VFd: an SR-IOV Hypervisor using DPDK

Alex Zelezniak

DPDK Summit Userspace - Dublin- 2017

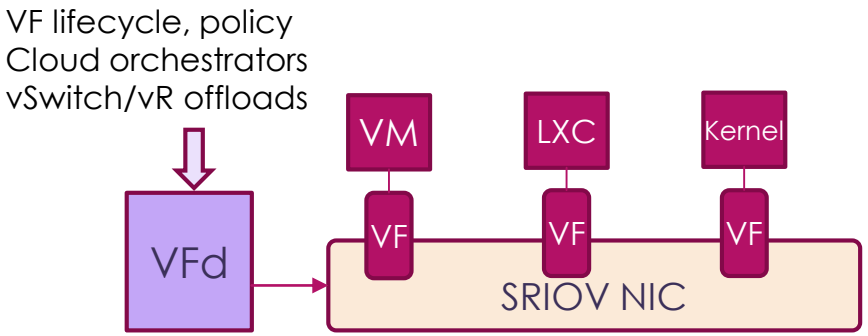
Current state

- Despite many improvements software overlays have fundamental inefficiencies for packet processing workloads
  - High performance network functions being realized via hardware virtualization (SR-IOV) for foreseeable future
  - SmartNIC and hardware offloads rely on SRIOV as the interface between tenants and the NIC
- Dynamic policy enforcement for needed resource management, security, and reliability in multi-tenant NIC sharing
  - For example: allow VF to change MAC, enter promiscuous mode if policy permits, etc.
- Today, there is no single policy enforcement point that takes on “hypervisor-like” functions for SRIOV NICs
  - Linux tools for SR-IOV don’t manage dynamic events – e.g., what to do if a VM tries to change MAC or set VLAN at runtime?
  - Also, kernel drivers don’t support resource allocation, configuration, and offload features in a standardized way

• Steering traffic using multiple VLANs	• Mirroring
• QoS (TC)	• Separate VLAN/MAC anti-spoofing control
• VF stats	• QinQ management
• BUM traffic management	• MAC filtering

VFd: a "hypervisor" for SR-IOV NICs

- Privileged software (driver) performing hypervisor function for SR-IOV network devices
  - Allocate/deallocate VFs
  - Flexibly allocate resources, e.g., queues, QoS classes, to VFs
  - Manage policy, e.g., VLAN steering, QinQ tagging, filtering, mirroring, anti-spoofing, in a single place
  - Configure VFs
  - Collect various PF/VF statistics
  - Flexible, user-space tool



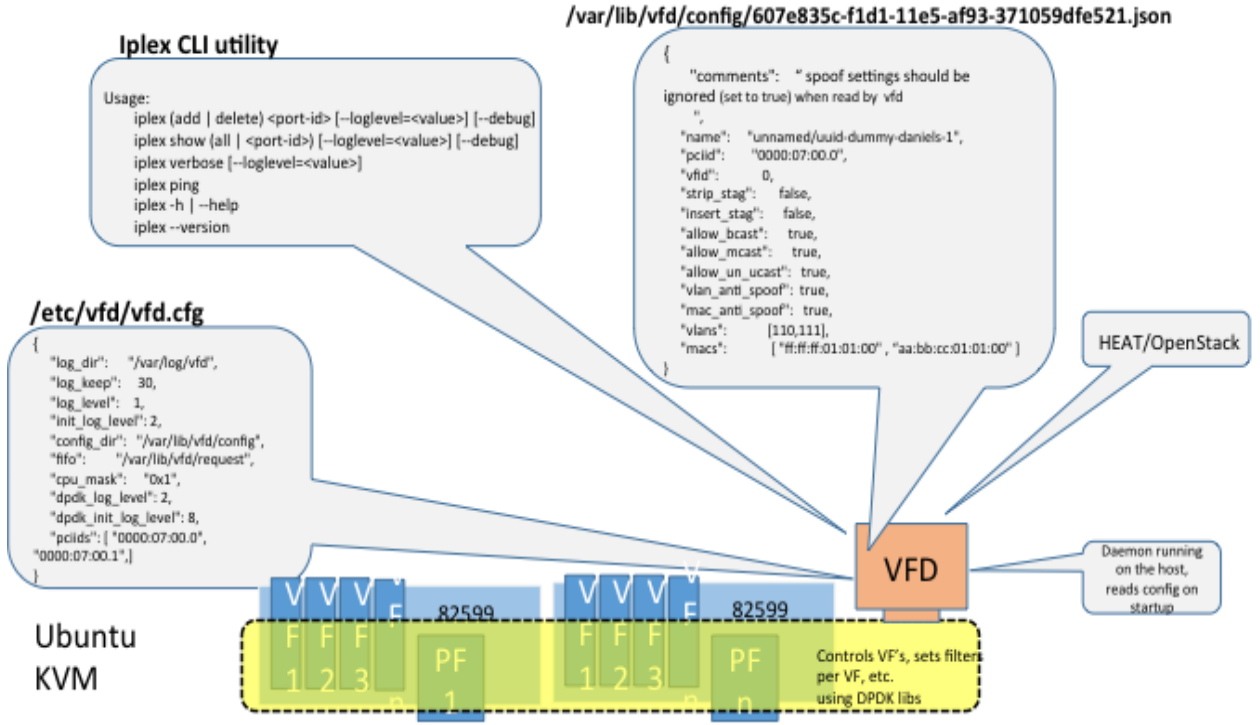
- Unfortunately, we hit some practical snags
  - Several of the functions needed are missing
  - Kernel doesn't standardize functions that do exist - each NIC vendor implements in their own way
  - No mechanisms for handling runtime events that are policy affecting
  - Many environments often run old kernels, and kernel upgrades is a major activity that could impact vast infrastructure. This impedes fast evolution in this rapidly changing space

Why DPDK?

- User space, rapid evolution
- SR-IOV and DPDK are both tools for high performance, so common target community
- Support from most major modern NICs where SR-IOV is involved
- NFV mindset

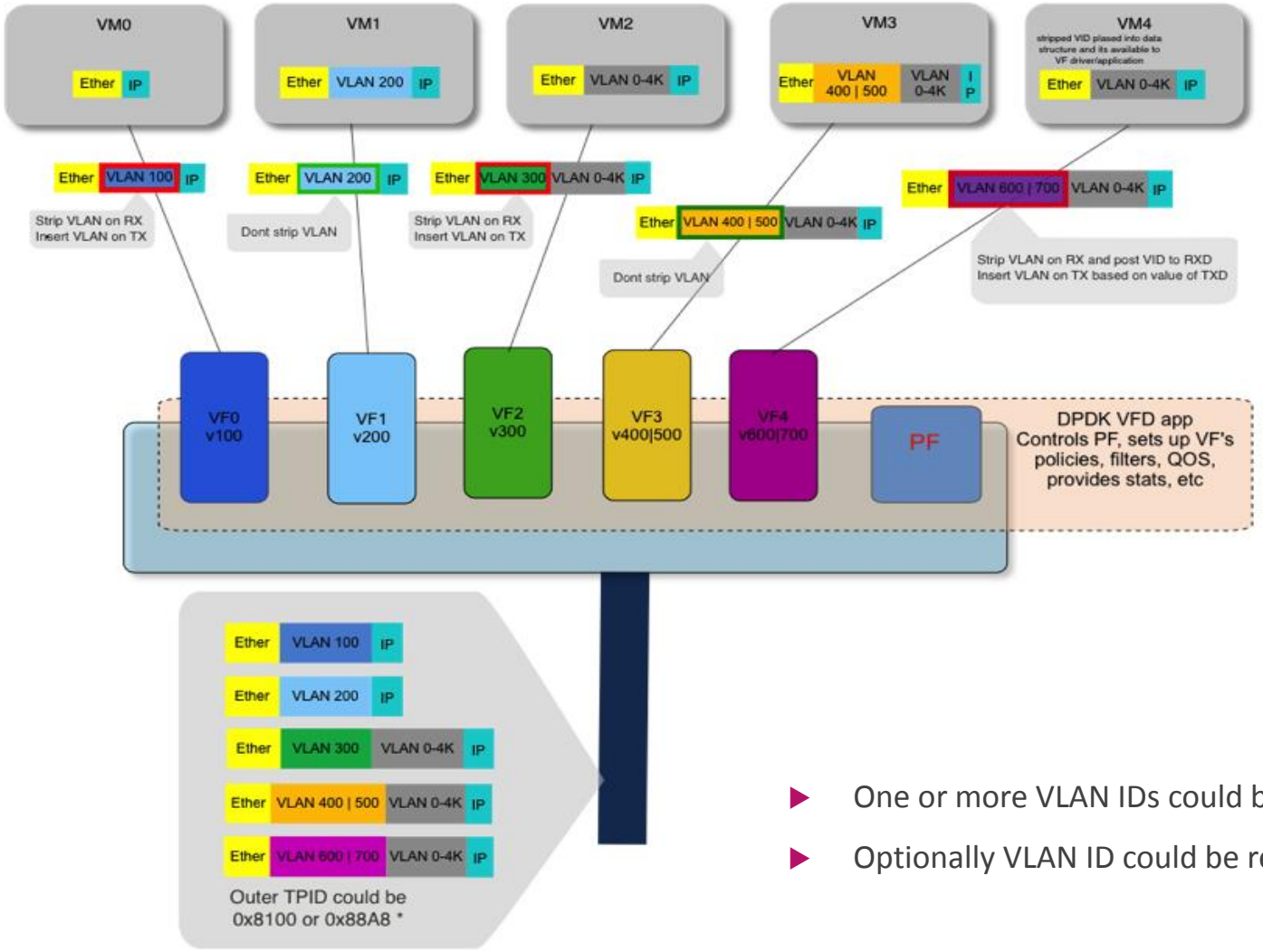
VFd architecture

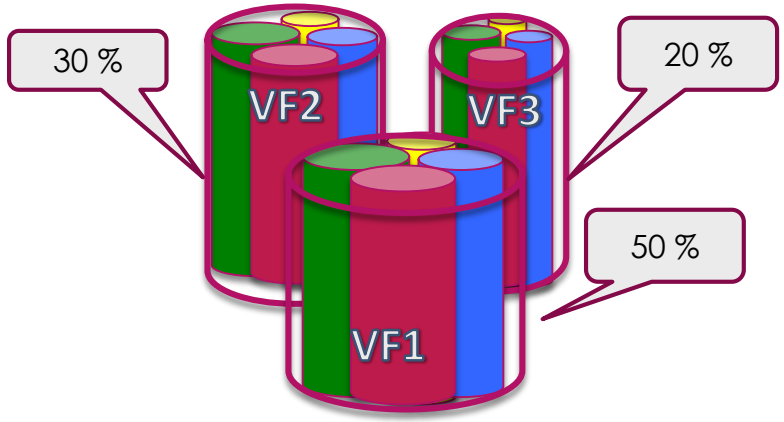
- Runs as a daemon process that can handle static configuration as well as dynamic events
- Backend is DPDK
- Front end tools allowing configuration and gathering statistics



PF/VF	ID	PCIID	Link	Speed	Duplex	RX pkts	RX bytes	RX errors	RX dropped	TX pkts	TX bytes	TX errors	Spoofed
pf	0	0000:05:00.0	UP	25000	1	0	0	0	0	0	0	0	0
vf	0	0000:05:02.0	UP			0	0			0	0	2	
pf	1	0000:05:00.1	UP	25000	1	0	0	0	0	0	0	0	0
pf	2	0000:81:00.0	UP	10000	1	0	0	0	0	20	1200	0	8
vf	0	0000:81:10.0	UP			0	0			20	920	0	
pf	3	0000:81:00.1	UP	10000	1	0	0	0	0	0	0	0	0
pf	4	0000:84:00.0	UP	25000	1	0	0	0	0	0	0	0	0
vf	0	0000:84:02.0	DOWN			0	0			0	0	0	
pf	5	0000:84:00.1	UP	25000	1	0	0	0	0	0	0	0	0

VFd packet steering/VLAN stripping/QoS





- Traffic classes with one strict-priority queue supported
- Packets placed to the appropriate queues based on PCP value
- Configurable Min/Max bandwidth values per TC/VF

- One or more VLAN IDs could be used to steer traffic to the VF
- Optionally VLAN ID could be removed on RX and inserted on TX

DPDK APIs used

- Uses “experimental” DPDK API
  - rte_pmd_ixgbe.h
  - rte_pmd_i40e.h
  - rte_pmd_bnxt.h

rte_pmd_[ixgbe | i40e | bnxt]_ping_vfs
rte_pmd_[ixgbe | i40e | bnxt]_set_vf_mac_anti_spoof
rte_pmd _[ixgbe | i40e | bnxt]_ set_vf_vlan_anti_spoof
rte_pmd _[ixgbe | i40e | bnxt]_ set_tx_loopback
rte_pmd _[ixgbe | i40e | bnxt]_ set_vf_unicast_promisc
rte_pmd _[ixgbe | i40e | bnxt]_ set_vf_broadcast
rte_pmd _[ixgbe | i40e | bnxt]_ set_vf_multicast_promisc
rte_pmd _[ixgbe | i40e | bnxt]_ set_vf_mac_addr
rte_pmd _[ixgbe | i40e | bnxt]_ set_vf_vlan_stripq
rte_pmd _[ixgbe | i40e | bnxt]_ set_vf_vlan_insert
rte_pmd _[ixgbe | i40e | bnxt]_ set_vf_vlan_filter
rte_pmd _[ixgbe | i40e | bnxt]_ get_vf_stats
rte_pmd _[ixgbe | i40e | bnxt]_ reset_vf_stats

Move to

“Generic” DPDK API

VFd status

- Supports ixgbe, i40e, bnxt devices
- Working on supporting QoS with more NICs
- Adding mirroring
- Improving operational support/troubleshooting
- Other vendors are working to contribute

Future

- Remove “experimental tag” from new API’s?
- Add generic APIs to DPDK?
- Add Netlink/sysfs/procfs to interface Linux tools?
- Variable number of queues per VF?
- Move complexity of VF configuration to the “SR-IOV Hypervisor“ simplifying creation of lightweight, portable VF?
- PF/VF reset/recovery?
- Standardized interface for SmartNIC offloads of hypervisor like functions – e.g., VFd as integration point for OVS, vRouter
- Who would benefit from using it? Cloud platform integrators, vSwitch/router projects, VNF vendors, …
- Who should think about contributing to it? NIC vendors, vSwitch/router projects, …
- How can you help?

Acknowledgments

- AT&T
  - E. Scott Daniels
  - Kaustubh Joshi
  - Dhanunjaya Ravada
  - John Craig
- Broadcom
  - Ajit Khaparde
  - Stephen Hurd
  - Venugopala Bhat
  - Hoan Do
  - Sudheer Vegesna
- Intel
  - Wenzhuo Lu
  - Bernard Iremonger
  - Aaron Rowden
  - Rahul Shah
  - Lian-min Wang
  - Jingjing Wu
  - Ferruh Yigit
  - Qi Z Zhang
  - Helin Zhang

Questions?

Alex Zelezniak

alexz@att.com

http://www.github.com/att/vfd