VPP: The ultimate NFV vSwitch (and more!)?

Franck Baudin, Principal Product Manager - OpenStack NFV - Red Hat
Uri Elzur, DNSG CTO - Intel

OpenStack Summit | Barcelona 2016

Agenda

- FD.io project and community overview
- FD.io Vector Packet Processing framework
- FD.io in a broader context
  - OPNFV and FDS
  - OpenStack and ML2
  - ODL and SFC
  - Containers
- Summary

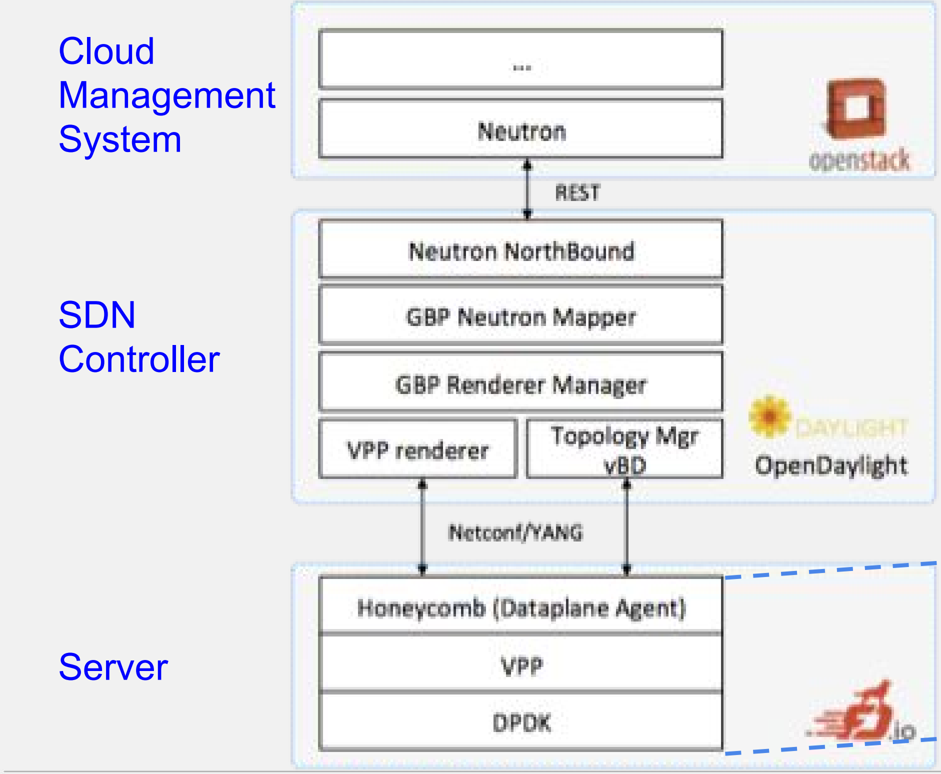
Key message:

FD.io is getting ready for production readiness and offers some interesting innovations

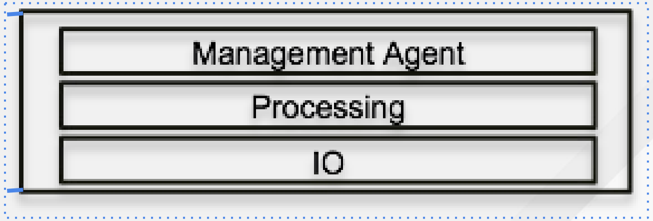
FD.io overview

Some slides adapted from multiple presentations at wiki.fd.io
The authors wish to also thank: Frank Brockners, Keith Burns, Joel Halpern, Ray Kinsella, Hongjun Ni, Ed Warnicke, Yi Yang, Jerome Tollet, Danny Zhou, …

FD.io in context



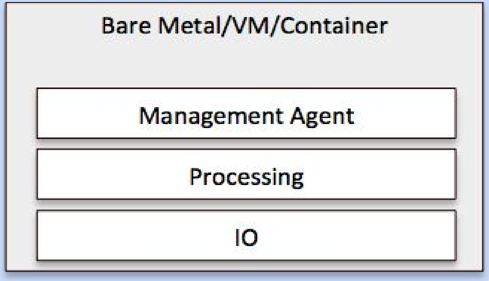
- FD.io
  - Offers a new high speed dynamic and programmable data plane adapted and optimized to the Server architecture
- VPP provides
  - IO, Processing and Management, for Bare Metal, VM or Container
  - IO: HW / vHW cores/threads
  - Packet Processing: Classify, Transform, Prioritize, Forward, Terminate
  - Management Agents: control/manage IO/Processing
  - Local and remote



Introduction Fast Data: .io

CISCO intel ERICSSON 6WIND HUAWEI BROCADE CAVIUM NETWORKS COMCAST INOCYBE MESOSPHERE NXP PLUMgrid PROJECT CALICO redhat

Scope


Bare Metal/VM/Container

Management Agent

Processing

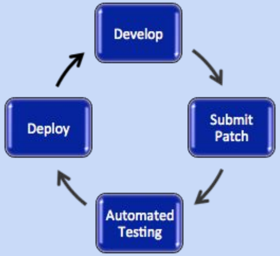
IO

Good Governance -> Fast Innovation

Modular Governance supports concept of independent sub-projects enabling:

- faster evolution
- independant work stream
- Greater flexibility

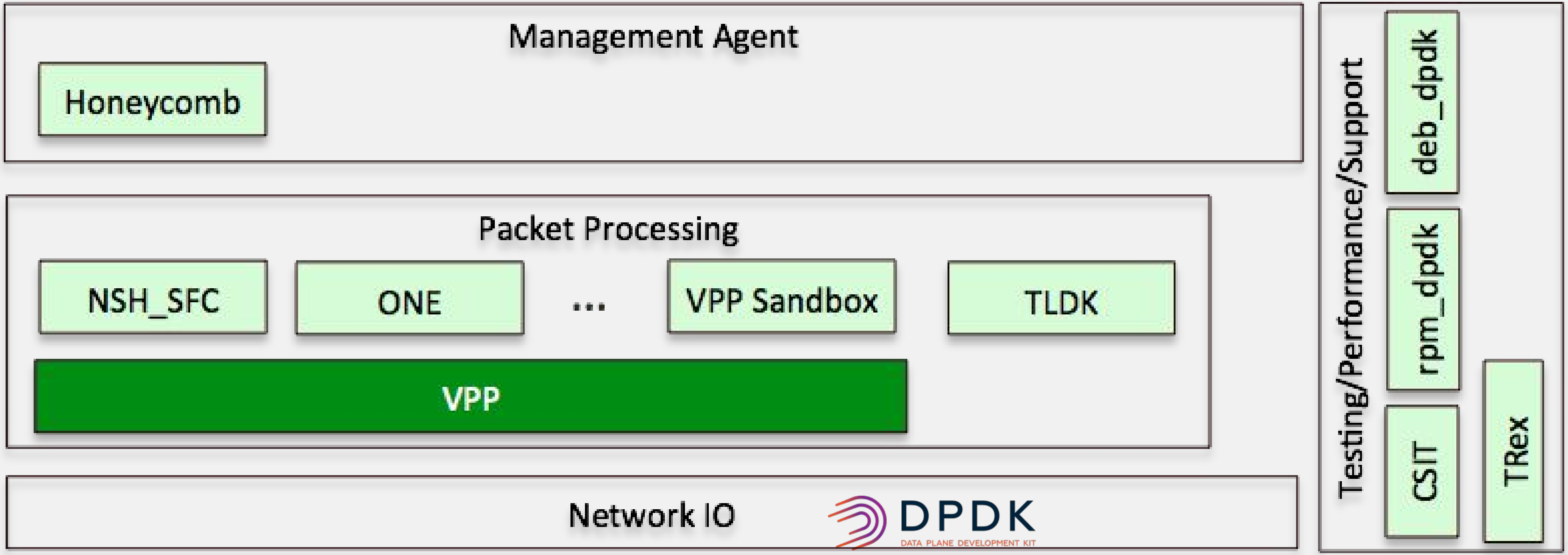
Continuous Performance Lab



FD.io projects

Legend:

- ⬜ - New Projects
- 🟩 - Core Projects



FD.io governance

Anyone May Participate – Not just members

Anyone can contribute code

Anyone can rise to being a committer via meritocracy

Anyone can propose a subproject

Technical Steering Committee

Fosters collaboration among sub-projects, but is not involved in day to day management of sub-projects

Approves new sub-projects, sets development process guidelines for the community, sets release guidelines for multi-project or simultaneous releases, etc.

Initial TSC will be seeded with representatives from Platinum Membership and core project PTLs with the goal of replacing representatives with Project Leads after the first year

Subprojects:

Composed of the committers to that subproject – those who can merge code

Responsible for sub project oversight and autonomous releases

Make technical decisions for that subproject by consensus, or failing that, majority vote.

Governing Board will Oversee Business Decision Making

Set Scope and Policy of Consortium

Composed of Platinum member appointees, elected Gold, Silver, and Committer member representatives

Examples of business needs include: budgeting, planning for large meetings (e.g. a Summit, Hackfest), marketing, websites, developer infrastructure, test infrastructure, etc.

fd.io committers/contributors

CISCO
intel
ERICSSON
6WIND
redhat
HUAWEI
COMCAST
QOSMOS
NXP
Netgate
INOCYBE
OPENDAYLIGHT DISTRIBUTION
UPC
KALRAY

Universitat Politècnica de Catalunya (UPC)

9 Months
Feb-Sept 2016
2159 commits
117 contributors

30 Days
Aug 21-Sept 20
301 commits
47 contributors

VPP: a packet processing framework for VNF and vSwitch

VPP architecture 1/3

Networking application SDK

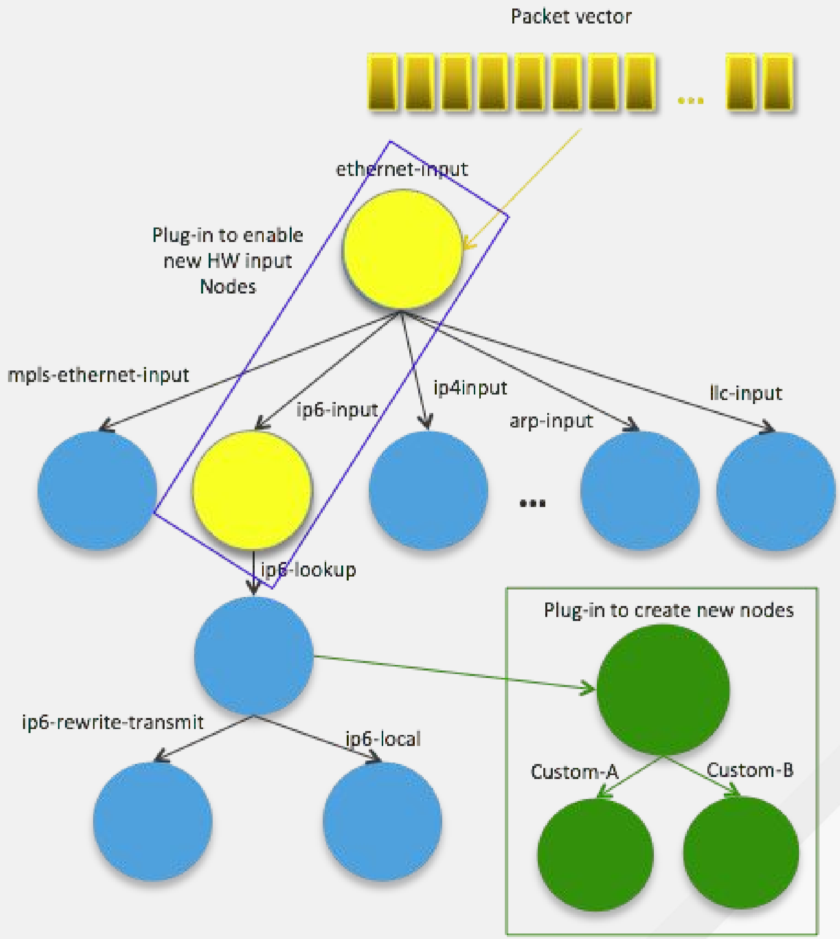
Protocol stack made of graph nodes

Vector based

- large bulk (256)
- dual loop (prefetch) for all nodes

Comparison with OVS

- Compiled graph vs OpenFlow
  - e.g. Stack vs Flow based (== cache)
- Ephemeral configuration vs OVSDB
  - ODL/OpenStack/XYZ side agent
- No kernel implementation, 100% userland



VPP architecture 2/3

Portability

Multiple architecture support: x86, ARM, PPC

OS portability thanks to clib

One “NIC” driver == one VPP input node:

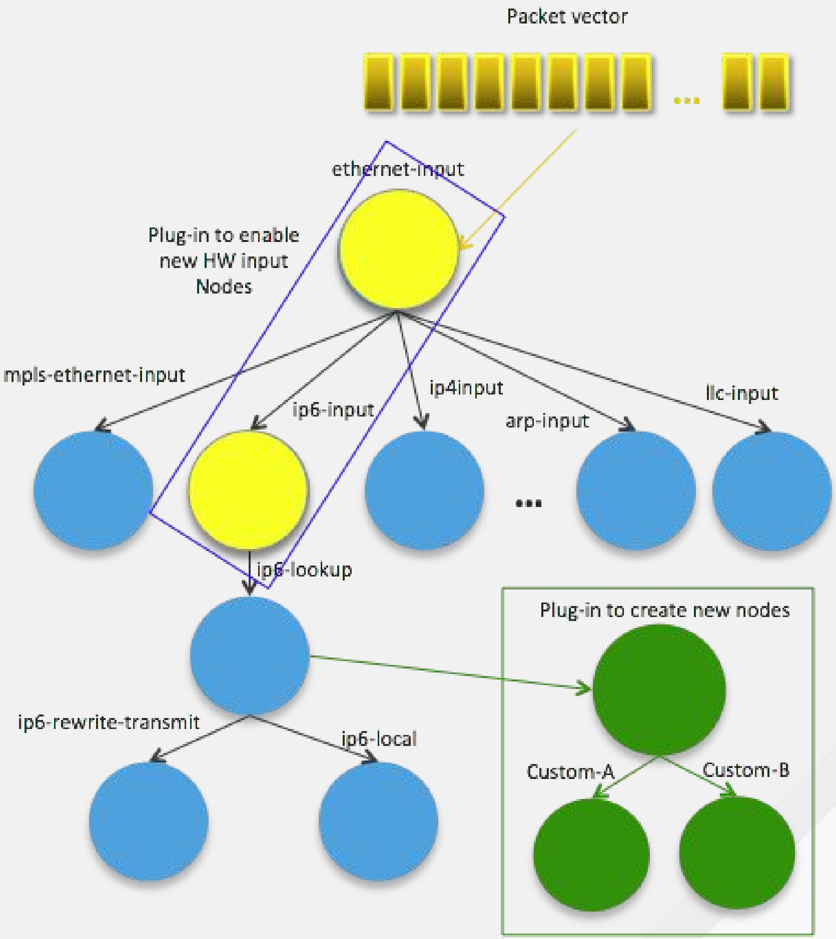
- DPDK[1], tun/tap, AF_packet, netmap, and even legacy PCI drivers (intel Niantic), vhost-user
- ssvm: SHM between two VPP instances, typically for containers use cases

Leverages DPDK HW accelerators (crypto, …)

Deployment models: bare metal, VMs, containers

Critical nodes for various CPU generation optimization

[1] DPDK patches are pushed upstream, zero patch goal



VPP architecture 3/3

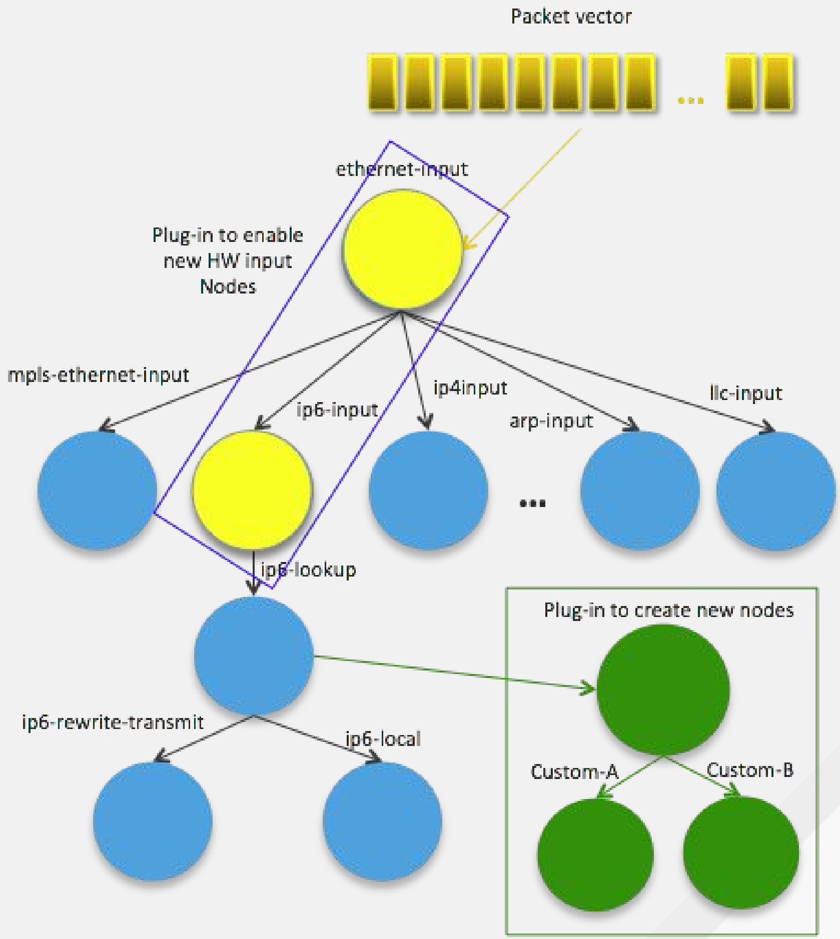
Modularity/flexibility

Plugins == subprojects

Plugins can:

- Introduce new graph nodes
- Rearrange packet processing graph
- Be built independently of VPP source tree
- Be added at runtime (drop into plugin directory)
- Extend control API
- All in user space

Permit to build: vSwitch, vRouter, CG NAT, ...



Per node statistics and counters

```
vpp# show run
Thread 1 vpp_wk_0 (lcore 18)
Time 2.8, average vectors/node 256.00, last 128 main loops 12.00 per node 256.00
  vector rates in 4.4518e6, out 4.4518e6, drop 0.0000e0, punt 0.0000e0
             Name                 State         Calls          Vectors        Suspends         Clocks       Vectors/Call
FortyGigabitEthernet81/0/1-out   active              47971        12280576               0          1.37e1          256.00
FortyGigabitEthernet81/0/1-tx    active              47971        12280576               0          2.11e2          256.00
dpdk-input                       polling             47971        12280576               0          1.24e2          256.00
ethernet-input                   active              47971        12280576               0          9.08e1          256.00
l2-input                         active              47971        12280576               0          3.72e1          256.00
l2-output                        active              47971        12280576               0          3.59e1          256.00
---------------
Thread 2 vpp_wk_1 (lcore 54)
Time 2.8, average vectors/node 16.04, last 128 main loops 0.00 per node 0.00
  vector rates in 5.9195e5, out 5.9195e5, drop 0.0000e0, punt 0.0000e0
             Name                 State         Calls          Vectors        Suspends         Clocks       Vectors/Call
FortyGigabitEthernet81/0/0-out   active             101774         1632928               0          3.59e1           16.04
FortyGigabitEthernet81/0/0-tx    active             101774         1632928               0          2.52e2           16.04
```

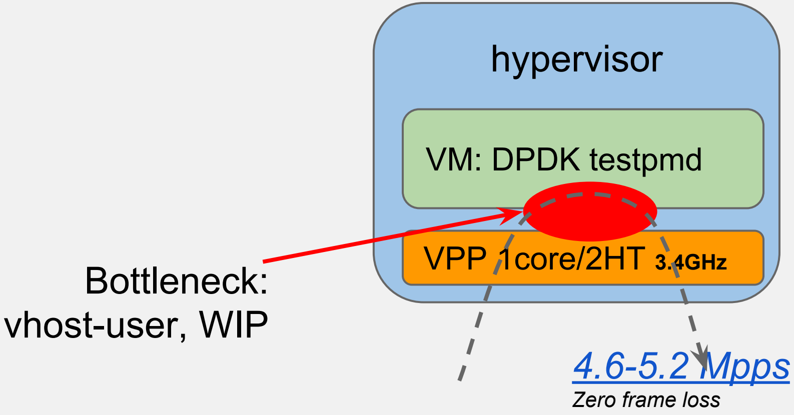
Debug: packet tracer

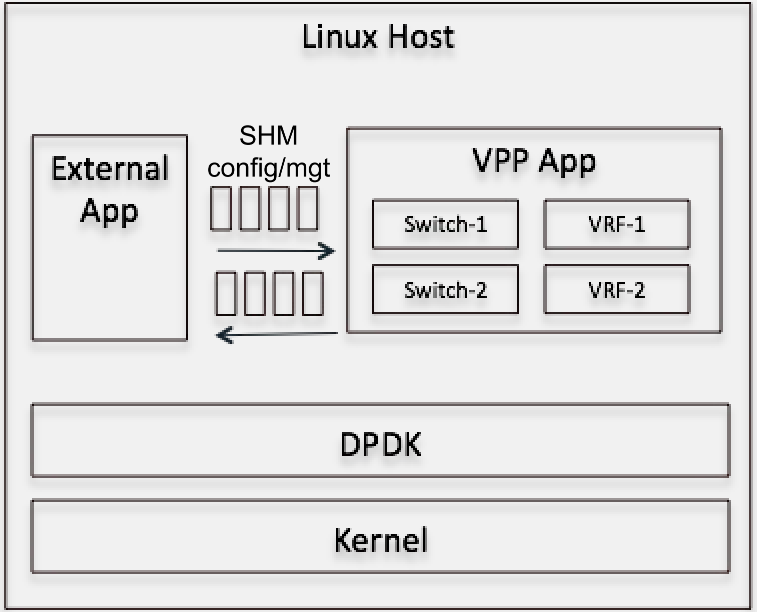
```
vpp# trace add dpdk-input 10
vpp# show trace
00:06:34:045368: dpdk-input
  FortyGigabitEthernet81/0/1 rx queue 0
  buffer 0x15210: current data 0, length 60, free-list 0, totlen-nifb 0, trace 0x0
  PKT MBUF: port 1, nb_segs 1, pkt_len 60
    buf_len 2176, data_len 60, ol_flags 0x0, data_off 128, phys_addr 0xbdb44300
    packet_type 0x191
    Packet Types
      RTE_PTYPE_L2_ETHER (0x0001) Ethernet packet
      RTE_PTYPE_L3_IPV4_EXT_UNKNOWN (0x0090) IPv4 pac
      RTE_PTYPE_L4_TCP (0x0100) TCP packet
  IP4: 3c:fd:fe:9d:7b:a9 -> 3c:fd:fe:9d:7b:a8
  TCP: 192.168.1.1 -> 192.168.0.1
    tos 0x00, ttl 4, length 46, checksum 0x62f8
    fragment id 0xd17f
```

```
00:06:34:045462: ethernet-input
  IP4: 3c:fd:fe:9d:7b:a9 -> 3c:fd:fe:9d:7b:a8
00:06:34:045494: l2-input
  l2-input: sw_if_index 2 dst 3c:fd:fe:9d:7b:a8 src 3c:fd:fe:9d:7b:a9
00:06:34:045500: l2-output
  l2-output: sw_if_index 1 dst 3c:fd:fe:9d:7b:a8 src 3c:fd:fe:9d:7b:a9
.../...
```

VPP as a vSwitch/vRouter

Alternative to OVS-DPDK or Contrail vRouter





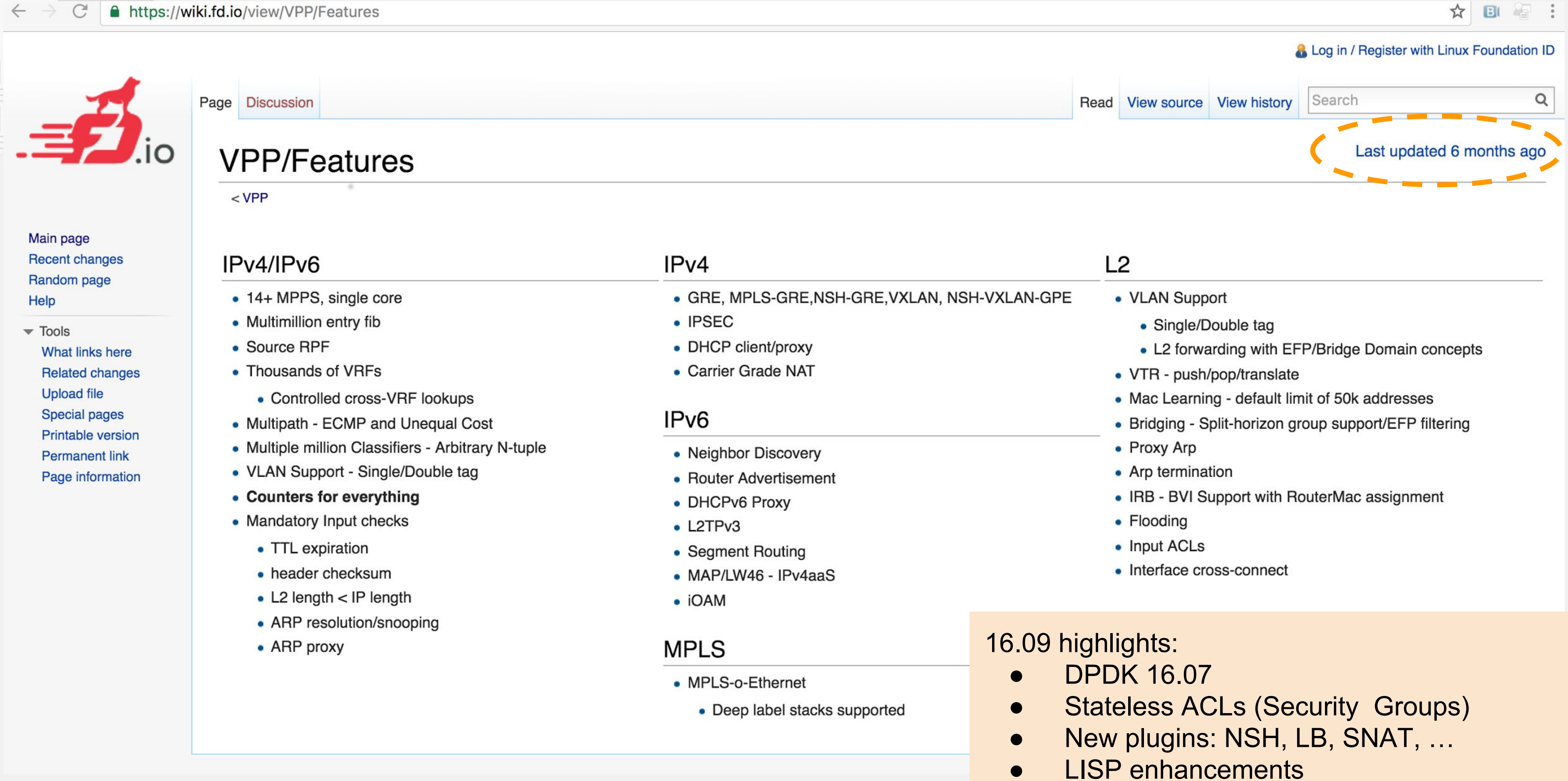
Configuration “stored/pushed” by external App

Neutron implementation based on bridges

But benchs run on port cross-connect, like OVS-DPDK benchs: OpenStack end to end benchs are WIP

VMs connected via vhost-user (like OVS-DPDK)

- Specific vhost-user implementation
- Moving to DPDK implementation or vice-versa



VPP/Features

Last updated 6 months ago

< VPP

IPv4/IPv6

- 14+ MPPS, single core
- Multimillion entry fib
- Source RPF
- Thousands of VRFs
  - Controlled cross-VRF lookups
- Multipath - ECMP and Unequal Cost
- Multiple million Classifiers - Arbitrary N-tuple
- VLAN Support - Single/Double tag
- **Counters for everything**
- Mandatory Input checks
  - TTL expiration
  - header checksum
  - L2 length < IP length
  - ARP resolution/snooping
  - ARP proxy

IPv4

- GRE, MPLS-GRE,NSH-GRE,VXLAN, NSH-VXLAN-GPE
- IPSEC
- DHCP client/proxy
- Carrier Grade NAT

IPv6

- Neighbor Discovery
- Router Advertisement
- DHCPv6 Proxy
- L2TPv3
- Segment Routing
- MAP/LW46 - IPv4aaS
- iOAM

MPLS

- MPLS-o-Ethernet
  - Deep label stacks supported

L2

- VLAN Support
  - Single/Double tag
  - L2 forwarding with EFP/Bridge Domain concepts
- VTR - push/pop/translate
- Mac Learning - default limit of 50k addresses
- Bridging - Split-horizon group support/EFP filtering
- Proxy Arp
- Arp termination
- IRB - BVI Support with RouterMac assignment
- Flooding
- Input ACLs
- Interface cross-connect

16.09 highlights:

- DPDK 16.07
- Stateless ACLs (Security Groups)
- New plugins: NSH, LB, SNAT, …
- LISP enhancements

VPP Gating CI: CSIT

TRex

Realistic Traffic Generator

Continuous **S**ystem **I**ntegration and **T**esting

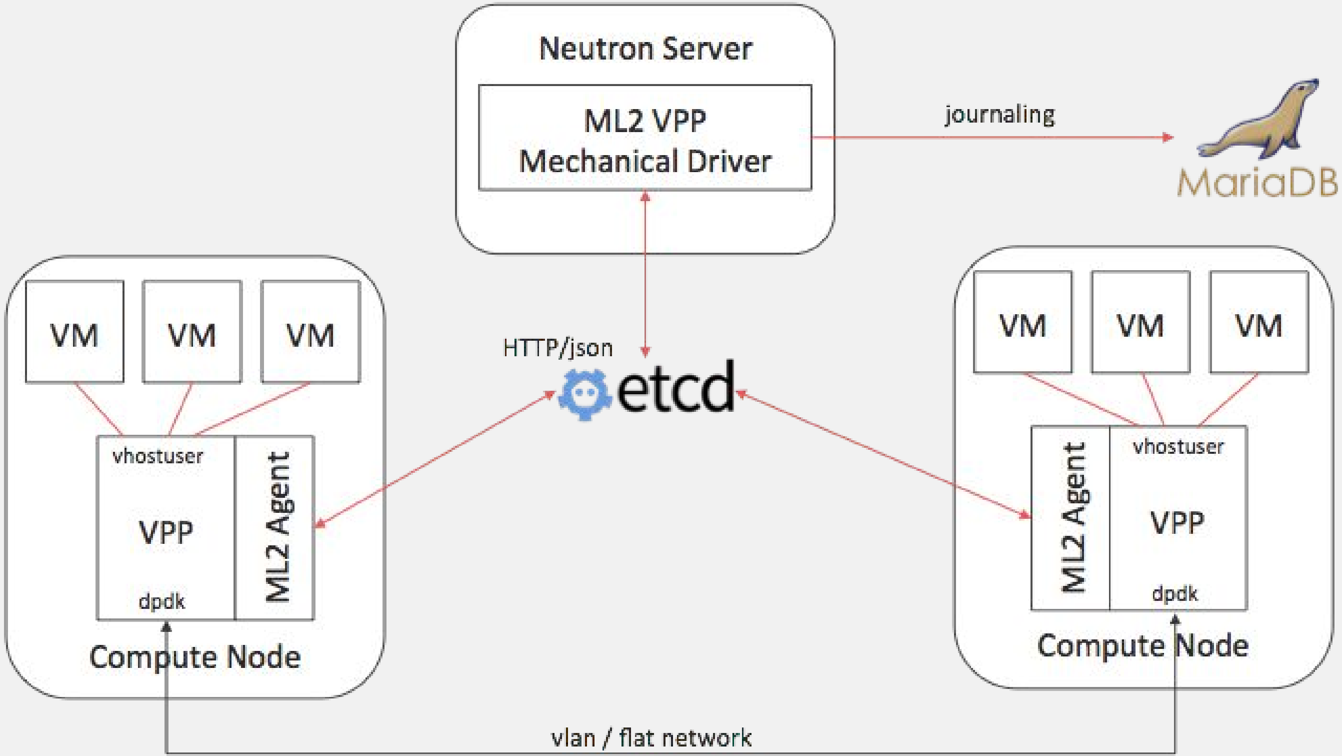
- CSIT: FD.io project hosting the test code
  - WIP: functional test code moving to projects
  - CSIT focus on performances
- Execution of CSIT test suites on LF FD.io virtual and physical compute environments (Continuous Performance Lab, aka CPL)
- Integration with FD.io continuous integration systems (Gerrit, Jenkins, ...)
- Gating CI: performances regression will discard a commit!

1. NIC devices and drivers
2. IPv4 data plane
3. IPv4 control plane
4. IPv4 encapsulations
5. IPv4 telemetry
6. IPv6 data plane
7. IPv6 control plane
8. IPv6 encapsulations
9. IPv6 telemetry
10. Ethernet L2 data plane
11. Ethernet L2 control plane
12. Ethernet L2 encapsulations
13. Ethernet L2 telemetry
14. MPLS data plane
15. NSH data plane

VPP: OpenStack and OpenDaylight

Announce October 5th: http://lists.openstack.org/pipermail/openstack-dev/2016-October/105148.html

1) VPP neutron ML2 plugin: architecture



VPP neutron ML2 plugin: features

Network Types: Flat, VLAN

Port connectivity:

- vhostuser ports for Virtual Machines
- Tap port for ”service connectivity”: DHCP (q-dhcp), Router (q-router)

Supported HA scenario

- VPP agent restart
  - resets VPP to a clean state
  - fetches any existing port data from etcd and programs the VPP state.
- ml2 driver restart
  - retrieves information from etcd
  - uses the journal to push as yet unpublished data to etcd

Installers: OPNFV APEX (based on TripleO), DevStack

Roadmap

- Security Groups / Anti Spoofing
- Tap-as-a-Service
- Enhanced automated testplan / testbed for unit testing

Radar

- *Integration with Telemetry systems*
- *Support for VXLAN*

2) OPNFV FDS project

OpenDaylight

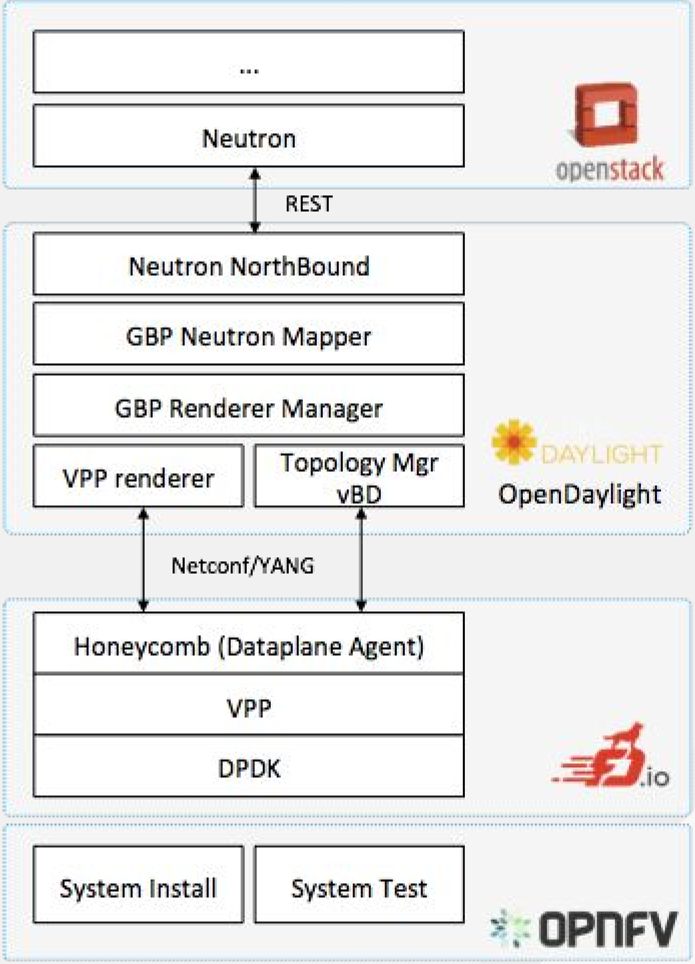
- GBP Neutron Mapper
- GBP Renderer Manager enhancements
- VPP Renderer
- Virtual Bridge Domain Mgr / Topology Manager

FD.io

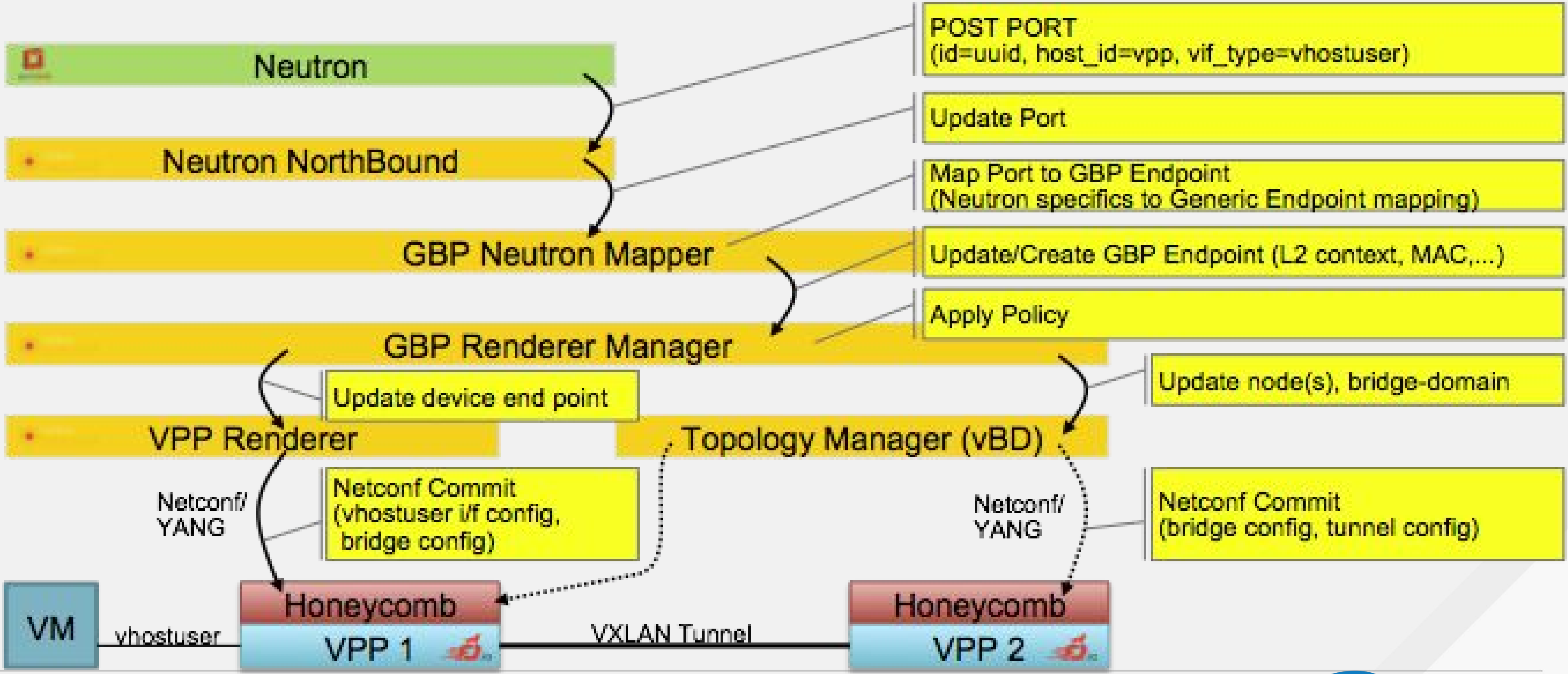
- HoneyComb – Enhancements
- VPP – Enhancements
- CSIT – VPP component tests

OPNFV

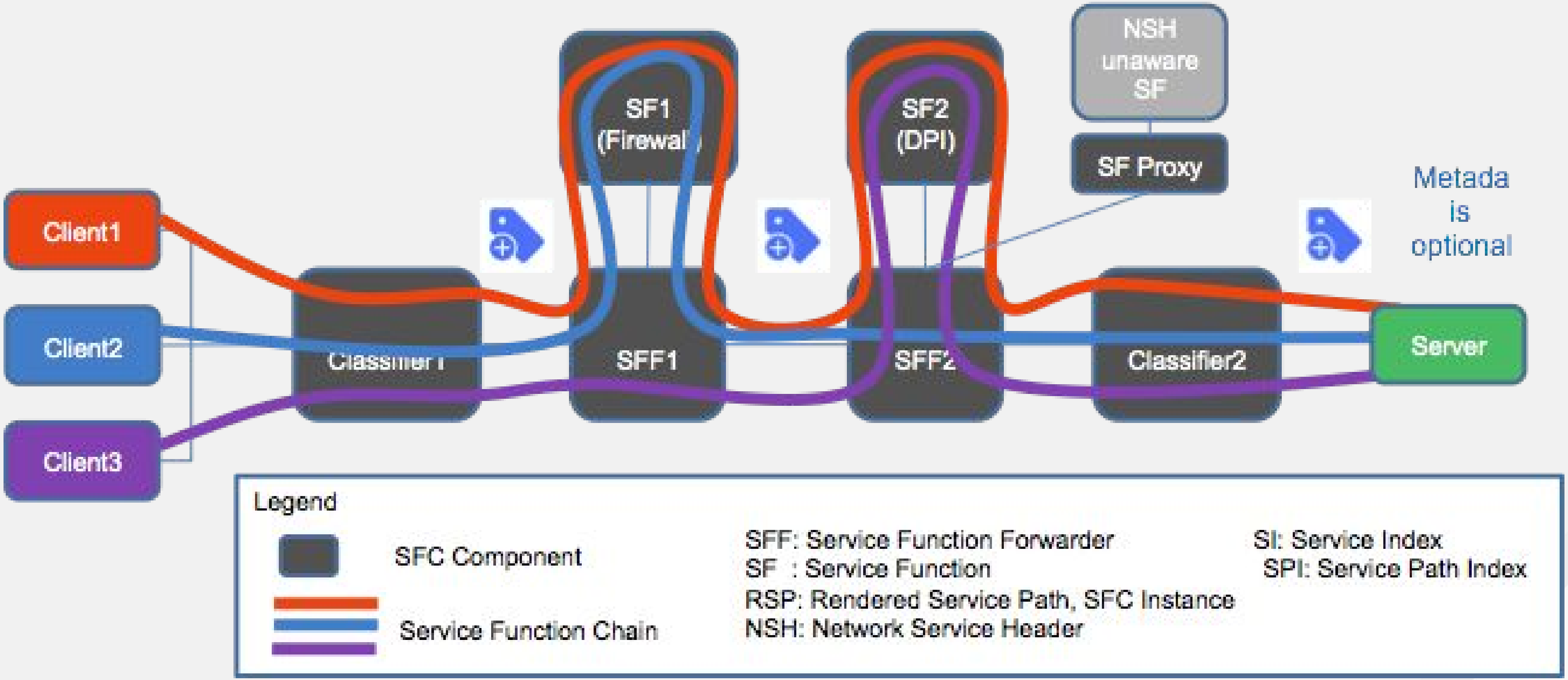
- Installer: Integration of VPP into APEX
- System Test: FuncTest and Yardstick system test application



Ex: Creating a neutron vhost-user port on VPP



3) Service Function Chaining (SFC) introduction



Legend

■ SFC Component

Service Function Chain

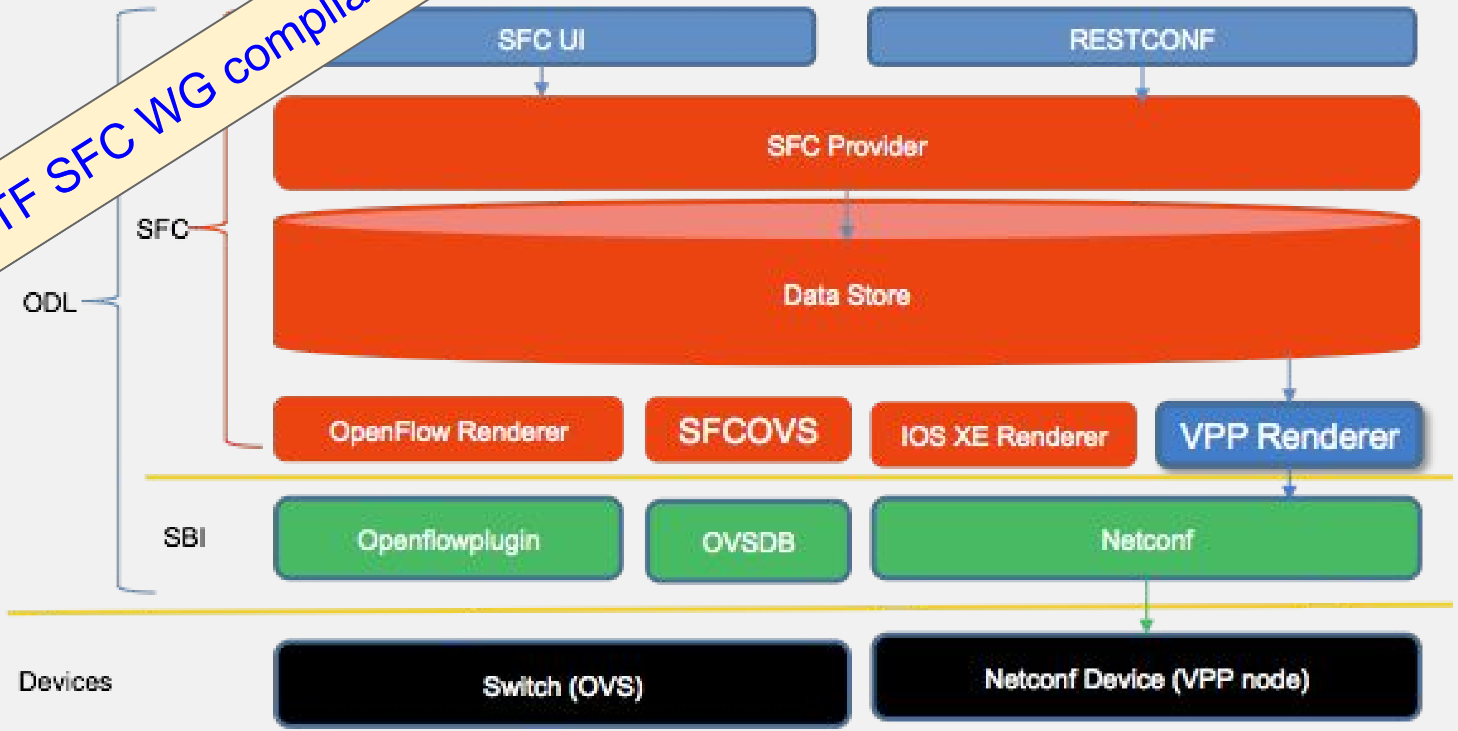
SFF: Service Function Forwarder
SF : Service Function
RSP: Rendered Service Path, SFC Instance
NSH: Network Service Header

SI: Service Index
SPI: Service Path Index

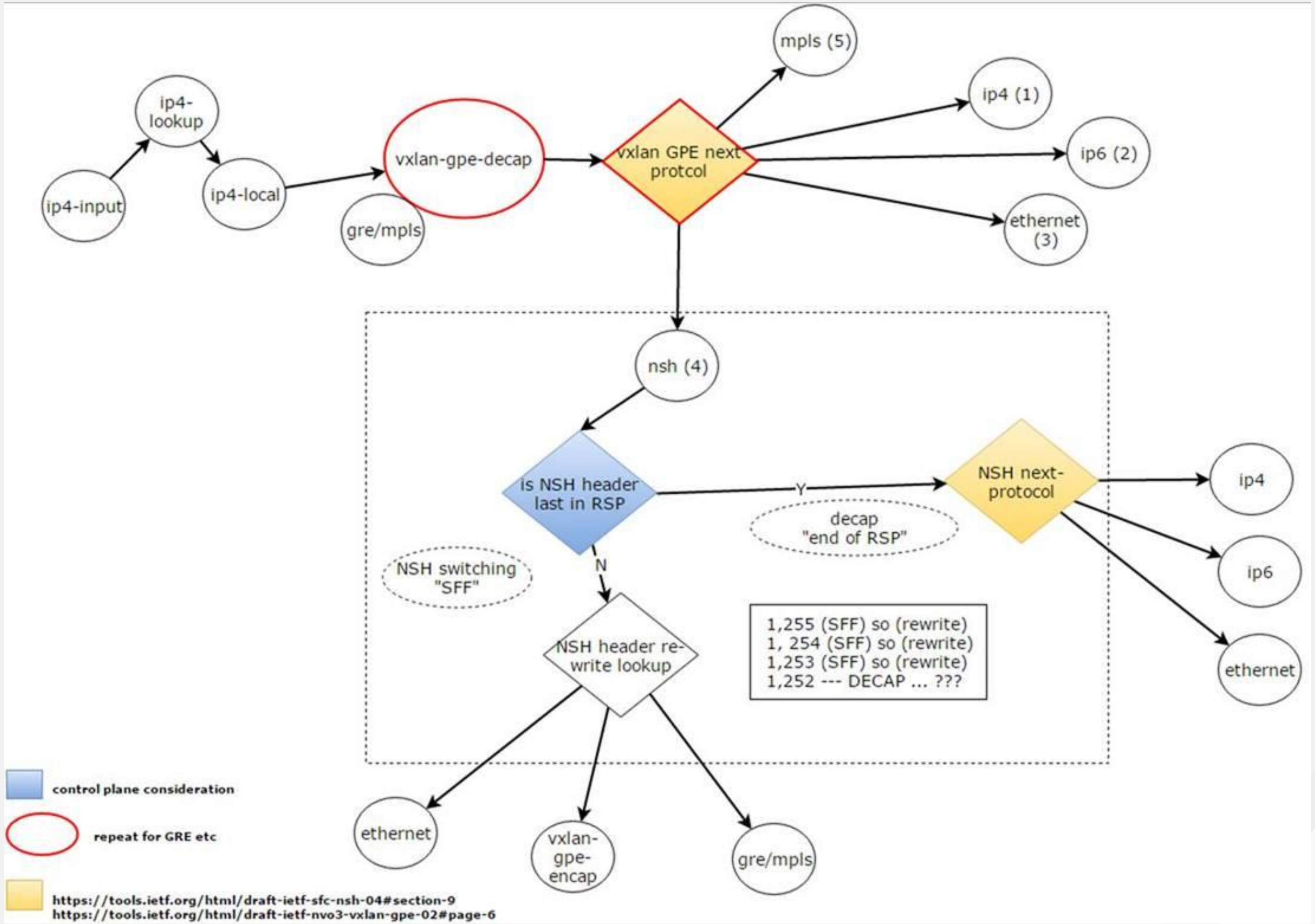
NSH draft: https://datatracker.ietf.org/doc/draft-ietf-sfc-nsh/

VPP renderer in ODL SFC architecture

Fully IETF SFC WG compliant



NSH_SFC



1,255 (SFF) so (rewrite)
1, 254 (SFF) so (rewrite)
1,253 (SFF) so (rewrite)
1,252 --- DECAP ... ???

🟦 **control plane consideration**

○ **repeat for GRE etc**

🟨 **https://tools.ietf.org/html/draft-ietf-sfc-nsh-04#section-9**
https://tools.ietf.org/html/draft-ietf-nvo3-vxlan-gpe-02#page-6

NSH_SFC: 16.09 first release

https://wiki.fd.io/view/NSH_SFC/Releases/1609/ReleasePlan

- SFF functionality
- NSH proxy over SF
- Transport:
  - VXLAN-GPE
  - GRE
- API
  - Automatically generated jar for java bindings

- Packaging
  - rpms/debs
  - apt/yum repo

Innovation: VPP and Containers

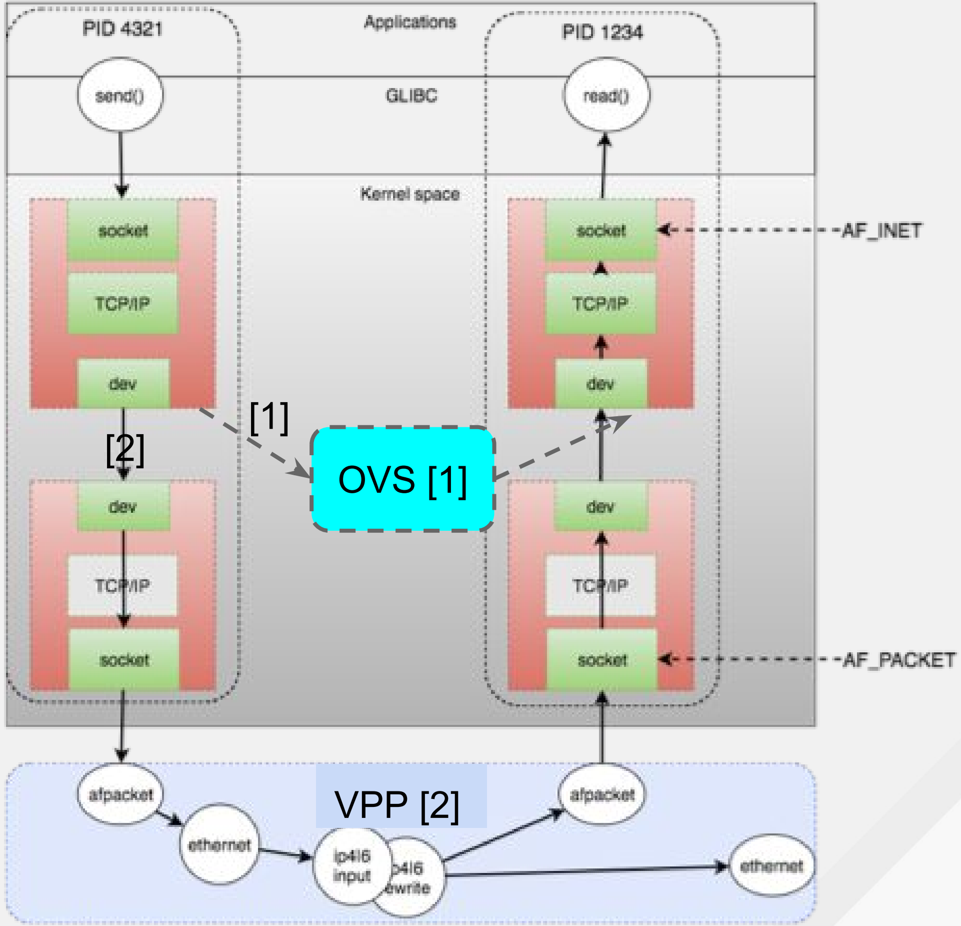
Today’s containers

Non NFV (~no DPDK)

[1] Today’s containers are typically connected by a pair of veth connected to OVS (kernel module)

[2] VPP already permit the same, but in userland [2]

- Functionally fine
- Obviously non optimal, [1] vs [2]
- … but what are the number?
- … and innovation/research in progress (next slide)

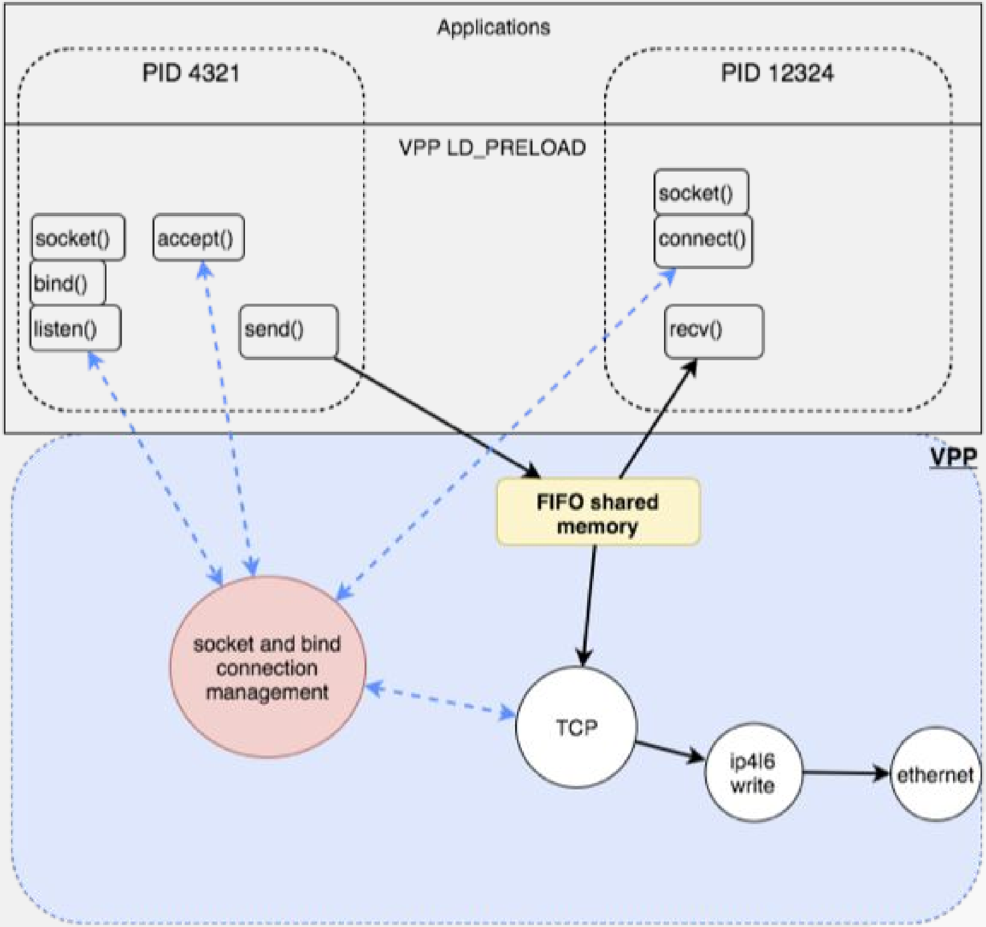


Container/Container TCP acceleration

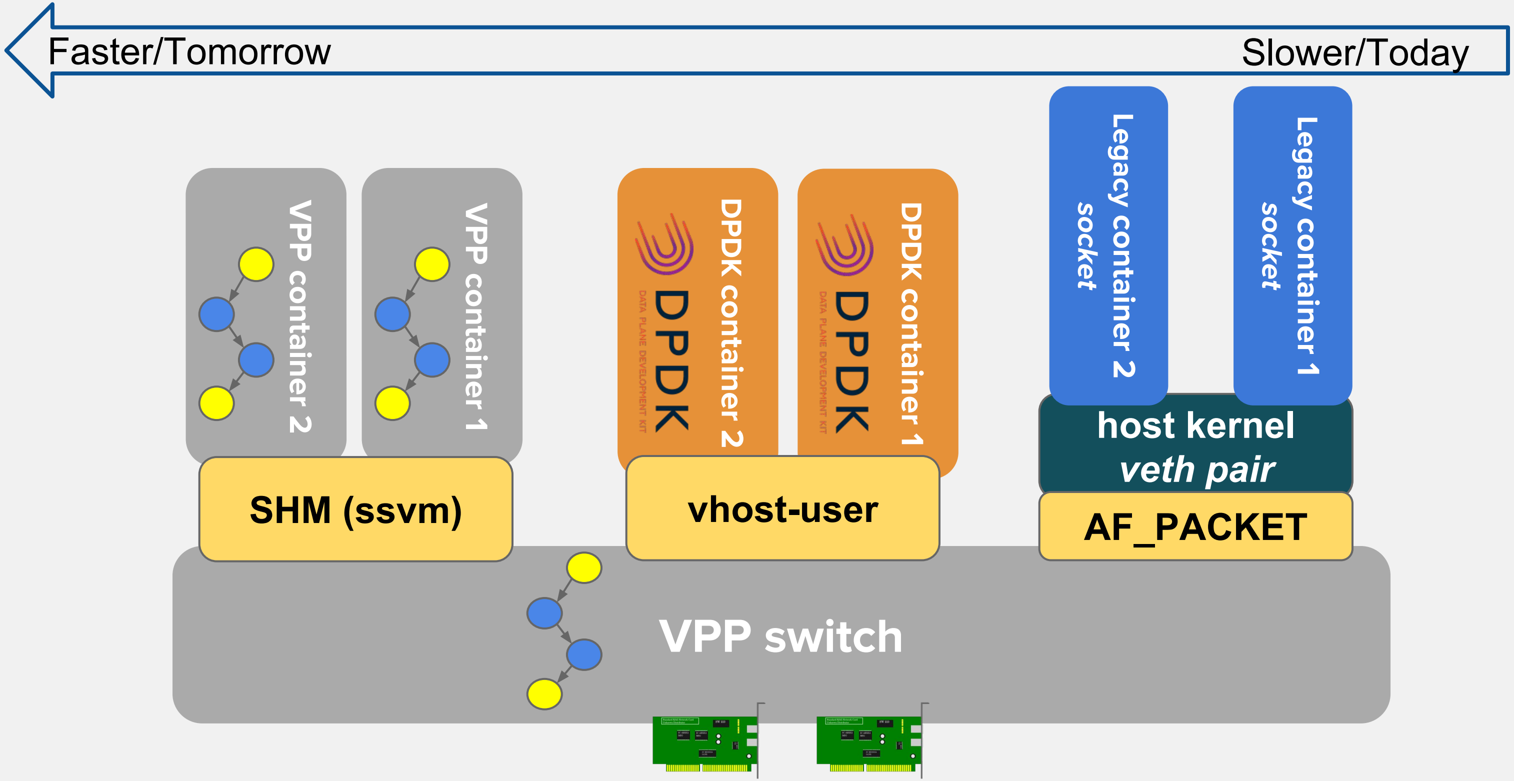
Non directly applicable to NFV

- Assuming that containers/microservices will heavily rely on REST, the Linux kernel TCP stack **may** become the bottleneck
- This approach permit to optimize **transparently** TCP **local** container / container communications
- Requires TLDK for remote communications

VPP provides a proper framework for such researches/innovations… so we can get numbers!

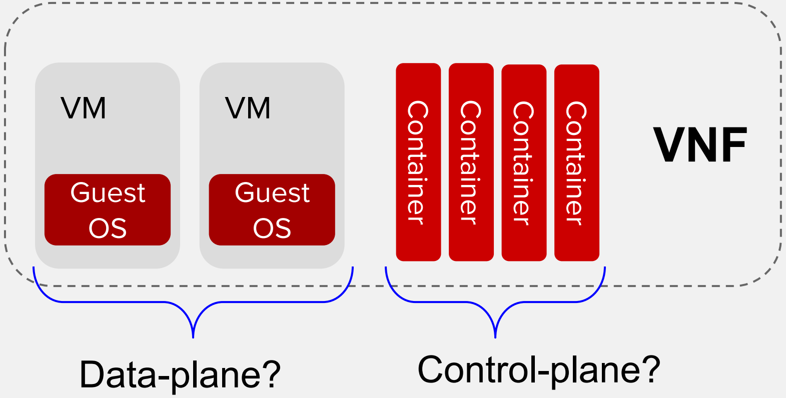


Containerized VNFs



VM or containers for my VNF?

- VPP supports VMs and containers
- Many VNFs are "simply" a 1:1 migration from blade-based PNFs into VMs
- Step wise evolution of such VNFs to containers will lead to hybrid VNFs with both VMs and Containers



Challenges:

- Orchestration
  - Isolation, trusted hosts
  - Cross-host deployment
  - Lifecycle
  - Failure modes, lifecycle
  - ...
- Design patterns
- Networking & service chaining
- ….

Key takeaways

Production vs Innovation

Why choosing?

Production's path

Counters/trace/documentation/training/community/open-weekly-calls/CSIT/Gating-CI

LTS, ABI/API stability: a bit too early…

… but definitely in the community's mind!

OPNFV/RDO integration for PoC: Ocata?

Not all features are there yet…

… but not so many are missing

Innovation's path

Portability

Modularity

Ease of protocol addition

Sandbox project

Containers multiple approaches

Cool stuff already there: NSH, LISP, …

Challenge: find the right balance!

Red Hat and Intel involvement beyond VPP/fd.io Putting all pieces together!

LINUX FOUNDATION

RDO

OPNFV

openstack CLOUD SOFTWARE

OPEN DAYLIGHT

DPDK DATA PLANE DEVELOPMENT KIT

CentOS®

Resources

Several schemas and content of this presentation have been borrowed from https://wiki.fd.io/view/Presentations

FD.io main hub: https://wiki.fd.io/view/Main_Page

FD.IO CSIT: https://wiki.fd.io/view/CSIT

VPP last release test report: https://wiki.fd.io/view/CSIT/VPP-16.09_Test_Report

VPP repos: https://wiki.fd.io/view/VPP/Installing_VPP_binaries_from_packages

VPP user demo: https://git.fd.io/cgit/vppsb/tree/vpp-userdemo/README.md

OpenStack Neutron VPP ML2 plugin: https://github.com/openstack/networking-vpp

OPNFV FDS project: https://wiki.opnfv.org/display/fds/FastDataStacks+Home

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

BREAKOUT SESSIONS - Thursday October 27th

Zuul v3: OpenStack and Ansible Native CI/CD	James Blair	11:00am-11:40am
Container Defense in Depth	Thomas Cameron, Scott McCarty	11:50am-12:30pm
Analyzing Performance in the Cloud : solving an elastic problem with a scientific approach	Alex Krzos, Nicholas Wakou (Dell)	11:50pm-12:30pm
One-stop-shop for OpenStack tools	Ruchika Kharwar	1:50pm-2:30pm
OpenStack troubleshooting: So simple even your kids can do it	Vinny Valdez, Jonathan Jozwiak	1:50pm-2:30pm
Solving Distributed NFV Puzzle with OpenStack and SDN	Rimma Iontel, Fernando Oliveira (VZ), Rajneesh Bajpai (BigSwitch)	2:40pm-3:20pm
Ceph, now and later: our plan for open unified cloud storage	Sage Weil	2:40pm-3:20pm

BREAKOUT SESSIONS - Thursday October 27th

How to configure your cloud to be able to charge your users using official OpenStack components!	Julien Danjou, Stephane Albert (Objectif Libre), Christophe Sauthier (Objectif Libre)	2:40pm-4:10pm
A dice with several faces: Coordinators, mentors and interns on OpenStack Outreach internships	Victoria Martinez de la Cruz, Nisha Yadav (Delhi Tech University), Samuel de Medeiros Queiroz (HPE)	2:40pm-4:10pm
Yo dawg I herd you like Containers, so we put OpenStack and Ceph in Containers	Sean Cohen, Sebastien Han, Federico Lucifredi	3:30pm-4:10pm
Picking an OpenStack Networking solution	Russell Bryant, Gal Sagie (Huawei), Kyle Mestery (IBM)	4:40pm-5:20pm
Forget everything you knew about Swift Rings - here's everything you need to know about Swift Rings	Christian Schwede, Clay Gerrard (Swiftstack)	5:30pm-6:10pm

Annexes

FD.io members

LINUX FOUNDATION

CISCO

intel

ERICSSON

6WIND

HUAWEI

BROCADE

CAVIUM NETWORKS

COMCAST

INOCYBE OPENDAYLIGHT DISTRIBUTION

MESOSPHERE

NXP

PLUMgrid

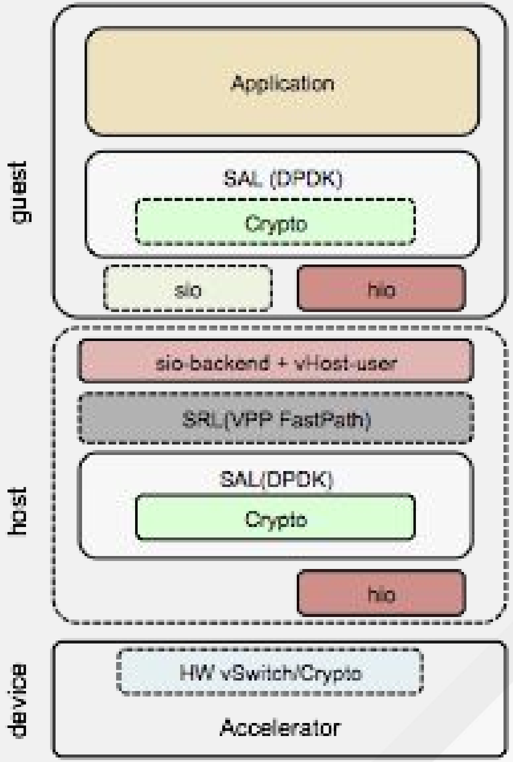
PROJECT CALICO SUPPORTED BY METASWITCH

redhat

OPNFV DPAAC and VPP Design

Software Routing Layer (SRL) can be VPP code for a FastPath design of L2/L3 forwarding

- VPP can also be in the guess for location fastpath support
- VPP can replace the vSwitch at the SRL layer and provide added functionality to the guest or host

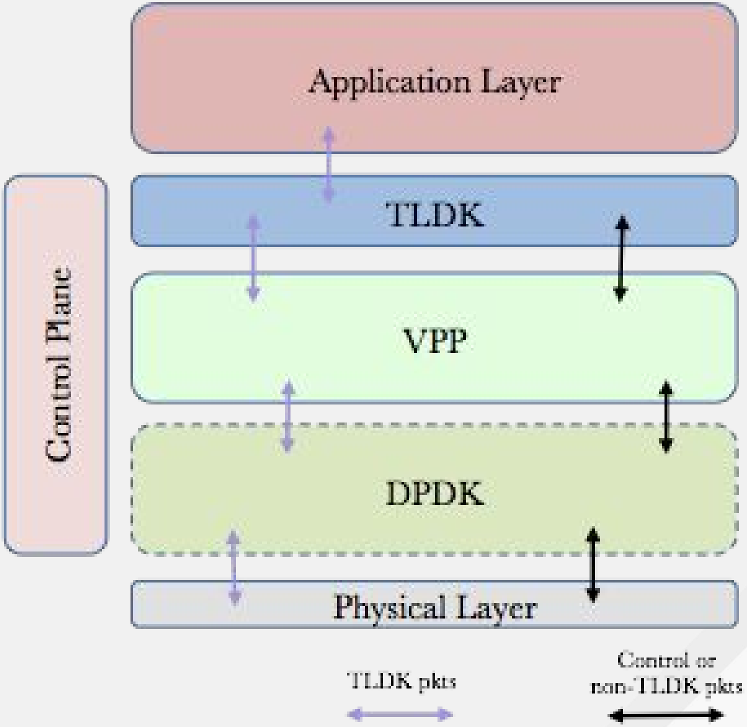


TLDK: Transport Layer Development Kit

- TLDK is not a normal network designed stack!
  - TLDK has turned the network stack upside down for better performance
- Network protocols are driven by the application needing the data
  - Normal network stack designs drive packet into the protocols, then to the application
  - In TLDK the packets are per-filtered to a given DPDK core/thread first
  - The application then drives the packets into the stack when it needs the data not before
- The design attempts to keep the CPU cache warm to reduce wasted cycles
- The goal is to move multiple packets thru the stack at a time, using the vector style packet processing
- Multiple packets at a time allows us to amortize packet processing overhead for higher throughput

TLDK use cases with VPP

- **TLDK:**
  - Handles packet I/O and protocol processing of packets
  - Application sets up the UDP/TCP protocol contexts and then calls I/O routines in TLDK to start processing packets
- **VPP Fastpath:**
  - Using VPP as the first layer for packet processing before packets are sent to the application layer
- **DPDK:**
  - DPDK provides the I/O abstraction to the physical layer for the network devices. The DPDK could be optional here only if some other I/O layer is used.
- **Physical Layer:**
  - Ports and other devices like crypto, compression, …
- **Control Plane:**
  - Not fully defined yet, but will need support in the future


Application Layer
TLDK
VPP
DPDK
Physical Layer
Control Plane
TLDK pkts
Control or non-TLDK pkts