HyperFlex: Demonstrating Control-plane Isolation for Virtual Software-Defined Networks

Arsany Basta, Andreas Blenk, Yu-Ting Lai, Wolfgang Kellerer
Chair of Communication Networks
Department of Electrical and Computer Engineering
Technische Universität München, Germany
Email: {arsany.basta,andreas.blenk,yuting.lai@tum.de,wolfgang.kellerer}@tum.de

***Abstract*—We present a demonstration of HyperFlex [1], a Software-Defined Networking (SDN) virtualization architecture with flexible hypervisor function allocation guaranteeing control-plane virtualization. Network Virtualization (NV) is expected to overcome the ossification of today's communication networks. SDN is seen as an enabler for programmable network control. In order to fully virtualize software-defined networks, not only the virtualization of the data-plane, but also the virtualization of the control-plane has to be considered. HyperFlex is a virtualization hypervisor that ensures isolated control-plane slices for virtual SDN networks. Control-plane isolation also protects the hypervisor resources from exhaustion. Furthermore, virtualization hypervisors have to be scalable and flexible in order to provide the best possible performance for the virtual software-defined networks. They should be able to adapt to the current state of the network and the divergent demands of virtual SDN networks. HyperFlex distributes the hypervisor functions flexibly and dynamically in order to adapt to the current network state.**

I. Introduction

Network Virtualization allows multiple network providers to run virtual networks on a shared physical network simultaneously. In case of virtual Software-Defined Networks (vSDNs), the virtual network operators gain the ability to run their own SDN controller. Each vSDN controller is able to make steering decisions for the network traffic independently. Thus, virtual network operators can tailor the use of network resources in a fine-granular manner according to their demands. This is expected to increase network resource efficiency, thus overcome the ossification of today's networks [2].

An SDN hypervisor realizes the abstraction of the physical network to virtual networks, i.e, data-plane virtualization, and the control interface to each virtual SDN controller, i.e., control-plane virtualization. As the hypervisor is logically placed between the virtual SDN controllers and the vSDNs, the hypervisor performance can influence the operation of vSDNs. While existing research work mostly focuses on data-plane virtualization, concepts for control-plane virtualization have been neglected [3]. However, control-plane virtualization concepts are necessary to achieve high performance for vSDNs. It has already been shown for SDN networks that performance degradation on the control-plane impacts the performance of the data-plane [4]. For example, in case of a virtual network serving web-browsing sessions, high latency for flow-rule installations increases necessary DNS resolutions. This directly leads to longer page load times, which means a decrease in performance on the data-plane.

In order to tackle these issues, we propose an SDN virtualization architecture, namely HyperFlex, based on flexible hypervisor function allocation that aims at control-plane virtualization [1]. HyperFlex's functions' realization and placement adapt to the data-plane and control-plane performance of the target platform. We demonstrate the flexible hypervisor function allocation on the example of control-plane virtualization.

II. HyperFlex Concepts

1) Hypervisor Function Decomposition: The first main concept relies on the decomposition of the hypervisor virtualization functions required to realize vSDNs. This approach of function decomposition is seen to be more optimal in terms of resource efficiency. If a distributed hypervisor layer is needed, individual virtualization functions could be distributed such that a tailored virtualization layer is achieved instead of duplicating the whole hypervisor instance along the network.

2) Platform Independence; Flexible Function Allocation: The second main concept is the hosting platform of the SDN hypervisor layer. The hypervisor functions are mostly realized by software hosted on servers. HyperFlex additionally hosts hypervisor functions on SDN network elements that interconnect hypervisor servers with vSDN controllers or the physical network, which we call the hypervisor SDN network. HyperFlex uses all available processing functions, e.g., traffic shapers or packet inspection, of the hypervisor SDN network elements to execute hypervisor functions. Thus, the virtualization layer can be spanned across multiple platforms, i.e., hardware servers and network elements. The hypervisor network is operated and controlled by a hypervisor SDN controller. The proposed architecture can flexibly allocate virtualization functions, on a per-function basis, among servers and SDN network elements of the hypervisor network.

3) Control-plane Virtualization: In virtualized SDN networks, the control-plane is shared among multiple vSDN controllers. Providing isolated slices on the SDN control-plane resources, including the hypervisor, is the third main concept of HyperFlex. This guarantees that the performance of a vSDN controller is not influenced by other vSDN controllers.

III. HyperFlex Setup

We demonstrate a prototype implementation and setup for our proposed HyperFlex architecture shown in Figure 1. We assume a scenario where two vSDNs are deployed through HyperFlex. We show the flexible hypervisor function allocation

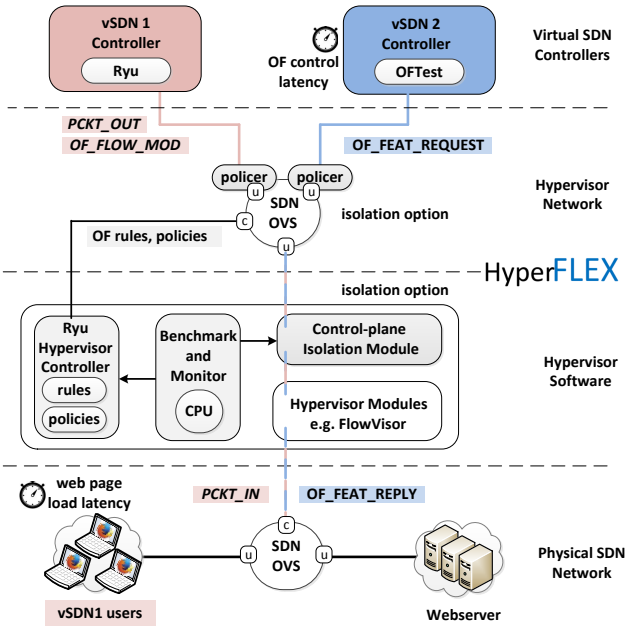

Fig. 1: HyperFlex Demonstration Setup

by adapting the control-plane isolation function according to the monitored hypervisor performance.

The setup can be classified into four different blocks. The virtual SDN controllers form the first block. We use *Ryu* [5], a python OpenFlow (OF) controller, to handle the flow setup for the users of vSDN1. OFTest [6], a test suite framework for OF switches, is used as the vSDN2 controller. OFTest is extended to generate an average sustainable rate of OF messages to test the hypervisor performance and control-plane virtualization.

The second setup block is the hypervisor software, where currently FlowVisor is installed to realize the core hypervisor functions, e.g., vSDN abstraction or OF message translation. We plan to develop our own hypervisor functions as future work. A software module is implemented to enforce control-plane traffic shaping prior to hypervisor processing. Shaping policies can be defined as OF message rate, packet rate or bitrate for each vSDN. Ryu is initialized to control the hypervisor network and to setup the control-plane isolation function, when triggered, on the hypervisor network elements. For benchmark and monitoring, a software module has been implemented to monitor the CPU utilization of FlowVisor's process. The observed CPU values are fed into the control-plane isolation software module, or alternatively, based on pre-defined threshold, the Ryu hypervisor controller is prompted to start the isolation function on the hypervisor network.

The third setup block is the hypervisor network realized by an OF Open vSwitch (OVS) [7]. OVS provides the capability to install rate limiting policies on the ingress ports. We use ingress policing to limit the control-plane flow towards the hypervisor to realize control-plane isolation. The physical SDN network is realized as an OVS controlled by FlowVisor. Finally, several users are started to measure data-plane performance, e.g., web page load latency for browsing application.

IV. Scenario and Use-Cases

For demonstrating HyperFlex's control-plane virtualization and flexble function allocation, the following scenario is considered. Users connect to vSDN1, run a browsing application and request a web page. The new flows need to be setup by vSDN1 controller through flow rules, i.e., *OF_FLOW_MOD*. Meanwhile, vSDN2 controller generates *OF_FEAT_REQUEST* messages to its vSDN and waits for *OF_FEAT_REPLY* messages, thus creating cross traffic on the shared control-plane and load on the shared hypervisor CPU resources. The data-plane performance is observed through the web page load latency for the users of vSDN1, while the control-plane performance is measured by the *OF_FEAT_REPLY* latency. The demonstrated use-cases are:

(a) Hypervisor CPU under-utilization: In this case, the hypervisor is able to handle control-plane traffic from both vSDN1 and vSDN2. This case is used as a reference to monitor the CPU utilization, data and control-plane latency.

(b) Hypervisor CPU over-utilization: This use-case can be triggered by generating *OF_FEAT_REQUEST* messages from vSDN2 at a high rate such that the hypervisor's CPU is overutilized. We show the performance degradation, in terms of latency compared to the reference underutilized case, for both data-plane, i.e., vSDN1 users, as well as control-plane, i.e., vSDN2 controller. This means that the action of vSDN2 controller influences the performance of vSDN1, which is unfair and should be avoided.

(c) Hypervisor Software Isolation: This corresponds to activating the control-plane isolation module at the hypervisor software. Control-plane shares are defined for each vSDN, e.g., 50% each. vSDN2 attempts to exceed its control-plane share with *OF_FEAT_REQUEST* messages. Having hypervisor software isolation, vSDN2 experiences high control-plane latency overhead, as in over-utilization, due to exceeding its control-plane share. The page load latency for the users of vSDN1 can be improved only if enough CPU is available for the additional processing of the isolation module at the hypervisor. Otherwise, the same data-plane latency for vSDN1 users as in over-utilization can be noted.

(d) Hypervisor Network Isolation: The page load latency is shown to be comparable to the reference underutilized case. This means that the control-plane is fully isolated and the cross influence between the vSDNs is evaded.

References

[1] A. Blenk, A. Basta, and W. Kellerer, "Hyperflex: An sdn virtualization architecture with flexible hypervisor function allocation," in *IFIP/IEEE IM*, 2015.

[2] T. Anderson, L. Peterson *et al.*, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.

[3] R. Sherwood, J. Naous *et al.*, "Carving research slices out of your production networks with OpenFlow," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, p. 129, Jan. 2010.

[4] A. Tootoonchian and Y. Ganjali, "Hyperflow : A distributed control plane for openflow," in *Proc. NSDI INM/WREN*, 2010.

[5] "Ryu SDN Framework." [Online]. Available: http://osrg.github.io/ryu/

[6] "OFTest." [Online]. Available: https://github.com/floodlight/oftest

[7] "Open vSwitch (OVS)." [Online]. Available: http://openvswitch.org/