Multi-domain service orchestration over networks and clouds: a unified approach

Balázs Sonkoly[1], János Czentye[1], Robert Szabo[2], Dávid Jocha[2], János Elek[2], Sahel Sahhaf[3], Wouter Tavernier[3], Fulvio Risso[4]

[1]*Budapest Univ. of Technology and Economics;* [2]*Ericsson Research;* [3]*Ghent University – iMinds;* [4]*Politecnico di Torino*

ABSTRACT

End-to-end service delivery often includes transparently inserted Network Functions (NFs) in the path. Flexible service chaining will require dynamic instantiation of both NFs and traffic forwarding overlays. Virtualization techniques in compute and networking, like cloud and Software Defined Networking (SDN), promise such flexibility for service providers. However, patching together existing cloud and network control mechanisms necessarily puts one over the above, e.g., OpenDaylight under an OpenStack controller. We designed and implemented a joint cloud and network resource virtualization and programming API. We demonstrate that our abstraction is capable for flexible service chaining control over any technology domains.

Keywords

NFV, SDN, multi-domain orchestration

1. INTRODUCTION

On top of cloud platforms, services can be created, managed and scaled on-demand. Efficient virtualization techniques and novel orchestration algorithms enable flexible operation and optimal usage of underlying resources. Besides virtual compute and storage resources, basic networking is also provided in order to connect the virtual machines. In contrast, traditional telecommunication services and carrier networks have a lot of limitations concerning service creation, service deployment or service provisioning due to built-in mechanisms strongly coupled to physical topologies and special purpose hardware appliances. Network Function Virtualization (NFV) opens the door between cloud technologies and carrier networks by providing software-based telecommunication services, which can run in virtualized environments over a wide range of general purpose servers. By these means, recent achievements from cloud researches can be leveraged and adopted in carrier environments.

Flexible service definition and creation may start by abstracting and formalizing the service into the concept of service chain or service graph. It is a generic way to describe high level services and to assemble processing flows for given traffic. Today, remarkable attention is focused on the concept of Service Function Chaining (SFC) and NFV. IETF, ETSI, MEF, Linux Foundation and several research projects address different aspects of SFC. For example, IETF is working on an SFC architecture and data plane elements; ETSI has proposed a management and orchestration framework for NFV; MEF has envisioned the concept of Lifecycle Service Orchestration; Linux Foundation launched the Open Platform for NFV Project (OPNFV), a carrier-grade open source reference platform.

UNIFY[1] is an EU-funded FP7 project, which aims at unifying cloud and carrier network resources in a common orchestration framework. We designed a novel SFC control plane capable of integrating *any* infrastructure domains including different NF execution environments, SDN networks or legacy data centers. Our architecture supports automated, dynamic service creation and multi-domain NFV orchestration. Central to our design is the definition of a joint virtualization and resource programming interface unifying cloud and network resources. In this demonstration, we highlight the power of joint virtualization and resource programming to control SFC deployments over multi-technology domains.

2. PROOF OF CONCEPT

In UNIFY, we have proposed a *joint, narrow waist SFC control plane* over compute (with storage) and network resources for flexible service creation. Our control plane design enables *recursive orchestration* of several types of resources and supports different even legacy technologies and migration between them. Furthermore, we have introduced a novel infrastructure element called *Universal Node* (UN) which is a COTS hardware based packet processor node with the capability of *i*) high performance forwarding and *ii*) running high complexity NFs in its virtualized environment. Our proof of concept prototype realizing the envisioned architecture is shown in Fig. 1.

Our virtualization is based on the concept of Big Switch with Big Software (BiS-BiS) elements, which is a forwarding element combined with software (compute and storage) resources capable of *i*) running NFs and *ii*) steering traffic transparently among infrastructure and NF ports. The key concept is that BiS-BiS combines cloud resources with forwarding behavior control. The virtualization view may contain arbitrary interconnection of BiS-BiS nodes. In turn, the control plane design does not enforce preemption of cloud or network resource control, like NF deployment *and* interconnection or interconnection *and* NF deployment. Finally, SFC programming is realized by *i*) assigning NFs to BiS-BiS nodes and by *ii*) editing the flowrules within BiS-BiS nodes. The task of the resource orchestrator is to map the configurations of different client virtualizations to a configuration at the underlying domain virtualizer. Within the mapping

[1]http://www.fp7-unify.eu

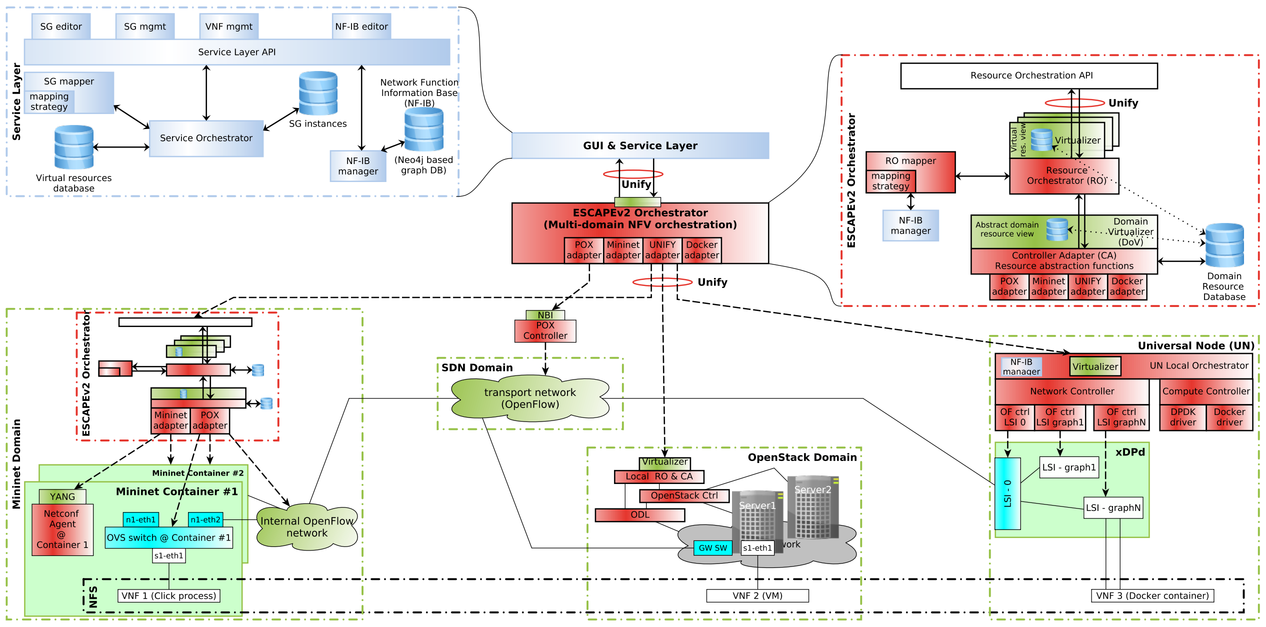

Figure 1: Joint SFC control plane architecture on top of different technological domains.

process, we introduced the concept of NF decomposition, i.e., an NF mapped to a BiS-BiS in the client virtualization can be replaced with an interconnection of NFs (components) during the mapping process [2]. The data model of the virtualizer is defined in Yang. In Fig. 1, virtualizers are denoted by green boxes and they provide virtual views (interconnected BiS-BiS nodes) for their managers (clients). Managers are resource orchestrators (shown by red boxes) configuring (controlling) the given view. The interface at the north of the virtualizer appears at multiple levels in Fig. 1. Indeed, the manager - virtualizer relationship is recursive; Unify domains can be stacked into a multi-level control hierarchy similar to the Open Networking Forum's SDN architecture. The recursive interface is the Unify interface.

The initial version of our open-source NFV orchestration framework was presented in [1]. It has been significantly redesigned in order to meet the design goals and requirements of the joint SFC control plane addressed by UNIFY. Now, ESCAPEv2 framework *i*) enables recursive orchestration by implementing Unify interfaces at north and south; *ii*) operates on our novel joint resource abstraction models; *iii*) supports UNIFY domains directly and several technological domains via adapters; *iv*) can be extended easily with additional plug and play components/algorithms, like NF implementations, network embedding algorithms, NF decomposition models.

On top of ESCAPEv2, we have implemented a simple service layer with GUI (see the upper left part of Fig. 1) where users can define service requests with their requirements, e.g., bandwidth or delay constraints between arbitrary elements in the service graph. The service layer contains a service orchestrator which is responsible for mapping the service request to the virtualizer. If a service orchestrator sees only *a single* BiS-BiS node then its orchestration task is trivial. Such setup allows delegation of all resource management to the lower layer Unify domain which is realized by the resource orchestrator component (see upper right part of Fig. 1). Here, the domain resource view is constructed by our domain virtualizer and controller adapter modules. At the infrastructure level, different technologies are supported and integrated with the framework. We kept our Mininet based domain orchestrated by a dedicated ESCAPEv2 entity via NETCONF and OpenFlow control channels. Here, the NFs are run as isolated Click processes. As a legacy data center solution, we support clouds managed by OpenStack and OpenDaylight. This requires a UNIFY conform local orchestrator to be implemented on top of an OpenStack domain. The control of legacy OpenFlow networks is realized by a POX controller and a corresponding adapter module. And finally, a proof of concept implementation of our Universal Node concept is also provided. UN local orchestrator is responsible for controlling logical switch instances (of e.g., xDPd) and for managing NFs run as Docker containers. The high performance is achieved by Intel's DPDK based datapath acceleration.

During the demo, we showcase *i*) how to create joint domain abstraction for networks and clouds; *ii*) how to orchestrate and optimize resource allocation over these unified resources; *iii*) how we can take advantage of recursive orchestration and NF decomposition; *iv*) how to deploy given service chains at different infrastructure components[2].

3. 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] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet. Network service chaining with efficient network function mapping based on service decompositions. In *1st IEEE Conference on Network Softwarization, NetSoft 2015*, 2015.

[2]Further information including video clips are available at: http://sb.tmit.bme.hu/mediawiki/index.php/UNIFY15

Demo Requirements

Equipment to be used for the demo

The following equipment will be brought by the authors.

- notebook #1: running the multi-domain orchestrator, service layer and GUI, POX controller
- notebook #2: running the Mininet domain with hosts (service access points)
- notebook #3: running the OpenStack domain
- notebook #4: running a Universal Node
- two small OpenFlow capable switches: realizing a simple OpenFlow transport network
- one switch: for control and management network
- external Ethernet adapters connected to the notebooks

Additionally we will need the following items.

- two large screens

Space needed

A large table with enough space for 4 notebooks, 3 small switches and 2 screens.

Setup time required

We will need approximately 30 minutes to set up the testbed.

Additional facilities needed

We will need power supply and extension cords.