OpenContrail, Real Speed: Offloading vRouter

Chris Telfer, Distinguished Engineer, Netronome

Ted Drapas, Sr Director Software Engineering, Netronome

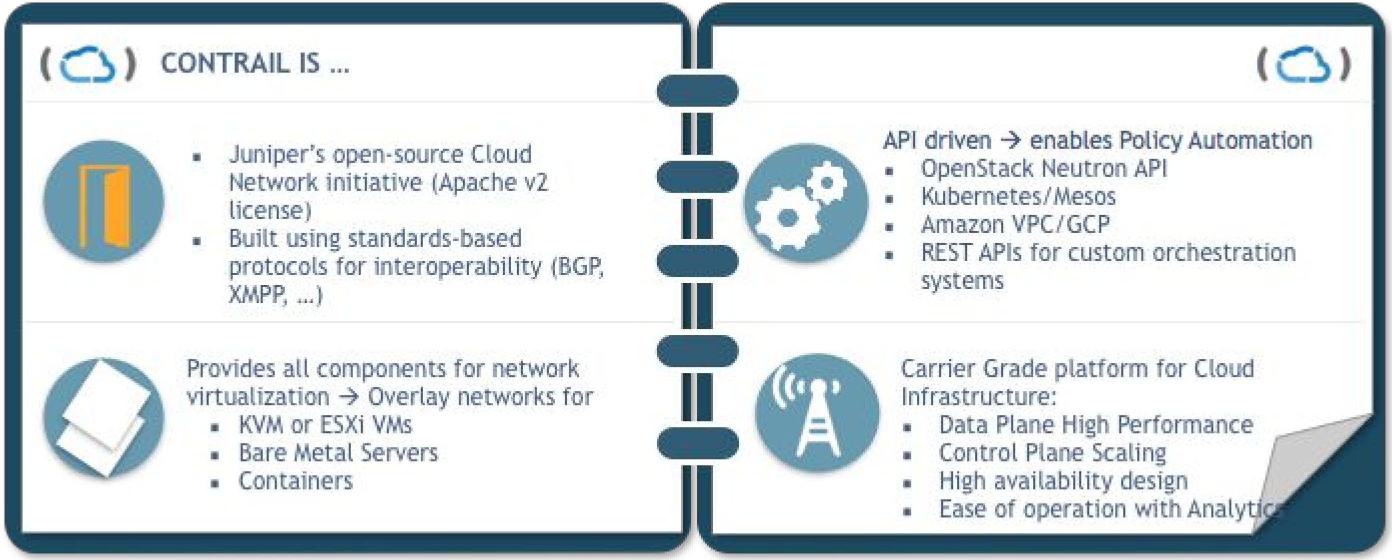
Agenda

- Introduction to OpenContrail & OpenContrail vRouter
- NFP Acceleration Strategy
- Software Architecture
- Results and Conclusions
- Questions

OpenContrail, Real Speed: Offloading vRouter

Introduction to OpenContrail and vRouter

What is Contrail?



CONTRAIL IS …

- Juniper's open-source Cloud Network initiative (Apache v2 license)
- Built using standards-based protocols for interoperability (BGP, XMPP, …)

Provides all components for network virtualization → Overlay networks for
- KVM or ESXi VMs
- Bare Metal Servers
- Containers

API driven → enables Policy Automation
- OpenStack Neutron API
- Kubernetes/Mesos
- Amazon VPC/GCP
- REST APIs for custom orchestration systems

Carrier Grade platform for Cloud Infrastructure:
- Data Plane High Performance
- Control Plane Scaling
- High availability design
- Ease of operation with Analytics

… For more information → opencontrail.org

Copyright and Courtesy of our partners Juniper Networks

What is Contrail?

OpenContrail consists of two main components

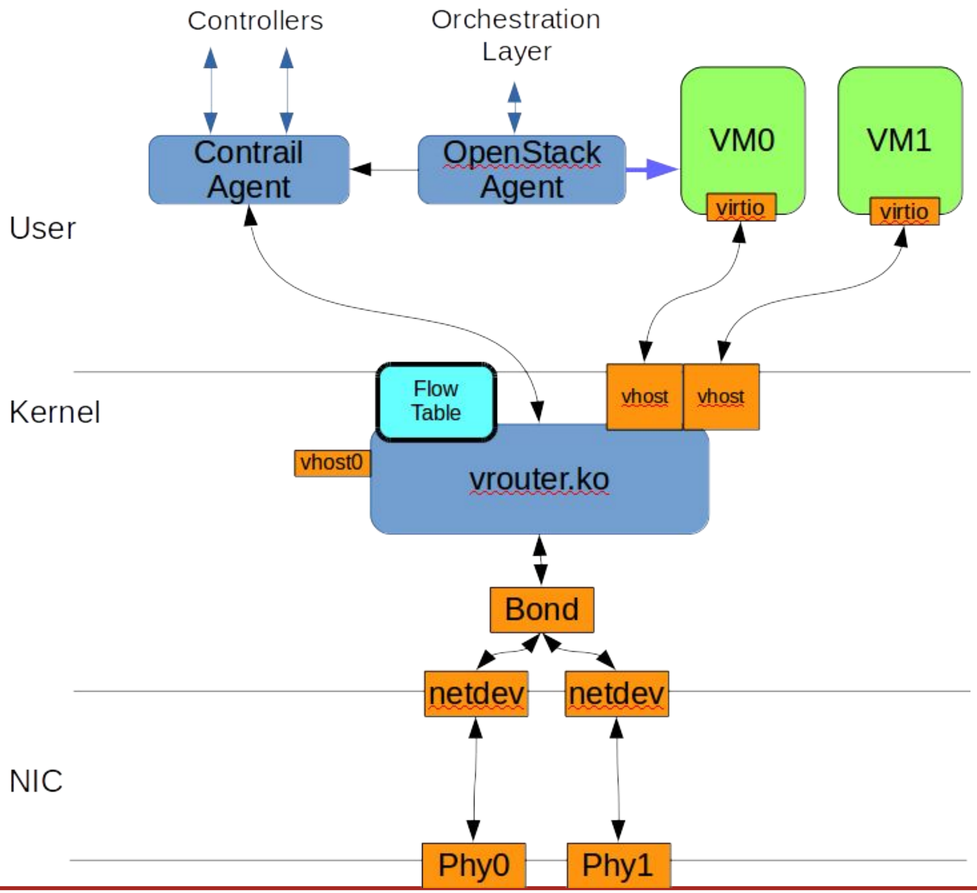
- OpenContrail Controller - Software Defined Networking (SDN) controller that is responsible for providing the management, control, and analytics functions of the virtual network
- OpenContrail vRouter - Programmable (by controller) datapath for managing data center networks

What is vRouter?

vRouter is the programmable datapath of OpenContrail

- Connects to an underlay network and manages overlay networks
- Similar in purpose to OVS, but designed specifically for datacenters
  - Assumes / optimizes the use of point-to-point tunnels over the underlay
  - UDP/MPLS (L2/L3), GRE/MPLS (L2/L3), VXLAN (L2 only)
- Is a “distributed device”
  - Acts as an L2 switch or L3 router on the overlay
  - Software sees one device, but it is spread across all compute nodes
  - Runs in the hypervisor
- Has both a kernel module and DPDK implementation

vRouter in a Compute Node



Problem Statement

Why the need for the NFP?

…..performance & overhead…..

- vRouter evolution without NFP
  - kernel vRouter - pkt delivery to VMs via kernel TAP interfaces - slow
  - ….so go to user space ….DPDK vRouter - significantly improves PPS compared to kernel vRouter, but at a significant price
  - DPDK vRouter achieves ~1.5 Mpps per core
- For the vRouter datapath to meet the needs of high-performance VNFs on compute nodes it must use significant CPU resources
- These resources therefore are not available for the VMs running on the compute node

Problem Solution

- Incorporating the NFP to offload the vRouter datapath (or any similar data path) on the host allows recovery of CPU resources
- Direct packet delivery from NFP into VMs can achieve higher performance than host-only software even with many dedicated cores

OpenContrail, Real Speed: Offloading vRouter

NFP Acceleration Strategy

Acceleration Requirements

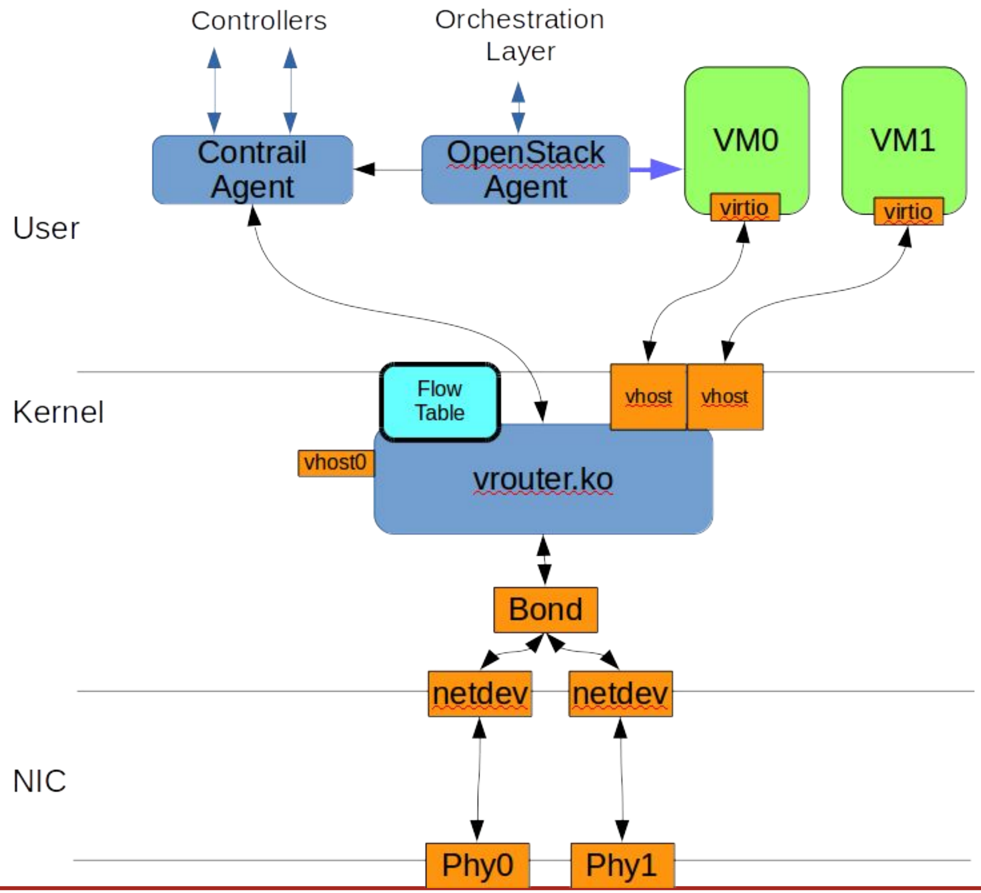
Two key goals: performance and transparency

- Main codebase should function regardless of acceleration
- End user ideally only sees a perf boost

Key Idea: NFP is an "ultra fastpath" parallel to vRouter

- BUT: NFP can't offload the full contrail logic
- Falls back to vRouter datapath if it can't handle a packet

Review: Compute Node

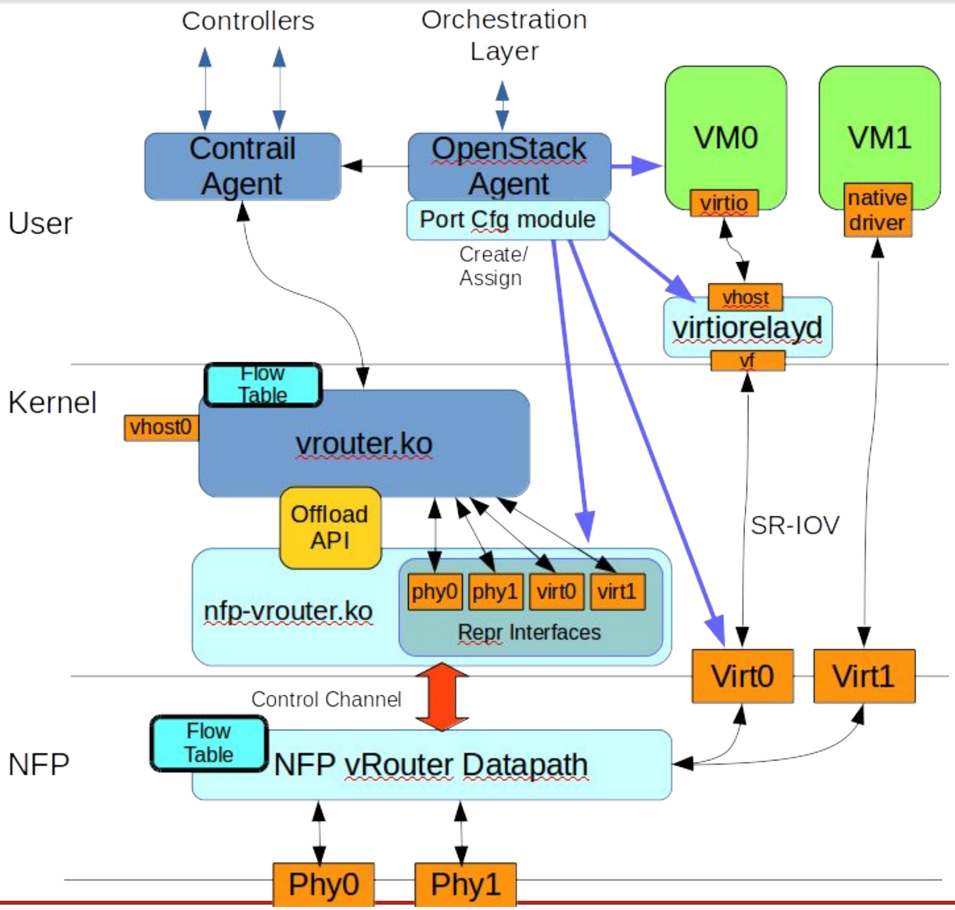


Integration Points

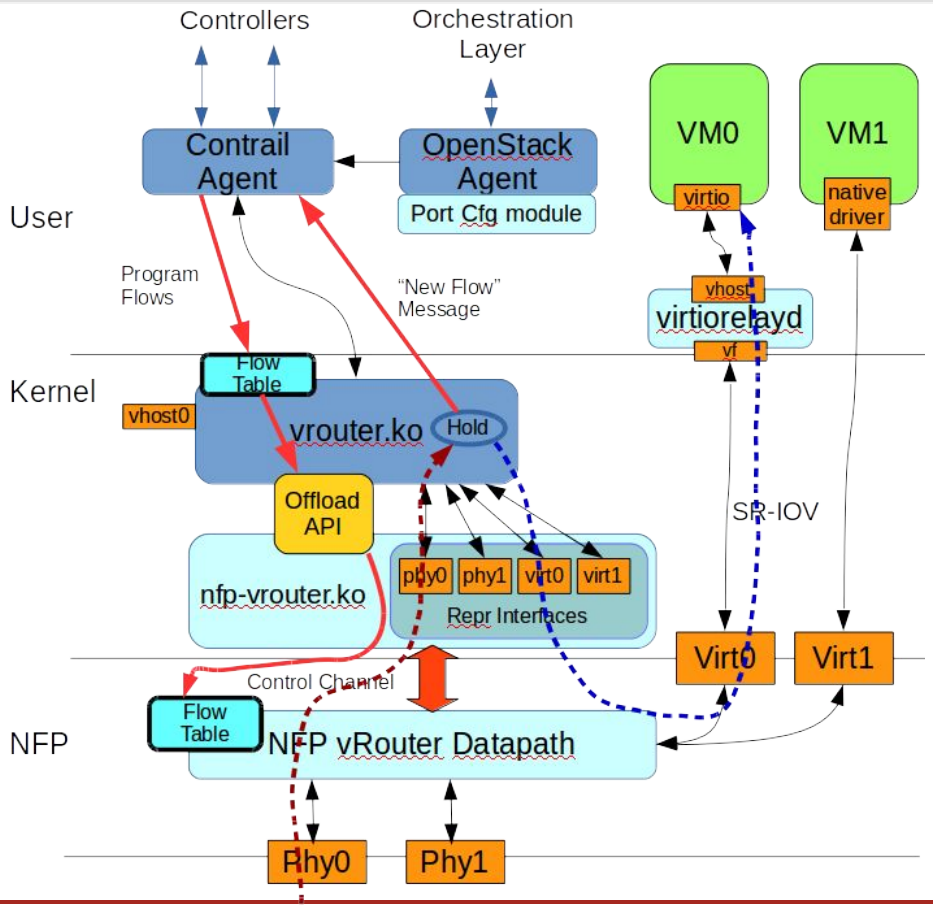
Our solution touches the vRouter codebase at 3 points:

1. Hardware offload API in the vRouter datapath
   - Intercepts and mirrors configuration changes to the NFP
   - Proxies stats requests to add in NFP counters
2. Representative Interfaces (RIs) in the Linux kernel
   - “Fake” netdevs that map to physical and virtual interfaces
   - Xmit on an RI sends the packet out the real interface via NFP
   - Arriving packet on an RI looks like it came from the corresponding interface
3. Port control changes for interface allocation/teardown
   - This was a script run by OpenStack but provided by JNPR

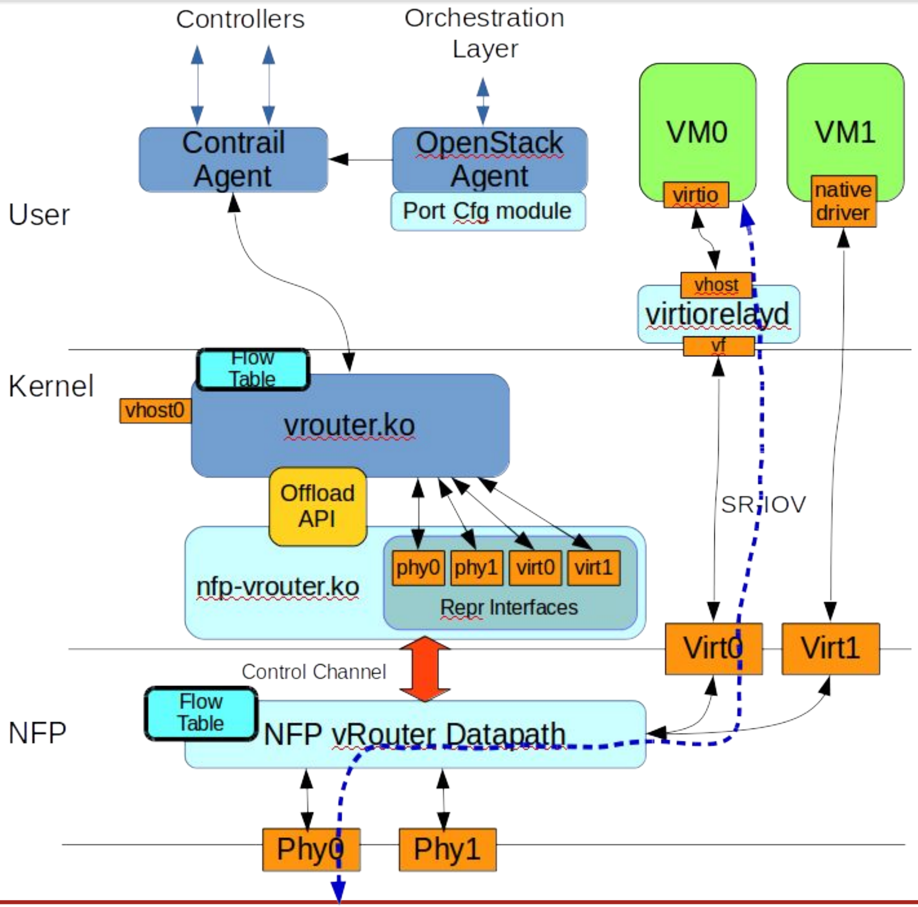
NFP Acceleration Approach



Day in the Life of a Packet



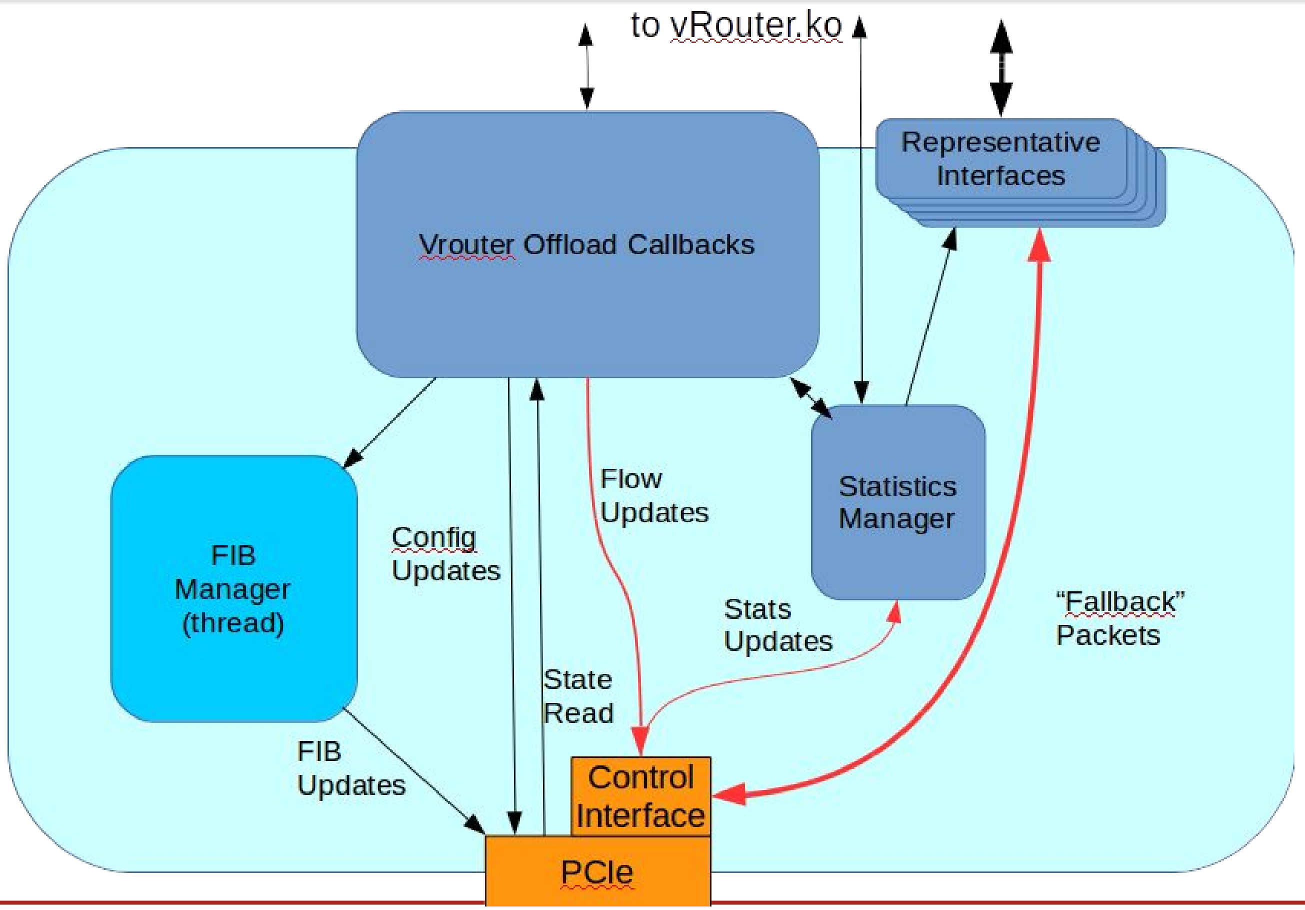
Day in the Life of a Packet (cont)



OpenContrail, Real Speed: Offloading vRouter

Software Architecture

Offload Module



Kernel Module Design

nfp-vrouter.ko module registers with vRouter offload API

Separate kernel thread to perform trie building ops

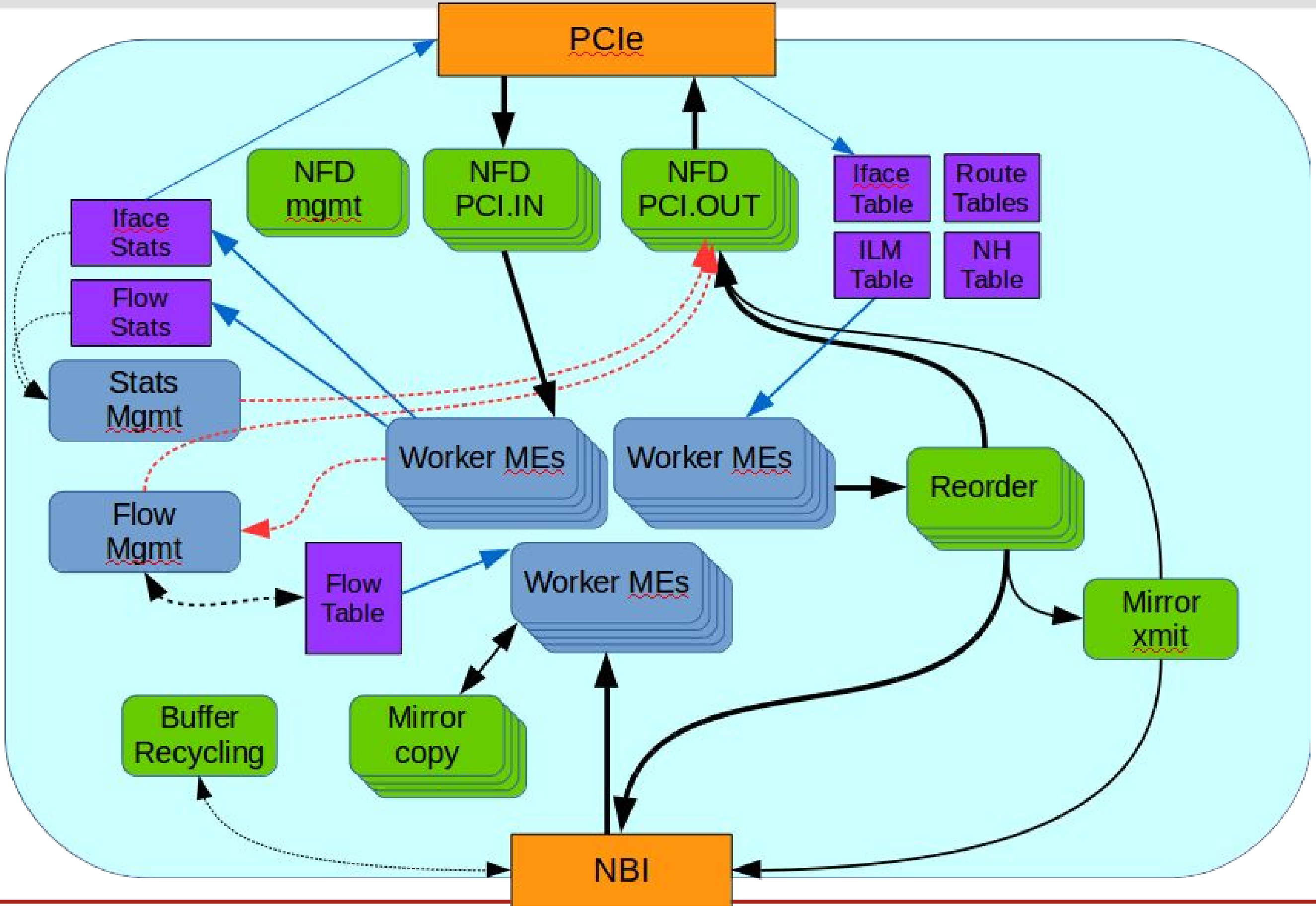
- Expensive operation: can’t hold up packets
- Fortunately also infrequent
- Accumulate and double-buffer routing tables

Stats manager updates vRouter and kernel statistics

Control channel carries both control messages and packets

- Both encapsulated in a common format
- RI packets muxed/demuxed over the control channel

NFP Firmware



Firmware Design

Main ME islands have a pool of run-to-completion workers

Workers access tables primarily read-only

- except for per-flow statistics

vRouter's update rate is less than 100,000 ops/sec

- Can do this on 1 or 2 NFP cores
- Preserves code store in the workers

Control messages demuxed to management subsystems

Several MEs to periodically poll, and push statistics

Separate mirroring MEs for copying, fragmentation and xmit

OpenContrail, Real Speed: Offloading vRouter

Results and Lessons Learned

Performance Results

Demonstrated moving over **20 Mpps** w/ imix distribution no-drop through VNFs utilizing 1 (mostly idle) core*

Sustained >27 Mpps in DPDK pktgen tests

Upcoming webinar will discuss methodology

*SR-IOV delivery to a service chain configured VNF

Conclusions

Programmable hardware enables transparent offload

- The impact to the dataplane is mirroring state updates to the NFP
- One can easily abstract this to be hardware-neutral

Acceleration is feasible even without offloading 100% of the dataplane

- Sophisticated fast paths follow the 90/10 rule as with most software: 90% of the work is done in 10% of the code
- With transparent offload you can pick and choose what gets offloaded and what falls back to the main datapath

Future Work

We plan to continue on this work as OpenContrail evolves as well

- Support full 4.0 Contrail feature set and OpenStack Newton
- Add QoS offload in the NFP
- Support for multi-card servers
- Accelerate container based SDN deployments
- Accelerate VMWare deployments
- Add support for custom VNFs offloaded within the dataplane
- Offload flow classification within the NFP

OpenContrail, Real Speed: Offloading vRouter

Questions