Packet Forwarding Using OpenFlow

Marcelo D. D. Moreira, Natalia C. Fernandes, Hugo E. T. Carvalho, Lyno Henrique G. Ferraz, Rodrigo S. Couto,
Igor M. Moraes, Miguel Elias M. Campista, Lus Henrique M. K. Costa, Otto Carlos M. B. Duarte
Universidade Federal do Rio de Janeiro - GTA/COPPE - Rio de Janeiro, Brazil

***Abstract*—OpenFlow is a network virtualization platform that separates the network control function from the forwarding function. OpenFlow defines a centralized element, called OpenFlow controller, that controls and programs shared forwarding tables in forwarding elements, called OpenFlow switches. The forwarding structure is highly flexible, because a packet is forwarded based not only on the destination IP address, but on user-defined header fields. This paper presents a performance evaluation of OpenFlow switches acting as a virtual network element in a personal computer.**

I. Introduction

The OpenFlow protocol defines a secure communication channel between OpenFlow switches and the controller, which uses this channel to monitor and configure OpenFlow switches [1]. The flow definition is generalized to an n-tuple of header fields, enabling a flexible forwarding structure based on multilayer wildcards. OpenFlow does not assume virtualized data planes on forwarding elements and, consequently, follows the model of one data plane shared by all virtual networks. Consequently, it is expected for OpenFlow performance the same performance of the native packet forwarding. OpenFlow, however, shows a disadvantage when a flow is not yet configured, because the first packet of unclassified flows must be forwarded to the controller. Then, the controller sets a path for the following packets of the flow in the chosen OpenFlow switches. This mechanism may introduce a significant delay, particularly if the traffic is mostly composed of small flows.

II. Experimental Results

We deploy OpenFlow in a Linux system and use native Linux performance as a reference to measure the overhead introduced by OpenFlow virtualization. We evaluate the performance of OpenFlow in a testbed composed of three machines. Our PC-based OpenFlow switch prototype forwards traffic from a traffic generator to a traffic receiver machine. Our prototype is an HP Proliant DL380 G5 server equipped with two Intel Xeon E5440 2.83 GHz processors and 10 GB of RAM, set up with just one logical CPU.

Our first experiments measure the forwarding rate achieved by OpenFlow forwarding solution. The packet forwarding rate analysis is accomplished with minimum (64 bytes) and large (1512 bytes) frames. We use 64-byte frames to generate high packet rates and force high packet processing in the OpenFlow switch and 1512-byte frames to saturate the 1 Gb/s physical link. Fig. 1 shows that OpenFlow performs close to Native Linux in router mode. Because the data plane is shared by all virtual networks, OpenFlow causes a low processing overhead.

Next, we analyze OpenFlow behavior with multiple networks. In this scenario, each network is represented as a flow of packets. If there is more than one parallel flow, the traffic is equally divided among the flows, maintaining a fixed aggregated traffic. The results shows that OpenFlow switch performance is similar to a software bridge running over Native Linux, maintaining the received packet rate close to the generated rate, despite the fact that the first packet of the flow must go to the OpenFlow controller.

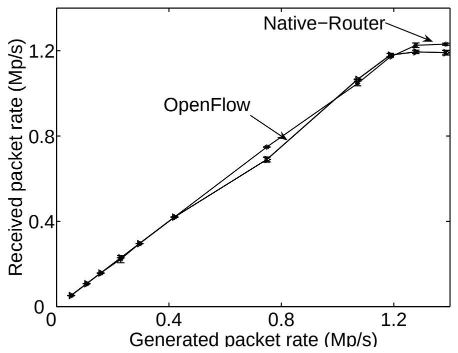

Figure 1. Packet forwarding rate, using 64-byte frames.

We also analyze the impact of OpenFlow on traffic latency. We create background traffic with different rates to be forwarded by OpenFlow switch. For each of those rates, an ICMP echo request is sent, from the generator to the receiver, to evaluate the round trip time (RTT) and the jitter according to the generated background traffic. The results show that OpenFlow packet processing introduces no measurable delay and jitter, and hence does not affect real-time applications.

III. Conclusions

OpenFlow network virtualization model follows the shared data plane approach by defining a centralized element that controls and programs the forwarding table in each network element. Our results demonstrate that the generalization of a flow to an n-tuple of header fields enables a flexible and yet performant forwarding structure. Our OpenFlow prototype forwards packets as fast as native Linux. Further experiments show that OpenFlow is a suitable platform for network virtualization, because it is proved to support multiple instances of virtual networks with no measurable performance loss.

Acknowledgments

This work was supported by FINEP, FUNTTEL, CNPq, CAPES, and FAPERJ.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S., and J. Turner. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.