Software-Based Networks: Leveraging high-performance NFV platforms to meet future communication challenges

K.K. Ramakrishnan
University of California, Riverside

(kk@cs.ucr.edu)

Joint work with: Timothy Wood (GWU), our students, collaborators
**Network Function Virtualization**

- Run network functions in software

![Diagram of network components]

- More flexible than hardware
  - Easy to instantiate new NFs
  - Easy to deploy NFs; Easier to manage NFs

- Network Service Providers are migrating towards a software-based networking infrastructure
Virtualization Overheads

- **Virtualization layer** provides (resource and performance) isolation among virtual machines.
- **Isolation** involves many functions such as access permissions (security), ability to schedule and share etc.
- **Network overhead** (packet delivery) is one of the most critical concerns.
- A generic virtualization architecture includes several critical boundaries – host OS, virtual NIC, guest OS, and guest user space—getting packet data there includes **memory copies**.

Our Contributions with NetVM

1. A virtualization-based high-speed packet delivery platform
   - for flexible network service deployment that can meet the performance of customized hardware, especially when involving complex packet processing

2. Network shared-memory framework
   - that truly exploits the DPDK (data plane development kit) library to provide zero-copy delivery to VMs and between VMs (containers)

3. A hypervisor-based switching algorithm
   - that can dynamically adjust a flow’s destination in a state-dependent and/or data-dependent manner

4. High speed inter-VM communication
   - enabling complex network services to be spread across multiple VMs

5. Security domains
   - that restrict access of packet data to only trusted VMs
OpenNetVM – NFV Open Source Platform

http://sdnfv.github.io

• Network Functions run in Docker containers
• DPDK based design, to achieve zero-copy, high-speed I/O
  • Key: Shared memory across NFs and NF Manager
• Created an open source version
• Multiple industrial partners evaluating use of OpenNetVM
  • Of course, there are many competitors (e.g., Fast Data Project (fd.io), etc.)
**OpenNetVM Architecture**

- **NF Manager** (with DPDK) runs in host’s User Space
- **NFs** run inside Docker containers
  - **NUMA-aware** processing
  - **Zero-copy** data transfer to and between NFs
  - **No Interrupts** using DPDK poll-mode driver
  - **Scalable** RX and TX threads in manager
  - Each NF has its own ring to receive/transmit a packet descriptor
  - NFs start in 0.5 seconds; throughput of 68 Gbps w/ 6 cores
Chained Packet Delivery

- Packets in memory do not have to be copied
- Applications in containers pass packet references to other NFs – through the descriptor ring
- Only one application can access a given packet at any time for writing – avoid locks
Trusted and Untrusted Domains

- Virtualization should provide security guarantees among VMs
- OpenNetVM provides a security boundary between trusted and untrusted NFs
- Untrusted NFs cannot see packets from OpenNetVM
- Grouping of trusted NFs via huge page separation
Performance w/ Real Traffic

- Send HTTP traffic through OpenNetVM
  - 1 RX thread, 1 TX thread, 1 NF = 48Gbps
  - 2 RX threads, 2 TX threads, 2 NFs = 68Gbps (NIC bottleneck?)
  - 2 RX threads, 5 TX threads, chain of 5 NFs = 38Gbps

- Fast enough to run a software-based core router; Middleboxes that function as a ‘bump-in-the-wire’

![Graph showing throughput in Gbps for different configurations: 1RX/1TX/1NF, 2RX/2TX/2NF, 5 NF Chain]
Negligible performance difference between processes and containers.

- OpenNetVM sees only a 4% drop in throughput for a six NF chain, while ClickOS falls by 39% with a chain of three NFs.
Service Diversity & Multiple Flows

- A typical NF platform may host NFs for many different service chains
- Each flow may need customized services
Service Diversity & Multi-Flows

- NF platforms host NFs for many different service chains
- Each flow may need customized services
- Many different flows, each with slightly different need

![Diagram of NF Platform with components such as Video Detector, Policy Engine, Quality Detector, Transcoder, Firewall, Scrubber, IDS, Cache, LB, DPI, and related connections.](attachment:image.png)
Monolithic NFs

- Multiple flows have to go through an NF
  - Scheduling packets: complex, multiple flows share packet queues
  - NF must classify flows? NF manager?
  - Manage flow interference
  - Scalability: avoid restriction of 1 core per NF

Need a **high speed** platform which can **isolate** and process flows with **fine granularity** and **efficiently** use resources
Goal: Per-Flow NFs

- Make the flow the scheduling entity
- Deploy a unique NF for each flow or class of flows
Flurries

- A scalable platform for unique, short-lived NFs

(ACM CoNext 2016)

- Run unique NFs per flow or per class of flows

Benefits:
- do flow-level performance management
- Flexible and customized flow processing
Flurries

A scalable platform for unique, short-lived NFs

Flurries contributions:
- Hybrid polling and interrupts to efficiently run 1000s of NFs
- Flow director maps flows to NFs; NFFlib recycles NFs
- Adaptive wakeup system and prioritized NF scheduling

Challenges
- How to move packets efficiently across service chains?
- How to run large numbers of NFs on a host?
- How to manage the mapping of flows to NFs?
- How to schedule NFs?
Flurries Performance: Benefit of Hybrid Polling & Interrupts

- Throughput drops as the number of NFs increases on the core for polling and netmap.
- Flurries achieves good performance even with large number of NFs.
Scale Out

- Run up to 80,000 NFs in a one second interval per host
- Achieve 30Gbps traffic rate and incur minimal added latency to web traffic
NFVnice

A user space control framework for scheduling NFV chains.
ACM Sigcomm 2017

• NFVnice in a nutshell:
  – Complements the existing kernel task schedulers.
    • Integrates “Rate proportional scheduling” from hardware schedulers.
    • Integrates “Cost Proportional scheduling” from software schedulers.
  – Built on OpenNetVM[HMBox’16, NSDI’14]: A DPDK based NFV platform.
    • Enables deployment of containerized (Docker) or process based NFs.
  – Improves NF Throughput, Fairness and CPU Utilization through:
    • Proportional and Fair share of CPU to NFs: Tuning Scheduler.
    • Avoid wasted work and isolate bottlenecks: Backpressure.
    • Efficient I/O management framework for NFs.
NFVnice: Building Blocks

- cgroups: (control groups) is a Linux kernel feature that limits, accounts for and isolates the resource usage (CPU, memory, disk I/O, network) of a collection of processes.
- Back pressure: Chain-aware scheduling: Avoid wasted work (within and across cores)
- ECN: End to End Bottleneck/congestion control (across nodes)
- I/O Mgt.: Efficient Disk I/O Mgmt. Library

Work-conserving and proportional scheduling (within each core)
Rate-Cost Proportional Fairness

• **What is Rate-Cost Proportional Fairness?**
  – Determines the NFs CPU share by accounting for both:
    • NF Load (Avg. packet arrival rate, instantaneous queue length)
    • NF Priority and per-packet computation cost (Median)

• **Why?**
  – Efficient and fair allocation of CPU to the contending NFs.
  – *Provides upper bound on the wait/Idle time for each NF.*
  – Flexible & Extensible approach to adapt any QoS policy.
Summary

• Networks are changing – moving to a software base
  • SDN’s centralized control
  • NFV’s software based implementations

• NetVM/OpenNetVM efforts enhance industry direction
  • NFV platform provides significant performance improvement
  • A more coherent and effective software network architecture
Getting OpenNetVM

• Source code and NSF CloudLab images at
  http://sdnfv.github.io/