Flexible, software-defined networking infrastructure

Red Hat Enterprise Linux enabling the clouds
Clouds Need Red Hat Enterprise Linux

Rashid Khan  
Networking Services

Russell Bryant  
Office of Technology

Wednesday, May 03, 2017
OS Matters
Red Hat Enterprise Linux Enables, Empowers, Excels, Enterprise

Red Hat Enterprise Linux (RHEL)
OS Matters!

Core operating system needs support
- Evaluation of patches regarding stability and impact (Hardening)
- Single point of support (no tennis match of bugs)
- Minimizes downtime with balance of stability and security (CVE)

Guarantee of API and ABI
- Applications will work after minor upgrades
- 3rd party kernel modules under kabi program will continue work
- Synchronization of user space with kernel features

Integration with layered products, and Ansible, and a whole portfolio of products

Dedicated to RHEL
- ~700 Developers
- ~400 QA

In addition
- Layered products
  - Developers
  - QA
- Support Services
  - ~14,000 people ready to ensure your success

Somethings like HW acceleration cannot be done without the OS!
Network Security, Isolation, Tunnels
Security and Isolation

For Multi-tenancy, Fairness, Enterprise readiness

- Robust Firewalling
  - Connection Tracking with NAT in OVS
  - NetFilter
- Network Namespaces
- L2 Security via MACsec
- L3 Security via IPsec

![IPsec vs MACsec Performance](IPsec_vs_MACsec.png)
Tunnels and Isolation

- VLANs (limited identifiers)
- VXLAN with HW offload with IPv6 also
- Geneve (more flexible)
- QinQ 802.1ad (great results)
Packets to/from Virtual Machines
Ready for NFV and Enterprise

- Fully integrated with
  - Red Hat Openstack Platform
  - Red Hat Virtualization

- Full support with
  - Network Namespaces
  - VXLAN
  - Geneve
  - Contrack
  - NAT
  - MQ
  - CPU Pinning
  - Kernel-DP / DPDK on Host
  - DPDK inside the Guest
  - Optimized for 10G
  - Tuning

Zero packet loss absolutely possible!

<table>
<thead>
<tr>
<th>Frame size</th>
<th>Mpps @0.002% loss</th>
<th>Gbps @0.002% loss</th>
<th>Mpps/core @0.002% loss</th>
<th>Mpps @0% loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>22.93</td>
<td>15.41</td>
<td>5.73</td>
<td>13.46</td>
</tr>
</tbody>
</table>

DPDK NOT “experimental” anymore

Intel 82599 NIC Max ~23Mpps
CPUs are needed for SW solutions

What about 25G? 40G? 50G?

Is it game over?
OVS HW offload emerging as an answer

Red Hat working with
- Mellanox
- Netronome
- Cavium
- Chelsio
- Others

RH Value add
- Open Source upstream solution (Must)
- Integrated solution with layered products (Must)
- Unified / Common API for Kernel and DPDK (Goal)
- No Vendor lock-in (Goal)
HW partner A

- Initial results look very promising
- Fully integrated solution with layered products achievable
- Physical-to-Virtual-to-Physical in this test
HW partner A

- Crawl, Walk, Run
- Huge Potential Of Success
- Physical-to-Virtual-to-Physical
- RHEL is a must for success
HW partner B

- Initial results look very promising
- Fully integrated solution with layered products achievable
- Physical-to-Virtual-to-Physical
HW partner B

- Crawl, Walk, Run
- Huge Potential Of Success
- Physical-to-Virtual-to-Physical
- RHEL is a must for success
Containers Networking
(BTW Containers are Linux!)
Container Versus VM Networking

Packet Flow

Userspace

- container
  - App
  - Networking Stack

Kernel

- Open vSwitch

NIC

VM

- OS executing App

vhost-user

DPDK Accelerated
- Open vSwitch

NIC

Packet Flow

Performance
Containers: DPDK or not DPDK?

Packet Flow

Userspace

- container
  - App
  - Networking Stack

Kernel

- Open vSwitch
- NIC

Networking Stack

- DPDK Accelerated Open vSwitch
- NIC

Performance

- Additional Context Switches

#redhat #rhsummit
Container Networking

- Namespaces provide private network stacks inside each container
- Kernel enforces privacy without need for heavy-weight virtualization
- Standard kernel network interfaces, iptables, routing, OVS, can all be used
Containers for NFV
Containers NFV (Needs / Requests)

- Multiple networks (SDN, physical, SAN, etc)
- Physical NICs and SR-IOV (DPDK inside container)
- Flexible IP addressing and overlapping IP networks (multi-tenancy)
- Flat architecture for line-rate processing and low latency
- NUMA and CPU affinity of containerized VFs
- Coordinating widely separated premises
- IPv6 support and availability, especially in public cloud
- Provide existing orchestration and optional micro-service features
Container NFV Proof of Concept Using OpenShift

- Static IPv6 addressing
- Service Function Chains
- Flexible IP addressing
- No bridging or SDN in packet fast-paths
- Multiple interfaces per container (NIC, VLAN, SDN)

Already Done!

- Overlapping IP networks
- NUMA/CPU affinity
- IPv6 SLAAC addressing
- Dynamic Service Function Chains/NSH
- Robust SRIO V

Under investigation:
OVN / SDN
Virtual Networking

- Decouple logical network topology from physical network

Logical Network Topology

Cloud Physical Network
Open Virtual Network (OVN)

- Provide common virtual networking implementation as a part of the base platform (RHEL)
- Comes with OVS and introduces no new dependencies
- Use lessons learned from previous OVS based virtual networking implementations
- Started upstream in 2015 and has now matured
Context - OpenStack Services

- API and common services
- Backend specific orchestration
- Backend specific host technology
Context - OpenStack Neutron

- **REST API**
  - Orchestration layer
    - (Translate Neutron config into configuration of a network across a deployment)
  - Per-host programmable virtual switch programmed by the orchestration layer

- **neutron-server**
  - ML2/OVS driver
  - Neutron agents
    - (OVS, L3, DHCP, Metadata)
  - Open vSwitch
Context - OpenStack Neutron with OVN

neutron-server

ML2/OVS driver
Neutron agents
(OVS, L3, DHCP, Metadata)

Open vSwitch

neutron-server

ML2/OVN driver
OVN services
(ovn-northd, ovn-controller, OVN DBs)

Open vSwitch
<table>
<thead>
<tr>
<th>Product</th>
<th>OVN Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack</td>
<td>neutron-server</td>
</tr>
<tr>
<td></td>
<td>ML2/OVN driver</td>
</tr>
<tr>
<td></td>
<td>OVN services (ovn-northd, ovn-controller, OVN DBs)</td>
</tr>
<tr>
<td></td>
<td>Open vSwitch</td>
</tr>
<tr>
<td>OpenShift</td>
<td>(Kubernetes)</td>
</tr>
<tr>
<td></td>
<td>OVN CNI plugin + kubernetes resource watcher</td>
</tr>
<tr>
<td></td>
<td>OVN services (ovn-northd, ovn-controller, OVN DBs)</td>
</tr>
<tr>
<td></td>
<td>Open vSwitch</td>
</tr>
<tr>
<td>RHV</td>
<td>(oVirt)</td>
</tr>
<tr>
<td></td>
<td>OVN network provider</td>
</tr>
<tr>
<td></td>
<td>OVN services (ovn-northd, ovn-controller, OVN DBs)</td>
</tr>
<tr>
<td></td>
<td>Open vSwitch</td>
</tr>
</tbody>
</table>
OVN Architecture
1) Logical Configure in Northbound Database

Neutron with networking-ovn

OVN
Northbound DB
2) ovn-northd Populates Southbound Database

Neutron with networking-ovn

OVN Northbound DB

OVN Southbound DB

ovn-northd
3) ovn-controller Generates Physical Flows

Neutron with networking-ovn

OVN Northbound DB

ovn-northd

OVN Southbound DB

ovn-controller

OVS

...
Thank You!

Questions?
Backup
## Differences

### Kernel Networking

**Pros**
- Feature Rich / Robust solution
- Ultra Flexible
- Integration with SDN
- Integration with OVS
- Supports Live Migration
- Supports Overlay Networking
- Full Isolation support / Namespace / Multi-tenancy

**Cons**
- Non line rate performance for tiny packets

### DPDK + Vhost-user

**Pros**
- Packets directly sent to user space
- Line rate performance with tiny packets
- Integration with OVS

**Cons**
- Everything has to be in user space

### Device Assignment

**Pros**
- Line rate performance
- Packets directly sent to VMs
- HW based isolation

**Cons**
- Limited number of VMs
- Not as flexible
- Less control from the host
- No OVS
- No SDN
- No live Migration of VMs
- No Overlay