Orchestrating an SFC-enabled SSL/TLS traffic processing architecture using MANO

Eduardo Sousa*, Vitor A. Cunha*, Marcio B. de Carvalho†, Daniel Corujo*, Joao P. Barraca*, Diogo Gomes*, Alberto E. Schaeffer-Filho†, Carlos R. P. dos Santos‡, Lisandro Z. Granville†, Rui L. Aguiar*
*Instituto de Telecomunicações, Portugal
†Institute of Informatics – Federal University of Rio Grande do Sul – Porto Alegre, Brazil
‡Department of Applied Computing – Federal University of Santa Maria – Santa Maria, Brazil
*{eduardosousa,vitorcunha,dcorujo,jpbarraca,dgomes,ruilaa}@av.it.pt, †{mbcarvalho,alberto,granville}@inf.ufrgs.br, ‡{csantos}@inf.ufsm.br

***Abstract*—The heterogeneity of 5G requirements commands more complex network architectures, imposing the need for network orchestration. ETSI NFV MANO is the standard which defines a common framework for vendors and operators to integrate their orchestration efforts. In this paper, we evaluated how an ETSI NFV MANO compliant orchestrator (OSM) fares while orchestrating an SFC-enabled SSL/TLS encrypted traffic processing architecture, which supports both edge and cloud deployments. A quantitative evaluation was carried-out, which assessed the responsiveness and overheads of OSM, as well as the actual functionality of our SSL/TLS processing architecture (with edge computing components). A qualitative evaluation was also carried-out, providing insight into the maturity of the current OSM release, what works well, what requires workarounds, and the actual limitations. A demonstration of the architecture evaluated in this work was accepted as a contribution to the ETSI OSM PoC Framework.**

I. Introduction

The evolution of operator's networks towards 5G, as defined by bodies such as NGMN and 5G-PPP, will require solutions for more complex network architectures. Use-cases requiring very low latency, such as Extreme Real-Time Communication and Ultra Reliable Communication, will require some form of Edge Computing or Cloud-Edge Hybrid environments to deliver applications within those constraints [1] [2]. In addition, these bodies also recognize the rising trends of adoption of Internet of Things (IoT) and Massive Machine Type Communications (mMTC) that will shift the scale of network operations. In this scenario, human intervention in network management and operation will no longer be suitable, requiring the use of some degree of automation in orchestration, as in European Telecommunications Standard Institute (ETSI) Management and Orchestration (MANO) approaches.

The conjunction of Broadband Access Everywhere and the embracing of business verticals within 5G, with highlight to the ability of selling network slices for a given vertical, opens new mobility experiences with direct application to businesses (that nowadays rely on VPN connections for intranet communications). One of the biggest challenges of intranet communications is assuring the security of the network as a whole, not only in regard to access control, data confidentiality and integrity, but actually dealing with existing end-to-end ciphered communications whose content may be nefarious (such as cloaked malware, employees inadvertently leaking data, or legitimate applications overreaching their data scope).

A future enterprise network architecture to process SSL/TLS encrypted traffic using Service Function Chains (SFCs) [3] was already proposed, which addresses not only the security dimension, but also allows to provide some functionality lost by using encryption (such as the ability to optimize content within those connections). Despite being mostly a Cloud solution, deployments in Cloud-Edge hybrid environments were already anticipated, which was shown in a remote office use-case within that work. However, manually orchestrating this architecture as a Network Service would be too complex and error prone (due to the number of functions to be configured, the need to introduce new processing profiles as new terminals/policies are enrolled into the system, and the growing number of Edges that can take part of the service). Because of this, in order to make the architecture practicable (as a 5G slice within a vertical), there is an unavoidable need to research ways to automatically orchestrate and manage it.

In this paper, we evaluated (from quantitative and qualitative perspectives) how an ETSI NFV MANO compliant orchestrator (Open Source MANO (OSM) release FOUR) fares while orchestrating an SFC-enabled SSL/TLS encrypted traffic processing architecture, which supports both edge and cloud deployments. In order to achieve this, we developed ways to interface our Network Functions (NFs) with the orchestration (through primitives), a key part of our evaluation. An outcome of this work is its public demo, which was accepted within the ETSI OSM Proof-of-Concepts (POCs) Framework.[1]

This paper is organized as follows. In Section II, we discuss information available from previous POCs of ETSI OSM and background information about the ETSI MANO standard. In Section III, we detail both the improvements made on top of the previous network architecture to consider edge resources and the development of the orchestration artifacts to automate the architecture's deployment and operation. In Section IV, we first present the quantitative analysis covering both the architecture functionality and the orchestration. In Section V, we present the qualitative analysis of our experience with the orchestration of a complex architecture using ETSI OSM. Finally, in Section VI, we draw the conclusions.

II. Related Work

Network Function Virtualization (NFV) is a paradigm that consists of the virtualisation of the funcionality provided by formerly physical network components [4]. This paradigm shift can improve manageability, flexibility, and scalability of network infrastructures because it breaks the relation between the network infrastructure components and their functionality. Indeed, Virtual Network Functions (VNFs) can be performed by Commercial Off-The-Shelf (COTS) hardware, introducing huge flexibility because of the integration/convergence of network and computational resources.

The NFV adoption leverages the coverage of orchestration efforts that can act potentially over full network topologies. Indeed, it allows the easy adaptation of network services to changes in traffic pattern or operator requirements benefiting from the elastic nature of virtualised resources. The importance of orchestration for future networks brought attention from researchers that tackled specific orchestration aspects, such as dependability [5], security [6] [7], multi-tenancy [8], and orchestrator geo-placement [9].

In order to help the integration of orchestration efforts conducted by operators and vendors, ETSI introduced ETSI NFV MANO Architectural Framework [10], where it provides a frame of reference, to how NFV should be managed and orchestrated. The NFV MANO Architectural Framework is composed of three components: NFV Orchestrator, VNF Manager and Virtualised Infrastructure Manager (VIM).

NFV Orchestrator has two primary concerns: Resource Orchestration and Network Service Orchestration. The NFV Orchestrator, to fulfill its functions as Resource Orchestrator, must verify the resources present and their usage across all VIMs. The Network Service Orchestration functions of NFV Orchestrator are more elaborate since it is responsible for all the actions that must be present to onboard and manage the Network Services and their components.

The VNF Manager is responsible for managing the lifecycle of VNFs. For this purpose, it provides a set of generic functions that apply to any VNF. It also must provide the capability of specification of functions that apply just to a specific VNF. These specific functions can be defined in the VNF package, which consists of the artifacts that actually perform the management tasks required by the VNFs.

VIM manage all the resources provided by the Network Function Virtualization Infrastructure (NFVI) (the MANO component that integrates with infrastructure components, such as OpenStack). The primary responsibility of the VIM is to control and manage the NFVI compute, storage and network resources, and these resources may be in an NFVI-Point-of-Presence (NFVI-PoP) or spread across multiple NFVI-PoPs. A VIM can also be something more than server pool; it can be more specialized like WAN Infrastructure Manager or a PNF Manager (the MANO component that integrates with Physical Network Functions (PNFs)). The ETSI NFV MANO Architectural Framework just concerns with the VIM exposed interfaces and not with its implementation.

ETSI provides an open source solution compliant to the framework: ETSI OSM. In order to evaluate the solution implementation, ETSI promotes a PoC Framework[1] in which OSM members and 3rd-party users demonstrate real-world usage of this orchestrator. Currently it features two PoCs, one which demonstrates a DevOps use-case for Service Function Chains and 5G slices, specifically the assurance of a given Service Level Agreement (SLA) through means of automatically deployable Virtual Test Agents (VTAs) which perform active monitoring of the service against the SLA specification. The other PoC explores Multi-Access Edge Computing scenarios, specifically showing that distributed multi-data center service scenarios are possible, with end-to-end service monitoring and an SDN-enabled VMware Integrated Openstack.

The Linux Foundation's ONAP, a closed-loop NFV orchestrator, also has its own set of demos[2] which are comprised of three major services: the vFirewall, whose the templates required to run the demo are supplied within the demos page; the vLB/vDNS demo, which shows a way to deploy Load Balancing and DNS scaling for the cloud; and lastly the vCPE demo, which shows a way to virtualize a Home Gateway.

The NFV paradigm is also known as an alternative to expensive commercial middleboxes. In this sense, in our previous work [3], we proposed an NFV approach to perform SSL/TLS Inspection (also known as Man-In-The-Middle (MITM)), which is a common functionality offered by commercial middleboxes to process encrypted traffic. Encryption was conceived to hide the content of communication when traversing the network. However, encrypted traffic also hinders the functioning of common network management solutions (*e.g.*, content cache, content optimization, content filter) because these solutions rely on the access to the content to work. In commercial middleboxes, the MITM functionality opens the traffic enabling its processing by their internal functions.

In this architecture, an SFC-enabled MITM forwards the opened traffic to be processed by an Service Function Chaining (SFC) composed by stock solutions for traffic processing acting as NFs. This approach recovers the functionality of these solutions that was lost due to encryption. However, as long as this approach improves effectiveness (recovered functionality), flexibility and scalability (compared to middleboxes), it is also intrusive (hinders privacy) and expensive (imposes performance degradation). In this sense, the SFC-enabled MITM needs to be incorporated into a network architecture that steers the traffic (Classifier functionality) to balance effectiveness, flexibility, scalability, privacy, and performance when processing encrypted traffic.

III. The Network Service and its orchestration

In this section we start by describing the changes made to the architecture of our Network Service (NS), followed by our proposed integration with OSM.

[1]https://osm.etsi.org/wikipub/index.php/OSM_PoC_Framework
[2]https://wiki.onap.org/display/DW/Running+the+ONAP+Demos

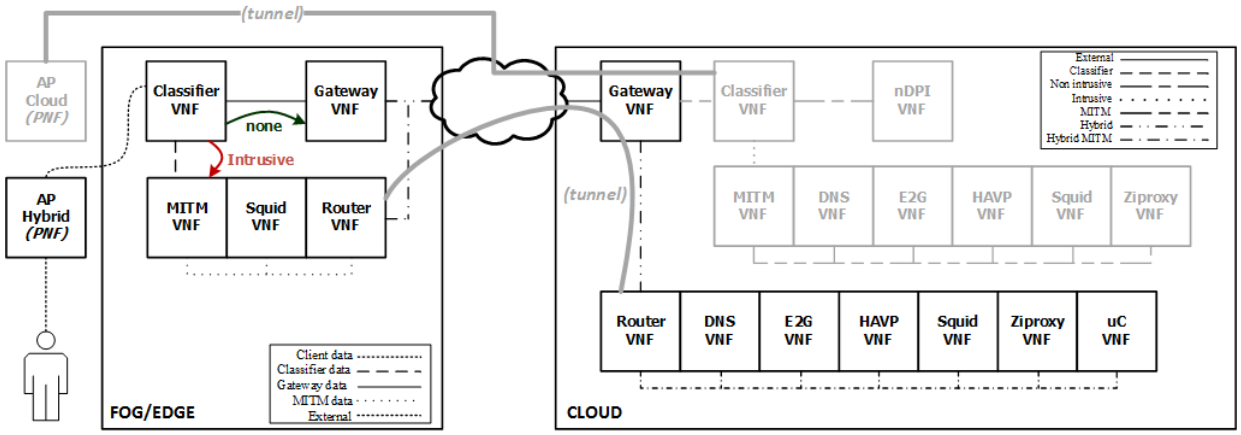

Fig. 1. Hybrid Network Architecture (Data-plane)

A. Changes to the Network Service

We have extended the SSL/TLS traffic processing architecture [3] to improve the SFC-enabled MITM so that it becomes aware of Cloud-Edge hybrid environments, particularly to allow its chains to span across locations, as depicted in Figure 1 (black lines show the hybrid deployment, gray lines show the cloud-only deployment; both will coexist). Because the MITM is transforming end-to-end encrypted traffic into clear-text versions of the protocols, in order to send this opened traffic across locations (Edge to Cloud, and vice-versa), we must provide some means to secure the data while in transit between the sites. There are two fundamental approaches to this matter. The first follows on the concept of slicing. If we are the full owner of the network (or have strong and enforceable slicing contracts with the provider), then we may be able to delegate this security to the underlying network. However, we propose to err on the side of caution and provide the choice of additional security mechanisms (in the chain extension between the Edge-Cloud), such as a new TLS based tunnel between the two sites. This approach assures confidentiality and data integrity between the sites and is actually more suitable for OTT providers, which are arguably a big target of the 5G verticals. It also allows for multi-operator deployments.

In order to have this flexibility, we had to develop a new component (uC for short), which receives the requests opened in the Edge and continues their SFC processing in the Cloud. When the response traffic is received, this component sends that to the Edge (the last NF of the chain in that location) so that it continues the SFC processing in the Edge side before returning the flow to the MITM, which will then send it to the user. In short, we propose to split SFC-enabled MITM in half, leaving the part that acts as server (to the user) in the Edge and the part that acts a client (to the remote) in the Cloud.

B. Proposed integration with OSM

We must specify configuration primitives according to the needed interactions between the OSS/BSS, the NFV-MANO, and the actual functions. We propose two categories of primitives, the ones that manage the connectivity of the NS, and the ones that manage actual functionality within the NS. The former determines service and functions reachability, both inside the NS (between NFs), and the external entities to which the service will be provided. The latter handles NF specific configurations, changing functionality parameters. Table I shows the proposed primitives to orchestrate the NS (as depicted in fig. 1). We must note there are two special primitives. First, the "hello-world" has no real use to the NS other than allowing the later evaluation of OSM overheads. Lastly, the "add-chain / del-chain" primitive will do some reachability configurations within the functionality part of the NFs, taking advantage of function specific proxy capabilities. In particular, the present NS allows the option of performing SFC as a proxy-chain of upstream proxies, which relaxes the requirements of the virtual infrastructure (ie., doing SFC even without native VNF Forwarding Graph capabilities).

IV. Quantitative Evaluation

We will start by presenting the quantitative results gathered from the execution of the primitives stated in the proposal and the functionality provided by the orchestrated Network Service. Then, we will conduct a remote office and data caps optimization case-study, to demonstrate the achieved functionality of the orchestrated Network Service.

Our evaluation was conducted using OSM Release FOUR along with Openstack Ocata (as the NFVI/VIM of choice). OSM was running in a virtual machine with 8 vCPUs and 32 GB of RAM, the Openstack controller was running in *Intel® Xeon® E5-2620 v4* CPU with 32 GB of RAM. The instantiated VMs had 1 vCPU and 2 GB of RAM available.

A. OSM Primitive Execution Times

We started by determining the overhead inflicted by OSM in each primitive, breaking it down each step of the way, as shown in fig. 2. We have observed that the step between the VNF Configuration and Abstraction (VCA) and the Juju Charm is the one that took the most time (over 2 seconds). This is likely due to the message queuing time and best-effort nature in which new requests are processed. The connection

TABLE I
PRIMITIVES DEVELOPED TO ORCHESTRATE THE EDGE SFC-ENABLED MITM

NS Primitive	**Short description**	**NFs**	**Parameters**
`hello-world`	Just to evaluate the orchestration overhead	*dummy*	- none
`add-tunnel` `del-tunnel`	Creates/deletes a tunnel between the Edge Gateway and the Cloud Classifier	Edge Gateway Cloud Classifier Router *(Edge/Cloud)*	- Tunnel endpoint IP - Tunnel technology *(ie. VxLAN/OpenVPN)* [*] optional tunnel keys
`add-route` `del-route`	Creates/deletes routes that influences each NF's routing table. Has the ability to perform NAT, mostly used in the cloud gateway, so that multiple terminations may share the same cloud IP.	*all*	- Network address - Mask [*] optional NATing rules
`add-range` `del-range`	Allows for client activation/deactivation within the Network Service, using its network range (from our own IPAM)	Classifier MITM	- Client IP - Mask
`add-rule` `del-rule`	Creates/deletes rules to decide which kind of traffic processing techniques is applied over matched traffic (intrusive or non-intrusive analysis, via selection of chain number)	Classifier *(Edge/Cloud)*	- 5-tuple - Action (SFC number)
`add-chain` `del-chain`	Configures/deconfigures the NFs which will be used within the SFC-enabled MITM	MITM	- Chain list coded in Base64
NF Primitive	**Short description**	**NFs**	**Parameters**
`add-exclusion` `del-exclusion`	Adds/deletes an URL to the Anti-virus white-list	HAVP	- URL
`add-filter` `del-filter`	Adds/deletes content filters, according to e2guarding filter syntax	e2guardian	- Extra filter file *(encoded in Base64)*
`clean-cache`	Cleans the Squid cache	squid	*(none)*
`add-profile` `del-profile`	Adds content optimization profiles, delete restores default	ziproxy	- Configuration file *(encoded in Base64)*

from the Charm to the NF was the next one, taking about 1.3 s. In part, this may be attributed to the use of SSH as the communication protocol to the NF. The other overheads are of a smaller order, with the connection between the Northbound Interface (NBI) and the VCA taking about 80 ms, and the return of an output from the NF to the Charm taking less than 5 ms. The full process, end-to-end, takes about 3.6 s.

We evaluated the execution time of each primitive in two ways. The first is the individual time, which shows how long it would take to perform that operation once we are already inside the NF (to have a control). This metric completely disregards any time taken to access the machine and be in a ready state to execute the command (it just shows how long the actual command takes to complete). Then, we show the total time elapsed when running the same operation through OSM, which is an end-to-end test that takes into account all orchestration overheads. The results are shown in table II. Outside of the waiting for the tunnels to actually be up (which is one order away, at around 650 ms), NFV-only orchestrator primitives are two orders away from the overheads inflicted by OSM (as evaluated previously in fig. 2). In the context of this NS, this is not really detrimental as the 3.6 seconds of overhead do not meaningfully impact the configuration's responsiveness needs, and are more than recouped by the capabilities made available by the orchestrator.

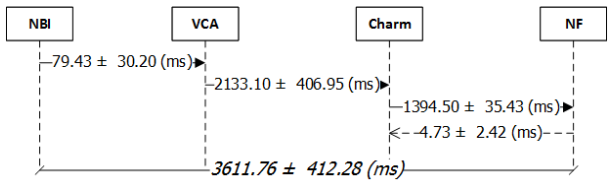

Fig. 2. OSM Primitive Overheads (ms)

We have verified that configuring a given chain is an operation which can be parallelized across the functions of that take part of the chain, meaning that outside of a few liveness checks performed by some NFs (of the next function of the chain), the total configuration time of a chain is close to the time it takes to configure the slower NF (plus the few liveness checks). MITM is the slowest function to be configured (on par with e2guardian), but given the MITM must take part of all intrusive chains, this means that no SFC configuration can ever be faster than about 3.86 s. Nevertheless, the worst case scenario (all NFs in a chain) takes just 5.46 s, being the increase attributable to the fact we have two Squids performing liveness checks of their (respective) next NF. When taking into account all OSM overheads, we can fully configure a chain (day-1) with all NFs (in a single go) in little under 10 s, a very good result for the initial provisioning of the service. Day-2 configurations, the ones that happen after the service is already running, should be made in a more proactive fashion (rather than reactive). Literature and some experimentation suggests that, most NFs herein used, allow for a smooth reconfiguration without causing any downtime of existing connections. Nevertheless, further study is required to determine the duration of a transient state while reconfiguring the NFs, which would be much lower than the seconds which

TABLE II
EXECUTION TIMES OF THE IMPLEMENTED SERVICE PRIMITIVES

Management		Individual Time (ms)	OSM time (ms)
add-tunnel	***VxLAN***		
del-tunnel	creation	33.21 ± 4.60	3644.96 ± 412.30
	actual-up	673.63 ± 147.36	4318.60 ± 437.85
	removal	30.52 ± 4.25	3642.28 ± 412.30
	OpenVPN		
	creation	82.27 ± 9.51	3694.03 ± 412.39
	actual-up	644.01 ± 148.43	4338.03 ± 438.29
	removal	66.38 ± 6.96	3678.14 ± 412.34
add-route		94.15 ± 5.55	3705.90 ± 412.32
del-route		91.83 ± 5.45	3703.59 ± 412.31
add-range		15.56 ± 3.54	3627.32 ± 412.29
del-range		15.88 ± 2.63	3627.64 ± 412.29
add-rule		92.92 ± 5.52	3704.67 ± 412.32
del-rule		99.86 ± 6.06	3711.62 ± 412.32

Management		Individual Time (s)	OSM time (s)
add-chain	***SFC***		
del-chain	all NFs	5.46 ± 0.05	9.07 ± 0.42
	empty	3.91 ± 0.02	7.52 ± 0.41
	(cloud squid+ziproxy)	4.64 ± 0.04	8.25 ± 0.41
	(edge squid + cloud squid+ziproxy)	5.00 ± 0.05	8.62 ± 0.42
	NFs		
	MITM	3.86 ± 0.02	7.47 ± 0.41
	DNS	1.59 ± 0.01	5.20 ± 0.41
	e2guardian	3.86 ± 0.02	7.47 ± 0.41
	HAVP	3.11 ± 0.05	6.72 ± 0.42
	squid	2.76 ± 0.02	6.38 ± 0.41
	ziproxy	3.39 ± 0.04	7.00 ± 0.41

are implied in table II (that evaluates day-1 configuration).

B. Remote Office and Data Caps Optimization Case-Study

The remote office scenario showcases the functionality achieved through the orchestration of this NS. In table III, we evaluated how 8 different SFCs (shown at the top of the table) improve data usage on the User link (U), Edge link (E) and Remote link (R) (shown at the middle, first with VxLAN to inter-connect the Edge to Cloud, then with OpenVPN). We have validated that deploying a Content Cache (Squid) at the Edge does indeed greatly reduce the data caps utilization on the Edge link (chains 2, 4, 6 and 8 – after the first request, in which the cache is populated, the data usage becomes about 1KiB per request). We have also demonstrated that using a Content Optimizer on the Cloud (chains 5 and 6) further helps to reduce the data usage on the Edge link (the page is now one order away from its original size, at about 133KiB down from aprox. 1500KiB). We have also concluded that, in terms of used data, the overheads difference of VxLAN vs OpenVPN is negligible for larger amounts of data (less than 1%). The largest difference was observed when using the Content Optimizer, in which the overheads of OpenVPN rose to about 3% (when compared against VxLAN). The first is likely attributable to the cryptographic overheads of OpenVPN, being the second just a side-effect of the compression within OpenVPN being less effective when the content was already optimized.

Lastly, the page load time (response time) follows the same trend as the data-usage of the Edge link. This means that the processing time overhead inflicted by the Content Optimizer is being beneficially compensated by the lower data usage of the link, making the chain with just the Ziproxy (5) faster than the control (1). When used in conjunction with content caches, the page load time becomes even more favorable (2, 4, 6 and 8), being peek performance achieved when we cache the optimized output of Ziproxy also on the cloud side (making way for gains in multi-edge environments). However, we have also observed that (unlike what happened with data usage) OpenVPN becomes slower (compared to VxLAN) as more data is transmitted within the tunnel. This is likely a consequence of the cryptographic overheads, which may be harder in computational time rather than in added packet data.

V. ETSI OSM QUALITATIVE EVALUATION

In this section, we describe the quantitative evaluation that was conducted to evaluate the suitability of ETSI OSM release FOUR to perform the NS orchestration. First, we describe the NS orchestration requirements which will be used as our evaluation methodology. Afterwards, we present the achieved results (pertaining to OSM itself) divided in three categories: what works well, what needs workarounds, and the limitations.

A. NS orchestration requirements

1) NS composability: which is the ability to compose a NS from the catalog of available VNFs, therefore delivering new functionalities by combining existing NFs. To achieve NS composability, the NSD must be capable of referencing different VNFDs, and organize them into a topology.

2) VNF independence: in order to have NS composability it is necessary that VNFs are independent of each other (ie. the existence or current operation of one does not affect the other), allowing for a loosely coupled architecture.

3) VNFD and NSD versioning: services and functions may improve over time, a natural consequence of the DevOps feedback loop. It is therefore needed to have multiple deployments with different versions to test the impact of changes, allowing testing and upgrade/downgrade of versions when needed.

4) Day 0/1/2 configuration: Day 0 configuration is the most elemental capability, which is the first configuration made during the startup of a new unit (VDU/"VM"). After all the units are up and running, it may be needed to perform another step of configuration to make them work together as a VNF and/or NS, which is called Day 1 configuration. However, day-to-day operations dictate that (often times) NS/VNF configurations need to be updated. These are Day 2 configurations, for instance enrolling new clients or allowing a new tunnel.

5) PNF configuration: NSs may require to integrate some sort of physical equipments (ie. Access Point). These are Physical Network Functions (PNFs) and they need to be configured by some means. The challenge with this kind of equipment is that their interfaces vary from manufacturer to manufacturer, so they may have something standard like SNMP, or some obscure protocol that only applies to a specific series of that manufacturer. Therefore, it is needed to have a PNF configuration framework within the orchestration, so that (1) equipment support can be easily added, and (2) those PNF configurations could be triggered in the Day 1/2 configuration scripts, as part of the service/function logic.

TABLE III
HYBRID SCENARIO - RESOURCES DEPLOYED IN EDGE AND CLOUD ENVIRONMENTS *(TLSv1.2 ECDHE_RSA_AES_256_GCM_SHA384)*

Network Functions deployed within the Service Function Chain – (E - Edge, C - Cloud)

NF	1	2	3	4	5	6	7	8
mitm(E)	✓	✓	✓	✓	✓	✓	✓	✓
squid(E)		✓		✓		✓		✓
squid(C)			✓	✓			✓	✓
ziproxy(C)					✓	✓	✓	✓
uC(C)	✓	✓	✓	✓	✓	✓	✓	✓

Data Transferred (VxLAN) (*Kbytes*) – (U - User link, E - Edge link, R - Remote link)

	1			2			3			4			5			6			7			8		
Req.	U	E	R	U	E	R	U	E	R	U	E	R	U	E	R	U	E	R	U	E	R	U	E	R
1	1505.8	1489.2	1492.2	1511.3	1492.4	1492.1	1513.4	1499.5	1494.1	1510.9	1501.8	1492.9	149.4	133.8	1494.0	148.7	134.3	1491.2	148.4	135.3	1490.6	149.1	135.0	1492.9
2	1514.6	1489.6	1492.3	1521.0	0.9	2.6	1517.9	1494.9	2.6	1524.3	1.0	2.6	148.9	133.7	1493.1	148.9	0.9	2.7	149.0	135.6	2.6	148.6	1.0	2.6
3	1505.8	1490.5	1493.0	1526.1	0.9	2.7	1505.4	1490.9	2.7	1518.1	1.0	2.6	149.1	133.9	1495.1	149.4	0.9	2.6	149.0	136.6	2.6	148.8	1.0	2.6
4	1524.2	1495.9	1492.5	1498.0	0.9	2.6	1524.4	1489.9	2.6	1520.8	1.0	2.6	148.9	133.8	1496.9	149.0	0.9	2.6	149.4	136.0	2.6	148.8	1.0	2.6
5	1513.5	1493.4	1494.6	1523.5	0.9	2.6	1528.0	1494.5	2.6	1524.1	1.0	2.6	149.2	133.9	1493.8	148.7	0.9	2.6	149.2	136.1	2.6	149.8	1.0	2.7

Data Transferred (OpenVPN) (*Kbytes*) – (U - User link, E - Edge link, R - Remote link)

	1			2			3			4			5			6			7			8		
Req.	U	E	R	U	E	R	U	E	R	U	E	R	U	E	R	U	E	R	U	E	R	U	E	R
1	1514.2	1531.1	1494.2	1519.6	1531.1	1491.2	1513.3	1532.2	1492.7	1516.3	1539.6	1492.9	149.3	137.3	1492.9	148.6	138.0	1493.4	148.6	138.2	1490.3	148.4	138.6	1493.3
2	1522.3	1531.1	1493.5	1512.3	0.9	2.6	1523.6	1532.1	2.6	1525.3	1.0	2.6	149.0	137.3	1491.1	149.0	0.9	2.6	148.4	138.1	2.6	149.9	1.0	2.6
3	1517.4	1531.1	1493.6	1523.3	0.9	2.6	1525.4	1532.4	2.6	1508.8	1.0	2.7	148.9	137.3	1494.2	149.4	0.9	2.7	148.9	138.0	2.6	148.4	1.0	2.6
4	1512.2	1531.1	1494.4	1526.6	0.9	2.6	1517.8	1532.1	2.6	1512.7	1.0	2.6	149.1	137.3	1492.5	149.0	0.9	2.6	149.2	138.1	2.7	149.3	1.0	2.6
5	1509.0	1531.1	1495.5	1525.8	0.9	2.6	1522.6	1532.1	2.6	1519.2	1.0	2.6	148.4	137.3	1493.1	148.9	0.9	2.7	149.1	138.1	2.7	149.0	1.0	2.6

Response Time (VxLAN/OpenVPN) (*ms*)

Req.	1	2	3	4	5	6	7	8
1	804/860	816/827	731/809	731/793	661/650	685/682	639/652	668/662
2	748/894	435/399	537/628	431/444	617/600	315/273	296/296	287/285
3	753/825	438/416	530/635	455/434	599/599	285/287	293/299	285/288
4	764/859	442/433	516/657	438/432	598/619	284/289	311/274	303/278
5	761/844	439/437	510/616	448/417	620/603	298/282	281/314	267/275

6) Automatic scaling and Performance management: scaling is a required capability for NSs that need to adjust the amount of resources accordingly to a variable load. It requires performance management information to trigger scale in/out events if necessary. It may seem natural since it just create more VDUs, but this requires particular attention because the new VDUs need configuration, the old VDU might have to be reconfigured, and the traffic needs to be split between the old and new VDU to distribute the load.

7) Scaling groups: is the capability to scale groups of VNFs that are dependent from the perspective of the NS functionality. It is needed to define which VNFs must be scaled together. A scaling group can be a composed by just one VNF, two or three VNFs or all the VNFs that compose the NS. The critical point is that the scaling group must scale all together.

8) VNF healing, update, and Fault management: VNF healing is a desired capability because it allows the operators to repair the internal state of VNFs, which might be behaving out of spec. There is another capability that must be present which is fault management, to help in anomaly detection. Fault management, when combined with VNF healing, helps to reduce downtime and prevent erroneous states. Another significant capability is VNF updates since it allows to patch the VNF, therefore allowing to solve performance or security problems that may have arisen or merely to update the implementation because the specification has changed.

9) NS lifecycle notifications: after the NS has been instantiated and Day 0/1 configurations have been successful, there might be a need to run Day 2 configurations. The OSS/BSS should initiate these configurations since it is the entity that holds the business logic and must decide what the configurations to run and their parameters. So there should be an NS lifecycle notification system to notify the OSS/BSS about the NS state, as proposed by ETSI.

10) SFC support: SFC seamlessly steers traffic at the network level (ie. without reconfiguring the NFs themselves). This allows for better resources and network optimization over the service path, which could be resolved on-the-fly according to policies defined in the NSD and enforced by the SDN-C.

B. Results

1) Works Well: During the orchestration of the proposed Network Service, the most crucial aspect that worked flawlessly and made the whole process easier was NS composability. The modeling provided by the OSM Information Model was adequate and allowed the composition of a Network Service with multiple VNFs, which did not need to be included in the same descriptor of Network Service. The fact that VNFDs were not packaged directly with the NSDs, facilitated updates of the VNFDs because the interfaces and the identifiers could be maintained, but the internal could be changed. VNF interfaces should be static once they are defined, also allowing for VNF independence. The lack of dependence between VNFs allows making changes without having significant concerns about the impact on the other VNFs. It is also essential for NS composability since they are composed of multiple VNFs combined with different orders, which can be made without adverse effects.

Day 0/1/2 configuration is another aspect that is supported by OSM through the use of Cloud-Init and VNF Proxy

Charms. Cloud-Init is a script that runs when a VDU starts up, and this script can do a myriad of operations, such as creating users and groups, update and delete files, amongst other stuff. A Proxy Charm is a Juju charm that runs inside a container that is being controlled by OSM. The VNF Proxy Charm runs after the VNF starts up and is responsible for Day 1/2 configurations. These configurations can be done using a multitude of protocols, such as SSH, SNMP or HTTP. To create a VNF Proxy Charm, the VNF developer can use Python or Ansible. Through the use of this configuration mechanism, the operator that deploys the NS is able to configure all the service primitives defined previously.

2) Workarounds: There are some capabilities, that although implemented, they do not work as intended or do not work at all. The first one is VNFD and NSD versioning. Even though versioning is included in the OSM Information Model, it is not respected. VNFD can have a version, but it is not referenced in the NSD. It is a problem because different versions of the same VNF cannot coexist with the same identifier. The workaround was done inserting a version directly in the identifier, but this, in turn, causes the problem that the same VNFD has multiple identifiers, when it should have one with multiple versions.

Another workaround was the use of Policy Based Routing along with Proxy Chaining to achieve Traffic Steering and Service Function Chaining. Although OSM supports SFC, it was not working due to problems with package versions.

3) Limitations: During the orchestration of the proposed NS, several limitations were found. Starting with the fact that performance and fault management are not working, so there is no way to have this information and act over it, for instance, triggering automatic scaling or VNF healing events.

Without automatic scaling, because there is no mechanism to trigger it, it only remains the manual scaling option, which is not present in OSM. Therefore, as scaling is not working, we cannot test whether scaling groups are working. Although they are in the Information Model and it is possible to define which VNFs should be scaled and in which quantities.

VNF healing and update are not working also. Although it is mentioned in ETSI NFV MANO Architectural Framework, it is not yet in the scope of OSM.

NS lifecycle notifications are another capability that is not yet supported in OSM Release FOUR, which affects the control over what is deployed and the timely fashion of when actions should be executed. For example, soon after our proposed NS is spawned, the client configurations should be added. Also, to be able to start using the service from the moment it is ready, there is the need of a mechanism to inform the OSS/BSS that the NS is ready to receive requests.

PNF configuration is a missing capability that would be useful for this NS use case since we could use vCPEs to establish the VxLAN tunnel between the edge and the cloud. There are also another PNFs that would be useful to be able to configure, such as APs that are in edge premises and could be used to redirect traffic to the classifier in the Fog/Edge. Therefore, PNF configuration would be a useful capability that is not yet present in OSM Release FOUR.

VI. Conclusion

We have evaluated (both quantitative and qualitative) how an ETSI NFV MANO compliant orchestrator (OSM release FOUR) performs while orchestrating an SFC-enabled SSL/TLS encrypted traffic processing architecture, which now supports both edge and cloud deployments. We have developed the required primitives to orchestrate both the NFs and the NS as a whole. The results show that day-0/day-1 configuration can be done in just a few seconds, which is more than enough for the initial provisioning of the service. We have also shown that SFC configuration can be done in a parallelized fashion (at worst, taking less than 10s with all overheads accounted). The successful use of the orchestration (and changes made to the NS architecture) was demonstrated in our data optimization use-case. A public demo of this work was accepted within the ETSI OSM POCs Framework.[1]

Acknowledgment

This work is supported by the European Commission Horizon 2020 Programme under grant agreement number H2020-ICT-2016-1/732497 - 5GinFIRE (Evolving FIRE into a 5G-Oriented Experimental Playground for Vertical industries), the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework [Project Smart EnterCom with Nr. 021949 (POCI-01-0247-FEDER-021949)] and by CAPES within the Ministry of Education of Brazil, under the project FCT number 0409/2016 entitled ”Um Ecossistema para Funções Virtualizadas de Rede”.

References

[1] NGMN Alliance, “NGMN 5G White Paper,” *NGMN*, 2015.

[2] 5GPPP Architecture Working Group, “View on 5G Architecture (Version 2.0),” *5GPPP*, 2017.

[3] V. A. Cunha, M. Carvalho, D. Corujo, J. P. Barraca, D. Gomes, A. E. Schaeffer-Filho, C. R. P. D. Santos, L. Z. Granville, and R. L. Aguiar, “An SFC-enabled approach for processing SSL/TLS encrypted traffic in future enterprise networks,” in *2018 IEEE Symposium on Computers and Communications (ISCC 2018)*, Natal, Brazil, Jun. 2018.

[4] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, “Network function virtualization: State-of-the-art and research challenges,” *IEEE Communications Surveys and Tutorials*, 2016.

[5] A. J. Gonzalez, G. Nencioni, A. Kamisinski, B. E. Helvik, and P. E. Heegaard, “Dependability of the NFV Orchestrator: State of the Art and Research Challenges,” *IEEE Communications Surveys Tutorials*, 2018.

[6] B. Jaeger, “Security Orchestrator: Introducing a Security Orchestrator in the Context of the ETSI NFV Reference Architecture,” in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 1255–1260.

[7] M. Pattaranantakul, Y. Tseng, R. He, Z. Zhang, and A. Meddahi, “A First Step Towards Security Extension for NFV Orchestrator,” in *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization (SDN-NFVSec ’17)*. New York, NY, USA: ACM, 2017, pp. 25–30.

[8] R. Muñoz, R. Vilalta, R. Casellas, R. Martinez, T. Szyrkowiec, A. Autenrieth, V. López, and D. López, “Integrated SDN/NFV Management and Orchestration Architecture for Dynamic Deployment of Virtual SDN Control Instances for Virtual Tenant Networks,” *J. Opt. Commun. Netw.*, vol. 7, no. 11, pp. B62—-B70, nov 2015.

[9] M. Abu-Lebdeh, D. Naboulsi, R. Glitho, and C. W. Tchouati, “NFV orchestrator placement for geo-distributed systems,” in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, 2017, pp. 1–5.

[10] ETSI, “Network Functions Virtualisation (NFV); Management and Orchestration,” *GS NFV-MAN 001 V1.1.1*, 2014.