SELFNET Virtual Network Functions Manager: A Common Approach For Lifecycle Management Of NFV Applications

G. Bernini, E. Kraja, G. Carrozzo, G. Landi, N. Ciulli
Nextworks, Pisa, Italy
e-mail: g.bernini@nextworks.it

***Abstract* - Deployment of services in 5G networks needs to follow highly dynamic and flexible approaches to reduce the provisioning time from 90 minutes to 90 seconds. Network Function Virtualization (NFV) is a key technology to enable very dynamic service lifecycles in 5G networks, including multiple recurrent provisioning and maintenance activities on installed services. The SELFNET research project, funded by the EC under the 5G Public Private Partnership within H2020 framework programme, envisages an advanced self-organized management infrastructure where NFV applications can be quickly plugged into cloud infrastructures and automatically configured. As such, SELFNET offers tools and systems to implement the new management infrastructure of the 5G networks. This paper presents the SELFNET approach to management of NFV applications lifecycle by means of a Virtual Network Function Manager (VNFM) and describes its experimental validation and demonstration.**

Keywords - NFV, Cloud, VNFs, MANO, VNFM, demonstration

I. INTRODUCTION

5G networks pose challenging requirements in terms of automation and performance for deployment of new services. When combined with the rapid development of the cloud market and the recent updates and advances on Network Function Virtualization (NFV) and Software Defined Networking (SDN) technologies, it becomes clear that network operators need to look at their infrastructures as more and more dynamic, flexible and reactive. As a consequence, the aspect of network management is being critical for the efficient deployment and maintenance of new services in 5G networks. Common and homogeneous mechanisms and procedures are needed to manage the whole lifecycle of NFV applications (i.e. instantiation, configuration, start, stop, scale, termination, etc.) irrespectively of their specific logic or function. This means that NFV applications need to be properly encapsulated to achieve a high degree of automation in the deployment of 5G services. NFV applications refer to Virtualized Network Functions (VNFs) implementing either control or data plane functionalities on top of a virtualized infrastructure managed by a cloud management system.

The SELFNET research project has been funded by the EC under the Phase 1 of the 5G Public Private Partnership within H2020 framework programme, to specifically address these network management challenges of 5G networks. It relies on a self-organized management platform composed of several layers and internal sublayers [1], as depicted in Fig. 1.

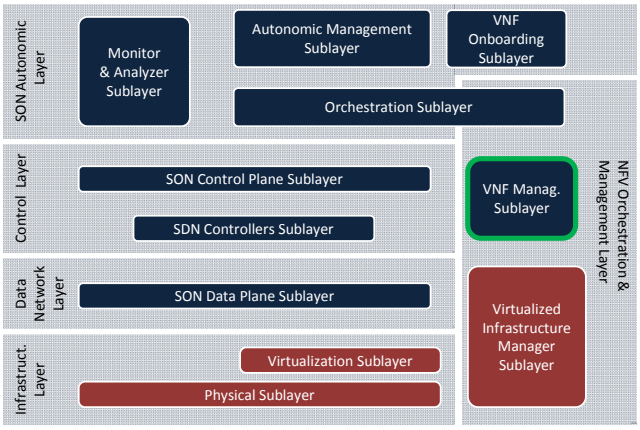

Fig. 1. SELFNET architecture overview.

The layers in Fig. 1 represent the main blocks of the SELFNET architecture, with a key role of generic lifecycle management of NFV applications provided by the VNF Manager (VNFM), which is the key focus of the demonstration described in this paper. Specific functions executed within each sublayer are as follows:

- *Infrastructure Layer*: for the physical (Physical sublayer) and virtual (Virtualization sublayer) resources to be deployed;
- *Data Network Layer*: contains all the data plane network functions including those that only have data plane functionalities, as well as the ones that have both control and data plane capabilities;
- *Control Layer*: includes the SELFNET control plane functions, as well as functions that include both control and data plane capacities. SDN Controllers are also part of this layer, exposing northbound services to network functions that contain only control plane features;
- *NFV Orchestration & Management layer*: for orchestration and lifecycle management of the virtual resources (at VIM) and of the VNFs (VNFM sublayer).
- *Self-Organized Network (SON) Autonomic Layer*: provides the SELFNET SON-related functionalities on

top of the abovementioned network functions and infrastructure. It includes sensors collection and analysis, machine-based decision making algorithms, as well as the orchestration functionalities to enforce decisions on the network domain;

This paper addresses specifically the lifecycle management of VNFs by means of the SELFNET VNFM. At first we present the SELFNET architecture concepts (Sec. **Error! Reference source not found.**) and the specific SELFNET VNFM design elements (Sec. II). In Sec. III the prototype demonstration is described to highlight how SELFNET sensors and actuators NFV applications are dynamically and automatically plugged into the SELFNET virtualized network infrastructure.

II. SELFNET VNFM

A. Architecture overview

The reference baseline for the NFV applications encapsulation in SELFENT is the ETSI MANO framework [2], with the VNFM as key component responsible for the lifecycle management of all the SELFNET sensor and actuator VNFs. ETSI MANO provides orchestration and lifecycle management for physical and software resources building the NFV Infrastructure (NFVI) on the one hand, and for VNFs on the other. Therefore ETSI MANO focuses on all virtualization specific management tasks within the NFV architecture, and it is built by three main components: the Virtual Infrastructure Manager (VIM), the VNFM and the NFV Orchestrator (NFVO).

The SELFNET encapsulation of NFV applications aims to provide a unified and common approach for the lifecycle management of sensor and actuator VNFs, thus exposing primitives to instantiate and control the VNFs towards other SELFNET components and layers, like the Orchestration sublayer. In the ETSI MANO architecture, the VNFM is responsible for the lifecycle management of all the VNF instances deployed within the virtualized infrastructure. The VNFM functions can be considered as generic and common functions applicable to any type of VNF, like instantiation and configuration, scale, modification and termination. Therefore, the VNFM is natively the most suitable candidate to implement all those mechanisms needed in SELFNET to encapsulate sensor and actuator VNFs and expose towards upper SELFNET layers unified VNF management primitives. The SELFNET VNFM implements and enhances the following VNF lifecycle management mechanisms and workflows specified by the ETSI MANO:

- VNF instantiation, including VNF configuration according to the VNF Descriptor (VNFD), which describes attributes and requirements to realize such VNF and instantiate it in the NFV infrastructure.
- VNF instance modification, that basically consists into an update of the VNF configuration
- VNF instance scale out/in (i.e. allocate or terminate Virtual Machines in support of a given VNF) and up/down (i.e. increase or decrease virtual resources for a given VNF), consisting into a structural modification of the VNF instance, in terms of virtual resources allocated
- VNF instance-related collection of performance measurements from the VNF
- VNF instance termination
- Management of integrity and consistency of the VNF instance through its lifecycle

Fig. 2 illustrates the high level SELFNET VNFM functional split and highlights its main functional entities and the interactions with external components of the SELFNET architecture. This functional split also includes those lifecycle management entities needed at the VNF side in support of the SELFNET VNFM operations.

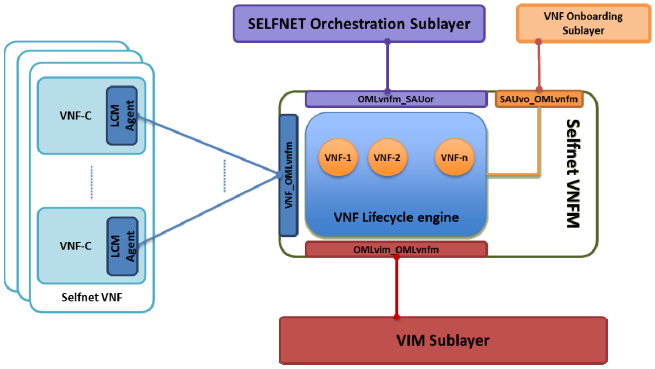

Fig. 2. SELFNET VNFM functional decomposition.

Each VNF is generally composed of a set of VNF components (VNF-C). A VNF-C is associated with a VNF software component, and a 1:1 correspondence exists between VNF-Cs and Virtual Machines. The way VNFs are structured into VNFCs is out of the scope of this work and generally depends on specific VNF vendor and developer factors. For each VNF-C, a lifecycle management (LCM) agent is embedded in the correspondent Virtual Machine to enable the communication with the SELFNET VNFM and implement the lifecycle actions on the VNF. The agent enables containerization of VNFs into encapsulated VNFs and provide a common interface towards the SELFNET VNFM. The core component of the SELFNET VNFM is the VNF Lifecycle Engine, which implements the per-VNF lifecycle management logic and intelligence. It takes care to maintain the state and integrity of each VNF, and for each request coming from the driver it applies and performs the proper actions. In particular, each VNF is maintained within this lifecycle engine with its own finite state machine to properly manage the evolution of its lifecycle. The SELFNET VNFM manages VNF states and lifecycle transitions according to the ETSI NFV VNF software architecture (SWA) specification [3].

B. VNF information model

As per ETSI MANO specifications, the instantiation and operational behavior of each VNF is described in a template called VNFD. The SELFNET VNFM also uses VNFDs to create VNF instances and to manage their lifecycle according to the attributes specified. In particular, during instantiation, virtual resources (VMs, storage, network, CPU) are assigned to

VNFs and VNF-Cs based on requirements and parameters included in the VNFD, which also contains resource allocation criteria, description of VNF composition (i.e. number and types of VNF-Cs), VNF functional scripts for specific lifecycle events. Different versions of a VNFD corresponds to different implementations of the same function, different versions to run in different execution environments (e.g. on different hypervisors, dependent on NFVI resources availability information, etc.), or different release versions of the same software.

When a VNF is instantiated, the SELFNET VNFM creates a VNF Record (VNFR) to index the virtual resources allocated to the given VNF instance and its VNF-Cs. The VNFR is created to model the VNF instance, and it is built based on the content of the related VNFD augmented with additional runtime VNF-C information depending on the given instantiation (e.g. virtual resource identifiers, lifecycle history, etc.). It also includes information to operate changes to the deployed VNF instance, e.g. in the case of a scalability update. Moreover, information related to the VNF-C instances associated to VDUs included in the VNFD represent the core part of the VNFR.

C. *Proof-of-concept prototype*

We developed the SELFNET VNFM prototype on top of the OpenBaton opensource project [4]. OpenBaton is compliant with the ETSI MANO specifications and workflows for VNF lifecycle management, and it can be easily integrated with existing cloud platforms and adapted to different types of VNFs. In particular, at the VIM level, OpenBaton supports the integration with (multi-site) OpenStack environments and it also provides an SDK to implement VIM-specific drivers. In summary, OpenBaton implements an NFVO with an integrated and general purpose VNFM.

The SELFNET VNFM has been developed starting from the OpenBaton general purpose VNFM, which has been enhanced with the following functionalities:

- Usage of *vnfm-sdk-rest* instead of *vnfm-sdk-amqp*: this enables the SELFNET VNFM to expose REST APIs at its northbound, thus easing the integration with the SELFNET Orchestration sublayer.
- Enhancement of the VNF instantiation procedure and interface to enable tenant awareness in the SELFNET VNFM. In particular the northbound REST API for VNF instantiation has been extended to include information about the tenant to which the VNF belongs. This information is then managed and kept by the SELFNET VNFM to isolate VNFs belonging to different tenants
- Implementation of VNF modification lifecycle management service (i.e. VNF re-configuration), that was not originally supported by the generic VNFM and has been implemented from scratch
- Enhancement of the native OpenBaton LCM agent to retrieve VNF performance statistics.

This SELFNET VNFM prototype follows the ETSI MANO lifecycle management approach where resource allocation during VNF instantiation is performed by the NFVO (i.e. the SELFNET Orchestration sublayer). Therefore it does not interface directly with the VIM, which is assumed to be OpenStack.

III. Experimental Validation and Demonstration

The SELFNET VNFM prototype has been validated and demonstrated to implement automated instantiations of sample dummy virtual Firewall VNFs. These sample VNFs emulate virtual Firewalls embedding a Netconf [5] server to configure sample rules through the LCM agent based on a dedicated YANG model. This validation has been carried out deploying the SELFNET VNFM on top of an OpenStack Liberty VIM controlling a single server NFV lab infrastructure. To enable the full lifecycle management operations, the native OpenBaton NFVO is also deployed on top of the SELFNET VNFM to take care of the Virtual Machines and virtual networks allocation in the OpenStack VIM. The overall demonstration scenario is depicted in Fig. 3.

According to the SELFNET VNFM operation, a dedicated VNFD has been prepared for the virtual Firewall VNF. This VNFD includes the reference to the lifecycle scripts to be executed by the LCM agent during the VNF instantiation and VNF modification (i.e. re-configuration). The OpenBaton NFVO, in particular by means of its dashboard, is used to trigger the VNF instantiation and re-configuration, which are the core lifecycle operations showcased in the demonstration.

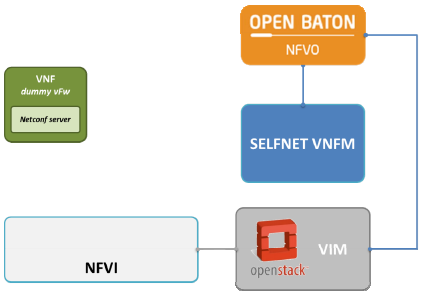

Fig. 3. Demonstration scenario.

The SELFNET VNFM demonstration basically covers two lifecycle management operations: instantiation and automated re-configuration. A step-wise description of the virtual Firewall VNF instantiation operation is described as follows (and depicted in Fig. 4):

1. virtual Firewall VNF instantiation request issued from the OpenBaton NFVO dashboard;
2. VNF instantiation request forwarded from the NFVO to the VNFM, including VNFD and LCM scripts (one for instantiation and one for re-configuration)

3. VNF instantiation request validated by the VNFM, with parsing of the VNFD and retrieval of virtual resources to be allocated for the VNF
4. Virtual resources allocation request issued from VNFM to NFVO
5. NFVO allocates virtual resources (Virtual Machine, subnets, ports, etc.) into the VIM for the virtual Firewall VNF
6. NFVO provides identifiers of virtual resources allocated in the VIM to the VNFM
7. VNFM associates resource identifiers to the VNF and generates the correspondent VNFR
8. VNFM saves LCM scripts into the VNF by interacting with the LCM agent, and triggers the execution of the LCM script for the instantiation operation by means of the LCM agent, that in this particular virtual Firewall case provides an initial configuration into the Netconf server embedded in VNF.
9. Once the execution of the LCM script for instantiation is performed, the LCM agent acknowledge the VNFM
10. VNFM sends a successful VNF instantiation response to NFVO.

Once a virtual Firewall VNF instance is deployed and configured, it can be re-configured at any time through the northbound REST API of VNFM. In the case of the virtual Firewall this operation may imply an addition of a new rule, as well as a deletion or modification of an existing one. A step-wise description of the virtual Firewall VNF re-configuration operation is described as follows (and depicted in Fig. 5):

1. VNF re-configuration request sent to the SELFNET VNFM, including new parameters to be used as inputs in the LCM script for re-configuration
2. VNFM processes the request to retrieve the Virtual Machine associated with the given VNF to apply the new configuration
3. VNFM triggers the execution of the LCM script for the re-configuration operation by means of the LCM agent.
4. LCM agent executes the script for re-configuration, which update the Netconf server configuration, in this case with a new virtual Firewall rule
5. Once the execution of the LCM script for re-configuration is done, the LCM agent acknowledge the VNFM, which in turn sends a successful VNF re-configuration response to the NFVO

CONCLUSIONS

This paper has presented the SELFNET VNFM architecture and prototype, which is a tool for NFV applications encapsulation and common lifecycle management of VNFs suitable for 5G networks and services. Experimental validation and demonstration are also described.

In future work we will test performances of the SELFNET VNFM in specific 5G use cases, which are under development in the SELFNET project. These use cases are specifically aimed at assessing prototype functions in specific 5G network scenarios of self-optimization of resources (to move resources across the network to cope with different user/application profiles over time), self-healing of network services (to rapidly react and repair failures in 5G networks) and self-protection (to automatically readapt security network service chains when new network sections are activated.

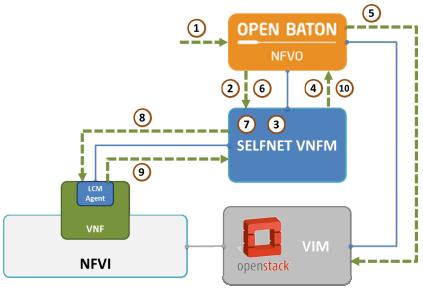

Fig. 4. VNF instantiation workflow

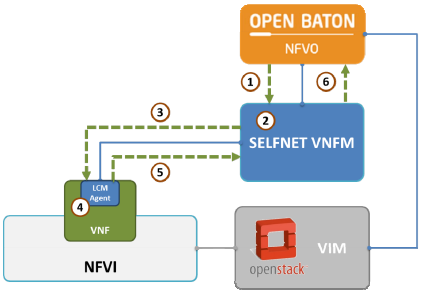

Fig. 5. VNF re-configuration workflow

ACKNOWLEDGMENT

This work was partially funded by the European Commission Horizon 2020 5G-PPP Programme under Grant Agreement number H2020-ICT-2014-2/671672 - SELFNET (Framework for Self-Organized Network Management in Virtualized and Software-Defined Networks).

REFERENCES

[1] P. Neves, et. al, "The SELFNET Approach for Autonomic Management in an NFV/SDN Networking Paradigm", International Journal of Distributed Sensor Networks Volume 2016 (2016), Article ID 2897479.

[2] ETSI GS NFV-MAN 001 V1.1.1, "Network Functions Virtualisation (NFV); Management and Orchestration", ETSI NFV ISG, December 2014.

[3] ETSI GS NFV-SWA 001 V1.1.1, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture", ETSI NFV ISG, December 2014.

[4] OpenBaton, http://openbaton.github.io/index.html .

[5] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network configuration protocol (NETCONF)," IETF RFC 6241, June 2011