Coupling VNF Orchestration and SDN Virtual Network Reconfiguration

Nemanja Đerić*, Amir Varasteh*, Arsany Basta*, Andreas Blenk*,
Rastin Pries†, Michael Jarschel†, and Wolfgang Kellerer*
*Chair of Communication Networks, Department of Electrical and Computer Engineering,
Technical University of Munich, Germany
Email: {nemanja.deric, amir.varasteh, arsany.basta, andreas.blenk, wolfgang.kellerer}@tum.de
†Nokia Bell Labs, Munich, Germany
Email: {rastin.pries, michael.jarschel}@nokia-bell-labs.com

***Abstract*—Network Function Virtualization (NFV) promises an efficient way of managing and orchestrating Virtual Network Functions (VNFs), where VNFs can be dynamically instantiated or migrated based on current service requirements. For instance, in order to improve the performance of mission critical services, VNFs can be instantiated closer to users or even be migrated over time to follow mobile users. However, this orchestration leads to changes, e.g, of the traffic requirements of the data and control plane of virtual network interconnecting VNFs. Particularly in virtualized Software-Defined Networking (vSDN) environments, these traffic changes require fast adaptations of the data and control plane isolation and abstraction policies, which guarantee predictable network operation and control. More in detail, the entity managing the virtualization (i.e., the SDN hypervisor) has to quickly reconfigure the policies of affected Virtual Networks (VNs) interconnecting VNFs. In this demo, we present *i)* the benefits of migrating a firewall VNF to a server, which is closer to its user, at runtime and show *ii)* how the migration is supported in a virtualized SDN environment.**

***Index Terms*—Virtual Network Function Orchestration, Network Virtualization, Virtual Network Reconfigurations, SDN Network Hypervisors**

I. Introduction

Network Function Virtualization (NFV) proposes a new way of managing and deploying network functions [1]. The network functions (e.g. firewall, NAT, load balancer) are decoupled from the dedicated hardware and realized as software instances, i.e., Virtual Network Functions (VNFs) running on commodity servers. The benefits of NFV are manifold [2]: *i) the consolidation of services* by using Commercial Off-The-Shelf (COTS) hardware instead of special purpose equipment, which can lead to reducing the network overprovisioning; *ii) the dynamic orchestration of VNFs* due to automation of VNF instantiation and migration based on the current service requirements, which becomes possible through cloud systems (another potential for improving cost factors); *iii) the dynamic update of virtual networks* based on changing requirements and workloads, i.e., the reconfiguration of virtual networks [3], which can help to increase the resource efficiency of the infrastructure [4].

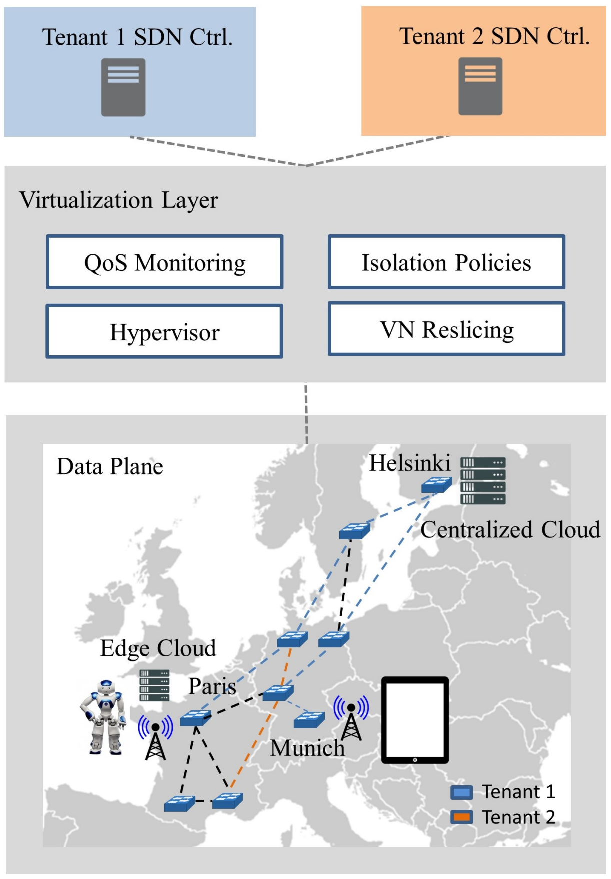

Fig. 1. The demonstration architecture and scenario show *i)* a European-based data plane topology with a centralized server located in the proximity of Helsinki, Finland, while the edge servers are positioned in the proximity of big cities and *ii)* the virtualization provided by the virtualization layer, i.e., HyperFLEX, and *iii)* two tenants that share the physical architecture, highlighted in blue and orange colours.

Unfortunately, the benefits can come with a prize: the observed end-to-end delay might *increase* due to the function softwarization (because of additional server virtualization layers) or due to placing functions at server locations that promise low resource costs but are far away from users [5]. Besides, updating virtual networks at runtime can lead to service interruptions or unacceptable control plane latencies [6]. This drawback, however, is not acceptable for applications with strict delay requirements, such as remote control of surgery robots or safety applications. Thus, an intelligent and service-aware VNF management and orchestration is needed.

VNF Orchestration within Virtual Software-Defined Networking: Software-Defined Networking (SDN) provides remote control of networking devices via control plane protocols (e.g., OpenFlow [7]). When realizing full Network Virtualization (NV) in SDN, a middle layer, also called SDN hypervisor, is logically placed between the SDN controllers of the tenants and the data plane forwarding devices [8]. With this approach, network tenants are able to use their own SDN controllers without any modifications. In particular, the virtualization layer takes care of tasks that are needed for predictable and seamless network operation, such as establishing the control and data plane isolation and resource abstraction policies [9].

Supporting Dynamic VNF Orchestration: Migrating VNFs can produce changes in the control and data plane traffic requirements of the Virtual Networks (VNs). Thus, SDN hypervisors have to provide the dynamic reconfiguration of the VNs to meet the changing demands. For instance, if a VNF is instantiated or migrated to a remote server that is not part of the initial VN, an SDN hypervisor has to reallocate network resources, i.e., establish additional paths to the new remote server and establish new isolation and abstraction policies. Besides, even if the VNF is migrated within the slice, the migration still changes the traffic patterns within the VN, which requires again updates of the virtualization policies.

Demonstration: In this work, we demonstrate the performance gain of migrating a firewall VNF instance from a consolidated general purpose server to a server closer to a firewall user. In our approach, we use a custom VNF orchestrator that considers end-to-end Quality of Service (QoS) requirements of the application running inside a virtual network. The virtualization layer, i.e., the SDN network hypervisor, is managed and orchestrated by utilizing HyperFLEX [10]. HyperFLEX provides control and data plane isolation with automated VN management. In this demo, it is extended to support dynamic VN reconfiguration as a result of VNF migrations.

II. Demo Setup and Scenario

The architecture of the demo is presented in Figure 1, where the control and data plane separation is illustrated. Starting from top to bottom, two virtually isolated software instances of the Ryu SDN controller software [11] are used to emulate SDN controllers belonging to two tenants.

Virtualization and VNF Orchestration: The HyperFLEX management layer enables NV for SDN in our demonstration. HyperFLEX provides control and data plane isolation, QoS monitoring, and automated management of VNs. In this demo, we extend it to support reconfigurations of VNs. That is, if a VNF is migrated in order to improve application QoS metric (e.g., delay), HyperFLEX firstly checks the available link (e.g., bandwidth) and node resources (e.g., CPU) towards the new server and establishes new virtualization policies accordingly. If there are no available paths that satisfy the required QoS demands, HyperFLEX checks whether reconfigurations of other VNs could free the demanded resources. Furthermore, isolation and abstraction policies are updated in order to meet the new VN requirements and in order to prevent performance interference among tenants. The VN reconfiguration process is completely abstracted (i.e., hidden) from the tenants. For this demonstration, an internally developed virtual firewall is used as a VNF in order to prevent malicious traffic entering the network.

Data Plane Topology: As illustrated in the lower part of Fig. 1, we use a European-based data plane network, which spans four countries (i.e., Finland, France, Sweden, and Germany). Most of the major cities are interconnected and the network is emulated using mininet [12]. We assume that one *centralized server* for the consolidation of services is located in Helsinki, Finland (the remote location), whereas *edge servers* are located in the proximity of the most major cities: e.g., Paris, Munich.

Scenario: We consider two tenants that are sharing the physical network, and we demonstrate a migration of a firewall VNF and the VN reconfiguration procedure for one tenant. The corresponding tenant requests a VN with two edge nodes, in order to control his humanoid NAO robot [13] located in Paris with a remote controller located in Munich. An Apple iPad is used to host the controller application. In both Paris and Munich, two Linksys WRT1900AC WiFi access points are installed to provide the access to the virtual network, connecting the robot and the remote controller to the backbone network. While the second tenant is generating dummy cross traffic.

In order to control the robot, two UDP streams need to be established. The NAO robot generates the first UDP traffic stream in order to provide a live video feed of the exact robot position to the remote controller. Based on the received information, the remote controller issues control instructions to the robot using the second UDP traffic stream. However, before reaching the destinations, both traffic streams have to pass through the firewall VNF, which is initially located in Helsinki.

III. Demo Presentation

In this demonstration we focus on presenting two main features: *VNF Orchestration* and *Dynamic VN Reconfiguration*.

VNF orchestration: Controlling the robot remotely based on the real-time video streaming is a highly sensitive application. Moreover, the additional delay coming from consolidation of services in a *centralized server* can potentially harm the performance of the application. In case the QoS requirements of the remote control application are not met, a firewall migration is triggered by the VNF orchestrator. The *firewall* is then migrated to the *edge server* in Paris. The migration of the firewall VNF should reduce the end-to-end delay and improve the experienced QoS. Fig. 2 shows the reduction of the network delay between the NAO robot and the remote controller before and after the VNF migration over time.

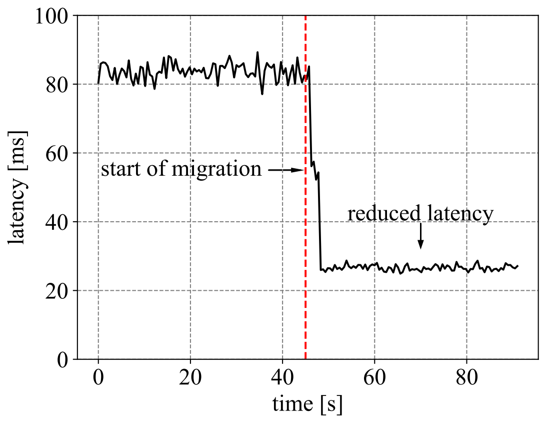

Fig. 2. Networking delay between the robot and the remote controller based on the given demonstration topology. During the first 46 seconds (left side of the red dashed line), the firewall VNF is located on the centralized server. The right side of the red line shows the network latency after the migration.

Dynamic VN Reconfiguration: After the VNF migration is triggered, HyperFLEX reconfigures the VN in order to meet the new requirements. The virtual links towards the *centralized location* in Helsinki are now redundant and can be removed, while the route connecting the edge servers in Paris and Munich is established. Additionally, isolation (e.g., bandwidth) policies spanning the used physical links between Munich and Paris are updated. The reconfiguration also ensures that the new VN requirements generated by the firewall migration do not affect other VNs.

IV. Conclusion and Future Work

The benefits of softwarization and consolidation of services are numerous. However, there are some parameters that should be considered: (1) locations far away from users can increase the networking delay, and (2) the processing time of software realizations of the networking functions is inherently higher compared to the hardware ones. Thus, Virtual Network Function (VNF) placement and management algorithms have to take service QoS requirements into the account.

In this demonstration, we showed the possible performance gains by migrating a softwarized VNF (i.e., a Linux-based firewall) from a centralized server to an edge server (closer to the user). In order to demonstrate the performance improvements, we supposed an application with a high QoS requirements: the remote control of a NAO robot over a secure network. We showed how the reduced latency is able to improve application quality greatly. Furthermore, we presented and discussed how the VNF orchestration should be realized in a virtualized SDN environment.

In the considered use case of this demonstration, the locations of the robot and the controller were static, since they were always connected to the same access points. As a future work, we plan to evaluate dynamic use cases like commercial aviation. Here, the positions of airplanes and the corresponding communication access points are well known in advance. Thus, including more granular and mobility-aware VNF migrations within the network could improve the performance and reduce the total operational cost. Moreover, a vehicular use case represents an even more challenging problem, as the movement of vehicles might not be predictable. Hence, more dynamic reconfigurations of VNs might become necessary for an efficient resource utilization.

Acknowledgment

This work has received funding from the CELTIC EUREKA project SENDATE-PLANETS (Project ID C2015/3-1) and is partly funded by the German BMBF (Project ID 16KIS0473). This work reflects only the authors view and the funding agency is not responsible for any use that may be made of the information it contains. The authors would like to thank reviewers for their fruitful comments.

References

[1] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying nfv and sdn to lte mobile core gateways, the functions placement problem," in *Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges*. ACM, 2014, pp. 33–38.

[2] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[3] A. Blenk and W. Kellerer, "Traffic pattern based virtual network embedding," in *Proc. ACM CoNEXT Student Workshop*, ser. CoNEXT Student Workhop '13. ACM, Dec. 2013, pp. 23–26. [Online]. Available: http://doi.acm.org/10.1145/2537148.2537151

[4] M. Henzinger, S. Neumann, and S. Schmid, "Efficient distributed workload (re-)embedding," in *ACM SIGMETRICS / IFIP Performance 2019*, June 2019. [Online]. Available: http://eprints.cs.univie.ac.at/6000/

[5] A. Varasteh, S. Hofmann, N. Deric, M. He, D. Schupke, W. Kellerer, and C. M. Machuca, "Mobility-aware joint service placement and routing in space-air-ground integrated networks," in *2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[6] A. Basta, A. Blenk, H. Belhaj Hassine, and W. Kellerer, "Towards a dynamic SDN virtualization layer: Control path migration protocol," in *Proc. IFIP/IEEE CNSM*. IEEE, Nov. 2015, pp. 354–359. [Online]. Available: http://ieeexplore.ieee.org/document/7367382/

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[8] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 655–685.

[9] N. Đerić, A. Varasteh, A. Basta, A. Blenk, and W. Kellerer, "Sdn hypervisors: How much does topology abstraction matter?" in *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 2018, pp. 328–332.

[10] A. Blenk, A. Basta, and W. Kellerer, "Hyperflex: An sdn virtualization architecture with flexible hypervisor function allocation," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 397–405.

[11] S. Ryu, "Controller," *URL:" https://osrg. github. io/ryu*, 2014.

[12] R. L. S. De Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*. IEEE, 2014, pp. 1–6.

[13] S. Robotics. (2006) Nao humanoid robot. [Online]. Available: https://www.ald.softbankrobotics.com/en/robots/nao