this paper to further detail the internal building blocks, and new functionality, and to improve its modularity and flexibility. Furthermore, this paper also presents experimental results over a real infrastructure as opposed to a simplified emulated one in [5].

Recently, several orchestration platforms have been developed, e.g., OpenSource MANO (OSM) [7], Cloudify [8], Open Network Automation Platform (ONAP) and OpenBaton, to automate the deployment and configuration of NS based on terms specified in NS descriptors. Despite the growing adoption of these orchestration platforms, the proposed 5GT-SO offers a number of additional features. Firstly, Cloudify and ONAP do not have any means to support network slicing, though preliminary efforts are ongoing. On the other hand, OSM claims to offer slicing support in its last release and Open Baton already supports an external component that provides a Network Slicing Engine (NSE). However, slicing support is limited to the Virtual Infrastructure Manager (VIM) level. Secondly, although such orchestration platforms support programming the network behavior between the nodes in a datacenter (DC) using VIM APIs, support for multi-datacenter interconnection through a WAN (Wide Area Network) transport is limited. In contrast, the 5GT-SO was designed bearing this scenario in mind. In fact, the 5GT-SO can decide the optimal placement of virtual network functions (VNFs) in distributed DCs considering the status of both cloud and network resources. Thirdly, none of the above orchestration platforms support orchestrating edge-computing services. Instead, the 5GT-SO also envisions Multi-access Edge Computing (MEC) as a key enabler to develop low-latency services allowing verticals to place their services at the edge and to provision time-critical applications. Finally, none of these orchestration platforms provides federation mechanisms, unlike the 5GT-SO. Federation allows deploying part of a NS in another administrative domain owned by another operator by peering 5GT-SOs (e.g., to accommodate spikes in demand).

In this paper, we present the details of the 5GT-SO software implementation focusing on the design of its internal

building block functionalities and their interactions, the workflows, and the interfaces. In particular, we show that the above 5GT-SO features result from (*i*) a WAN Transport Engine that can effectively deal with different types of WAN resources allowing multi-site, multi-VIM NS deployments; (*ii*) a Resource Orchestration engine that embeds multiple Placement Algorithms [6] deciding the optimum location of VNFs and the allocation of required network and cloud resources; and (*iii*) a Service Orchestration engine able to coordinate NS deployment and lifecycle management along with federation. Moreover, we show the advantage of our modular design able to integrate multiple MANO platforms, either open source or proprietary. Finally, we present the experimental evaluation and profiling of service deployment time as seen from the 5GT-SO in the context of an entertainment use case Proof of Concept (PoC). We show how service creation time is decreased from hours (as in current manual procedures, which are the norm in current operator networks) to minutes and that its main component is the creation of VNFs and post-deployment configuration (one order of magnitude higher than other deployment steps). The difference could be much higher when WAN connections through heterogeneous infrastructure need to be established due to the additional network management departments involved and their manual inter-department order handling.

The rest of the paper is organized as follows. Section II presents the 5GT-SO software design. Section III describes the 5GT-SO building blocks interaction and workflow for NS instantiation. Section IV details the 5GT-SO implementation. Section V reports experimental service creation time results. Finally, Section IV provides the concluding remarks.

II. Service Orchestrator Software Architecture

Figure 1 presents the architecture of the 5GT-SO and its relation with the 5GT-VS and 5GT-MTP. It is in charge of receiving the network service instantiation requests from the vertical slicer, which includes the network service descriptor and instantiation parameters (e.g., deployment flavor). It also handles the service and resource orchestration workflows.

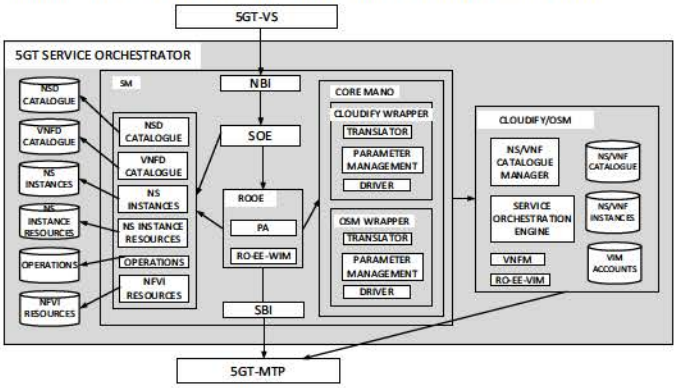

Figure 1. 5G-T Service Orchestrator architecture

In this direction, the 5GT-SO receives the service structure and requirements from the 5GT-VS and requests the available infrastructure (e.g., network topology, available computation power) from the 5GT-MTP, and through the placement algorithm (PA), which maps these requirements to the infrastructure. Eventually, the service components, VNFs and virtual links (VLs), requested by an NS are deployed over the underlying infrastructure in locations where they fulfill the requirements towards a successful service delivery.

A. Architecture overview

The aim of 5GT is to design a flexible service orchestrator architecture capable of integrating, and so, exploiting, the functionality already available from well-known and production-level MANO platforms (e.g., OSM and Cloudify) as far as the high-level functionality described above is concerned. In this way, the focus is put on the innovative functionality brought by the project (e.g., federation).

At a high-level, its architecture is organized in four main building blocks. First, the service manager (SM) is a novel building block introduced by 5GT, which is the brain of the 5GT-SO, as it handles service requests, and has an end-to-end view of what is offered by the 5GT-MTP. Thanks to this global view, it is also in charge of stitching Point-of-Presence (PoP) and WAN network connections in an automated way when multiple NFVI PoPs are involved in providing a network service, which is a functionality hitherto not offered by open source MANO platforms. It also embeds the service orchestration and resource orchestration logic and dispatches relevant tasks to the other building blocks according to the operational workflow it handles. Second, the Core MANO platform (i.e., OSM or Cloudify) will handle the computing resources by interacting with the 5GT-MTP. Since the SM uses the API of each MANO platform and they are different, a wrapper is needed for this translation and adaptation. It will also handle the specificities of each orchestration platform in a way that does not alter the internal workflow. Third, even if the placement algorithm is part of the resource orchestrator of the 5GT-SO, it is highlighted here as one main building block, since it is accessible through a well-defined REST API. In this way, external algorithmic modules can be easily tested and its performance compared. Finally, the databases are in charge of maintaining the state of the system in terms of service offering, instantiated network services, 5GT-MTP resources used, etc.

This 5GT-SO design has several advantages. It brings flexibility in terms of development. In fact, large software platforms are powerful, but they are complex and heavy given the huge amount of functionality brought. By introducing the SM, 5GT exploits the power of these platforms whilst allowing adding new functionality easily, hence focusing on the novel concepts to be evaluated. Furthermore, we believe this approach makes the 5GT code more survivable and evolvable, since it can be adapted to new MANO platform releases by just modify the wrapper to adapt to the new API. In this way, the 5GT functionality can also be exploited with newer releases. Moreover, other core MANO platforms may be integrated in the future by developing the corresponding wrappers.

B. Service Manager

In Figure 1, we can find the main building blocks of the Service Manager component, which acts as the brain of the 5GT-SO. These building blocks are:

- **Database (DB) module**: this module contains several submodules to interact with each of the external databases containing the system status information. Currently, the available external databases are:
  - *NSD (Network Service Descriptor) catalogue*: stores the information of NS descriptors.
  - *VNFD (VNF Descriptor) catalogue*: stores the information of VNF descriptors.
  - *NS Instance Database*: contains information of NS instances.
  - *NS Instance Resources Database*: stores information of resources used by an NS Instance.
  - *Resources Database*: stores information of the status of the different resources (networking, computing and storage) of the underlying transport Infrastructure.
  - *Operations database*: stores information about NS operations, i.e., instantiate, terminate an NS.
- **Service Orchestration Engine (SOE):** this module receives the requests from the NBI. For the NS query-related operations it interacts with the appropriate databases to fulfil the request and retrieve NS instance information status. If the requests involve 5GT-MTP resources management (e.g. for NS instantiation or termination requests), the SOE delegates the operation to the ROOE module.
- **Resource Orchestration Engine (ROOE):** it handles the requests that are related to 5GT-MTP resources management, i.e., NS instantiation and termination. It interacts with the Placement Algorithm (PA) submodule, the coreMANO module and the Resource Orchestration Execution Engine WIM (RO-EE-WIM) submodule to accomplish the request, where WIM stands for wide area network infrastructure manager. The ROOE submodules perform the following functionalities:
  - *Placement Algorithm (PA):* determines the placement of the virtual network functions and their required allocation of resources during the instantiation of NSs based on its characteristics.
  - *Resource Orchestration Execution Engine WIM (RO-EE-WIM):* it is submodule in charge to communicate with the SBI to handle network resources management operations at the 5GT-MTP.
- **CORE MANO:** it allows the interaction between the SM and the available open source orchestration platforms. Current supported orchestration platforms are Cloudify [8] and OpenSource MANO [7]. The different wrappers manage the requests coming from the ROOE to handle compute and storage resources operations at the 5GT-MTP.

Finally, northbound/southbound interface modules act as interfaces with the corresponding block of the 5G-TRANSFORMER architecture.

III. SERVICE ORCHESTRATOR OPERATION

Figure 2 presents the network service instantiation workflow implemented by the 5GT-SO.

First, a NSD of an NS is on-boarded on the 5GT-SO. The on-boarding is performed via the 5GT-SO NBI and the SOE of the SM. The procedure is executed manually by uploading all necessary descriptors on the 5GT-SO repository (NSD catalogue). A unique identifier is generated for each on-boarded NSD.

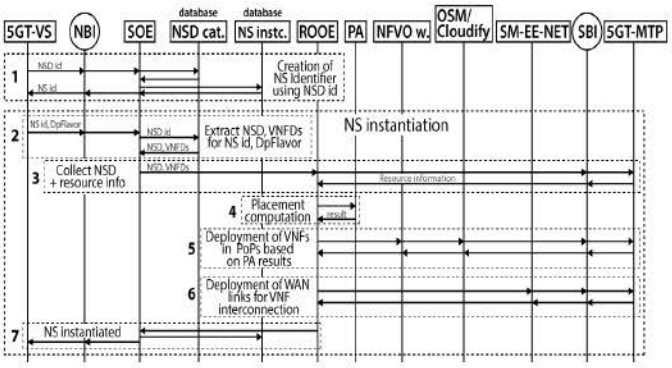

Figure 2. Network service Instantiation workflow

After a successful on-boarding procedure, the 5GT-VS issues a request for creation of an NS instance identifier (1). The SOE of the SM receives the request and checks the NSD catalogue for the existing NSD. On positive response, the SOE creates and generates a new NS instance identifier into the Instance DB. The created NS instance identifier is returned to the 5GT-VS.

The 5GT-VS at any given moment sends request for instantiation of the NS to the 5GT-SO using the NS instance identifier and a deployment flavor and an instantiation level id (2). The SOE receives the request and extracts the matched NSD and the involved VNFDs from the NSD and VNFD Catalogues. The extracted NSD and VNFDs along with the flavor and instantiation level IDs are forwarded to the ROOE requesting it to properly instantiate the NS according to the available resources (3). Upon receiving the request, the ROOE obtains information of available aggregated resources from the 5GT-MTP. The obtained information on aggregated available resources plus the received information from the SOE, are delivered to the PA module (4). The PA extracts service requirements from the NSD (e.g., VNFDs, VLs, etc.) and computes the placement of the VNFs and Virtual Links based on the available aggregated resources. Different PAs can be used for different types of NS instances, hence using different optimization objectives (e.g. latency, cost) depending on the service requirements. The generated result from the PA is forwarded to the NFVO Wrappers via the ROOE. The PA result consists of a request for specific VNFs and Virtual Links deployment over the available resources (5). Depending on the used NFVO (Cloudify or OSM), the NFVO/CoreMANO Wrappers adapt the parameters and deliver the request to the Cloudify or OSM NFVO. The Cloudify/OSM orchestrates the deployment of all VNFs over the available resources (including VMs, logical links, etc.) through interaction with the 5GT-MTP on the 5GT-SBI.

Once all the VNFs are deployed by Cloudify/OSM, the ROOE obtains all the information regarding the connection points (e.g., IP addresses) of the VNFs and issues a request to

the SM-EE-NET module for creation of all VLs interconnecting the VNFs (6). With all VNFs instantiated and interconnected with VLs, the NS instantiation is completed (7). Information regarding the NS is stored in the Instances DB.

For the duration of the service instantiation procedure, the 5GT-VS polls the SOE for the status of the operation. Once the NS is successfully instantiated, the 5GT-VS receives positive answer on the next polling request. Later on, the 5GT-VS can query for information of the NS instance (e.g., Service connection point).

IV. Service Orchestrator Implementation

The prototype of the 5GT-SO has been implemented following the software architecture described in section II. All the code is open source. In particular, the novel components developed as part of the Service Manager (SM) have been released under Apache 2.0 license and they are available in the project git repository [9] .

The SM is implemented in Python 3.5 and adopts MongoDB as database backend for its repositories. The Service Manager is a collection of stateless components, where the status of all the managed entities is maintained updated through the MongoDB repositories. All the SM modules access the MongoDB repositories to get a shared view of the internal status of the whole system. Concurrent requests about active lifecycle actions on new or existing Network Services are managed in an asynchronous manner, with per-service dedicated threads. The feasibility of each requested action is verified based on the current status of the related service, as reported in the database, and refused in case of conflicts.

The system design of the Service Orchestrator is based on a modular approach, where the entire prototype is structured in several components that interact with each other via REST APIs. The same kind of interaction is also adopted with the external components, like the 5GT-VS and the 5GT-MTP.

In particular, the interaction with the legacy NFVOs (i.e., Cloudify and OSM) is wrapped by the Core MANO module, which hides the NFVO-specific REST APIs and information models exposing a unified interface towards the other modules of the Service Manager. A similar approach is adopted at the SBI, which is modelled as an extension of the interface specified in ETSI NFV IFA 005 [10] and, internally, VIM- and WIM-specific drivers translate into the HTTP messages exposed by the 5GT-MTP REST APIs. It should be noted that the standard IFA 005 defines the interaction with a generic VIM, thus including primitives related only to resources managed by cloud platforms like OpenStack, such as Virtual Machines (VMs), virtual networks, or virtual storage. In 5GT this concept has been extended to cover also the interaction with WIMs, thus introducing a set of messages for the advertisement of network topology and the request of network paths to interconnect PoPs with a given Quality of Service.

At its NBI, the Service Manager implements a REST server based on the HTTP protocol and Json language for message encoding. The information models for NSDs, VNFDs, and Network Service lifecycle actions (e.g., instantiation or queries of Network Service instances) follow the ETSI NFV IFA specifications 014 [11], 011 [12] and 013 [13], respectively. Extensions have been implemented to describe additional characteristics of the Network Services, like end-to-end service latency, or specify service constraints or policies to be applied at runtime. The Core MANO component is thus in charge of translating all Json-based descriptors exchanged at the Service Manager NBI into the NFVO-specific format. For example, TOSCA-based descriptors are used for Cloudify, while Yaml- or Json-based descriptors are used for OSM, each of them with its own proprietary information models.

REST APIs are also used internally in the Service Manager, for example allowing the ROOE module to interact with different PAs. This approach allows supporting multiple resource orchestration algorithms which can be developed by third parties and easily integrated in the system. Depending on the specific criteria to be enforced for the selection of the resources, different PAs can be selected at runtime, even on a per-service basis, according to operator's policy or constraints provided by the vertical. The ROOE-PA API has been designed using Swagger and it specifies both the Information Model (IM) and the operations that the PAs can perform (e.g., computing where to deploy a requested service). The ROOE and PAs use client and server stubs generated with Swagger that implement such operations and IM. Data such as the resources requested by the service VNFs, and available infrastructure computing and bandwidth resources, are present in the Json that follows the IM and exchanged in every operation. Forcing the ROOE and every PA to implement a common API, eases the integration of new PAs in the system and the development and maintenance of the ROOE entity.

The whole 5G-TRANSFORMER architecture (and its software architecture implementation) is modularized to clearly separate conceptually different functionality. This enables serving complex scenarios involving potentially multiple stakeholders (e.g., datacenter providers, transport providers, virtual infrastructure providers, 5GT service providers). Such scenarios also include integrating multiple technology domains under the same 5GT-MTP, or multi-administrative domain scenarios, enabling service federation, in which parts of a complex network service are provided by peer domains in a way that is completely transparent to the 5GT-VS, and so, the vertical. We believe, this modular architecture makes the 5GT platform, and more specifically, that of the 5GT-SO, evolvable to scale and adapt to future needs.

V. Service Orchestrator Evaluation

The 5GT project has several vertical use cases including automotive and eHealth. The entertainment use case is used as example to evaluate 5GT-SO behavior in this paper. More specifically, the architecture and implementation explained above is evaluated in this section by deploying a virtual Content Delivery Network (vCDN) service, with particular focus on the 5GT-SO processing and its interaction with the underlying infrastructure. This use case aims to implement a video streaming content delivery service consisting of several VMs. Other use cases considered in the project have their own NSD, which would result in equivalent behavior in terms of

service deployment time. The only envisioned difference could be the constrained resources when deploying at edge data centers or hosts (e.g., MEC applications), which could result in higher deployment times due to longer VNF creation times (analysis left as future work).

In this case, the request for deploying the vCDN network service arrives at the 5GT-SO northbound interface as generated by the 5GT-VS. The total service creation time and its most significant steps are measured to analyze what are those having a bigger contribution to it.

The management of the vCDN service with the 5GT-SO platform enables a dynamic edge deployment that exploits the virtualization capabilities by deploying the components as VNFs. This flexibility allows on the one hand to deploy additional services, such as live feeds or video-on-demand content. On the other hand, the 5GT-SO also allows adapting the service to the demand by dynamically using additional resources for an existing service. An industrial adoption of a solution such as the 5GT-SO would allow the provisioning of rich media services in sport events within a small time interval.

A. Network setup

The vCDN service scheme is presented in Figure 1 and consists of the following elements:

- Origin Server: VM storing video content
- Edge CDN Server: cache VM serving end users requests
- Web Server: VM storing web-based video player and static content to be downloaded by the end user.

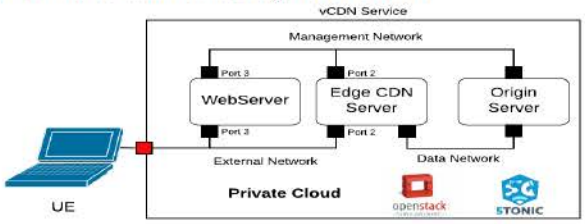

Figure 1.vCDN service structure

All the software components were deployed at the 5TONIC [14] lab, which runs OpenStack as a virtualization platform. Bare metal servers are equipped with Intel Xeon E5-2609 processors and DDR4 RAM (per-server RAM varies from 16GB to 128GB). No resource overcommitting for CPU or memory was configured for these deployments. Each server has 4 10Gbps network adapters. 5GT-SO software components were also deployed in one of the servers and vCDN images were uploaded into the OpenStack image store (Glance).

B. Service creation time vs. infrastructure load

The results show the total service creation time of the vCDN for different CPU loads. This load was generated with the Stress utility, which performed a loop that computed the square root of a random number by configuring several workers concurrently to load a specific number of cores of the CPU. The experiment was repeated ten times for each CPU load and a statistical analysis is presented in Figure 4.

The maximum and the minimum of all the samples is represented by the upper and lower whiskers, respectively. The band inside the boxes represents the median of the samples for each CPU load, and the spacing between the upper and lower parts of each box measures the degree of dispersion based on the third and the first quartiles, respectively. The cross in each box is the average of the data.

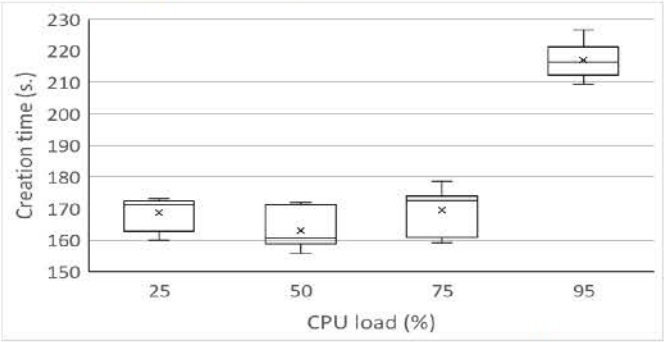

Figure 2. vCDN service creation time vs. CPU load

The graph shows an increasing trend of service creation time with load, which is moderate up to 75% CPU load (ranging from 160 s. to 170 s.), whilst there is a steep increase for very high loads (95%) reaching an average of 217.02 s. The dispersion of values (measured by the difference between the max and the min or that between the 1st and 3rd quartiles) also shows an increasing trend up to 75% load. However, for 95% CPU load, dispersion decreases, which may be due to the fact that the server works almost in saturation. Therefore, it is always backlogged, and so, the range of values obtained is much higher than with lower loads, but closer to each other, because there is less variety of ways in which tasks can be impacted. On the other hand, the probability of finding spare cycles that can be consumed right away increases with lower loads, resulting in lower creation times than when some backlog is found, hence the wider range.

Based on the experience of the vertical, a traditional CDN deployment used in a major sports event in South Korea included the deployment of an Edge CDN Server, Origin Server and the Web Server used in this evaluation. The CDN components could be deployed in dedicated hardware appliances, with a service creation time of around 2 hours, or in a Public Cloud where the creation of the service takes at least 30 minutes. In order to offer the service to more users, more Edge servers must be deployed, requiring a bigger CDN infrastructure, hence higher total service creation.

Furthermore, this difference is much higher when service deployment also requires manually configuring the WAN transport over heterogeneous infrastructure. In this case, in addition to those departments handling cloud resources and service deployment per se, other network management departments of the operator would be involved, which would handle orders manually and would schedule network configuration jobs in batches at night [15]. This seems to be the norm in current operational networks. With the use of our developed 5GT-SO, in this scenario, the service creation time is decreased from days to minutes. In this sense, the 5GT-SO will enable a much faster and more flexible deployment of the service, bringing more benefits as the CDN service grows, a

key aspect in the scope of media delivery in major sport events, and one of the key performance indicators of 5G networks.

C. *Service Creation time profiling*

During vCDN service instantiation, all supporting infrastructure-specific elements, like VM ports, security groups, etc. were created automatically at runtime. Thus, creation time of these elements also contributes to overall service creation time and is included in the resulting output. In the last step, when all infrastructure components are deployed, configuration is automatically applied to the relevant vCDN components. Additionally, the origin server with video content is pre-provisioned, and thus, this time is not included in the table. The component split of the vCDN service creation time using |Cloudify as a 5GT-SO core MANO component is presented in Table 1. This corresponds to a scenario where only residual CPU load is present in the hosts used, i.e., only regular operating system processes are running in the servers and the Stress utility is not used.

Table 1. vCDN service creation latency

	Component name	Duration, sec
1	Service Manager processing	1.68
2	Security group creation	5.87
3	Data Network creation	5.87
4	External Network creation	4.71
5	Management Network creation	5.89
6	Key pair creation	5.87
7	Port3 at External Network creation	6.78
8	Port2 at External Network creation	6.97
9	Port2 at Management Network creation	8.14
10	Port3 at Management Network creation	6.98
11	Port2 at Data Network creation	6.97
12	Edge CDN Server creation	61.10
13	Web Server creation	63.71
14	Edge CDN and Webserver configuration	47.73

The service creation is executed sequentially in the following way. First, service manager processing includes the time since the request from the VS is received until the wrapper and Cloudify come into action, including processing at SOE, ROOE, interaction with the databases, and the PA placement calculation. Second, security group is created. Third, networks and the key pair is executed in parallel, and then, the creation of network ports takes place in parallel. Once finished, creation of the servers starts. Finally, when the servers are instantiated, the configuration starts.

As an output of the profiling it may be concluded that the most time consuming operations are related to VM management, i.e., instantiation and post-deployment configuration. Thus, for services requiring lower deployment times, a more lightweight virtualization technology, like containers, may be more suitable.

VI. Conclusions and Future Work

This paper shows how the 5G-TRANSFORMER platform, and, in particular, its service orchestrator, offers end-to-end network service orchestration over mobile transport and computing platforms. Furthermore, its software architecture is flexible in the sense that it allows integrating multiple MANO platforms (e.g., OSM, Cloudify) by only developing a wrapper that adapts to their respective API while maintaining the rest of the operational workflow as is. This will also allow federating heterogeneous domains through service orchestrator peering. Moreover, the results presented in this paper show that service creation time is reduced from hours (if manual intervention is required) to minutes (thanks to the automation offered by the 5GT-SO platform), and the difference may be even higher if complex transport network configuration is also required. Additionally, the profiling also shows that the biggest component is the creation of virtual network functions in the corresponding nodes and their post-deployment configuration (one order of magnitude higher than other deployment steps).

The focus of this paper is on service provisioning time assuming the process develops without any issue. Analysis of failure handling procedures is left as future work as well as deployment in more complex and heterogeneous setups.

Acknowledgment

This work has been partially funded by the EC H2020 5G-TRANSFORMER Project (grant no. 761536) and grants TEC2017-88373-R (5G-REFINE) and 2017 SGR 1195.

References

[1] 5G-TRANSFORMER project, available at: http://5g-transformer.eu/

[2] A. de la Oliva et al., "5G-TRANSFORMER: Slicing and Orchestrating Transport Networks for Industry Verticals," in IEEE Communications Magazine, vol. 56, no. 8, August 2018.

[3] X. Li et.al., "Service orchestration and federation for verticals", in Proceeding of IEEE WCNC workshops (WCNCW), April 2018.

[4] L. Valcarenghi et al, "A Framework for Orchestration and Federation of 5G Services in a Multi-Domain Scenario," in Proceedings of the Workshop on Experimentation and Measurements in 5G (EM-5G'18), collocated with ACM Conext conference, December 2018

[5] J. Baranda et al. Deploying a containerized ns-3/LENA-based LTE mobile Network Service through the 5G-TRANSFORMER platform, IEEE NFV-SDN 2018, November 2018.

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

[7] Open Source Mano (OSM) project. Available at: https://osm.etsi.org

[8] Cloudify Orchestration Platform. Available at: https://cloudify.co

[9] 5G-TRANSFORMER project code repository. Available at: http://github.com/5g-transformer

[10] ETSI ISG NFV-IFA 005, "Network Function Virtualisation (NFV), Management and Orchestration; OR-VI reference point- Interface and Information Model Specification", v2.1.1, Apr. 2016.

[11] ETSI ISG NFV-IFA 014, Network Functions Virtualisation (NFV); Management and Orchestration; Network Service Templates Specification, August 2017.

[12] ETSI ISG NFV-IFA 011, Network Functions Virtualisation (NFV); Management and Orchestration; VNF Packaging Specification, 2016.

[13] ETSI ISG NFV-IFA 013, Network Functions Virtualisation (NFV); Management and Orchestration; OS-MA-NFVO reference point - Interface and Information Model Specification, Oct. 2016.

[14] 5TONIC laboratory. Available at: https://www.5tonic.org/

[15] J. Mangues-Bafalluy et al., Experimental framework and evaluation of the 5G-Crosshaul control infrastructure, Elsevier CSI, Jan. 2019 (online)