Multi-layered service orchestration in a multi-domain network environment

Attila Csoma[1], Balázs Sonkoly[1], Levente Csikor[1], Felicián Németh[1], András Gulyás[1]
Dávid Jocha[2], János Elek[2], Wouter Tavernier[3], Sahel Sahhaf[3]
[1] *BME, Hungary;* [2] *Ericsson Research, Hungary;* [3] *iMinds, Belgium*

***Abstract*—In this demo, we show a novel method to multi-layer service orchestration in a multi-domain network. This method is a basic implementation of the three layered concept with multi-layer orchestration designed by the UNIFY project. A global orchestrator is capable of instantiating service elements, i.e., virtual network functions (VNFs), in separate domains. Dedicated local orchestrators in different infrastructure domains are responsible for setting up new VNF instances and configuring the underlying network. Our implementation is based on the ESCAPE prototyping framework and an OpenStack (OS) data center with the OpenDaylight (ODL) controller.**

I. Introduction

In today's networks, services are strongly coupled to the physical topology, the capabilities of expensive middleboxes and the placement of special purpose hardware elements. As a consequence, service provisioning and service deployment have several limitations in terms of dynamicity, scalability, flexibility and optimal usage of resources. Network Functions Virtualization (NFV) is an effort to make telecommunication services and service components software-based as much as possible[1]. By this means, whole services or service elements can run in virtualized environment on a wide range of general purpose hardwares which makes service deployment, configuration and operation easier. Moreover, the usage of different types of resources (e.g., compute and storage) can be optimized in a more flexible way and several tools are available from the Cloud world. Besides packet processing tasks assigned to (physical or virtual) network functions, steering traffic flows between these service elements is an indispensable part of service provisioning and SDN enables to realize it efficiently. Furthermore, we can invoke service chains or more generally service graphs in order to describe high level services in a generic way and to assemble processing flows (series of network functions) for given traffic[2].

UNIFY (http://www.fp7-unify.eu) is an EU-funded FP7 project[3], which aims at unifying Cloud and carrier networks by developing an automated, dynamic service creation architecture based on a *dynamic fine-granular service chaining* model leveraging Cloud virtualization techniques and SDN. The architecture proposed by UNIFY comprises three relevant layers (see left part of Fig. 1). *Service layer* is aware of the service logic, handles service requests, and is responsible for SLAs. The multi-level *Orchestration layer* is responsible for mapping service requests to available resources exposed by multiple domains with different capabilities. The unified orchestration modules at different levels optimize the usage of different types of resources based on an abstract network and resource view (global or local). *Infrastructure layer* contains physical and virtual resources including compute, storage and networking resources.

In [1], we have presented a prototyping framework, named ESCAPE, for this architecture supporting single-layer service orchestration. In this demonstration, we extend that framework with multi-layer orchestration over multiple domains. Our unified orchestration framework supports light-weight, virtual domains realized by Mininet [2] and data centers managed by OpenStack [3] as well. It is capable of setting up and configuring service chains on demand, mapping virtual network functions (VNFs) to resources, steering traffic according to chains' policies, and providing real-time management information on running VNFs. The system makes use of widely used tools, such as Click [4], POX [5], OpenDaylight [6] and NETCONF integrated into a common framework.

II. Architecture

The main components of our framework are shown in Fig. 1. Following UNIFY project's approach, the orchestration is spanning through multiple domains and service graphs can be instantiated by third parties. For this reason, we developed communication interfaces and an abstract view of a whole domain which can be adopted by a global orchestrator to use the resources in a separate domain through its local orchestrator. Our framework currently supports Mininet-based virtual domains and data centers managed by OpenStack/OpenDaylight called as OS/ODL domain. VNFs are implemented in Click and in case of a Mininet domain, they run as distinct processes with configurable isolation models, while in OS/ODL domain, virtual machines are deployed to run Click processes. We have a VNF catalog storing available and deployable network functions. The infrastructure comprises OpenFlow switches and VNF containers (managed nodes) hosting VNFs, while a dedicated controller application (implemented in POX) is responsible for traffic steering.

Our aim is to provide an abstract view of our OS/ODL domain for the ESCAPE orchestrator. We expose the whole domain as a simple node with the capability of service graph instantiation which is managed solely by the domain's local orchestrator. By the global orchestrator of ESCAPE, this domain is treated as another VNF container. The global orchestrator has a global view of the available resources and the capabilities of each node running in its domain. It is capable of

[1]ETSI has a dedicated working group on NFV.
[2]IETF has a dedicated working group (Service Function Chaining Working Group) dealing with several aspects of the service chaining architecture.
[3]This work was conducted within the framework of the FP7 UNIFY project, which is partially funded by the Commission of the European Union.

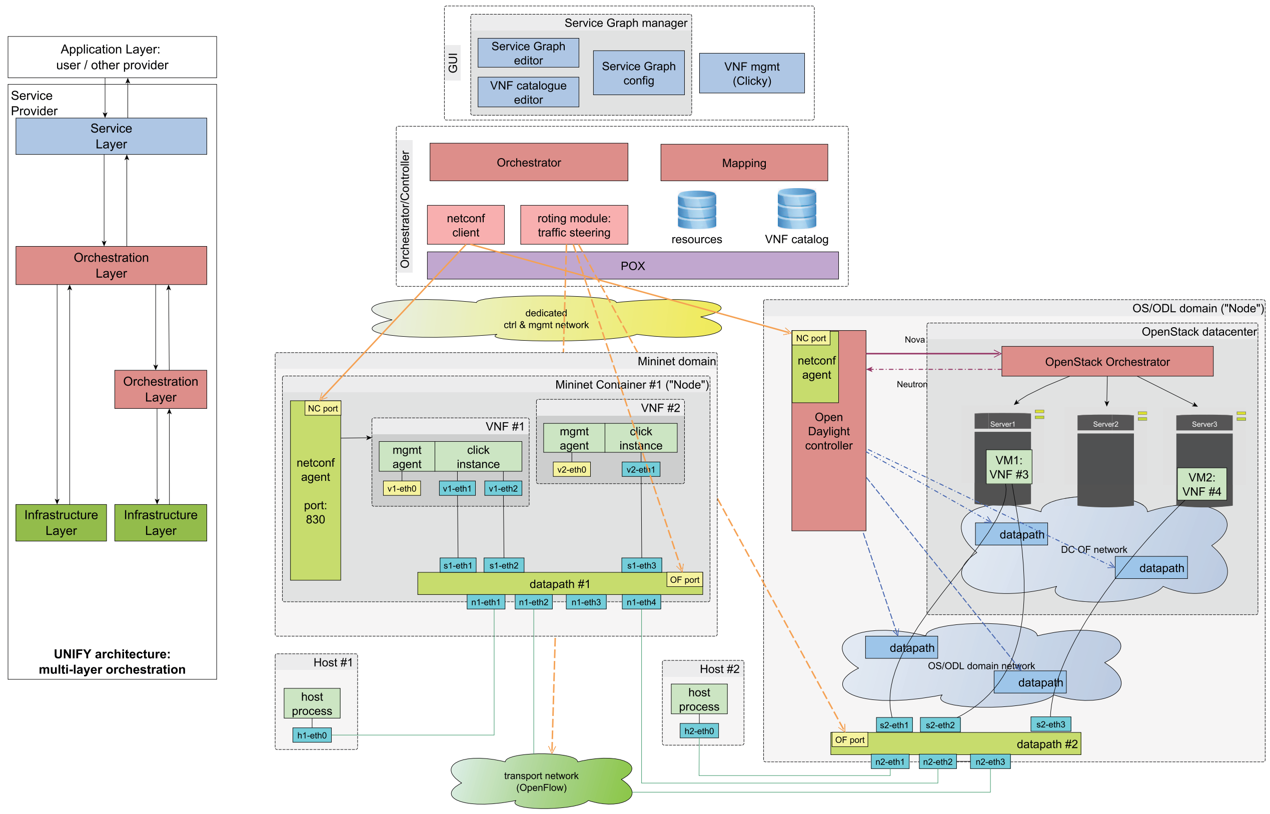

Fig. 1. Logical view of the demo scenario. On the left the UNIFY project's concept with multi-layer orchestration, on the right our demo's logical elements. The corresponding main components to the layers are the GUI, the global orchestrator and the Mininet and OS/ODL domains, the latter with a local orchestrator.

partitioning a service graph into multiple subgraphs which can be given to the OS/ODL domain for further decomposition and instantiation. In the simplest case the subgraph could comprise only one network function which would be instantiated in the OS/ODL domain's data center. After successful instantiation of the subgraph, the OS/ODL domain's local orchestrator notifies the global orchestrator and provides information on access to the newly created network function.

Similarly to Mininet containers, OS/ODL domain has two control interfaces implemented by dedicated components. The first one is an OVS switch steering traffic between the edge of the domain and specific ports on which the instantiated network functions are reachable. The second one is a NETCONF agent module which is implemented in the ODL controller and tightly integrated with its framework. The plugin calls the REST API of the OS ("Nova") to request a network function which is then initiated in the data center as a virtual machine. After the network function is booted up and the relevant network configurations are deployed in the data center's network, the ODL controller sets up an overlay topology in the domain's network and creates a new port in the OVS switch which is directly connected to the overlay. This port's id is signaled back to the global orchestrator through the NETCONF protocol. Note that the edge domain OVS switch has only one master controller which is the global orchestrator. The ODL controller has the right to create ports on this switch but nothing more. In our implementation the OS cloud's internal networking is controlled by the same ODL controller which is responsible for the domain network. The OS makes network configuration requests to the ODL via it's "Neutron" REST API. Then ODL configures the DC's internal datapath elements via OpenFlow.

During the demo, a complex service described by a service graph (with given requirements) will be requested via the service layer's global orchestrator's GUI. This complex service will be decomposed by the orchestrator to smaller blocks. Some of these blocks will be instantiated in Mininet containers, while others in the OS/ODL cloud which makes further local orchestration. This decomposition and instantiation details are hidden from the requestor of the high level complex service.

References

[1] A. Csoma, B. Sonkoly, L. Csikor, F. Németh, A. Gulyás, W. Tavernier, and S. Sahhaf, "ESCAPE: Extensible service chain prototyping environment using mininet, click, netconf and pox," in *Proc. of the ACM SIGCOMM 2014*, 2014.

[2] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *ACM HotNets 2010*.

[3] "Openstack: Open source cloud computing software," 2014. [Online]. Available: https://www.openstack.org/

[4] E. Kohler *et al.*, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, Aug. 2000.

[5] "The POX controller," 2014. [Online]. Available: https://github.com/noxrepo/pox

[6] "Opendaylight sdn controller," 2014. [Online]. Available: http://www.opendaylight.org/