TableVisor: An Emulation Layer for Multi-Table OpenFlow Switches

Steffen Gebert*, Michael Jarschel§, Stefan Herrnleben*, Thomas Zinner*, Phuoc Tran-Gia*
*University of Wuerzburg, Wuerzburg, Germany
§Nokia, Munich, Germany
Email: {steffen.gebert|stefan.herrnleben|zinner|trangia}@informatik.uni-wuerzburg.de, michael.jarschel@nokia.com

I. Introduction

Software Defined Networking (SDN) changes the way, how network devices operate: complex forwarding logic is implemented in easy-to-maintain, scalable and hardware independent software, while packet processing at line rate is done by relatively simple hardware devices. The softwarization of network elements is also the goal of Network Functions Virtualization (NFV). Instead of implementing logic in integrated and expensive middleboxes, network functions are running as virtualized software instances on commodity hardware. Due to performance challenges and slow market introduction of SDN hardware, recent research suggests programmable data paths, i.e., Protocol Independent Forwarding (OF-PI) [1]. This combines both approaches: complex logic implemented in software, while programmable hardware does packet processing. Similar to the split of responsibility in SDN, this offers a split between data and control plane also for NFV scenarios.

Many network functions can already be implemented using OpenFlow-based switching hardware and corresponding controller applications, often based on switches containing multiple flow tables. Compared to single-table switches, multiple tables remedy the explosion of flow table entries or are even required for implementing particular use cases [2]. However, costly implementation of multiple tables in hardware switches lead to a diversification of switching hardware and features.

The lack of available switching hardware offering large, versatile or even freely programmable flow tables still hinders development of SDN solutions to use cases requiring multiple flow tables or pipeline processing. To remedy this issue, this paper introduces *TableVisor* (TV). TV is a proxy between multiple interconnected (single-table) OpenFlow switches forming a desired processing pipeline or Table Type Pattern (TTP) [3] and an OpenFlow controller. It represents the connected switches to the controller as a single switch with the desired configuration of tables. By rewriting particular messages exchanged through the control channel, e.g., *flow-mod*s including references to particular tables, the controller application can be programmed as if a multi-table switch were connected. TableVisor thus offers a way to create controller applications relying on multiple tables with specific features even before compatible devices are available. This simplifies prototype development, as existing hardware can be used.

This demonstration shows TableVisor enabling a multi-table SDN application to control an emulated multi-table switch offering MPLS support, while these tables are realized using multiple hardware switches offering diverse capabilities.

II. TableVisor

TableVisor[1] acts as a proxy layer between an OpenFlow controller and switches (cf. Figure 1), similarly to other existing frameworks extending OpenFlow's capabilities, i.e., FlowVisor [4], or Hyperflex [5]. TableVisor's goal, however, is fundamentally different: First, multiple switches establish an OpenFlow secure channel connection to TableVisor. Afterwards, TableVisor connects to the configured controller, pretending to be one single switch. Further, TV advertises in the answer to the controller's *features-request* that this switch comprises a certain number of tables, which actually reflects the number of connected switches[2].

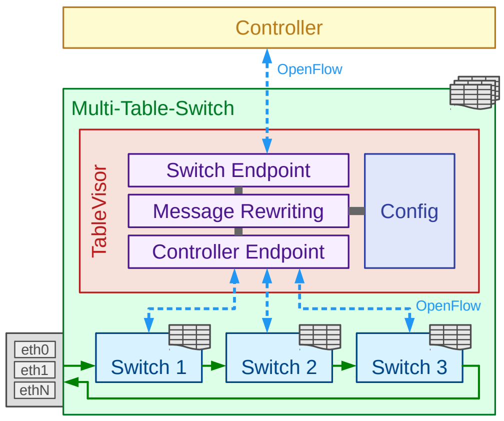

Fig. 1. Architecture of TableVisor.

All communication between controller and switch is passed through the proxy layer. Depending on the type of message, it will be answered directly by TableVisor (*hello*, *features-reply*), modified in such a way that particular fields are rewritten (*packet-in*, *flow-mod*), or a response merging data from multiple switches will be returned (*flow-stats*).

By letting TableVisor take care of rewriting OpenFlow messages, the controller-side implementation can act as if a single switch offering the combined set of tables provided by the connected hardware switches is connected. Further, this allows to explore multi-table use cases that are not yet realizable with current hardware due to a small number or lack of flow table capabilities.

[1]https://github.com/lsinfo3/TableVisor, licensed under Apache 2.0 License
[2]we are working towards using more than one table per connected switch

III. Example: Future Mobile Network

Instead of the currently static configuration of mobile networks, future mobile networks need to dynamically adapt resources according to the current situation. This includes planning of which data center should be used to provide virtualized resources [6], as well as a dynamic instantiation of resources close to the end users [7].

Programmable hardware on the user plane, controlled by software instances running in the cloud, is key to such flexibility and meeting the performance requirements. Similar to OpenFlow switches, these advanced network elements should offer packet forwarding capabilities in hardware, while control plane logic is running in software instances in cloud data centers. A widely discussed scenario in this area is the separation of user and control plane of the LTE mobile gateways, i.e., Serving Gateway (SGW) and Packet Data Network Gateway (PGW). To handle traffic of mobile users according to the 3GPP standard, the network elements realizing the user plane need to process GPRS Tunneling Protocol (GTP) packets. However, such programmable, GTP-enabled devices do not yet exist. With TableVisor, the development of the control plane applications and indeed their testing can already begin after the processing pipeline for such a device is specified – in parallel to the hardware development process.

IV. Demonstration

For the purpose of this demonstration, we chose to realize a simple MPLS Label Edge Router with a data plane processing pipeline consisting of switches abstracted by TableVisor.

The demonstration setup consists of the following parts:

- Traffic generator *Spirent TestCenter C1*, sending MPLS traffic and receiving the decapsulated IP traffic.
- Ryu controller with MPLS-based multi-table forwarding application, assuming one multi-table switch.
- Multiple OpenFlow hardware switches comprising different capabilities: One MPLS-capable *NEC PF5240*, multiple *HP ProCurve 2920*, connected via Ethernet.
- TableVisor, emulating one multi-table switch towards the controller, realized by all available switches.

The setup is illustrated in Figure 2. The table-based pipeline implemented by the controller's MPLS application is reflected by the physical connectivity of the hardware switches. TableVisor abstracts these switches towards the controller and emulates one multi-table switch. All flow entries sent by the controller are rewritten to match the pipeline setup of the connected hardware switches.

The flow tables of the different switches are programmed such that incoming traffic is checked for an MPLS ethertype in the first switch. Such a match does not require any MPLS capabilities. Once MPLS traffic is detected, it is forwarded to the second, MPLS capable switch. This one executes the *Pop MPLS header* action for labels associated with egress traffic. Remaining tables then provide functionality for IP routing and selecting the appropriate egress port, as well as next hop.

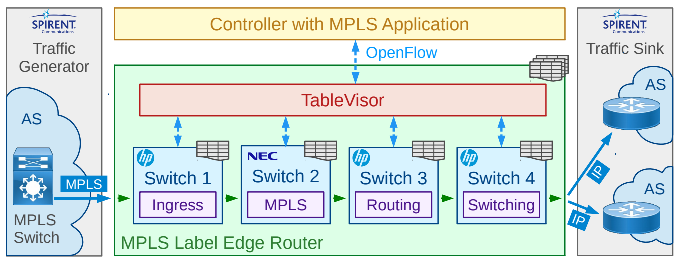

Fig. 2. Demo setup consisting of traffic generator and emulated multi-table OpenFlow switch. Particular tables are realized using one switch per table.

V. Conclusion & Outlook

This work first demonstrates the capabilities of TableVisor to emulate a hardware-accelerated multi-table OpenFlow switch. Especially tunneling protocols like GTP or MPLS require multiple flow tables. The described demonstration illustrates the capabilities of TableVisor to support building a prototype setup comprising a number of flow tables that might not be available in a single OpenFlow hardware switch. Similar to the first steps towards OpenFlow hardware offering MPLS support [8], parts of a setup, i.e., the GTP processing, can be implemented using specialized components, like a NetFPGA.

Future extensions of TableVisor aim to support Table Type Patterns including the mapping between different TTPs [9], which can reflect the physical connectivity of switches. This way, controller implementations can be tested before other hardware supporting the desired TTP might be available.

Acknowledgments

This work has been carried out in the framework of the CELTIC EUREKA project SASER-SIEGFRIED (Project ID CPP2011/2-5), and it is partly funded by the BMBF (Project ID 16BP12308). Hardware devices partially provided by DFG CRC 1053 MAKI. The authors want to thank Boris Henning and Volker Pianowski from Spirent Communications GmbH, as well as Peer Hasselmeyer and his colleagues from NEC Laboratories for their support.

References

[1] "OF-PI: A Protocol Independent Layer," Open Networking Foundation, Tech. Rep., Sep. 2014.

[2] "The Benefits of Multiple Flow Tables and TTPs," Open Networking Foundation, Technical Report, Feb. 2015.

[3] "OpenFlow Table Type Patterns," Open Networking Foundation, Specification, Aug. 2014.

[4] R. Sherwood, G. Gibb, K.-K. Yap *et al.*, "FlowVisor: A Network Virtualization Layer," Tech. Rep., 2009.

[5] A. Blenk, A. Basta, and W. Kellerer, "HyperFlex: An SDN Virtualization Architecture with Flexible Hypervisor Function Allocation," in *IFIP/IEEE IM*, 2015.

[6] A. Basta, A. Blenk, M. Hoffmann *et al.*, "SDN and NFV Dynamic Operation of LTE EPC Gateways for Time-Varying Traffic Patterns," in *Mobile Networks and Management*, 2015, vol. 141.

[7] S. Gebert, D. Hock, T. Zinner *et al.*, "Demonstrating the Optimal Placement of Virtualized Cellular Network Functions in Case of Large Crowd Events," in *SIGCOMM*. ACM, 2014.

[8] J. Kempf, S. Whyte, J. Ellithorpe *et al.*, "OpenFlow MPLS and the Open Source Label Switched Router," in *23rd International Teletraffic Congress*, 2011.

[9] C. Beckmann, "ONF Blog: The Evolving TTP Architecture," 07 2015. [Online]. Available: https://www.opennetworking.org/?p=1811&option=com_wordpress&Itemid=316