OpenNetVM: A Platform for High Performance NFV Service Chains

Demo Paper

Mykola Yurchenko, Patrick Cody, Aaron Coplan, Riley Kennedy, and Timothy Wood
George Washington University

K. K. Ramakrishnan
University of California Riverside

ABSTRACT

Middleboxes in traditional networks relied on purpose-built hardware/software appliances to run data plane services, making it difficult for networks to evolve. OpenNetVM seeks to address this problem by offering a flexible Network Function Virtualization framework designed for rapid development and deployment of virtualized network functions. By efficiently chaining together a series of network functions, complex services can be created and adapted to meet changing needs. OpenNetVM provides an easy to use platform for NF development and research into network function management. This demo will present the improvements we have made to OpenNetVM since its initial open source release two years ago, including new functionality such as more efficient service chaining and a TCP stack to allow integrated deployments of middleboxes and end host applications.

CCS CONCEPTS

• **Networks → In-network processing**;

KEYWORDS

Network Function Virtualization

1 INTRODUCTION

Today's networks are filled with a range of middleboxes from simple switches to complex intrusion detection systems [4]. These network functions were traditionally provided by expensive, specialized hardware, which leads to high costs for network operators and makes network expansion a costly procedure.

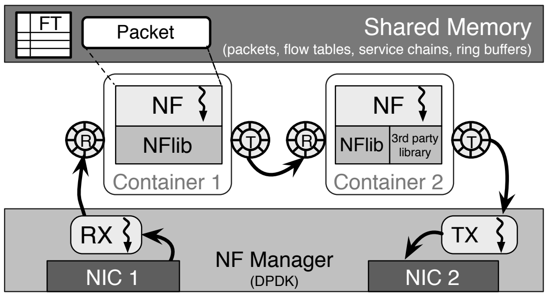

Figure 1: OpenNetVM Architecture

Network operators would like networks to be more flexible, while still maintaining high performance and low-cost, so that a network could easily evolve as requirements change. Replacing services currently provided by hardware with software implementations based on Network Function Virtualization has a number of benefits, including the desired flexibility and cost-reduction. Because virtual network functions can easily be added and removed, network services can adapt more quickly, without needing to physically change any hardware. Furthermore, deployment and scalability become easier, as creating new services to enhance a network is simply a matter of launching a new program, instead of needing to physically install hardware.

OpenNetVM [5] is an open source NFV platform designed to accelerate research and development of virtual network functions by creating an abstraction layer over the Linux Foundation's Data Plane Development Kit (DPDK). Through these abstractions, OpenNetVM allows network functions (NFs) to easily be chained together and run on the same host, allowing for complex service chains between NFs with minimal latency. Each NF can be launched as its own light-weight Docker container with unique dependencies, simplifying deployment. In this demo we will present how OpenNetVM can be used by researchers and developers to easily prototype new network services and explore research into NF management.

SOSR '18, March 28–29, 2018, Los Angeles, CA, USA
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5664-0/18/03.
https://doi.org/10.1145/3185467.3190786

2 OPENNETVM DEMO

Network functions are often deployed as service chains, where several NFs work together to provide complex services. By intentionally designing NFs to be modular and single-function, they can then be reused in a wide variety of use-cases. The danger in using multiple NFs, however, is the potential for high performance costs if data needs to be copied between each NF. This problem is exacerbated by longer service chains. For long and complex service chains to be performant, data must be directly shared between NFs, to minimize unnecessary latency.

OpenNetVM provides a framework for operators to run multiple network functions on a server and link them together to form complex chains, as shown in Figure 1. Each NF is bound to its own CPU core and is run as a separate process or light-weight Docker container, but packet-copying is avoided using shared memory. Through DPDK's zero-copy IO functionality, even long chains of NFs can maintain high throughput. To further reduce latency, each NF is also responsible for its own local sending (TX) and receiving (RX) operations. This reduces the total number of cores required to run OpenNetVM, as no core needs to be dedicated solely to packet transferring, and it maximizes the total number of NFs that can be run on a single host.

The OpenNetVM manager provides load balancing between NFs, flexible flow management, and service name abstractions. To send and receive packets from the host's NICs, it creates TX and RX threads to mediate access between NFs and the NIC. Upon receiving packets, it distributes them to NFs, potentially following an SDN Controller provided service chain. NFs are able to directly communicate with each other, or the manager can be used as an intermediary in order to enforce security policies. Finally, the manager retains full control of the NICs, and handles sending packets out on the network for other NFs.

Sample Network Functions: OpenNetVM provides a library of NF examples to build simple service chains and guide NF development. We currently maintain 12 NFs including basic network functions that handle forwarding, monitoring packets, bridging together NIC ports, and resolving ARP requests. We provide a Layer 4 Load Balancer and multiple NFs that showcase different flow operations. Here we overview two examples and evaluate their performance.

The Speed Tester NF measures the throughput of the system by generating either fake packets or replaying PCAP files to simulate real traffic. To stress test packet movement through ONVM, a service chain of Speed Tester NFs can be run. Because there is no data copying and each NF handles its own sending and receiving of packets, we get high throughput even for long NF chains. Our measurements show that a chain of length two has a maximum throughput of 32 million packets per second, while extending the chain to seven NFs only incurs a 10% throughput drop.

OpenNetVM also supports Snort, a common intrusion detection system used by network administrators. We have modified the Snort data acquisition module (DAQ) so that it can receive packets from ONVM using zero-copy [3]. OpenNetVM's support for load balancing across replicated NFs allows Snort to linearly scale performance, meeting a maximum throughput of nearly 7 Gbps with ten replicas when using a computationally intensive ruleset. In contrast, standard Snort with the kernel interface barely reaches 2 Gbps with the same number of replicas.

Many advanced network functions require access to a TCP stack, so we have ported the high performance mTCP [2] and mOS [1] libraries to work with ONVM. This allows OpenNetVM to act as an endpoint capable of running applications such as the lighttpd web server. Running mTCP with OpenNetVM makes it easy to setup complex network services, such as running multiple servers and middleboxes integrated on a single machine.

Demo Plans: We will demonstrate how OpenNetVM can be easily used by researchers and developers. We will show the basic steps to start the management layer and run network functions. We provide both console and web statistics to visualize NF performance. Finally, we provide experiment templates on the NSF CloudLab platform, thus making it fast and easy to setup OpenNetVM.
Demo Video: **https://youtu.be/FHx5iwafxKI**
Source code: **https://github.com/sdnfv/openNetVM**

Acknowledgments: This work was supported in part by NSF grants CNS-1422362 and CNS-1522546.

REFERENCES

[1] Muhammad Asim Jamshed, YoungGyoun Moon, Donghwi Kim, Dongsu Han, and KyoungSoo Park. 2017. mOS: A Reusable Networking Stack for Flow Monitoring Middleboxes. In *USENIX Symposium on Networked Systems Design and Implementation.*

[2] EunYoung Jeong, Shinae Wood, Muhammad Jamshed, Haewon Jeong, Sunghwan Ihm, Dongsu Han, and KyoungSoo Park. 2014. mTCP: a Highly Scalable User-level TCP Stack for Multicore Systems. In *USENIX Symposium on Networked Systems Design and Implementation.*

[3] Guyue Liu, K. K. Ramakrishnan, Mike Schlansker, Jean Tourrilhes, and Timothy Wood. 2017. Design Challenges for High Performance, Scalable NFV Interconnects. In *Proceedings of the Workshop on Kernel-Bypass Networks (KBNets '17).*

[4] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. 2012. Making Middleboxes Someone else's Problem: Network Processing As a Cloud Service. *SIGCOMM Comput. Commun. Rev.* 42, 4 (Aug. 2012).

[5] Wei Zhang, Guyue Liu, Wenhui Zhang, Neel Shah, Phillip Lopreiato, Gregoire Todeschi, K.K. Ramakrishnan, and Timothy Wood. 2016. OpenNetVM: A Platform for High Performance Network Service Chains. In *Proceedings of the 2016 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization.* ACM.