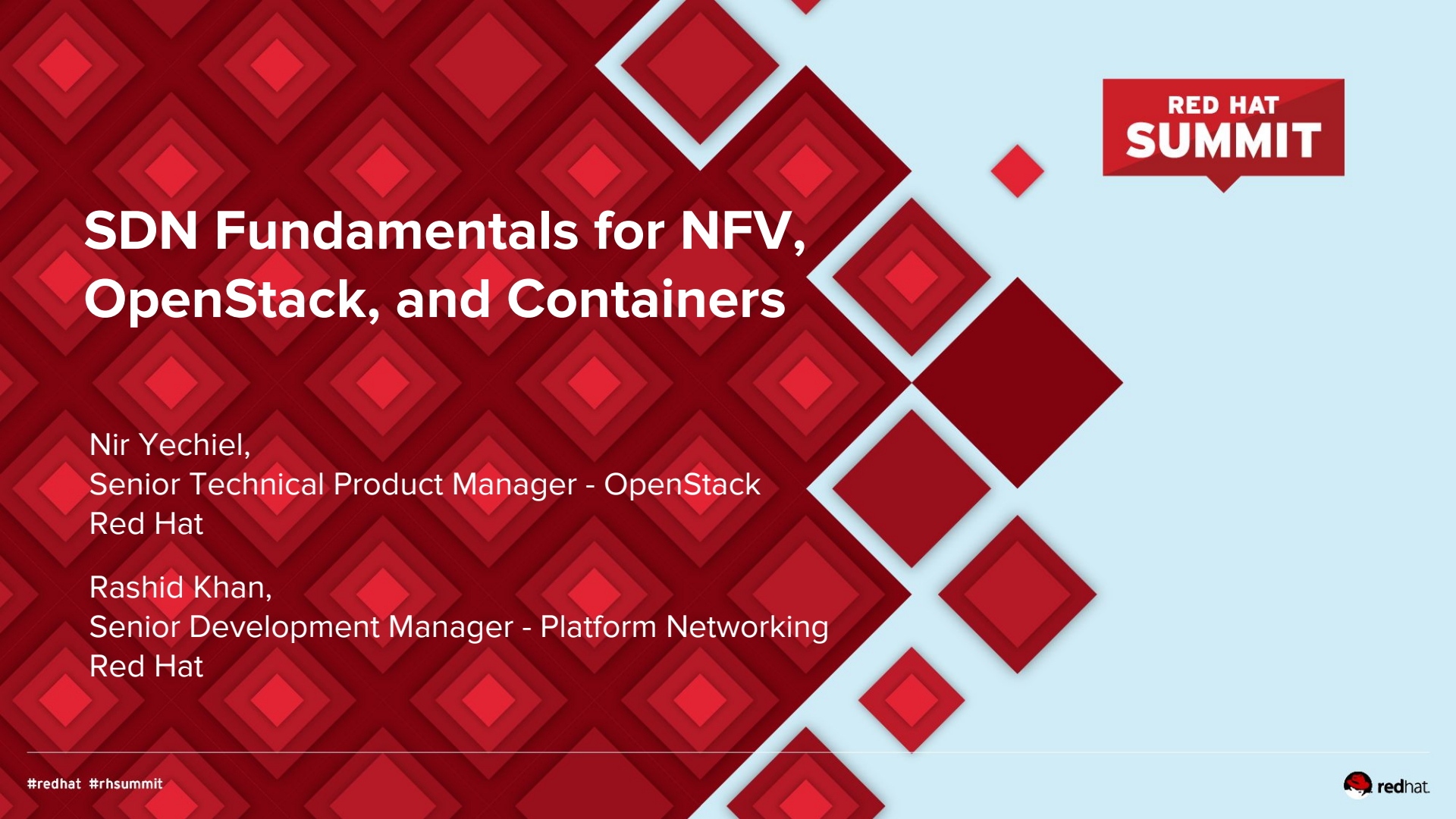


SDN Fundamentals for NFV, OpenStack, and Containers

Nir Yechiel,
Senior Technical Product Manager - OpenStack
Red Hat

Rashid Khan,
Senior Development Manager - Platform Networking
Red Hat

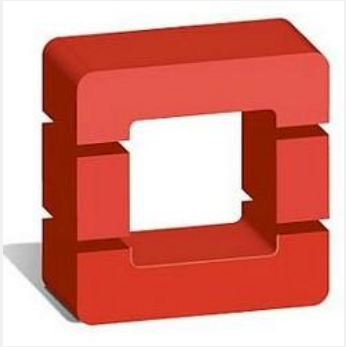
Agenda

- Setting context: OpenStack, SDN, and NFV
  - OpenStack overview
  - NFV Infrastructure
  - Red Hat approach to SDN/NFV
- Key RHEL networking features and tools…
  - and how they apply to OpenStack Networking (Neutron)
- Accelerating the datapath for modern applications
- Q&A

SETTING CONTEXT

OPENSTACK, SDN, AND NFV

What is OpenStack?



- Fully open-source cloud “operating system”
- Comprised of several open source sub-projects
- Provides building blocks to create an IaaS cloud
- Governed by the vendor agnostic OpenStack Foundation
- Enormous market momentum

Red Hat OpenStack Platform

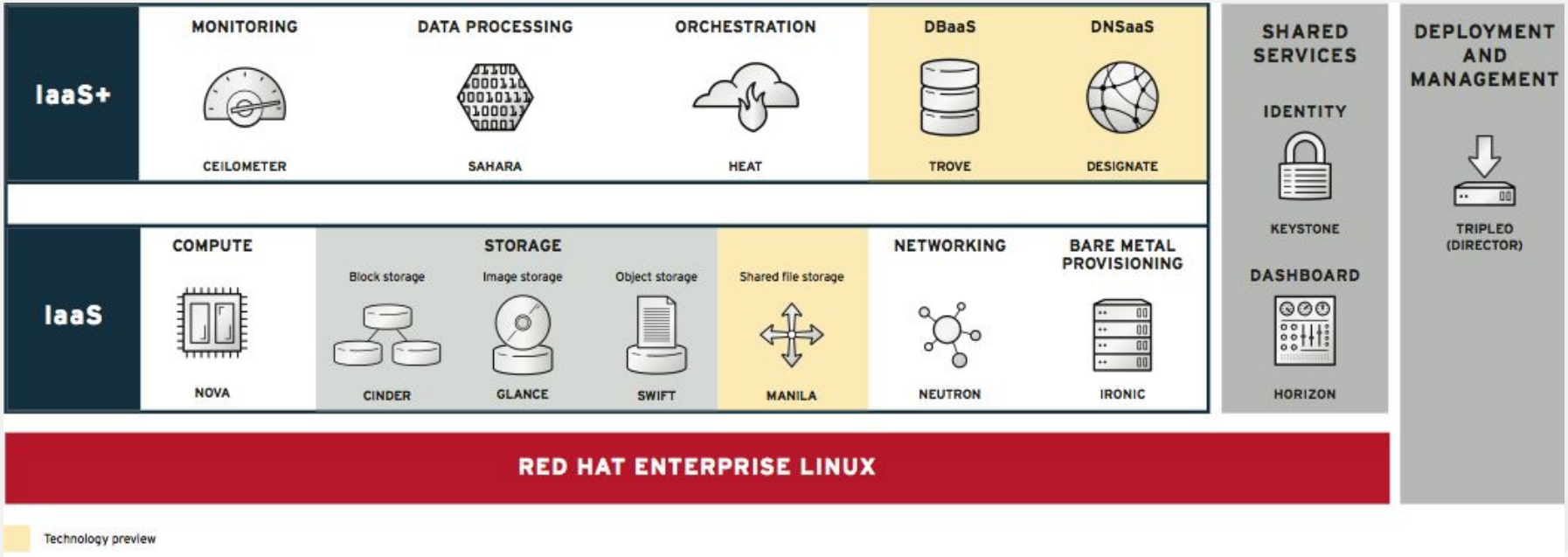
February 2015
Red Hat OpenStack Platform 6 (Juno)
RHEL 7

August 2015
Red Hat OpenStack Platform 7 (Kilo)
RHEL 7

April 2016
Red Hat OpenStack Platform 8 (Liberty)
RHEL 7

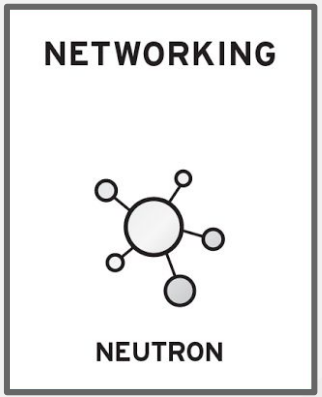
Source: https://access.redhat.com/support/policy/updates/openstack/platform

Red Hat OpenStack Platform 8

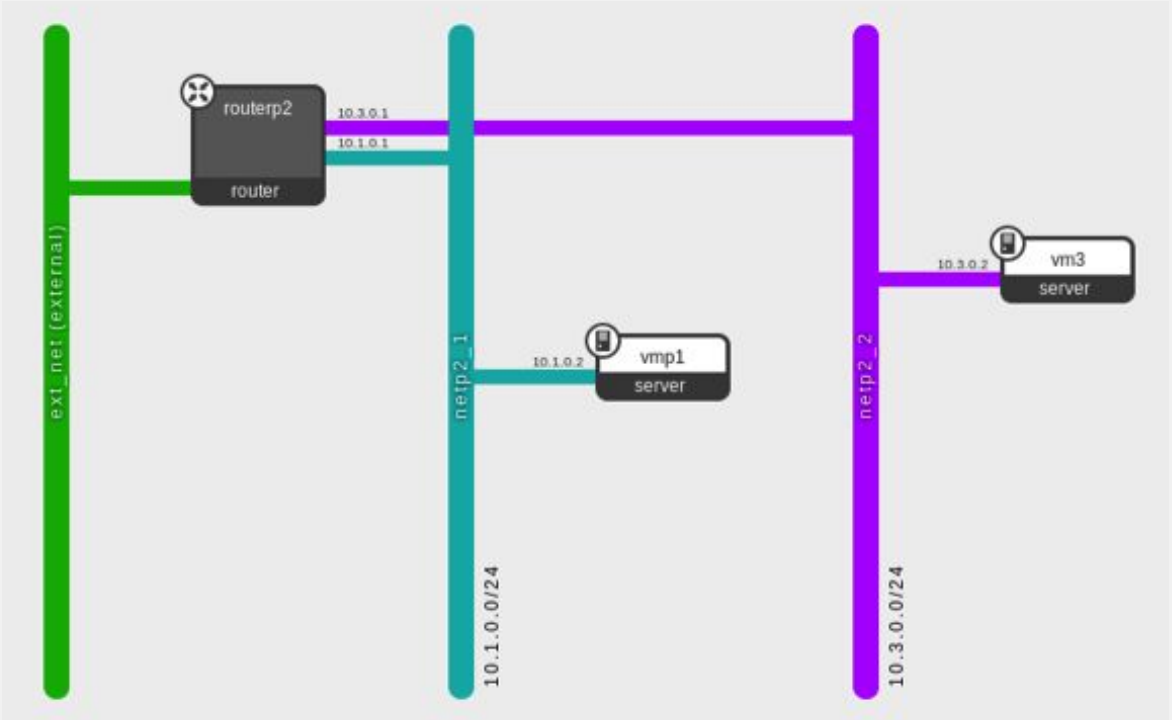


What is Neutron?

- Fully supported and integrated OpenStack project
- Exposes an API for defining rich network configuration
- Offers multi-tenancy with self-service

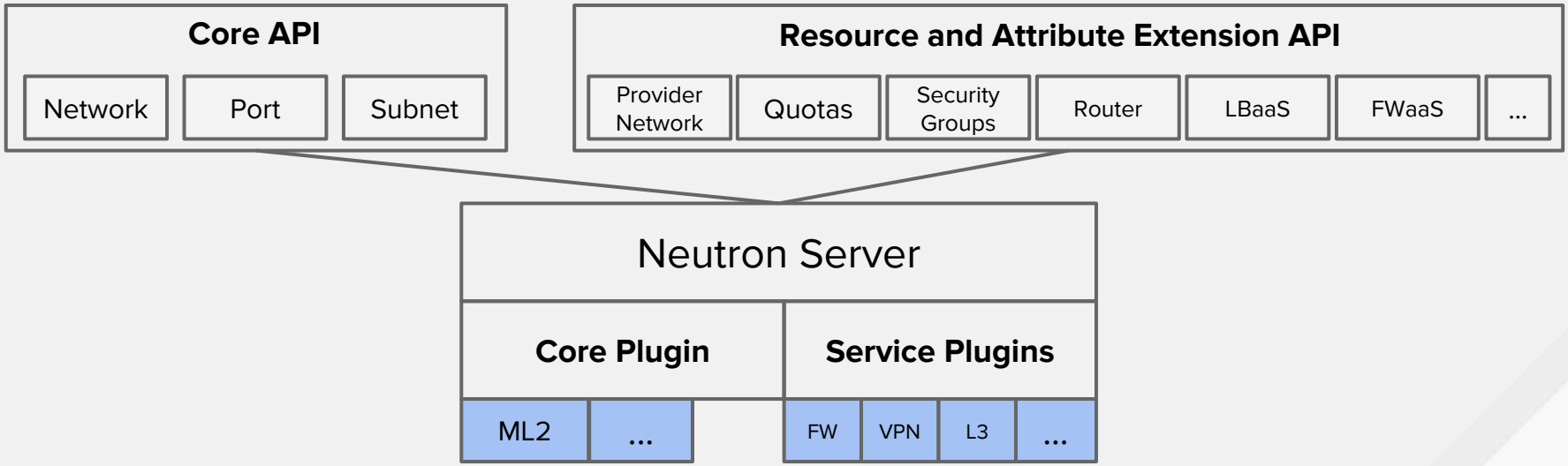


Neutron - Tenant View



What Neutron is not?

- Neutron does not implement the networks
  - Uses the concept of plugins



Neutron Key Features

- L2 connectivity
- IP Address Management
- Security Groups
- L3 routing
- External gateway, NAT and floating IPs
- Load balancing, VPN and firewall

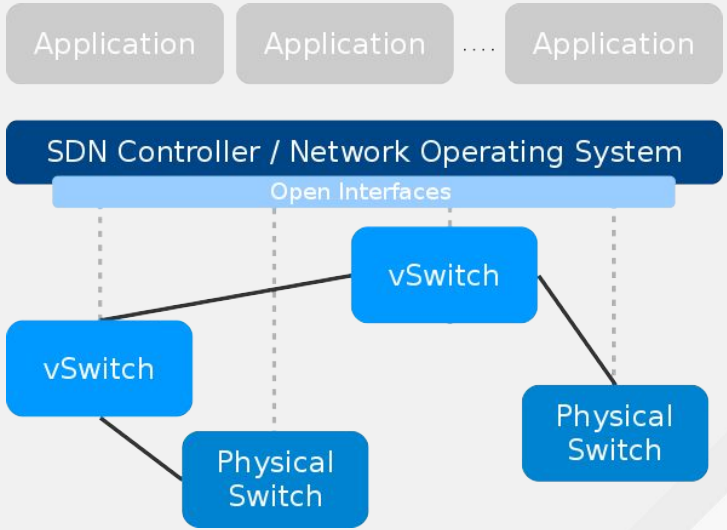
Red Hat Neutron Focus

- ML2 with Open vSwitch Mechanism Driver (today)*
  - Overlay networks with VXLAN
- ML2 with OpenDaylight Mechanism Driver (roadmap)
- Broad range of commercial partners
  - Through our certification program

*Focus of this presentation

Software Defined Networking

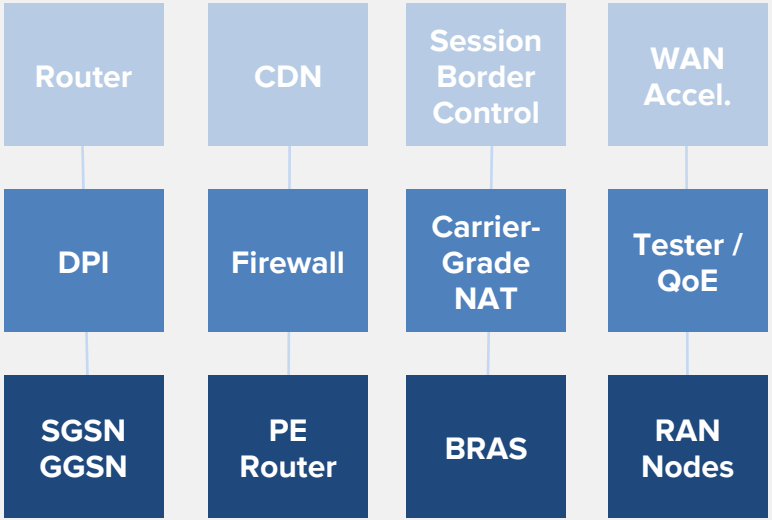
- Different things to different people
  - Separation of control plane and forwarding plane
  - Open standard protocols
  - Well-known, stable API
  - Application awareness
  - Programmability
  - Agility

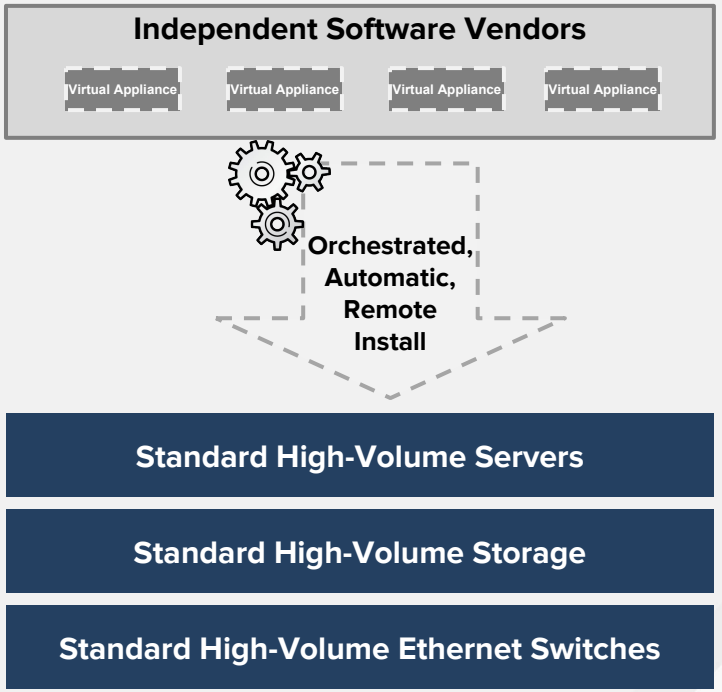


Software Defined Networking

- OpenFlow != SDN
- We see the opportunity to push the virtual network edge to where it belongs – the hypervisor

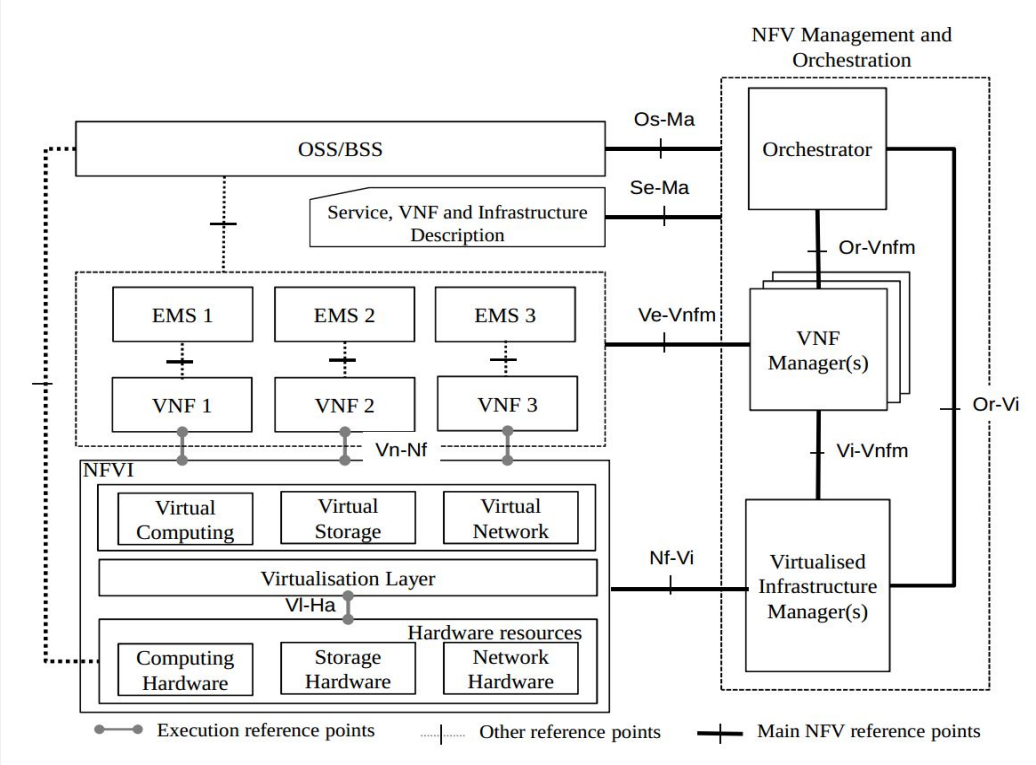
Network Functions Virtualization





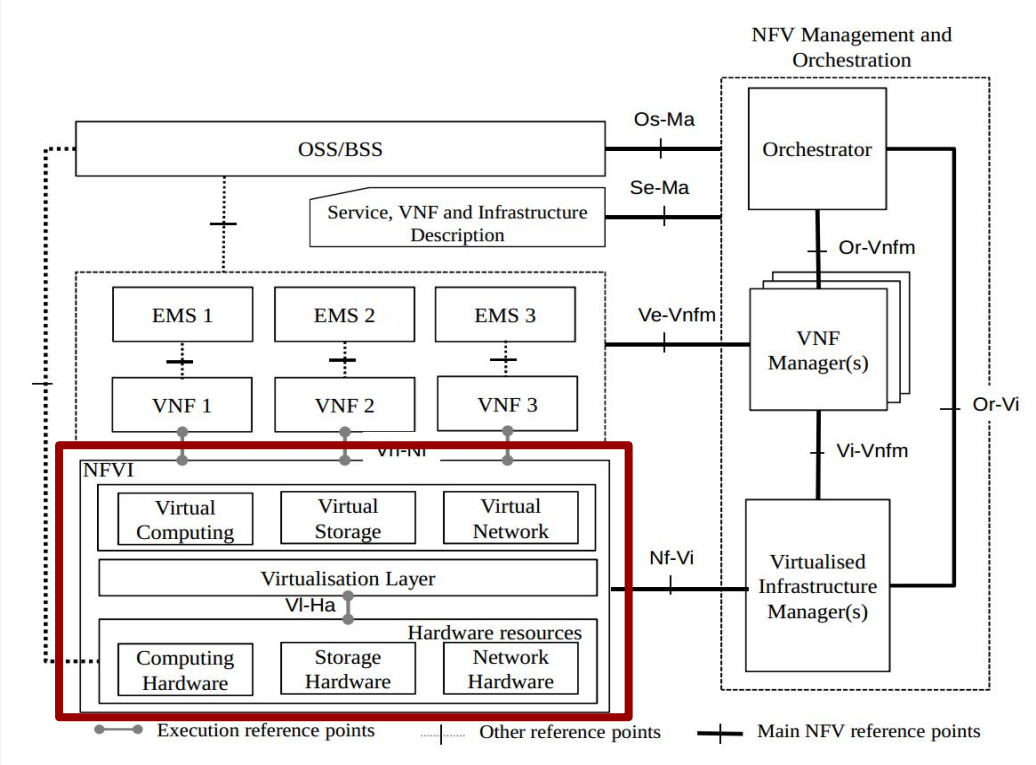
- Fragmented non-commodity hardware
- Vertical design
- Physical install (per appliance, per site)

Network Functions Virtualization



Source: NFV ISG

Network Functions Virtualization

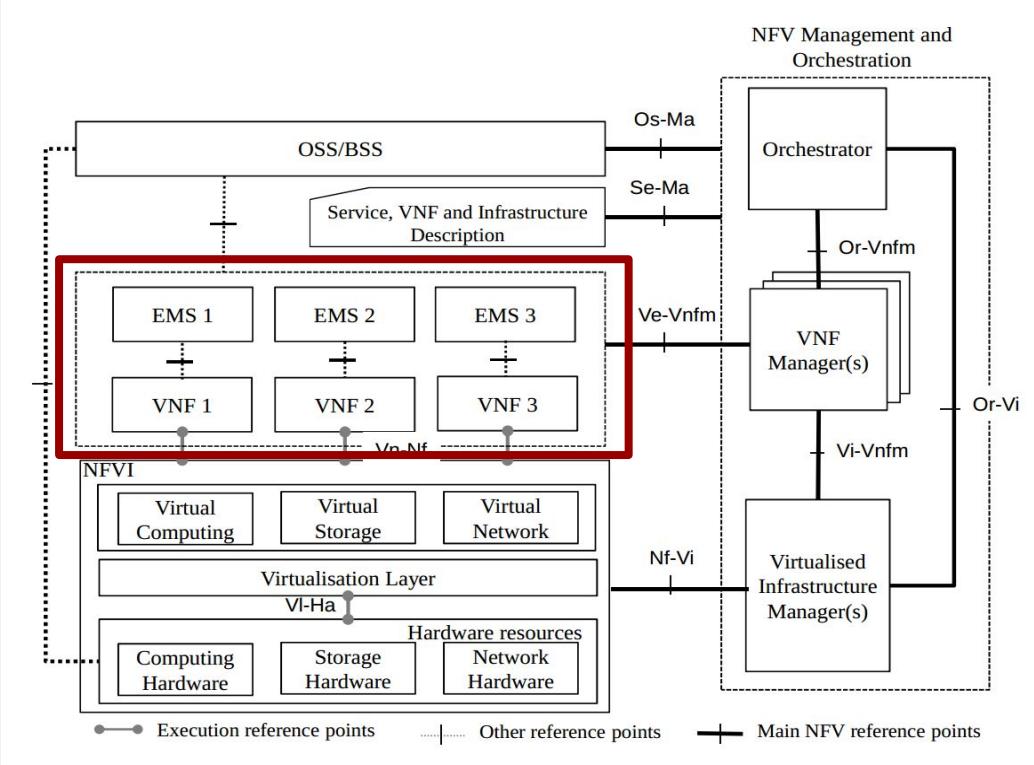


NFVI

Source: NFV ISG

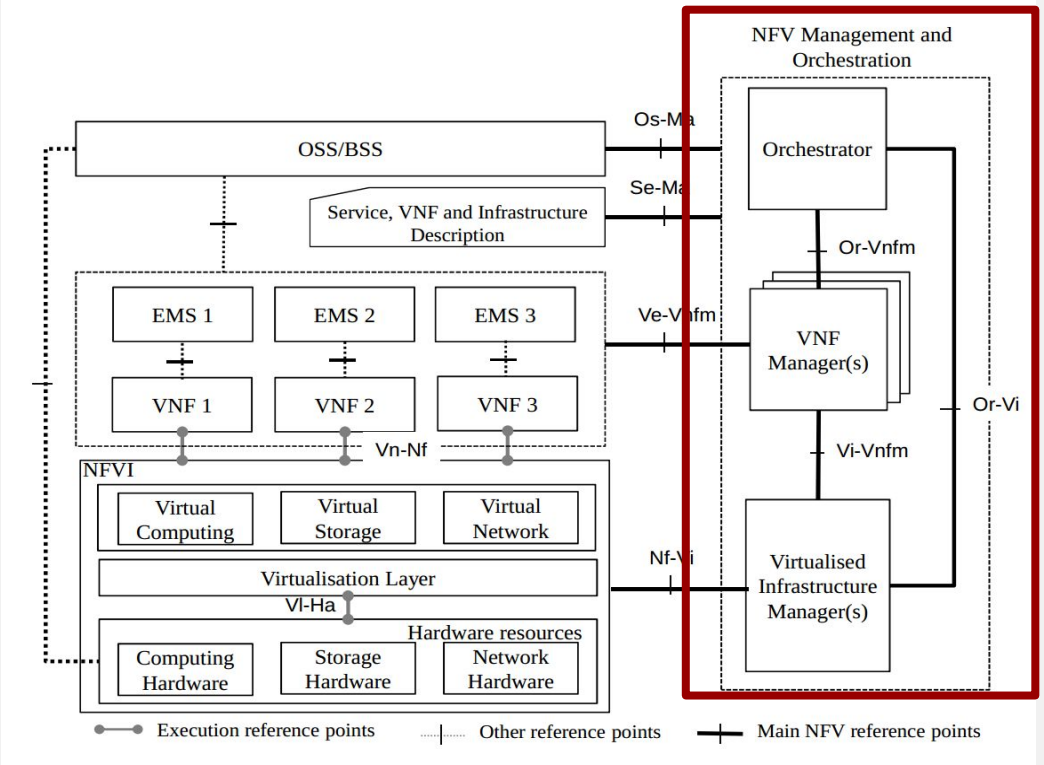
Network Functions Virtualization

VNF



Source: NFV ISG

Network Functions Virtualization



MANO

Source: NFV ISG

NFV Loves OpenStack

- OpenStack is the de-facto choice for VIM on early NFV deployments
- OpenStack's pluggable architecture - key for NFV
  - e.g storage, networking
- Open standards and multi-vendor
- Many network equipment providers (NEPs) and large telco operators are already involved in the community

NFV Infrastructure - Just OpenStack?

OpenStack

libvirt

DPDK

Open vSwitch

QEMU/KVM

Linux

Red Hat NFV Focus

- Our focus is on the infrastructure:
  - NFVI
  - VIM
  - Enablement for VNFs
- Partner with ISVs for MANO
- Partner with hardware providers
- No separate product for NFV
  - We don’t make an OpenStack version for NFV, but rather make NFV features available in Red Hat OpenStack Platform and across the entire stack

An SDN/NFV Stack

It's all about:

➔ Develop the right capabilities on the platform/Operating System level

➔ Leverage and expose the capabilities across the entire stack

Single Root I/O Virtualization (SR-IOV)	Network namespaces
Open vSwitch	Data Plane Development Kit (DPDK)
Overlay networking technologies	Multiqueue support for Virtio-net
iptables	conntrack
...	

KEY RHEL NETWORKING FEATURES

AND HOW THEY APPLY TO NEUTRON

SR-IOV

Single Root I/O Virtualization (SR-IOV)

- Allows a device, such as a network adapter, to separate access to its resources among various PCIe hardware functions: physical function (PF) and one or more virtual functions (VFs)
- Enables network traffic to bypass the software layer of the hypervisor and flow directly between the VF and the virtual machine
- Near line-rate performance without the need to dedicate a separate NIC to each individual virtual machine
- Supported with RHEL 7, available starting with Red Hat OpenStack Platform 6

SR-IOV Networking in OpenStack

- Implemented with a generic ML2 driver (sriovnicswitch)
  - NIC vendor defined by the operator through `vendor_id:product_id`
  - New Neutron port type (vnic-type ‘direct’)
  - VLAN and flat networks are supported
- Optional agent for advanced capabilities (requires NIC support)
  - Reflect port status (Red Hat OpenStack Platform 6)
  - QoS rate-limiting (Red Hat OpenStack Platform 8)
- Supported network adapters are inherited from RHEL
  - See https://access.redhat.com/articles/1390483
- Coming soon with Red Hat OpenStack Platform: PF Passthrough

Open vSwitch

Open vSwitch (OVS)

- Multi-layer software switch designed to forward traffic between virtual machines and physical or logical networks
- Supports traffic isolation using overlay technologies (GRE, VXLAN) and 802.1Q VLANs
- Highlights:
  - Multi-threaded user space switching daemon for increased scalability
  - Support for wildcard flows in kernel datapath
  - Kernel based hardware offload for GRE and VXLAN
  - OpenFlow and OVSDB management protocols

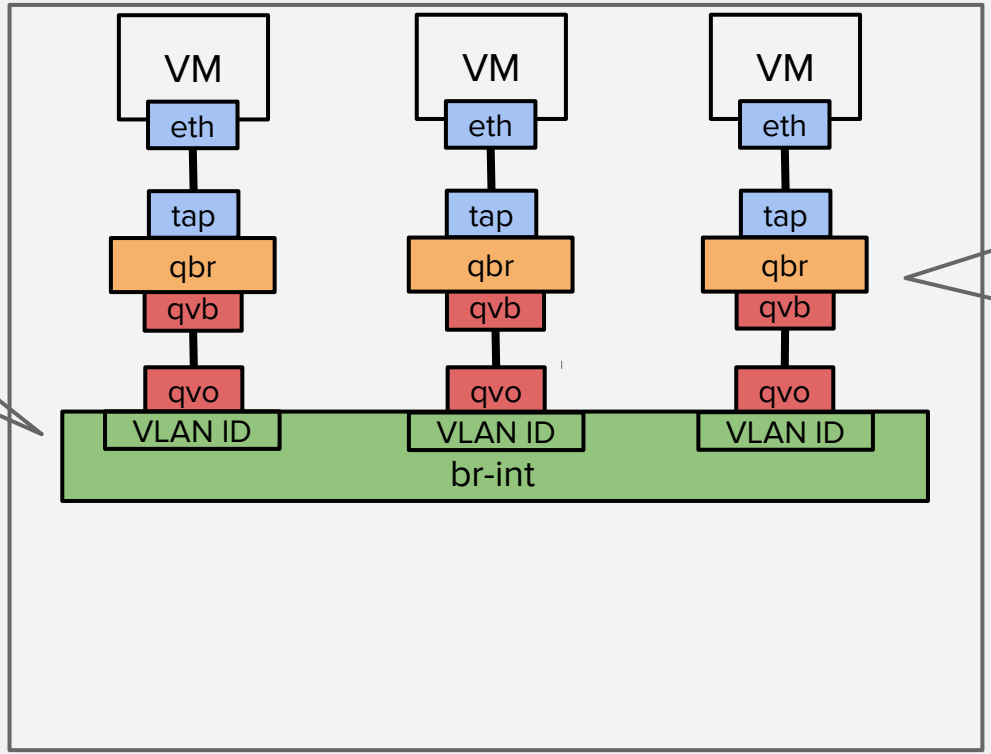
Open vSwitch (OVS)

- Kernel module ships with RHEL, but the vSwitch is supported on Red Hat OpenStack Platform and OpenShift by Red Hat
- Integrated with OpenStack via a Neutron ML2 driver and associated agent
- New with Red Hat OpenStack Platform 8: technology preview offering of DPDK accelerated Open vSwitch (more later)

OVS - L2 Connectivity in OpenStack

- Between VMs on the same Compute
  - Traffic is bridged locally using normal MAC learning
  - Each tenant gets a local VLAN ID
  - No need to leave ‘br-int’
- Between VMs on different Computes
  - OVS acts as the VTEP
  - Flow rules are installed on ‘br-tun’ to encapsulate the traffic with the correct VXLAN VNI

OVS - Compute View



Tenant flows are separated by internal, locally significant, VLAN IDs. VMs that are connected to the same tenant network get the same VLAN tag

OVS can't directly attach a TAP device where iptables rules are applied; bridge device is necessary - offers a route to the kernel for filtering

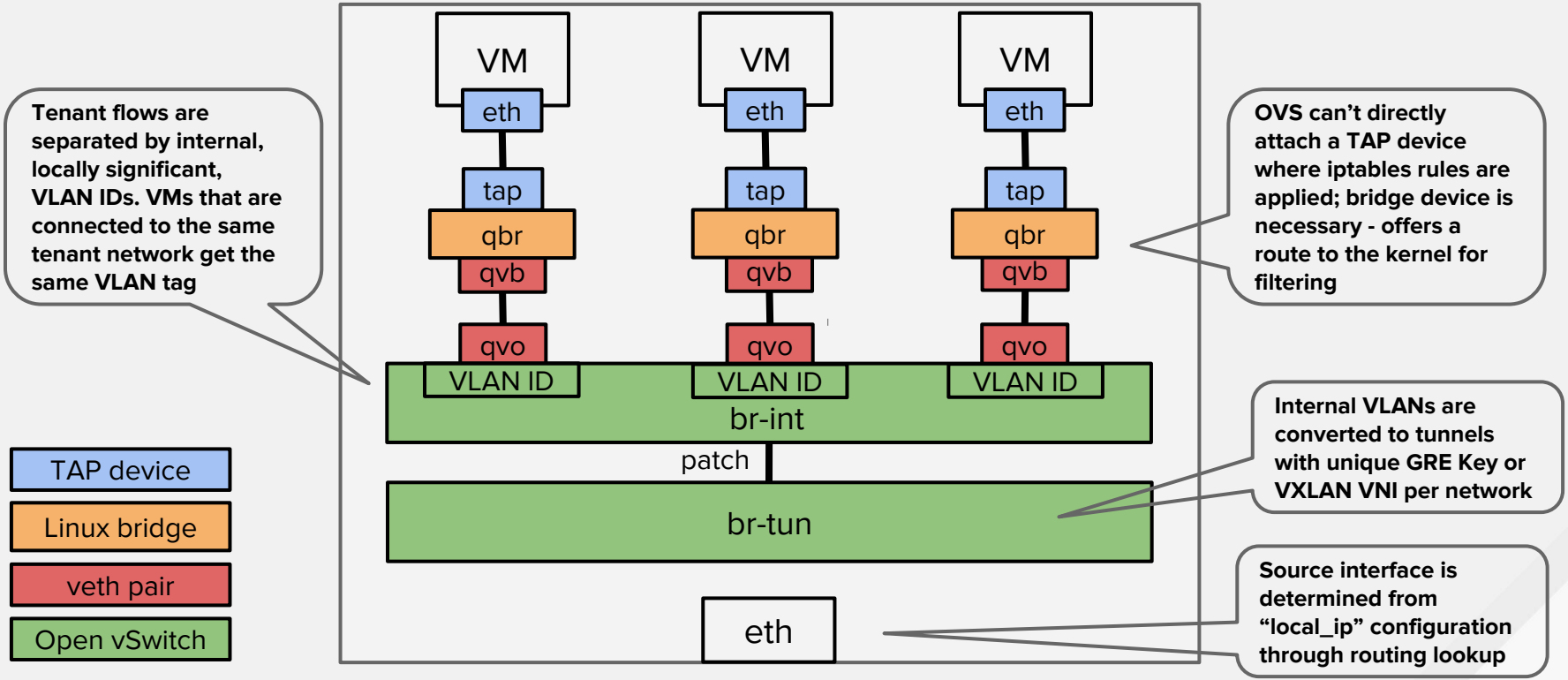
TAP device

Linux bridge

veth pair

Open vSwitch

OVS - Compute View



Overlay Networking

Overlay Networking Technologies

- Virtual Extensible LAN (VXLAN)
  - Common encapsulation protocol for running an overlay network using existing IP infrastructure
  - RHEL supports TCP/IP VXLAN offload and VXLAN GRO
  - Recommended encapsulation for tenant traffic network in Red Hat OpenStack Platform
- Generic Routing Encapsulation (GRE)
  - Can be used as an alternative to VXLAN
  - Hardware checksum offload support using GSO/GRO
- Network adapter support is inherited from RHEL
  - See https://access.redhat.com/articles/1390483

Network Namespaces

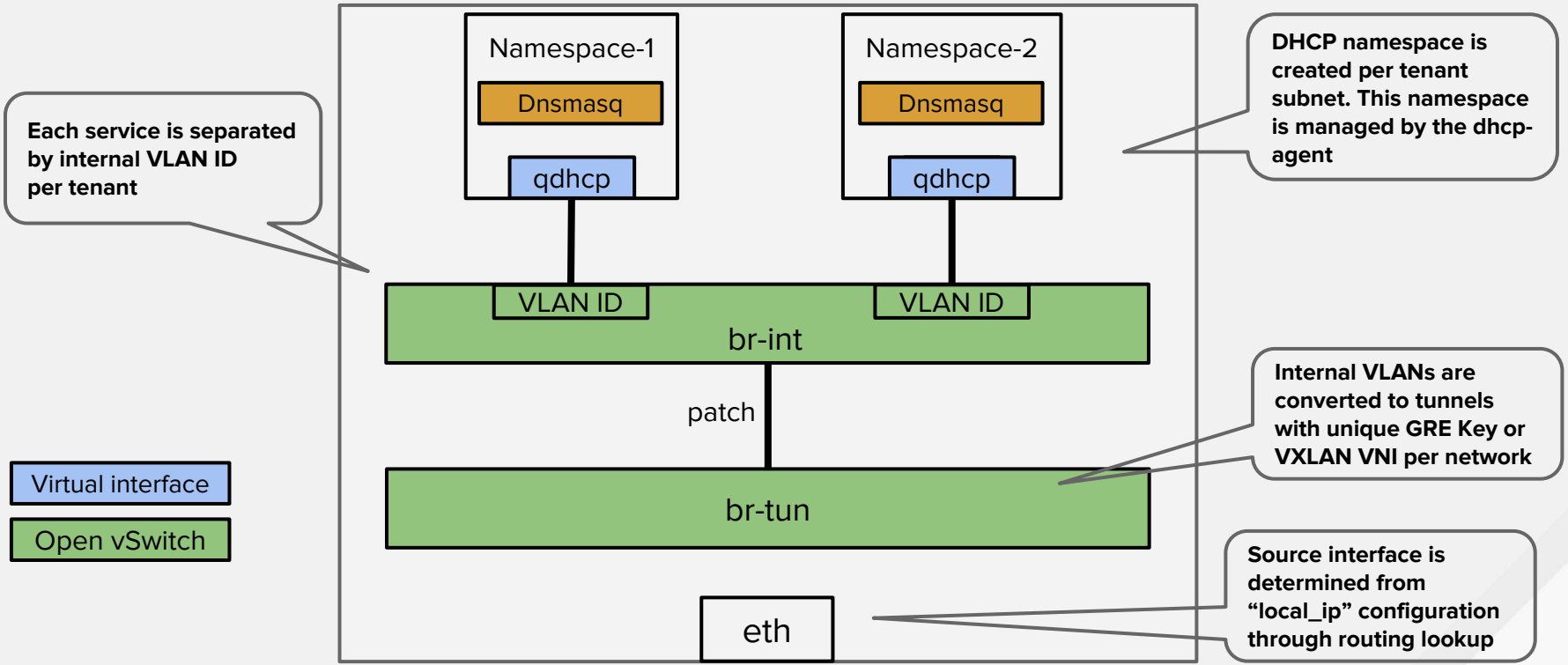
Network Namespaces

- Lightweight container-based virtualization allows virtual network stacks to be associated with a process group
- Create an isolated copy of the networking data structure such as the interface list, sockets, routing table, port numbers, and so on
- Managed through the iproute2 (ip netns) interface

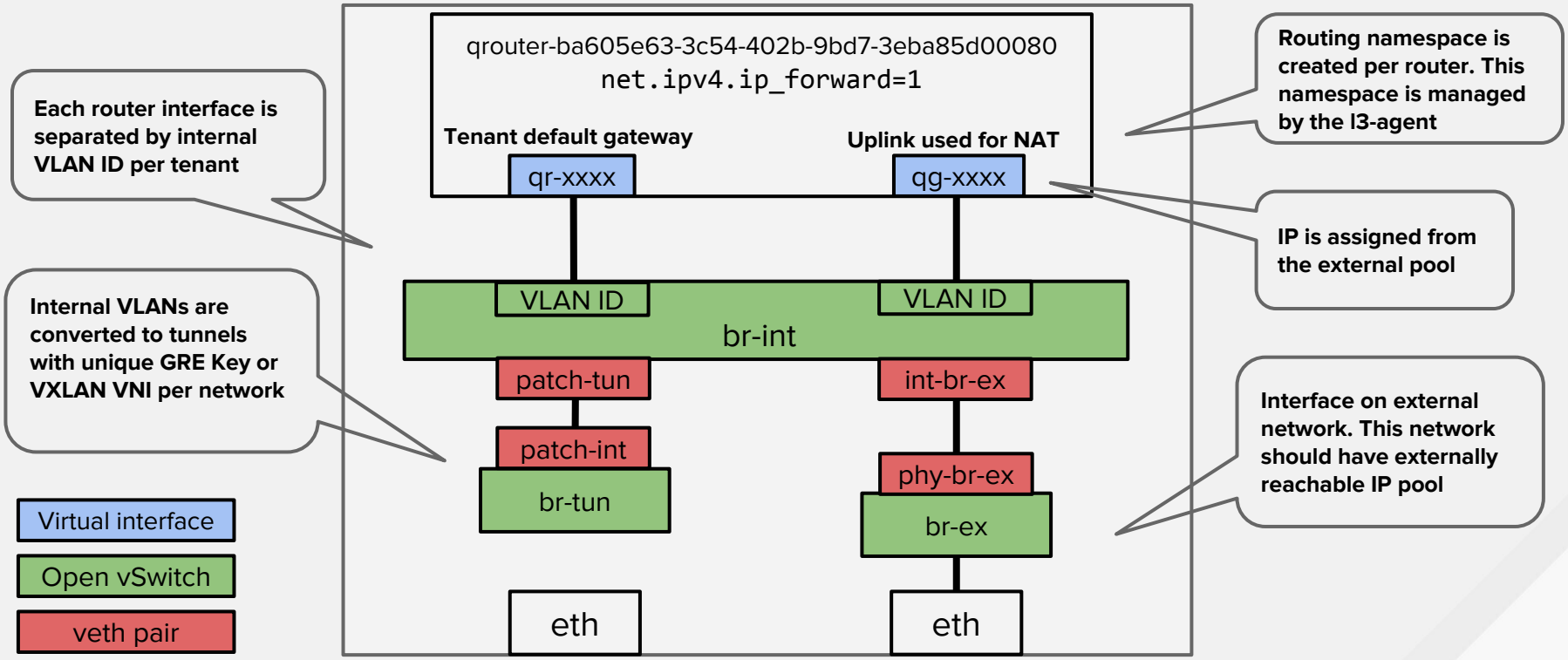
Network Namespaces in OpenStack

- Fundamental technology which enable isolated network space for different projects (aka tenants)
  - Allows overlapping IP ranges
  - Allows per tenant network services
- Used extensively by the l3-agent and dhcp-agent

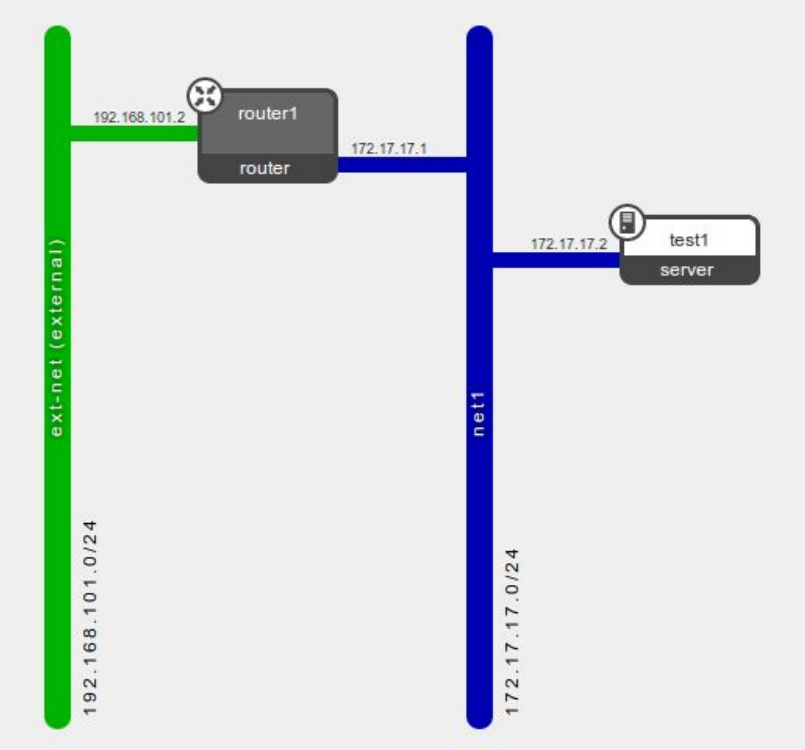
DHCP Namespace in OpenStack

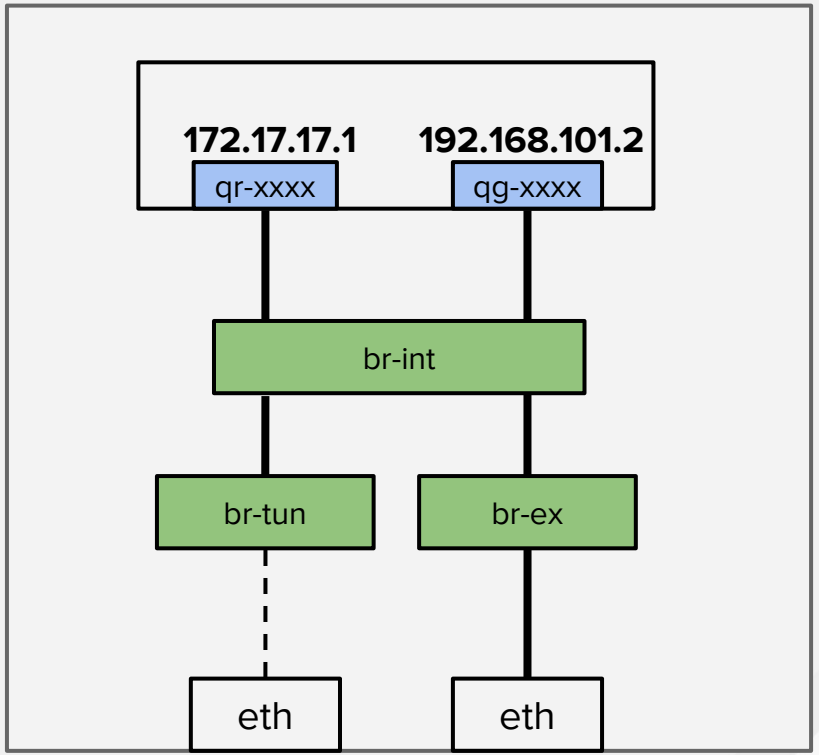


Routing Namespace in OpenStack

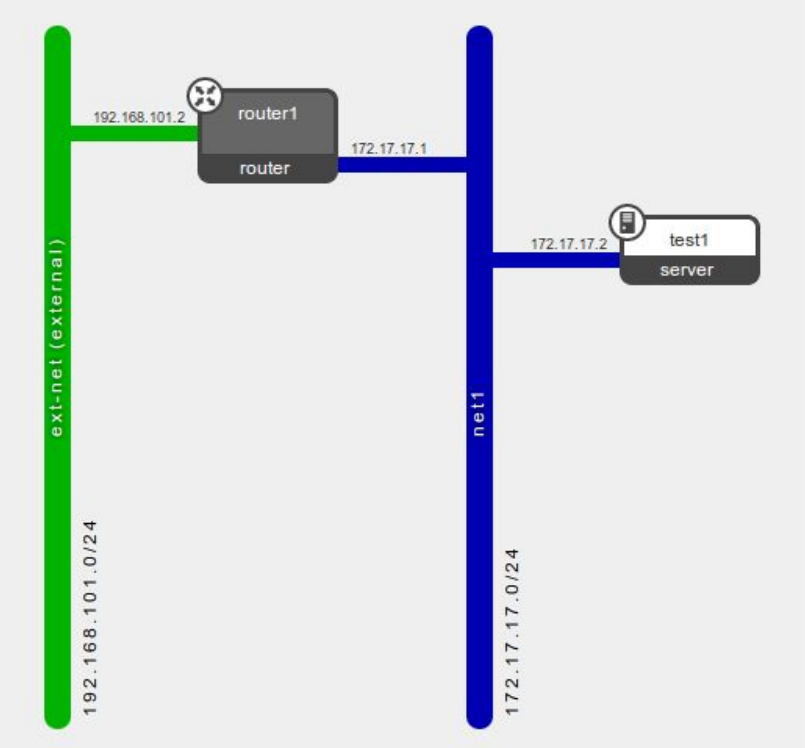


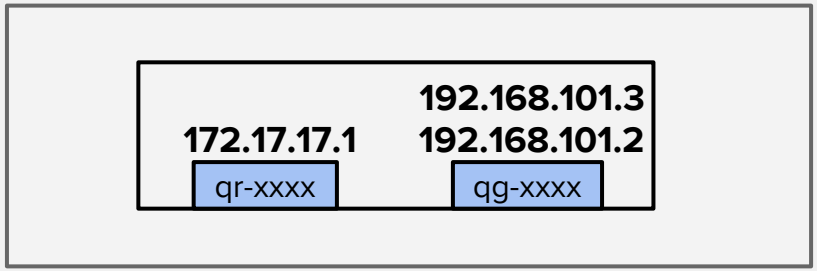
Routing - Example





Routing - Example





Default SNAT -
-A quantum-l3-agent-snat -s 172.17.17.0/24 -j SNAT --to-source 192.168.101.2

Floating IP (1:1 NAT) -
-A quantum-l3-agent-float-snat -s 172.17.17.2/32 -j SNAT --to-source 192.168.101.3
-A quantum-l3-agent-PREROUTING -d 192.168.101.3/32 -j DNAT --to-destination 172.17.17.2

DPDK

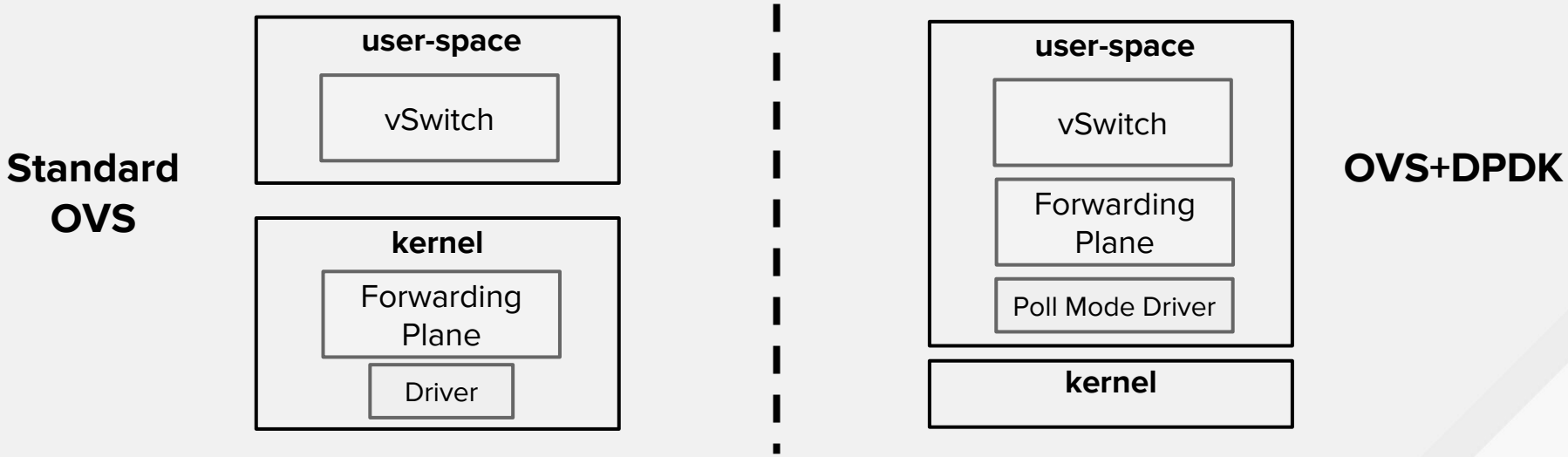
DPDK

- The Data Plane Development Kit (DPDK) consists of a set of libraries and user-space drivers for fast packet processing. It enables applications to perform their own packet processing directly from/to the NIC, delivering up to wire speed performance for certain cases

- DPDK can be utilized with Red Hat OpenStack in two main use-cases:
  - DPDK accelerated VNFs (guest VM)
    - A new package (`dpdk`) is available in the RHEL 7 “Extras” channel
  - Accelerated Open vSwitch (Compute nodes)
    - A new package (`openvswitch-dpdk`) is available with Red Hat OpenStack
    - DPDK is bundled with OVS for better performance

DPDK accelerated OVS (Tech Preview)

- DPDK accelerated Open vSwitch significantly improves the performance of OVS while maintaining its core functionality

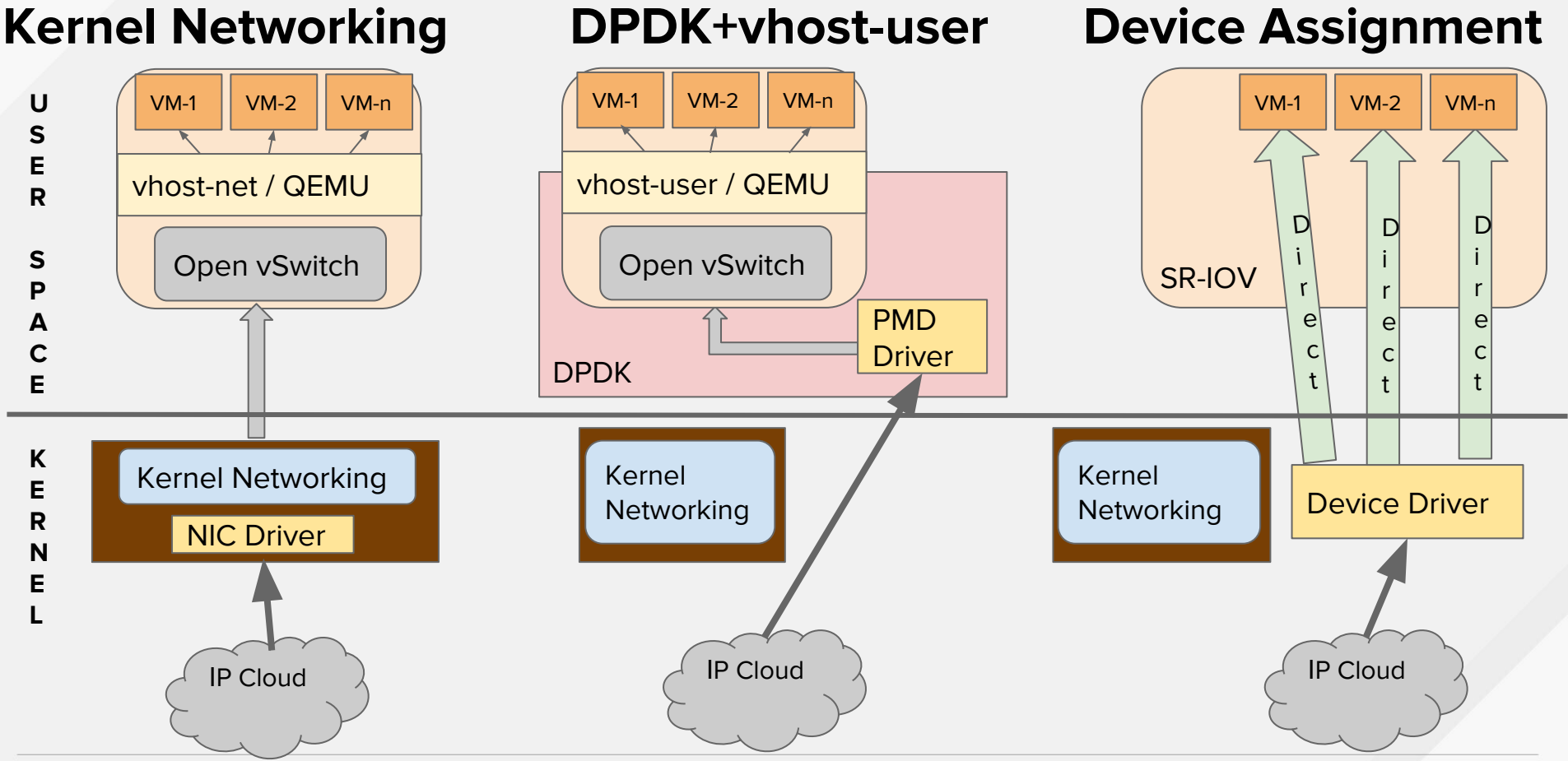


DPDK accelerated OVS (Tech Preview)

- Provided as a separately installable package from the standard OVS
  - It’s not possible to run OVS and OVS+DPDK on the same host
- Device support is limited to in-tree PMDs only, which don't rely on third party drivers and/or non-upstream modifications
  - Currently includes e1000, enic, i40e, and ixgbe
- The OVS agent/driver were enhanced to support DPDK datapath
  - `datapath_type=netdev`
  - Resulting in correct allocation of VIF types and binding details

ACCELERATING THE DATAPATH

FOR MODERN APPLICATIONS


Kernel Networking
DPDK+vhost-user
Device Assignment
USER SPACE
KERNEL
VM-1
VM-2
VM-n
vhost-net / QEMU
Open vSwitch
Kernel Networking
NIC Driver
IP Cloud
vhost-user / QEMU
Open vSwitch
PMD Driver
DPDK
Kernel Networking
IP Cloud
SR-IOV
Direct
Direct
Direct
Kernel Networking
Device Driver
IP Cloud

Comparison

OVS+kernel

Pros

- Feature rich / robust solution
- Ultra flexible
- Integration with SDN
- Supports Live Migration
- Supports overlay networking
- Full Isolation support / Namespace / multi-tenancy

Cons

- CPU consumption

OVS+DPDK

Pros

- Packets directly sent to user space
- Line rate performance with tiny packets
- Integration with SDN
- CPU consumption

Cons

- More dependence on user space packages

WIP

- Live Migration
- IOMMU support

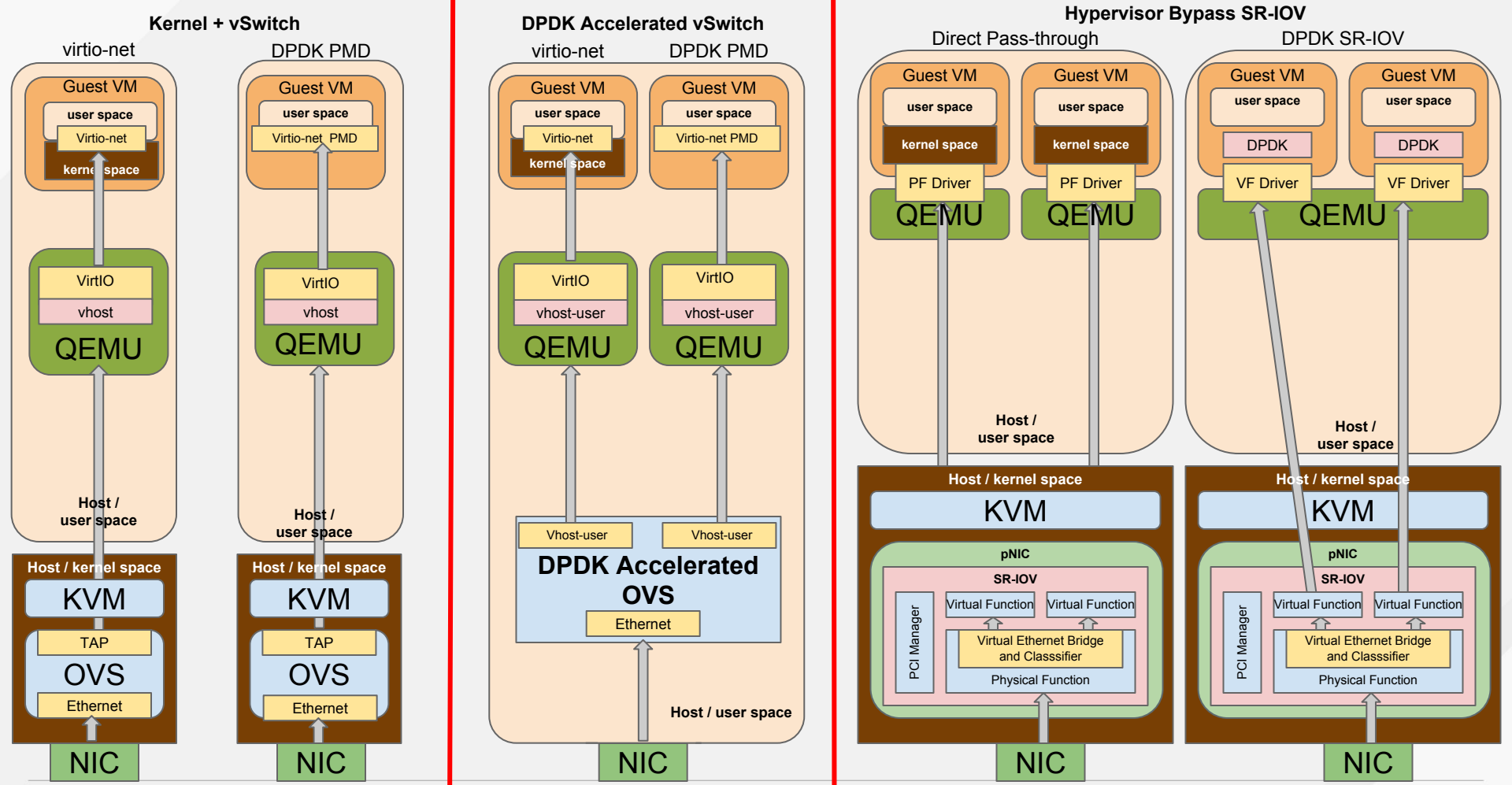
SR-IOV

Pros

- Line rate performance
- Packets directly sent to VMs
- HW based isolation
- No CPU burden

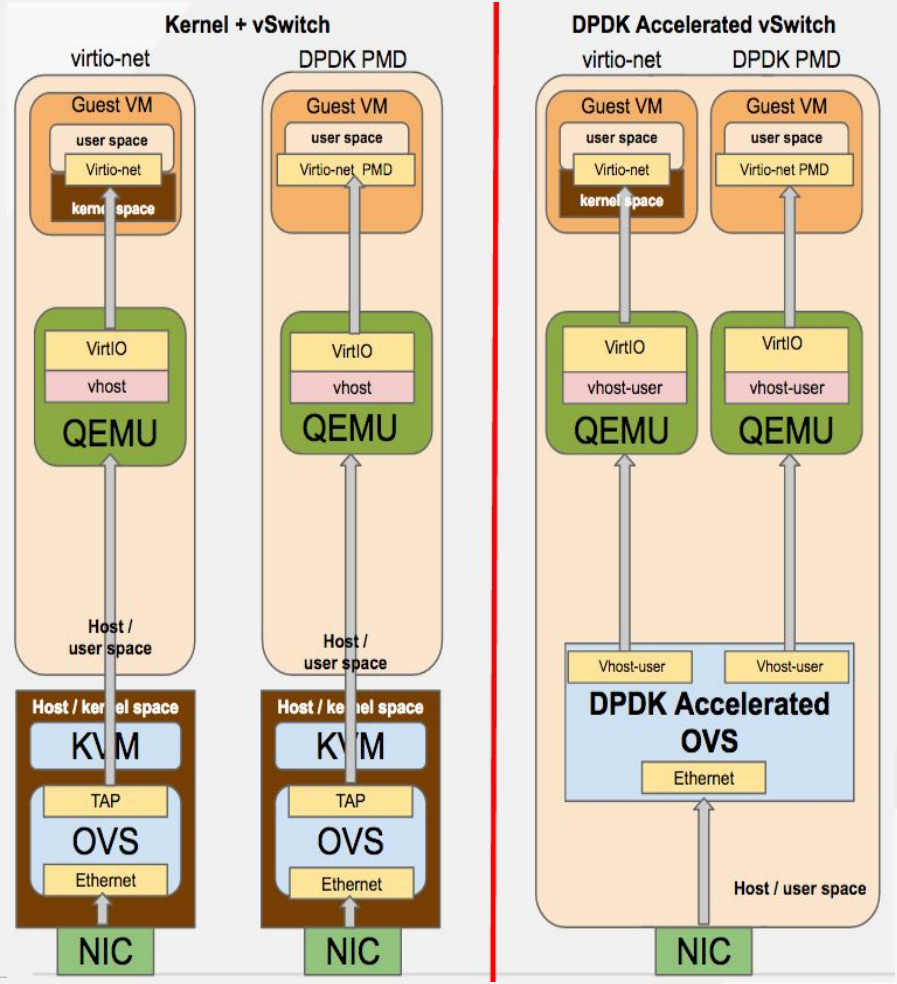
Cons

- 10s of VMs or fewer
- Not as flexible
- No OVS
- Need VF driver in the guest
- Switching at ToR
- No Live Migration
- No overlays


Kernel + vSwitch
virtio-net
DPDK PMD
Guest VM
user space
Virtio-net
kernel space
Virtio-net PMD
VirtIO
vhost
QEMU
Host / user space
Host / kernel space
KVM
TAP
OVS
Ethernet
NIC
DPDK Accelerated vSwitch
vhost-user
Vhost-user
DPDK Accelerated OVS
Hypervisor Bypass SR-IOV
Direct Pass-through
DPDK SR-IOV
PF Driver
DPDK
VF Driver
pNIC
SR-IOV
PCI Manager
Virtual Function
Virtual Ethernet Bridge and Classsifier
Physical Function

Red Hat's Value add

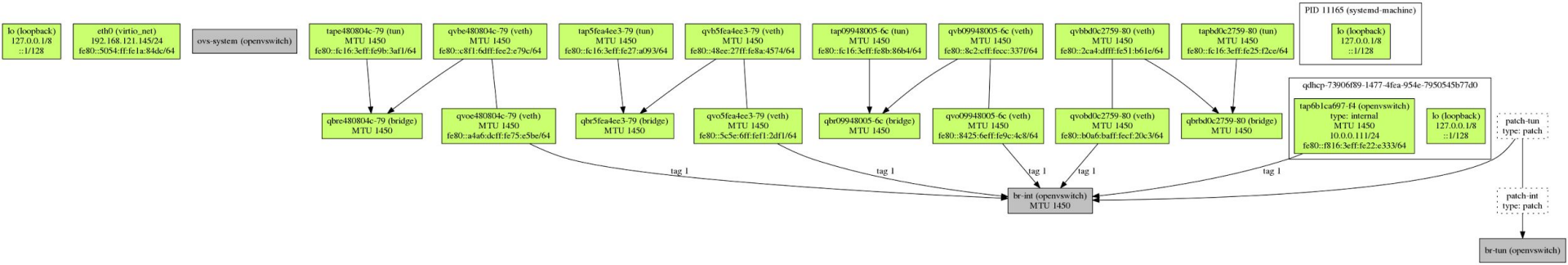
- State of the art open source technology
- Support / Debuggability
  - OVS + DPDK Tcpdump
  - Command line simplification
  - OpenFlow table names
  - OpenFlow dump readability
  - Cross layer debugging / tracing tools
  - Plotnetcfg
- Integration and performance tuning
  e.g multi-queue, connection tracking,
  VirtIO expected perf increase by 35%



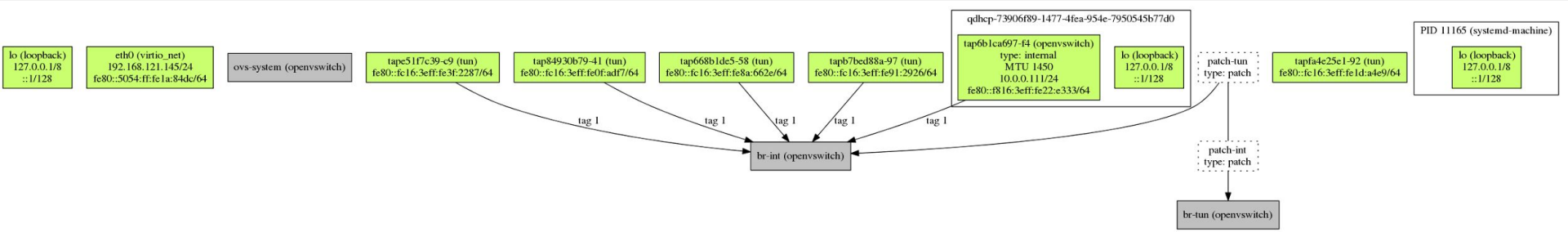
Security Groups and OVS/conntrack

https://github.com/jbenc/plotnetcfg
Performance info in backup section

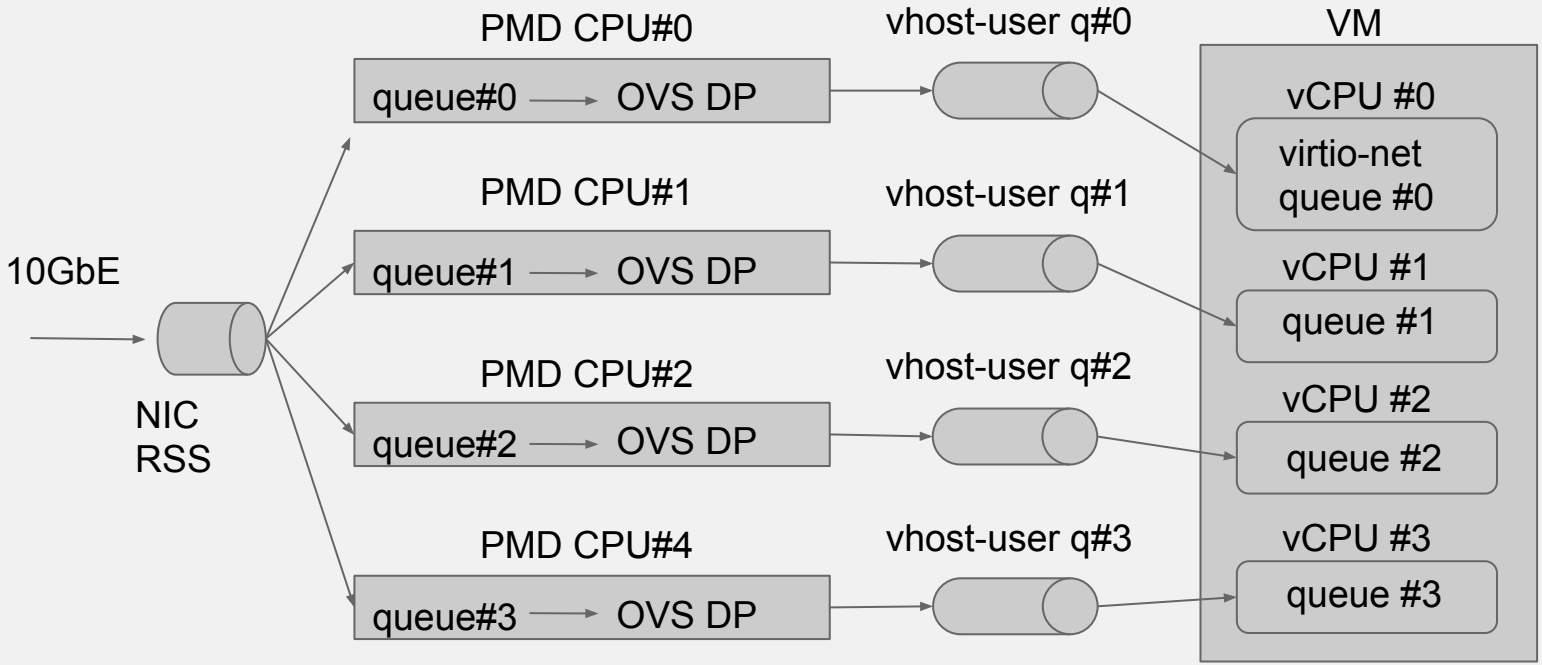
BEFORE conntrack (extra layer of Linux bridges)



AFTER conntrack (native OVS)

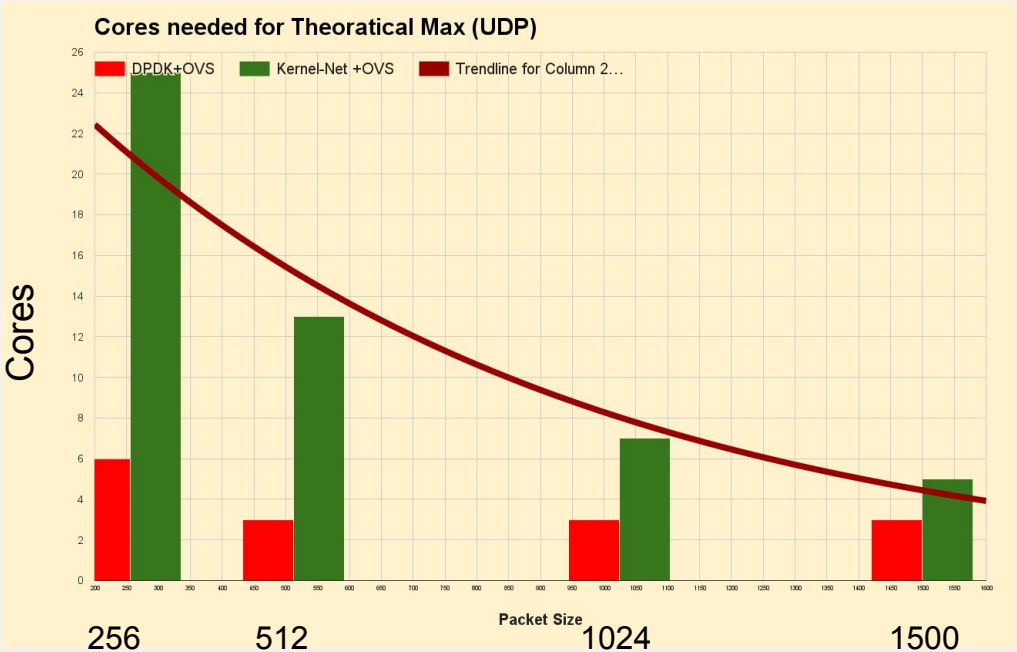


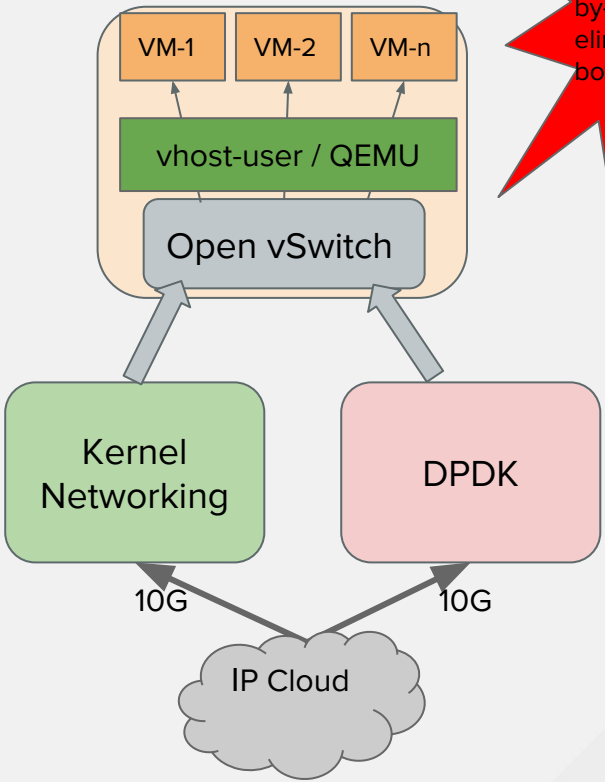
vhost-user Multiple Queues



12 Mpps

Line Rate Performance

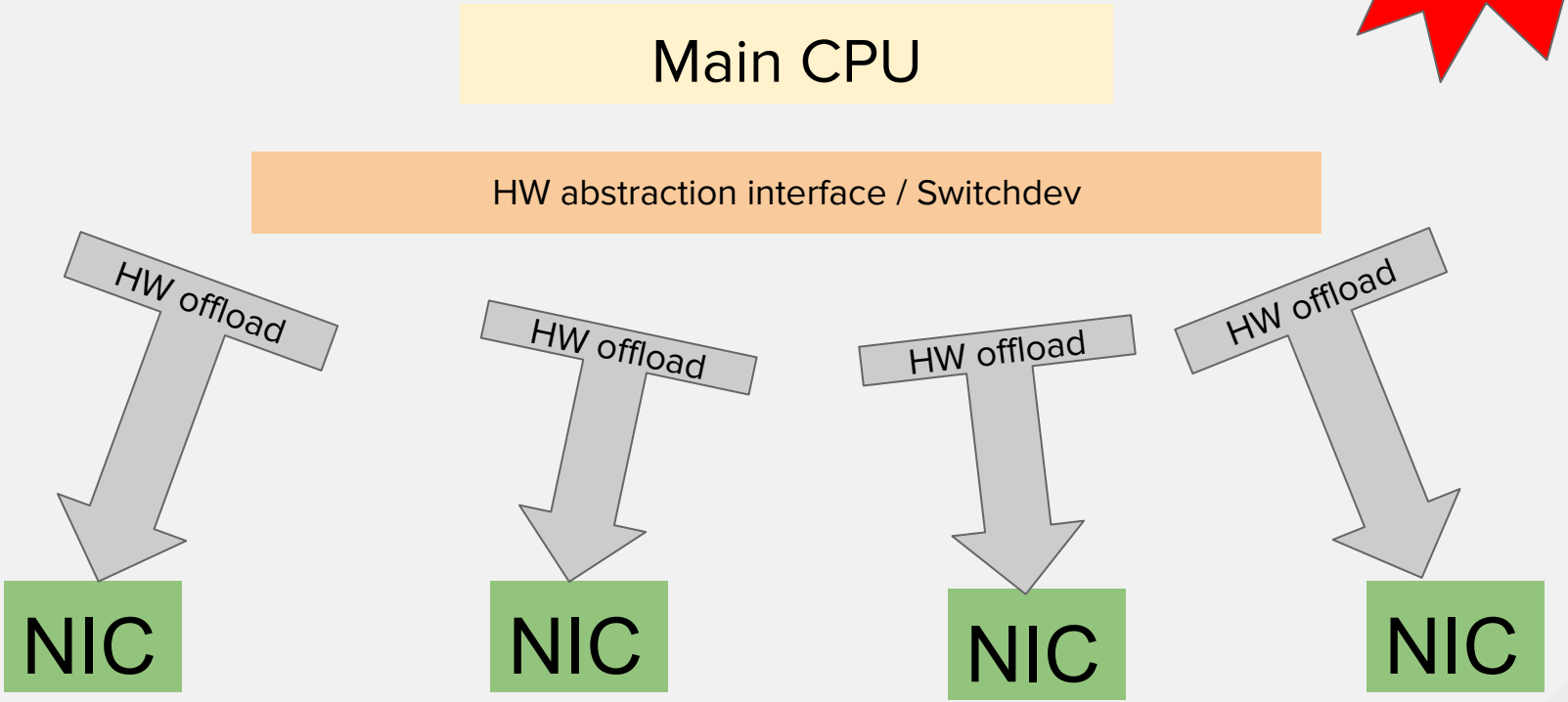




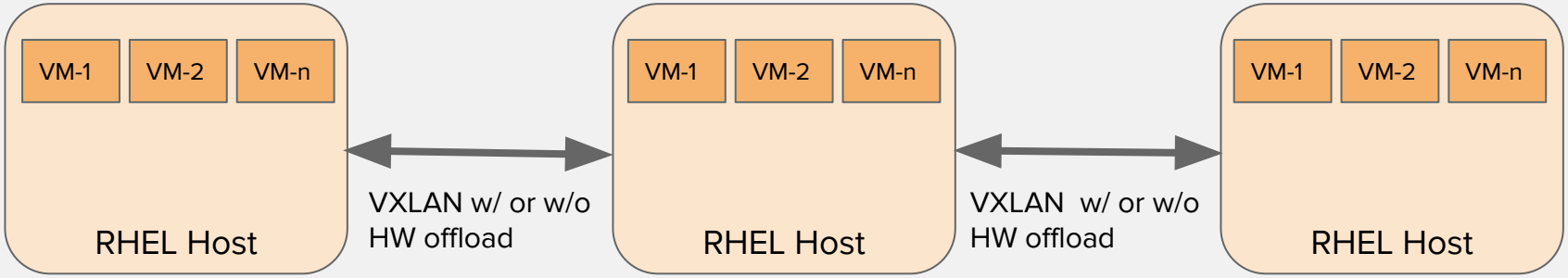
Upstream and RHEL experts working side-by-side to eliminate bottlenecks

More Help is on the Way...

Red Hat works with HW vendors



Theoretical Max Performance - Host to Host

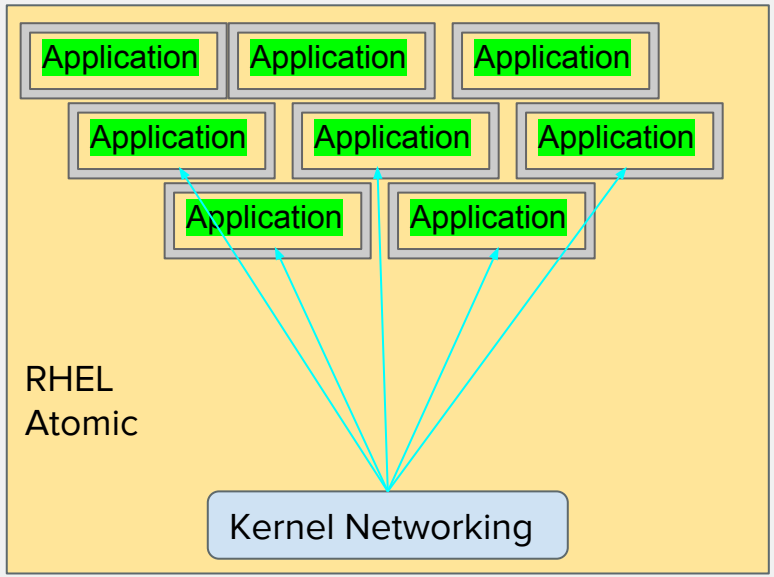


Intel ixgbe

- 9.1 Gb/s w/ HW offload
- 4.8 Gb/s w/o HW offload

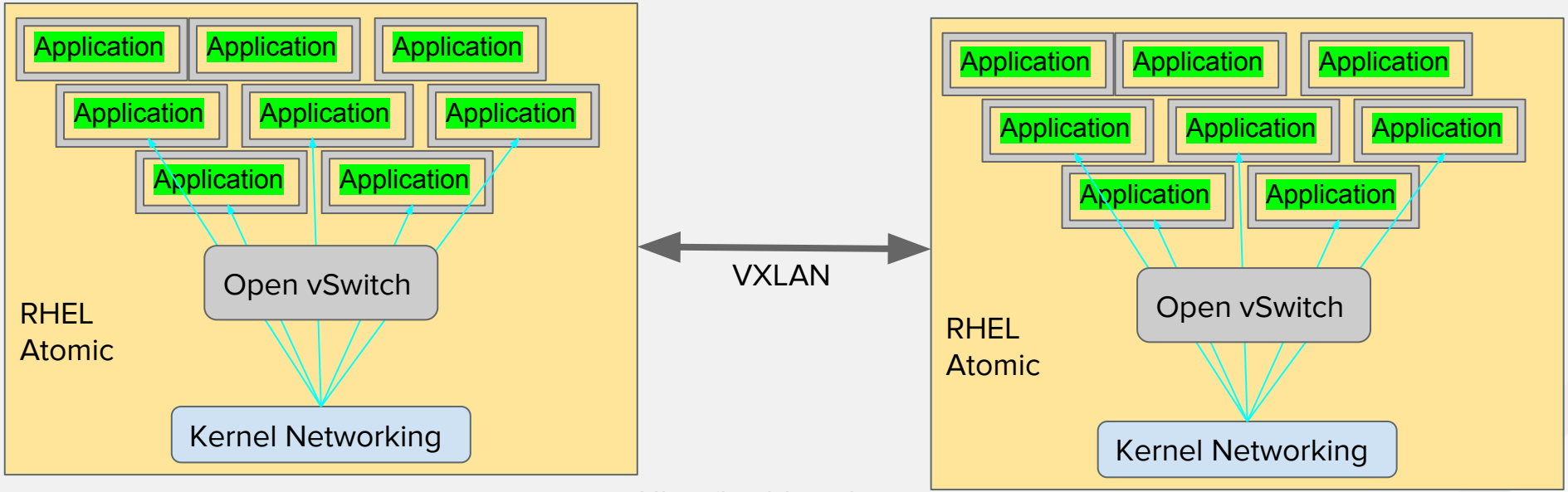
Red Hat works with NIC vendors to provide line rate performance for 10G and 40G, with HW offload

Containers Networking at Theoretical Max



Max throughput at memory transfer rate within the host

Containers Networking at Theoretical Max



Ultra flexible solution
Public / Private / Hybrid clouds

Container

Further Reading

- Red Hat Solution for Network Functions Virtualization (NFV)
- Co-Engineered Together: OpenStack Platform and Red Hat Enterprise Linux
- SR-IOV – Part I: Understanding the Basics
- SR-IOV – Part II: Walking Through the Implementation
- Driving in the Fast Lane: CPU Pinning and NUMA Topology
- Driving in the Fast Lane: Huge Pages
- Scaling NFV to 213 Million Packets per Second with RHEL, OpenStack, and DPDK
- Boosting the NFV datapath with Red Hat OpenStack Platform
- Troubleshooting Networking with Red Hat OpenStack Platform: meet ‘plotnetcfg’

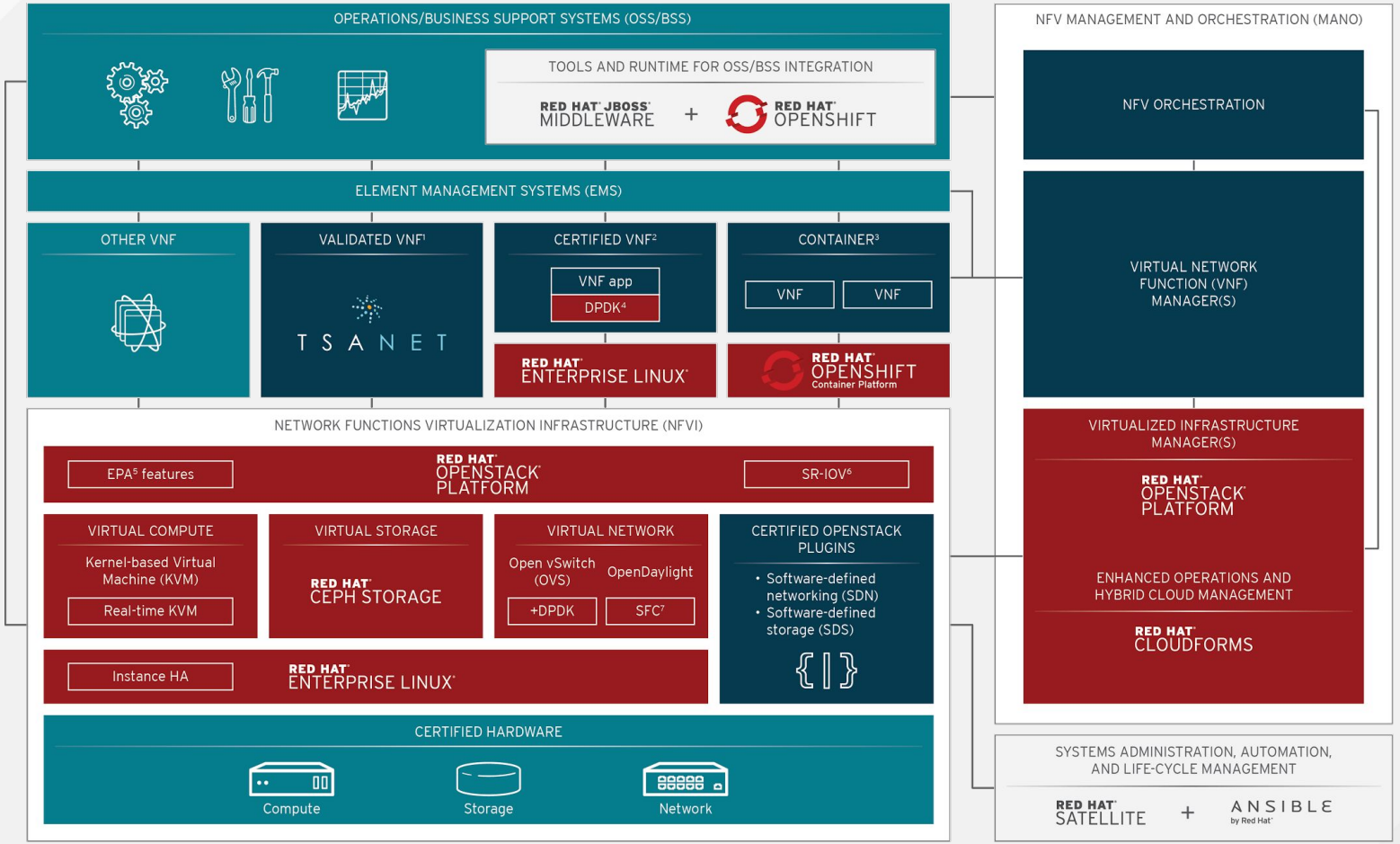
Questions?

Don’t forget to submit feedback using the Red Hat Summit app.

✉ nyechiel@redhat.com

✉ rkhan@redhat.com

BACKUP


OPERATIONS/BUSINESS SUPPORT SYSTEMS (OSS/BSS)
TOOLS AND RUNTIME FOR OSS/BSS INTEGRATION
RED HAT JBOSS MIDDLEWARE + RED HAT OPENSHIFT
ELEMENT MANAGEMENT SYSTEMS (EMS)
OTHER VNF
VALIDATED VNF1
TSANET
CERTIFIED VNF2
VNF app
DPDK4
CONTAINER3
VNF
VNF
RED HAT ENTERPRISE LINUX
RED HAT OPENSHIFT Container Platform
NETWORK FUNCTIONS VIRTUALIZATION INFRASTRUCTURE (NFVI)
EPA5 features
RED HAT OPENSTACK PLATFORM
SR-IOV6
VIRTUAL COMPUTE
Kernel-based Virtual Machine (KVM)
Real-time KVM
VIRTUAL STORAGE
RED HAT CEPH STORAGE
VIRTUAL NETWORK
Open vSwitch (OVS)
OpenDaylight
+DPDK
SFC7
CERTIFIED OPENSTACK PLUGINS
• Software-defined networking (SDN)
• Software-defined storage (SDS)
Instance HA
RED HAT ENTERPRISE LINUX
CERTIFIED HARDWARE
Compute
Storage
Network
NFV MANAGEMENT AND ORCHESTRATION (MANO)
NFV ORCHESTRATION
VIRTUAL NETWORK FUNCTION (VNF) MANAGER(S)
VIRTUALIZED INFRASTRUCTURE MANAGER(S)
RED HAT OPENSTACK PLATFORM
ENHANCED OPERATIONS AND HYBRID CLOUD MANAGEMENT
RED HAT CLOUDFORMS
SYSTEMS ADMINISTRATION, AUTOMATION, AND LIFE-CYCLE MANAGEMENT
RED HAT SATELLITE + ANSIBLE by Red Hat

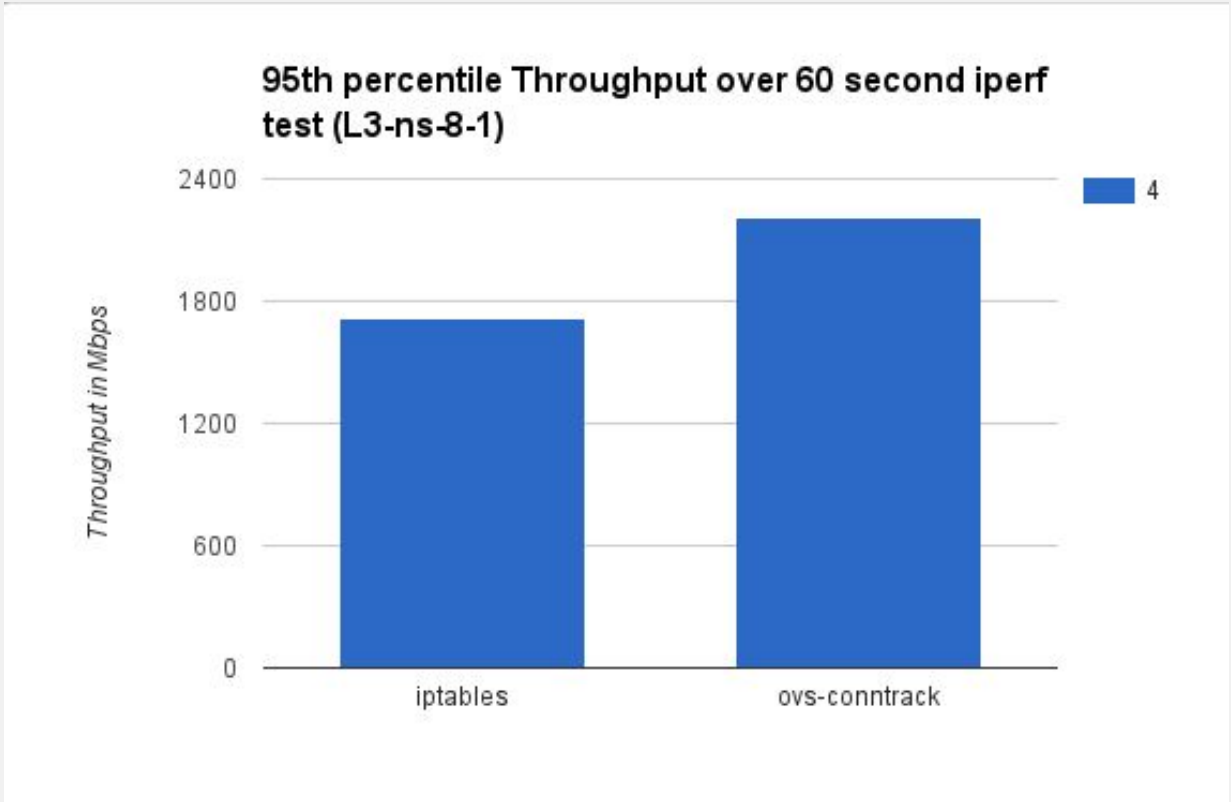
Multiqueue Support for Virtio-net

- Enables packet sending/receiving processing to scale with the number of available virtual CPUs in a guest
- Each guest virtual CPU can have it's own separate transmit or receive queue that can be used without influencing other virtual CPUs
- Provides better application scalability and improved network performance in many cases
- Supported with RHEL 7, available with Red Hat OpenStack Platform 8
  - `hw_vif_multiqueue_enabled=true|false` (default false)
  - Nova will match number of queues to number of vCPUs

Packet size vs Core background data

- Two socketed Intel Haswell Intel(R) Xeon(R) CPU E5-2690 v3
- 2.60GHz
- 12 core per socket for a total of 24 cores
- Hyperthreading disabled
- 256 GB RAM
- Two 10GbE interface 82599 based NIC
- Up to 1% acceptable packet loss (smaller loss shows similar results)
- Unidirectional traffic
- RHEL 7.2 in the host and the guest

Security Groups iptables vs conntrack (1)



Security Groups iptables vs conntrack (2)