OpenNetVM: Flexible, High Performance NFV (Demo)

Wei Zhang* Guyue Liu* Wenhui Zhang* Neel Shah* Phil Lopreiato* Gregoire Todeschi‡
K.K. Ramakrishnan† Timothy Wood*
*The George Washington University ‡INP ENSEEIHT †University of California Riverside

Abstract

Network Function Virtualization promises to enable dynamic management of software-based network functions. We envision a dynamic and flexible network that can support a smarter data plane than just simple switches that forward packets. This network architecture supports complex stateful routing of flows where processing by network functions (NFs) can transform packet data, customized on a per-flow basis, as it moves between end points. This demo will present OpenNetVM, a highly efficient packet processing framework that greatly simplifies the development of network functions, as well as their management and optimization. OpenNetVM runs network functions in lightweight Docker containers that start in less than a second. The OpenNetVM platform manager provides load balancing, flexible flow management, and service name abstractions. OpenNetVM uses DPDK for high performance I/O, and efficiently routes packets through dynamically created service chains. We will demonstrate how the research community can easily build new network functions and rapidly deploy them to see their effectiveness in high performance network environments.

I. Introduction

Network Function Virtualization (NFV) will enable a vast array of in-network software functions running efficiently in virtualized environments. These functions range from lightweight software switches, function-rich firewalls, high performance proxy engines to complex intrusion detection and protection systems (IDS/IPS). Network operators seek to deploy these as parts of complex service chains comprising multiple network functions (NFs). By allowing processing capabilities to be instantiated on demand exactly when and where they are needed, complex services can be flexibly embedded in the network. Combined with Software Defined Networking (SDN), NFV promises to enable customized packet processing and routing for individual packet flows.

We seek to accelerate NFV research and development. To this end, we have developed OpenNetVM, a platform for high performance NFV. OpenNetVM builds upon our NetVM [1] NFV platform to support network functions running in Docker containers. OpenNetVM retains the shared memory framework of NetVM, and its ability to support complex network service chains at high performance. OpenNetVM provides flexible management capabilities so that NFs can be easily deployed and combined into dynamic service chains.

This work is supported in part by the US NSF under grants CNS-1422362 and CNS-1522546.

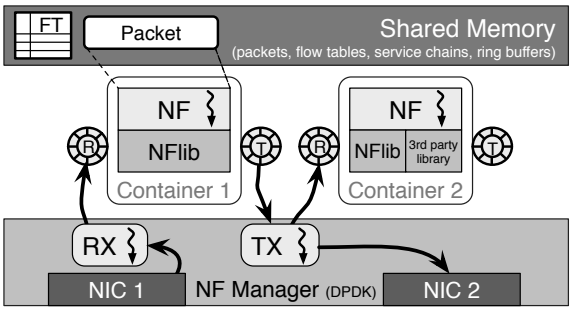

Fig. 1. The NF Manager creates a shared memory region to store packets and meta data such as the flow table and service chain lists. Packets are moved between NFs by RX and TX threads that copy packet descriptors into an NF's receive (R) and transmit (T) ring buffers. NFs run in isolated containers that encapsulate dependencies.

We complement other efforts to provide high performance network I/O such as Intel's Data Plane Development Kit (DPDK) and netmap [2, 3]. These high performance I/O libraries allow developers to build efficient NF prototypes. However, they do not assist with the composition of NFs nor in their management. Thus, while such low-level tools to build network functions are becoming available, we still lack a platform that provides the higher level abstractions needed to compose them into service chains, control the flow of packets among them, and manage their resources.

Our demo of OpenNetVM will illustrate some of its key features:

Container-based NFs: Network functions run as standard user space processes inside Docker containers, making them lighter weight than virtual machines, but still allowing a versatile linux development environment.

NF and flow management: OpenNetVM's management framework tracks what network functions are running and how they are composed into service chains. An SDN Controller or individual NFs can update a flow table to steer packets between NFs.

Efficient I/O: OpenNetVM uses DPDK to bypass the kernel, allowing zero-copy access to DMA'd packets from a user space poll-mode driver. We extend this to support zero-copy I/O in service chains of multiple NFs using a shared memory accessible to each Docker container. The only information that may need to be copied across NFs is a small packet descriptor, which also allows us to pass critical state information between NFs.

Scalability and Optimizations: NFs can be easily replicated for scalability, and the NF Manager will automatically load balance packets across threads to maximize performance. The framework is carefully optimized with techniques such as cached flow table lookups to avoid bottlenecks occurring in the management layer.

Combining these features results in an efficient and easy to use platform available for the community's use.[1] Our demo will present the OpenNetVM architecture, demonstrate its performance, and provide information on how research groups can make use of it in their work.

II. OpenNetVM Architecture

As shown in Figure 1, OpenNetVM consists of the NF Manager, which interfaces with DPDK-capable NICs, maintains flow tables, and facilitates communication between NFs; it includes a library, NFLib, which provides the API within an NF to interact with the manager; and user-written NFs that make use of that API. The NF Manager typically runs on the host, although it may be run inside a container if desired. The NFs, compiled together with NFlib and any desired 3rd party libraries, run as Docker containers.

OpenNetVM can very efficiently move packets from the NIC into an NF for processing. This is illustrated in Figure 2, which shows how OpenNetVM performs under a realistic web traffic workload of HTTP packets. We show that with this traffic mix, which includes 44.3% of 64 bytes, 18.5% of 65-1023 bytes, and 37.2% of 1024-1518 bytes, we can achieve nearly 70 Gbps. In our experiment, the packets are sent to an NF, which immediately forwards the packets back out the NIC port. When configured with one RX thread, one TX thread, and one NF, OpenNetVM achieves 48 Gbps. If we start a second NF and add another RX and TX thread to the manager, OpenNetVM automatically load balances flows across the NFs to achieve 68 Gbps.

If we reconfigure the system to create a linear service chain of five NFs, with 2 RX and 5 TX threads. This setup gets a throughput of 40 Gbps. Note that the chain of five NFs increases the load on the manager by 5X—the TX threads are steering packets at a total rate of 34 million packets per second. This configuration consumes all of the cores on our server, but we expect that with additional cores the entire service chain could be replicated to further increase throughput.

These experiments illustrate the potential of using OpenNetVM as a platform for software routers even deep in to a core network or a data center. We believe OpenNetVM can serve as a platform for network resident functions (acting essentially as a 'bump in the wire'), and high performance network routers and switches at arbitrary points in the network. This also enables the convergence of cloud data centers and the network.

III. Conclusion

OpenNetVM allows researchers and developers to prototype high performance network functions and experiment with their management. Using containers to host NFs greatly simplifies deployment by multiple vendors since they are self-contained and modular. Further, containers consume fewer resources than regular virtual machines and can start up in less than a second. In conjunction with an orchestrator, complex service chains can be created and network functions placed dynamically. By coordinating with an SDN controller, a flow can be appropriately routed through the service chain. OpenNetVM is carefully optimized with lock-free data structures, NUMA-awareness, and zero-copy I/O to maximize performance while still providing isolation. We will demonstrate the effectiveness of the platform and help researchers get started using it for their own work.

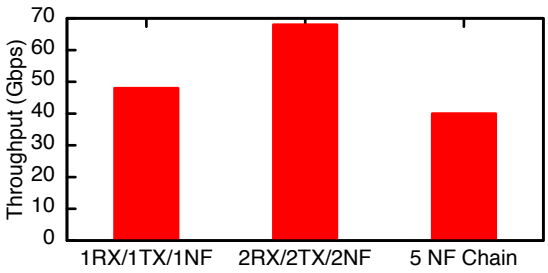

Fig. 2. OpenNetVM achieves nearly 70 Gbps on real traffic using only six cores.

[1] Source code and NSF CloudLab images at http://sdnfv.github.io/

References

[1] J. Hwang, K. Ramakrishnan, and T. Wood. NetVM: High Performance and Flexible Networking using Virtualization on Commodity Platforms. In *Symposium on Networked System Design and Implementation*, NSDI 14, Apr. 2014.

[2] Intel. Data plane development kit.

[3] L. Rizzo. netmap: A Novel Framework for Fast Packet I/O. In *USENIX Annual Technical Conference*, pages 101–112, Berkeley, CA, 2012. USENIX.