Deploying elastic routing capability in an SDN/NFV-enabled environment

Steven Van Rossem*, Wouter Tavernier*, Balazs Sonkoly †, Didier Colle*,
Janos Czentye †, Mario Pickavet* and Piet Demeester*
*Ghent University, IBCN, INTEC, Gaston Crommenlaan 8 bus 201, B-9050 Gent, Belgium
Email: {steven.vanrossem, wouter.tavernier, didier.colle, mario.pickavet, piet.demeester} @intec.ugent.be
† Budapest Univ. of Technology and Economics
Email: {sonkoly, czentye}@tmit.bme.hu

***Abstract*—SDN and NFV are two paradigms that introduce unseen flexibility in telecom networks. Where previously telecom services were provided by dedicated hardware and associated (vendor-specific) protocols, SDN enables to control telecom networks through specialized software running on controllers. NFV enables highly optimized packet-processing network functions to run on generic/multi-purpose hardware such as x86 servers. Although the possibilities of SDN and NFV are well-known, concrete control and orchestration architectures are still under design and few prototype validations are available. In this demo we demonstrate the dynamic up- and downscaling of an elastic router supporting NFV-based network management, for example needed in a VPN service. The framework which enables this elasticity is the UNIFY ESCAPE environment, which is a PoC following an ETSI NFV MANO-conform architecture. This demo is one of the first to demonstrate a fully closed control loop for scaling NFs in an SDN/NFV control and orchestration architecture.**

***Index Terms*—Elastic Router, NFV, SDN, UNIFY**

I. Introduction

Traditional telecom services are highly dependant on physical topologies and vendor-specific hardware. To overcome this, SDN (Software Defined Networking) and NFV (Network Function Virtualization) principles are getting commonly adopted to design new platforms for creating, managing and scaling network services on-demand, in a faster and more resource-optimized way. In this context, *elasticity* denotes the ability of assigning hardware resources dynamically, as response to fluctuations in eg. service demand or resource availability. By monitoring specific KPI's (Key Performance Indicators), the service can deploy more or less NF's (Network Functions), only consuming the resources needed for its real-time average performance. In worst-case conditions, or as the service configuration changes, the number/type of resources can be altered.

Using new possibilities enabled by SDN/NFV, a service creation platform is being worked out in the UNIFY project[1]. It aims at unifying cloud and carrier network resources in a common orchestration framework. In this paper, we demonstrate the basic mechanisms which enable an elastic service management, by using an elastic router as illustrating example.

II. Architecture

A router can for example be offered as a service itself, Routing-as-a-Service (RaaS), or be deployed as a functional part of a more advanced service. Let's envision a user-requested VPN service, where a business is connecting its offices over the public internet. It would need a L3VPN connecting N private LAN networks over the Internet, each requiring a full-duplex throughput R. In this context, the provider would need an elastic routing function in his offered VPN service such that:

(i) The number of ports N scales to the number of requested VPN islands by the enterprise user.
(ii) It can dynamically adapt its traffic processing capacity to the required service needs (throughput R).
(iii) Hardware resources are assigned dynamically, optimized in function of the current service requirements (N, R).

A. Elastic Router

The elastic router consists of one or more *SDN-enabled (virtual) switches* in the forwarding plane. On the control plane, an *OpenFlow Controller (OF Ctrl)* is configuring the SDN switches and gathers flow statistics. An extra *Ctrl App* on top of this controller monitors the service's KPI's and can trigger an elastic topology change, deploying more or less SDN switches. A configuration interface via the *Ctrl App* enables the service user to configure parameters in eg. the IP routing table. This is illustrated in figure 1.

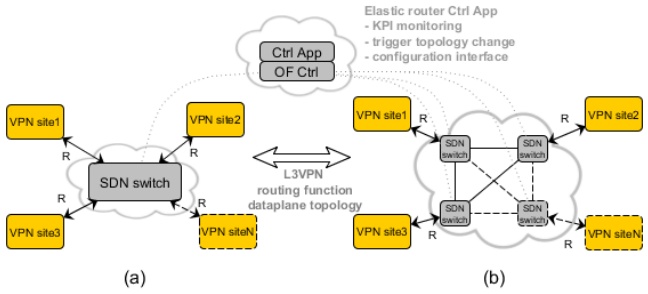

Fig. 1. VPN service with an elastic router providing N ports. The dynamic creation of extra ports is illustrated by the dashed links/blocks. The *Ctrl App* triggers the change between (a) single switch instance and (b) a new switch topology (VLB) with distributed processing.

978-1-4673-6884-1/15/$31.00 ©2015 IEEE

NFV enables the deployment of these (virtual) NF's *(SDN switches, OF Ctrl, Ctrl App)* on servers in a datacenter. The required processing rate of the elastic router determines the needed resources for these VNF's (Virtual Network Functions), especially for the forwarding elements.

In case of a single switch instance, where N ports require receiving and sending at rate R, the processing rate of the switch must be NR. When N, R increase, we can imagine that the resources of a single server will fall short. To still fulfill the SLA (Service Level Agreement) with these servers, we need to alleviate this NR requirement. This can be done by load-balancing traffic among multiple switch instances on different servers, which increases the throughput the elastic router can process. Implementing logical routers using multiple router/switch instances is a known research area [2]. Many examples are also found in datacenter network topologies (eg. Clos/Fat-Tree, Jellyfish). In this paper we focus on direct VLB (Valliant Load Balancing). This is the topology shown in figure 1b.

We decompose the single switch instance to a VLB topology, where we shift from a single switch with N ports, to a full mesh of N switches. In its simplest case, it can be shown that if all flows originating from a node have rates no bigger than $\frac{R}{(N-1)}$, this node can send all the flows along direct paths, directly to its destination node, effectively lowering the switch required processing rate to $2R$ [3].

B. Network Function Profiling

If we assume that the maximum processing throughput R is specified in the SLA, then it can be noted that by rate-limiting the links to the elastic router, the worst-case processing rate is known. Therefore also the needed hardware resources and topology can be predicted at service deployment. However, when average traffic rates vary below the worst-case limit, the elastic router can dynamically optimize its resource use by changing its dataplane topology during service run-time.

This requires a mapping function which translates any high-level performance parameters (SLA specifications including processing rate or latency) to low-level infrastructure resource requirements (vCPU, memory, bandwidth, delay). We call this mapping a VNF profile. The VNF profile can be used to inform the Service Provider which resources are needed for the required performance when deploying the service. The profile can be provided by the VNF developer or it could be benchmarked by another party, VBaaS, VNF Benchmarking as a Service, as described in [4].

In the elastic router, a VNF profile would be useful for the *SDN switch* VNF's. That way, the maximum packet processing rate of the deployed switch instance is known and can be anticipated upon. It is used in the *Ctrl App* to set its threshold for changing the switch topology.

C. Network Function State Migration

In our elastic router service, the flow tables of every dataplane switch should be migrated or merged upon a topology change. Also flow statistics should not be reset, as they might be useful for the Service Provider to measure eg. datavolume. The *Ctrl App* initiates the topology change and has a direct interface to the *SDN switches* via the *OF Ctrl*. It is therefore best positioned to set flow rules in each new SDN switch instance, including any user configuration settings. The elastic router service could be further enhanced with specialized API's for reliable state migration, as proposed in the openNF framework [5]. This however, would require additional implementation modifications in all the NF's.

D. Automatic elasticity provisioning using NFV/SDN

The emulation environment used to deploy the elastic router service is developed in the aforementioned UNIFY project[1]. A high-level breakdown of this architecture is shown in figure 2. We distinguish 3 main layers in this model:

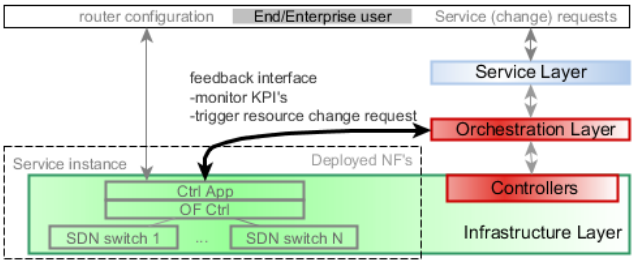

Fig. 2. Elastic router deployed in the Unify framework. The *feedback interface* enables the running service to dynamically control its resources.

- The user requests the VPN service via the *Service Layer*. Here, the high-level requirements (number of ports N, throughput R) are translated to the required NF's (*SDN switches*, *OF Ctrl*, *Ctrl App*) and their interconnections.
- The *Orchestration Layer* finds an optimal way to map the required NF's to the available infrastructure (the available servers to deploy the NF's).
- The *Infrastructure layer* comprehends all available hardware resources (compute, storage and network nodes). Different *Controllers* are each handling the resources in their domain (eg. the compute controller finds available servers to deploy the NF's, the network controller steers the traffic between the deployed NF's).

The *feedback interface* from the Ctrl App to the Orchestration Layer further enhances the elastic service. It creates a fully closed control loop by enabling the running service to monitor and control its own needed resources (like deploying more *SDN switch* instances), without any intervention via the service layer or the user. It allows a quick reaction to changing conditions as resource control requests are handled locally, unlike with a higher-layered management entity. It can be noted that the *feedback interface* is in fact similar to the interface between Service and Orchestration Layer. They both provide a way to monitor and configure running/new services (by changing/adding interconnected NF's on the Infrastructure Layer).

ESCAPE is a PoC that implements the above architecture [6]. It virtualizes available network resources for the deployment of NF's in the context of a network service (deployable in Mininet, via OpenStack or as Docker containers). An SDN network domain with an OpenFlow controller (POX) is steering the network traffic between the NF's.

III. Demonstration

The *Infrastructure layer* in the demo is situated in 2 domains: Mininet and Openstack, demonstrating the multi-domain orchestration capabilities of the ESCAPE framework. The SDN-enabled switches are deployed as open vSwitch instances in a Mininet environment with a custom API acting as infrastructure *Controller* allowing to add/remove switches. Another custom *Controller* is able to deploy virtual machines in an Openstack environment. The *Ctrl App* and *OF Ctrl* are implemented by an OpenFlow controller instance (Ryu). This process runs in a virtual machine, started in the Openstack domain. On each of the N ports we generate $N-1$ ingress traffic flows of rate r (equally distributed to the $N-1$ other ports). As shown in figure 3, the *Ctrl App* monitors the switches' processing rate and triggers, through the *feedback interface*, a topology change.

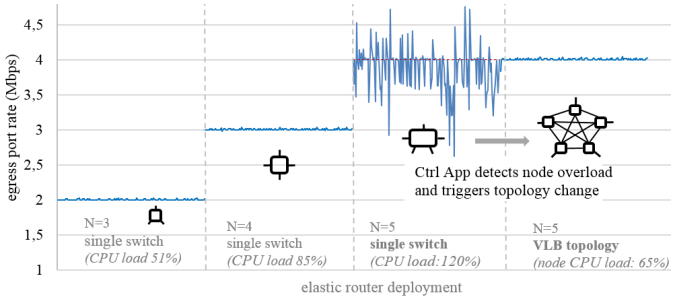

Fig. 3. Elastic router demo (r=1Mbps) showing the ability to set the number of ports N via the *service layer* and scale resources adaptively to R via the *feedback interface* (The used topology and CPU load is displayed as info).

The threshold for the topology change is depending on the packet processing capacity of the SDN switch. For the demo, we assume the VNF profile of the *SDN switches* to be hard coded in the *Ctrl App* by the VNF developer. The main KPI for the *Ctrl App* to monitor, is the ingress packet rate of a switch. When this exceeds the threshold, the switch instance can no longer meet the service performance request and the packet processing is offloaded to multiple switches in a VLB topology.

The demo will deploy the elastic router service using the ESCAPE framework and 2 laptops, each with its own infrastructure domain (Mininet and Openstack). A GUI shows how the router topology is changing in function of the traffic load as shown in figure 4.

IV. Conclusion

The elastic router use case is a simple an clear example that helps to identify following aspects of an elastic service deployment:

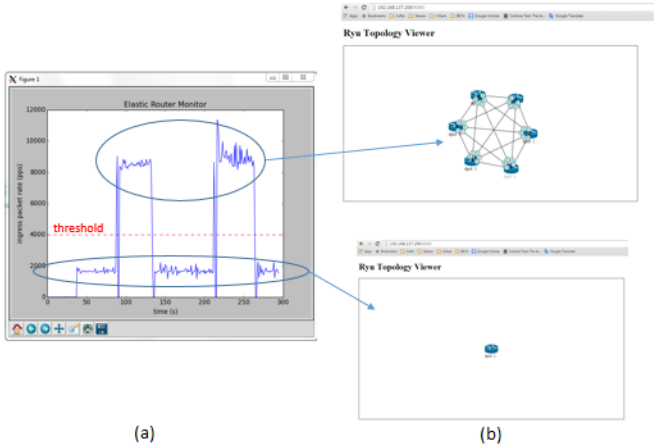

Fig. 4. Info provided by the *Ctrl App*: (a) plot showing the ingress packet rate and the threshold (b) visual representation of the switch topology currently deployed.

(i) Mapping of high-level SLA's to hardware resources via a VNF profile.
(ii) A Ctrl App function that monitors KPI's and optimizes resource use in function of real-time processing needs.
(iii) A feedback interface that allows quick changes in the running service.
(iv) Reliable state migration during changes in a running service.

An architecture is proposed and used to deploy an elastic router. The generic design of the deployment platform, makes it easily extendible to manage all kinds of network services. It can be noted that, using NFV, the functionality of the forwarding nodes in this topology can also be augmented from simple packet switching to more advanced packet processing (like firewalling), which will further increase the load on the generic/multi-purpose servers in the datacenters. This further justifies the relevance of elasticity in innovative service deployment platforms.

Acknowledgment

This work was conducted within the framework of the FP7 UNIFY project, which is partially funded by the Commission of the European Union.

References

[1] A. Császár, W. John, M. Kind, C. Meirosu, G. Pongrácz, D. Staessens, A. Takacs, and F.-J. Westphal, "Unifying cloud and carrier network: Eu fp7 project unify," in *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*. IEEE, 2013, pp. 452–457, http://www.fp7-unify.eu/.

[2] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "Routebricks: exploiting parallelism to scale software routers," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009, p. 15.

[3] H. Liu and R. Zhang-Shen, "On direct routing in the valiant load-balancing architecture," in *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, vol. 2. IEEE, 2005, pp. 6–pp.

[4] R. S. Raphael Vicente Rosa, Christian Esteve Rothenberg, "VBaaS: VNF benchmark-as-a-service," in *Proceedings of the 4th European Workshop Software Defined Networks (EWSDN)*. EWSDN, 2015.

[5] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "Opennf: Enabling innovation in network function control," in *Proceedings of the 2014 ACM Conference on SIGCOMM*. ACM, 2014, pp. 163–174.

[6] B. Sonkoly, J. Czentye, R. Szabo, D. Jocha, J. Elek, S. Sahhaf, W. Tavernier, and F. Risso, "Multi-domain service orchestration over networks and clouds: a unified approach," in *Proceedings of the 2015 ACM conference on SIGCOMM*. ACM, 2015.