Transport PCE
Network Function Virtualization

R. Vilalta[(1)], R. Muñoz[(1)], R. Casellas[(1)], R. Martínez[(1)], V. López[(2)], D. López[(2)]

[(1)] Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), ricard.vilalta@cttc.es

[(2)] Telefónica I+D

Abstract *We propose a Transport PCE architecture to deploy a Transport PCE by means of Network Function Virtualization. Virtual PCEs are deployed on demand, but they are perceived as a single Virtualized Network Function. We present the benefits by experimental validation.*

Introduction

Network Functions Virtualization (NFV) aims at using IT virtualization techniques to virtualize entire classes of network node functions. A Virtualized Network Function (VNF) consists of a network function running as software on several virtual machines instead of having custom hardware appliances for the proposed network function[1].

A transport Path Computation Element (PCE) is a transport network function, which is able to perform constrained path computation on a graph representing a network (Traffic Engineering Database - TED)[2]. The PCE global architecture and communication protocol (PCEP) have been standardized by IETF. The PCE can be run as an application on top of Commercial Off-The-Shelf (COTS) equipment[3]. The initial driver for the deployment of PCEs was the increasing complexity of path computation. The PCE architecture has been extended to support scalability constraints with the introduction of the hierarchical PCE. Using hierarchical PCE architecture, each PCE is considered as a single network function.

In this paper, we propose the adoption of the NFV architecture to deploy a PCE dedicated to path computation of a transport network as a VNF. Although the NFV architecture has successfully been demonstrated for mobile networks, there have been only few attempts to introduce this architecture to core networks. A PCE NFV orchestrator is introduced, so that the proposed transport PCE NFV is be able to handle intense peak loads of path computation requests. The NFV orchestrator dynamically deploys virtual PCEs (vPCEs) on demand to keep the quality of the VNF (e.g., in terms of latency, request processing time, dedicated algorithms, etc.). A vPCE is a PCE instance, which is run as a software application on a cloud computing environment (e.g., a virtual machine).

We also introduce a PCE DNS[4] in order to offer the deployed vPCEs as a single VNF perceived by the different Path Computation Clients (PCC).

Finally, a PCE front-end/back-end architecture is proposed to bypass the limitations of the presented approach.

Proposed Architecture

In this section, the proposed transport PCE NFV architecture is described (Fig. 1.a). A PCE NFV Orchestrator is the entity responsible for the deployment of the PCE as a VNF. The PCE NFV Orchestrator consists of three separated modules: PCE VNF provider, Virtual IT resources and PCE computation load monitoring.

The PCE VNF provider implements the necessary logic for deploying the necessary vPCE in order to guarantee the quality of the VNF. In order to guarantee the quality, the PCE VNF provider interacts with the PCE computation load monitoring module in order to obtain the necessary data to decide to deploy a new instance of a vPCE or to delete one, via the virtual IT resources module. Thus, the PCE VNF is the responsible for deploying the logic of the orchestrator.

The Virtual IT resources module is responsible for managing a cloud infrastructure (e.g., OpenStack). The cloud infrastructure shall allow the dynamic deployment and release of virtual machines with custom images running vPCE as an application. The cloud infrastructure must assign to the vPCE a new IP address from a set of available ones. This IP address is parsed and the PCE DNS is notified with the new IP address for a new available vPCE.

Finally, the PCE computation load monitoring module is the responsible for monitoring the quality of the VNF. The monitored parameters are a set of the PCE monitoring parameters defined in [5], which are exposed by the vPCEs, by means of an HTTP server. One of these parameters is the mean path processing time. If the mean path processing time exceeds a certain threshold, the

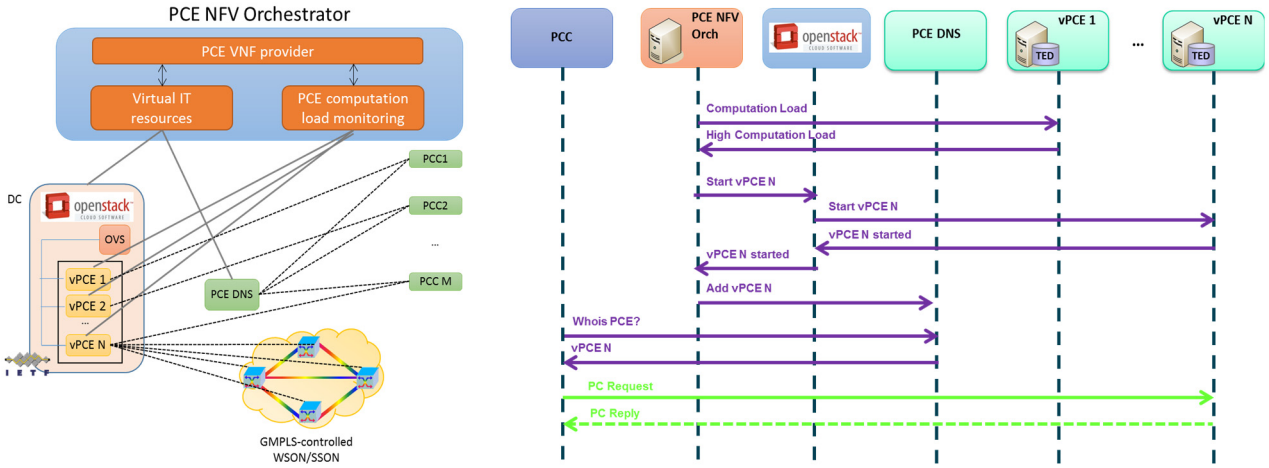

Fig. 1: (left) Proposed Transport PCE NFV architecture, (right) PCE NFV Orchestrator message exchange

PCE VNF could deploy a new vPCE to reduce the peak request load in the PCE VNF.

As a PCE discovery mechanism, a PCE DNS is proposed. DNS is a query-response based mechanism. A Path Computation Client (a PCC) can use DNS to discover a PCE only when it needs to compute a path and does not require any other node in the network to be involved. In case of an intermittent PCEP session, which are systematically opened and closed for each PCEP request, a DNS-based query-response mechanism is suitable. Moreover, DNS supports load balancing where multiple vPCEs (with different IP addresses) are known in the DNS for a single PCE server name and are seen for the PCC as a single resource. Requests are load-balanced among vPCEs without any complexity at the PCC.

The messages exchanged between the different elements of the proposed architecture are displayed in Fig. 1.b. It can be observed, that the PCE NFV Orchestrator is the responsible for checking the different quality parameters to the deployed vPCEs. Once these quality parameters are received, the PCE VNF provider module within the PCE NFV Orchestrator is the responsible to determine whether a new vPCE is required.

If a vPCE is selected to be deployed, the Virtual IT resources module will deploy a new virtual machine with the vPCE image, will assign a new IP address to the vPCE and once the vPCE is started, the Virtual IT resources module will notify the new vPCE IP address to the PCE DNS.

Once a PCC requires a new path computation, first will issue a DNS query to the PCE DNS. The PCE DNS is responsible to load balance the different vPCEs, so returns a single IP address corresponding to one of the vPCEs. Finally, the PCC establishes a path computation session with the corresponding vPCE.

Experimental Performance

The experimental performance of the proposed Transport PCE NFV architecture has been evaluated in the Cloud Computing Platform of the ADRENALINE Testbed at CTTC. An OpenStack[6] cloud has been deployed on top of a Custom Server using an Asus Z9NA-D6 board with 2 Intel Xeon E5-2410 processors and 32 Gb RAM.

The proposed NFV orchestrator has been developed in Python, and the PCE has been described in [2]. The PCE DNS server has been setup using bind9, which is the standard linux DNS server. All the deployed vPCE where sharing a static network view of a typical Spanish 14-node 44-link Flexi-grid DWDM network. In the future, BGP-LS could be used in order to dynamically synchronize the TED of the different vPCEs.

The deployed vPCEs allows the measurement of the rolling mean (we use a 10 request window) processing time of a request (time between a request is received and responded) via HTTP through an XML response.

Every new instance of vPCE is deployed by means of the OpenStack nova API, which is responsible for virtual machines management on an OpenStack cloud. All deployed virtual machines share a common file repository for ease of synchronization. We have prepared a vPCE snapshot, which is able to easily run a vPCE.

The OpenStack neutron API, responsible for

a)

Source	Destination	Protocol	Info
10.1.7.130	10.1.7.129	DNS	Standard query A pce.lab.cttc.es
10.1.7.129	10.1.7.130	DNS	Standard query response A 10.1.7.131 A 10.1.7.132
10.1.7.130	10.1.7.131	PCEP	OPEN MESSAGE
10.1.7.131	10.1.7.130	PCEP	OPEN MESSAGE
10.1.7.131	10.1.7.130	PCEP	KEEPALIVE MESSAGE
10.1.7.130	10.1.7.131	PCEP	KEEPALIVE MESSAGE
10.1.7.130	10.1.7.131	PCEP	PATH COMPUTATION REQUEST MESSAGE
10.1.7.131	10.1.7.130	PCEP	PATH COMPUTATION REPLY MESSAGE
10.1.7.130	10.1.7.131	PCEP	CLOSE MESSAGE
10.1.7.130	10.1.7.129	DNS	Standard query A pce.lab.cttc.es
10.1.7.129	10.1.7.130	DNS	Standard query response A 10.1.7.132 A 10.1.7.131
10.1.7.130	10.1.7.132	PCEP	OPEN MESSAGE
10.1.7.132	10.1.7.130	PCEP	OPEN MESSAGE
10.1.7.132	10.1.7.130	PCEP	KEEPALIVE MESSAGE
10.1.7.130	10.1.7.132	PCEP	KEEPALIVE MESSAGE
10.1.7.130	10.1.7.132	PCEP	PATH COMPUTATION REQUEST MESSAGE
10.1.7.132	10.1.7.130	PCEP	PATH COMPUTATION REPLY MESSAGE
10.1.7.130	10.1.7.132	PCEP	CLOSE MESSAGE

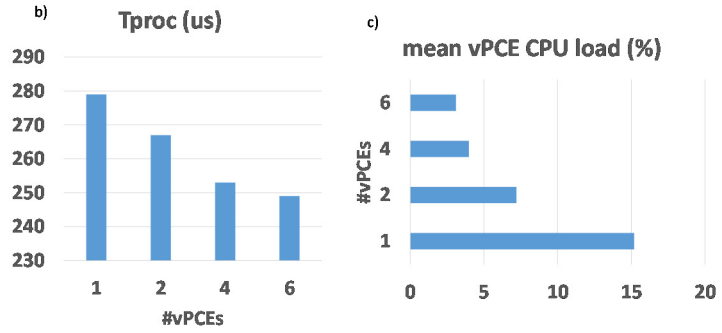

Fig. 2: a) Wireshark of different PCEP sessions established to different vPCEs. b) Mean request processing time (microseconds). c) Mean vPCE CPU load (%)

network configuration, assigns to the vPCE an IP address, which is later added as a possible resolution for pce.lab.cttc.es to the PCE DNS.

The PCC is responsible for issuing a DNS query, when a new path computation request is issued. When the PCE DNS receives a DNS query, it applies a simple load balancing algorithm by returning a different vPCE IP address for each query. Finally, the PCC establishes a PCEP session to the assigned vPCE. Fig. 2.a. shows the standard PCEP session including OPEN, KEEPALIVE, PCRequest, PCReply and CLOSE messages.

In order to stress the proposed architecture, a PCC requests 500 requests per second. Each path computation request is randomly selected between two endpoints of the described flexi-grid network. The mean request processing time (Tproc) is measured as a mean of the previously defined request processing time of the current vPCEs.

We have requested 10000 Path Computation Requests for each measurement. When a single vPCE (acting as a PCE) was deployed the Tproc was 279 microseconds. It can be observed that when more vPCEs have been deployed the measured Tproc is reduced. For example, for 6 vPCEs deployed, Tproc is 248 us (Fig. 2.b).

Fig. 2.c shows the mean measured CPU load at a single vPCE, when different vPCE have been deployed. The measured CPU load tends to be balanced by the different vPCEs, when 2 vPCE are deployed the mean CPU load is of 7.2 %. If there are 6 vPCEs deployed the CPU load is of 3.1%. It can be observed, that if more vPCEs are deployed, the computational load is balanced between them, allowing a faster mean request processing time.

Conclusions

We have presented a transport PCE NFV architecture, which is able to guarantee a mean request processing time within a detected peak of path computation requests. The proposed architecture exploits the benefits of NFV. We have experimentally evaluated the mean request processing time, demonstrating the benefits of the presented approach.

Further research shall be done on request peak detection, the usage of pre-deployed vPCEs and finally the introduction of the front-end/back-end PCE architecture to bypass the need for a PCE DNS, so that dedicated vPCEs and more complex request allocation algorithms can be exploited.

Acknowledgements

EU FP7 project STRAUSS (FP7-ICT-2013-EU-Japan 608528) and Spanish MINECO project FARO (TEC2012-38119).

References

[1] NFV whitepaper, ETSI, 2013.

[2] R. Casellas et al., “Applications and Status of PCE”, JOCN, vol. 5 n.10, 2013.

[3] Y. Yoshida et al., “First international SDN-based Network Orchestration of Variable capacity OPS over Programmable Flexi-grid EON”, ThA.2 OFC 2014.

[4] Q. Wu et al., “PCE discovery using DNS”, draft-wu-pce-dns-pce-discovery-05, IETF.

[5] JP. Vasseur et al., “A Set of Monitoring Tools for PCE-Based Architecture”, RFC 5886

[6] X. Wen et al., “Comparison of open-source cloud management platforms”, FSKD 2012.