ETSO: End-To-End SFC Orchestration Framework

Marouen Mechtri
Orange Labs
Paris, France
marouane.mechteri@orange.com

Chaima Ghribi, Oussama Soualah, Djamal Zeghlache
UMR 5157 CNRS Samovar, Telecom SudParis
Université Paris Saclay, France
{chaima.ghribi, oussama.soualah, djamal.zeghlache}@telecom-sudparis.eu

***Abstract*—This demo presents a SFC orchestrator that provides intelligent, dynamic and automated VNFs and network deployment over heterogeneous NFV and SDN enabled Cloud environments. The proposed architecture is based on the ETSI NFV reference architecture and TOSCA standards. In this demo, reliable VNF placement and chaining is ensured using the algorithm proposed in [1] "A Link Failure Recovery Algorithm For Virtual Network Function Chaining".**

I. ETSO ORCHESTRATION FRAMEWORK

The End-To-End SFC Orchestration Framework (ETSO), which is an extension of our work in [2], is a modular orchestration solution that addresses NFV challenges. Our orchestration framework combines requirements from both NFV and SDN. It also enables VNF orchestration in heterogeneous NFV and SDN environments consisting of different Cloud orchestrators (e.g OpenStack Heat, Cloudify ...) and SDN controllers such as OpenDaylight and ONOS. We rely on a family of VNF placement and chaining algorithms that brings intelligence to the orchestrator. Placement decisions are taken according to different strategies to meet SLA requirements, use resources efficiently and handle failure recovery. We have also extended the Tosca standard language that describes service templates for cloud applications to deal with the network service description challenge.

Figure 1 depicts our proposed VNF orchestration framework. The architecture can be easily extended thanks to its modularity and plugin model and the used REST APIs for key interfaces. The functions and role of each module are specified next along with their relationships to ensure automated network services chaining.

SFC Orchestrator: is the service instantiation and management engine that ensures orchestration of all components required for VNF service graph deployment. This is achieved on the basis of the tenant or user request for distributed and interconnected network services.

Tosca Parser: interprets the user request and verifies the request correctness for validation prior to any orchestration by the SFC orchestrator. This module is based on the native TOSCA parser and extends the TOSCA normative types, initially limited to Cloud applications, by adding new specific nodes and features to support NFV.

Request Manager: builds the data input files and information necessary for the operation of all other modules.

Intelligent Placement Module: is an essential component of the architecture as it embeds a family of optimization algorithms to intelligently place VNFs in underlying NFV Infrastructures (NFVI) and to steer traffic flows across the VNFs while using efficiently the infrastructure resources.

Simulator: this component allows scientists and developers of VNF chaining and placement algorithms to integrate their solutions in the overall framework to evaluate performance in a realistic experimental setting (on real hardware) and compare the results with other algorithms. Some algorithms are already integrated in the framework and can be selected, invoked and activated for analysis and comparison purposes with the ability to reproduce the simulation scenarios conditions and settings.

Monitoring Module: monitors the allocated resources and the NFVI. The Monitoring Module feeds the orchestrator with the information (resources status and availability) needed to ensure efficient VNF placement and failure recovery.

NCT[1] Translator: is responsible for translating the user requested topology defined though the TOSCA specification to the appropriate template language of the Cloud Orchestrator (e.g. OpenStack Heat, Cloudify...).

NCT Manager: is responsible for VNFs instantiation on a single Cloud or multiple Cloud environments. This module can invoke different Cloud orchestrators to deploy and configure VNFs thanks to its plugin oriented design.

SFC Manager: is responsible for the SFC or network resources instantiation. The SFC manager can handle, interact and communicate with different SDN controllers (such as ODL, ONOS,...) thanks to dedicated plugins.

II. SFC INSTANTIATION USING ETSO FRAMEWORK

In this section, we will highlight the interactions between the different components of the proposed architecture and describe how the orchestrator invokes other modules to provide a reliable VNF placement and chaining. First, when the orchestrator receives a request (written in yaml) it invokes the Tosca parser module for validation. After that, the orchestrator relies on the request manager to generate three data input files. One for the NCT translator, another for the intelligent placement and a third one for the SFC manager module. Note that the generation of the placement input file is fundamental since the placement algorithm need not only the requested capacities of the NCT resources (which exist in the NCT template) but also the ordered list of the VNFs representing

[1]Network Connectivity Topology (NCT) specifies the virtual network function (VNF) nodes that compose the global service and the connection between them

978-3-901882-89-0 @2017 IFIP

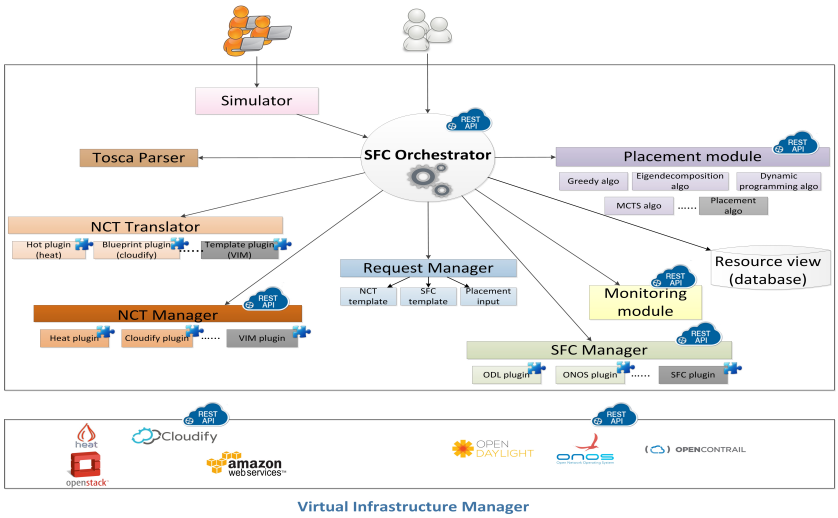

Fig. 1. SFC Orchestrator Framework

the requested SFCs (information described only in the SFC template).

The orchestrator calls next the intelligent placement module to make VNF placement and chaining decisions using one of the available algorithms (e.g. algorithm in [1]). This placement module invokes the monitoring module to get an overview on the infrastructure status, required as an input to the optimization.

In this demo, we use the OpenStack heat as the cloud orchestrator. To reflect the optimization decisions made by our algorithms and to impose VM placement to OpenStack Heat, we assigned to each physical server a unique zone. The idea is to disable the decisions of the OpenStack scheduler. In fact, in heat template we can assign an availability zone to a VM but we can not specify the server name in the VM location.

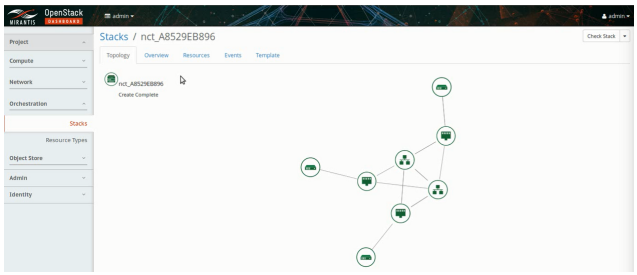

Fig. 2. NCT instantiation

Following placement decisions, the orchestrator updates the NCT template with the VNFs location information and calls the NCT translator module to transform the NCT template to a HOT (Heat Orchestration Template) template. After that, the orchestrator invokes the NCT manager to instantiate the NCT. The NCT manager relies on the heat plugin to instantiate the requested resources in OpenStack. Fig. 2 shows a screenshot of the instantiated NCT graph in the cloud platform. This graph is composed by three requested VMs and the associated networks and ports that interconnect the VMs between them.

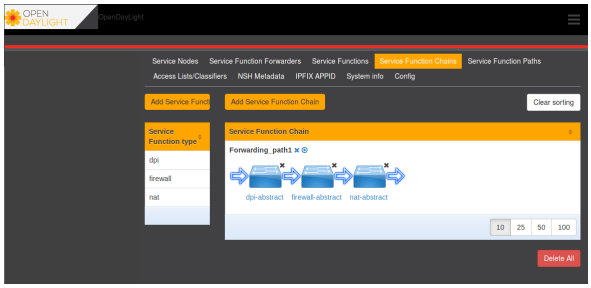

Fig. 3. Forwarding Path and SFC instantiation

Finally, the orchestrator retrieves information about all the instantiated resources (VMs IP addresses) and updates the SFC template with connectivity information to invoke the SFC manager module that instantiates the SFCs and their associated forwarding paths. The SFC manager relies on the ODL plugin to invoke the SFC feature of OpenDaylight in order to instantiate the requested SFCs and forwarding paths. Fig. 3 shows the instantiated forwarding paths composed by a dpi, firewall and nat services. Note that each VNF component is instantiated in a VM.

III. Conclusion

We present the ETSO Framework for end-to-end SFC chaining and show how it ensures automated and reliable SFC deployment over NFV and SDN enabled Cloud environments.

References

[1] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "A Link Failure Recovery Algorithm For Virtual Network Function Chaining," in the 14th IFIP/IEEE International Symposium on Integrated Network Management, 2017, Lisbon, Portugal.

[2] M. Mechtri, I. G. Benyahia, and D. Zeghlache, "Agile service manager for 5g," in 2016 IEEE/IFIP Network Operations and Management Symposium, NOMS 2016, Istanbul, Turkey, April 25-29, 2016, 2016, pp. 1285–1290. [Online]. Available: http://dx.doi.org/10.1109/NOMS.2016.7503004