SFC with Nirvana Stack

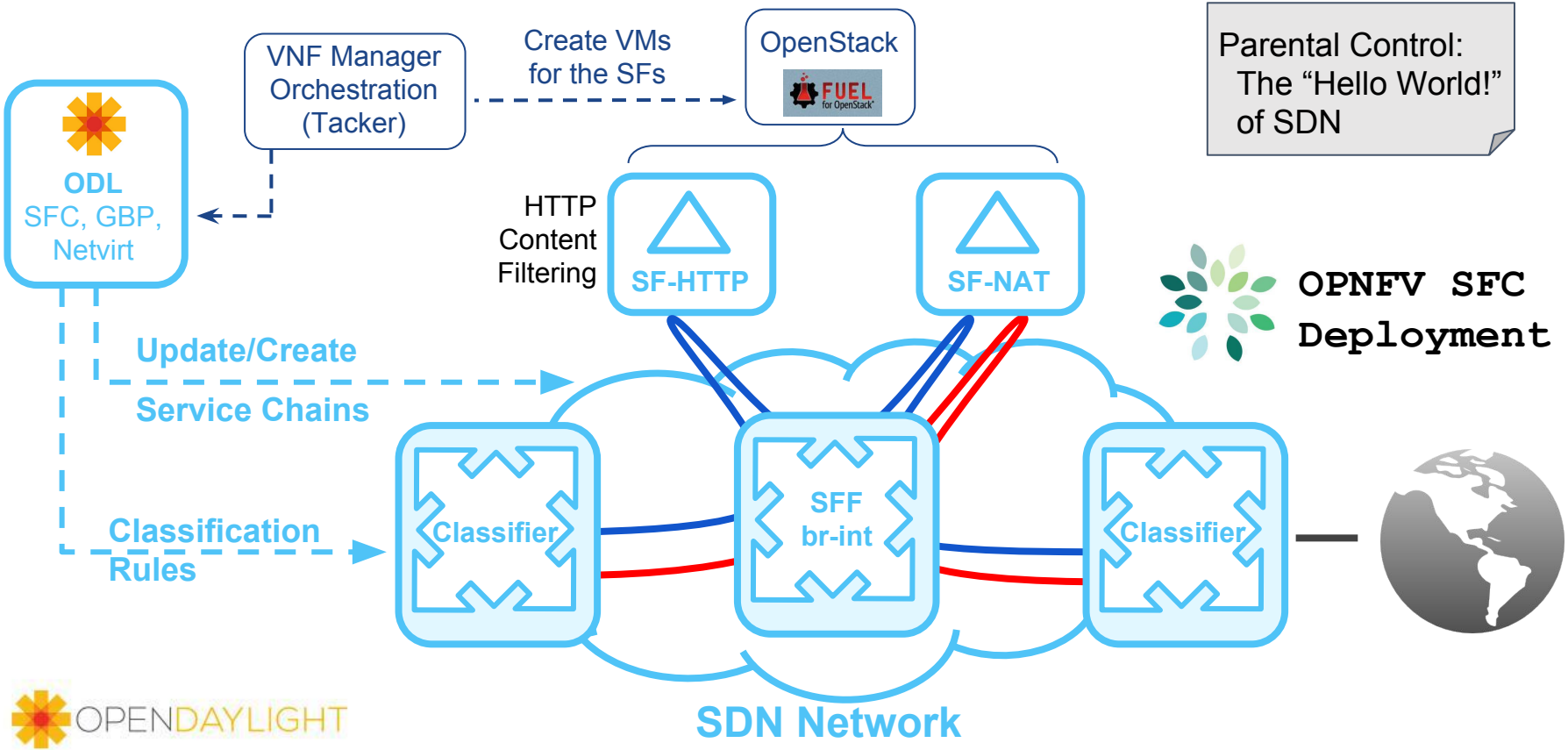
Brady Johnson
brady.allen.johnson@ericsson.com

Danny Zhou
danny.zhou@intel.com

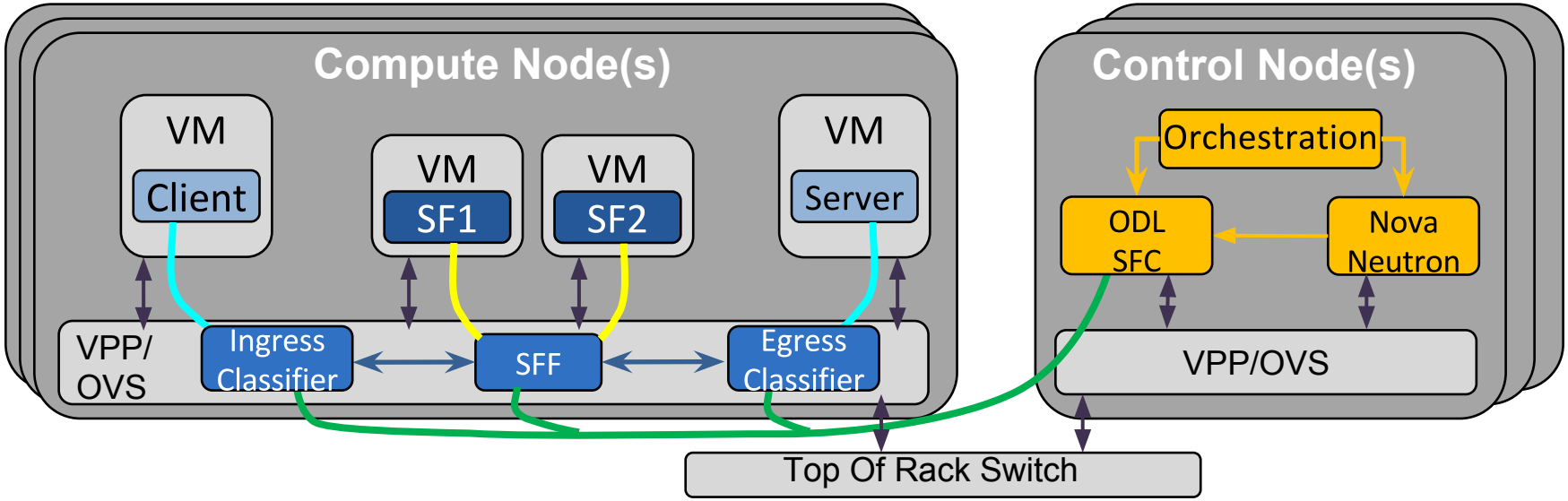
Agenda

- SFC Introduction
- SFC use cases
- SFC status and plans
- SFC integration with VPP

Introducing OPNFV SFC with a simple Use Case: Parental Control



OPNFV SFC deployment illustrated



Legend

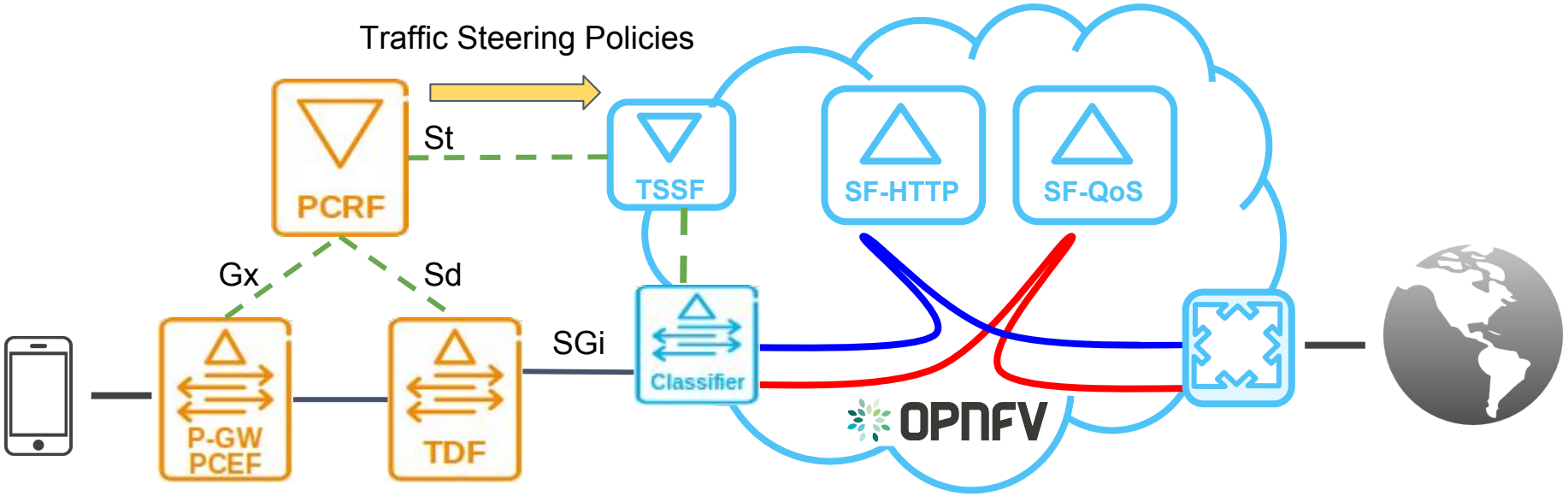
- VxLAN tunnel SF/SFF
- OpenFlow 1.3/OVSDB
- Classifier encaps VxLAN-GPE NSH
- Original packets, no encap

OPENDAYLIGHT

SFC Use Cases

- Service Chaining with GiLAN
- vCPE and Service Chaining
- Inter-Data Center Service Chaining

SFC Use Case: Service Chaining with GiLAN



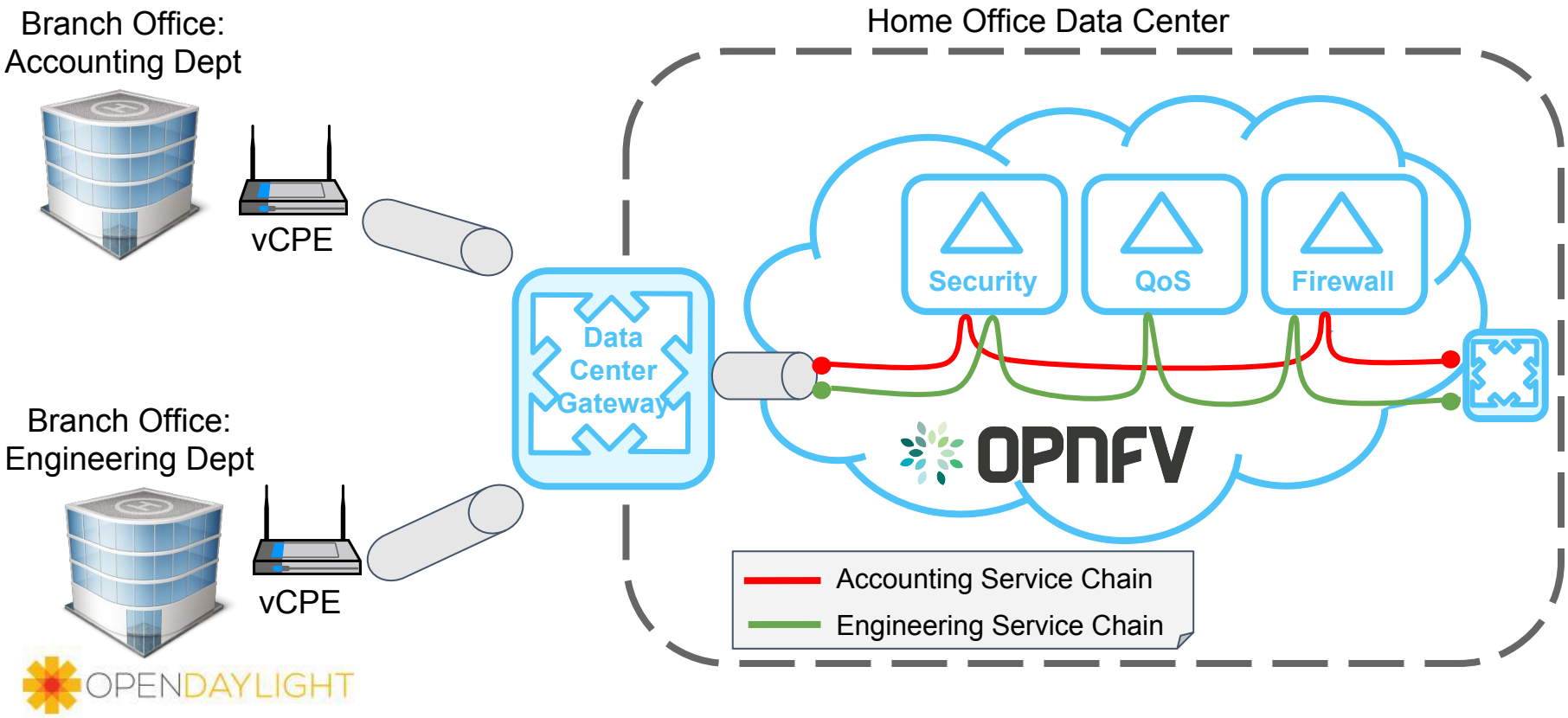
GiLAN nodes

PCEF: Policy and Charging Enforcement Function
PCRF: Policy and Charging Rules Function
TDF: Traffic Detection Function
TSSF: Traffic Steering Support Function

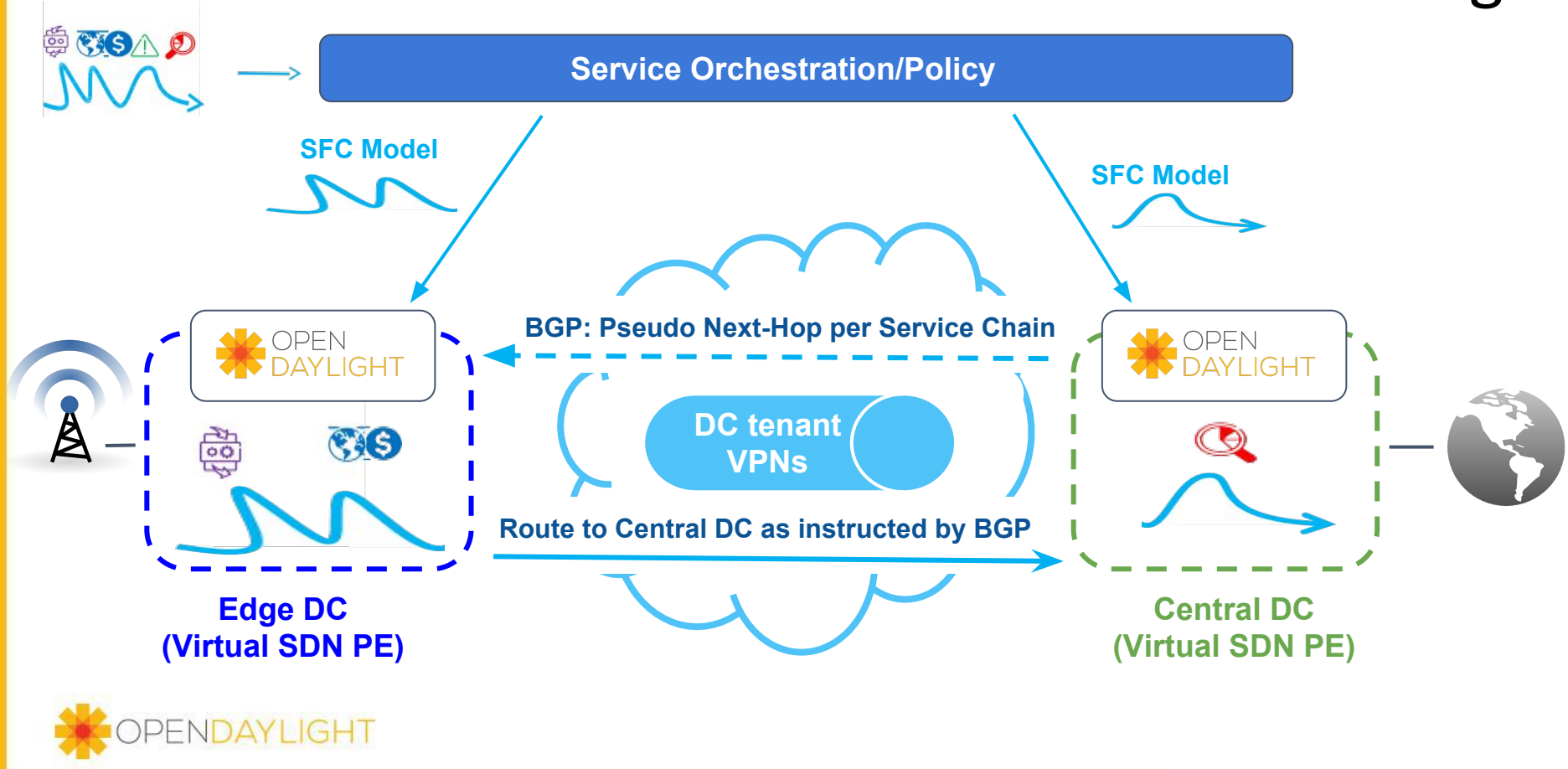
Standard 3GPP Interfaces

Gi: LTE/GGSN Internet Interface
Gx: Policy/Charging rules provisioning
Sd: App. Detection and Control Rules
St: Service Steering Policy Interface

SFC Use Case: vCPE and Service Chaining



SFC Use Case: Inter Data Center Service Chaining



Status and Plans

- Current Status
- Coming in OPNFV SFC Euphrates
- Gaps to work in Nirvana Stack

Current status of OPNFV SFC

OPNFV SFC Danube 2.0 (latest release)

- OPNFV SFC uses OpenDaylight Boron SR2: Netvirt and SFC
- Using a branched version of OVS 2.6 with NSH
  - https://github.com/yyang13/ovs_nsh_patches.git
- Tacker used as VNF Manager and Orchestration
  - Using a branched version of Tacker: https://github.com/trozet/tacker
  - Manages VMs via OpenStack
  - Orchestrates ODL SFC and Netvirt classifier config via ODL RESTconf northbound
- ODL SFC
  - Currently only uses OVS in OPNFV
  - ODL SFC already works with FD.io VPP, but they’re not integrated in OPNFV yet
- ODL Netvirt
  - Currently using Netvirt as the Openstack Neutron backend to manage VM networking
  - Currently using the Netvirt SFC classifier
- Openflow Application Coexistence
  - ODL SFC and Netvirt coordinate Openflow table usage to avoid conflicts

Coming in OPNFV SFC Euphrates

- Integration with OPNFV Fast Data Stacks project
  - OPNFV Fast Data Stacks integrates FD.io VPP with Openstack
  - We started this integration in OPNFV Danube and will finish in Euphrates

- Coming with OpenDaylight Carbon
  - Start using Group Based Policy (GBP)
    - GBP will be used as the SFC classifier
    - GBP also implements Openstack Neutron backend to manage VM networking
  - Continue using Netvirt
    - Switch to the New-Netvirt
  - Start using Genius
    - Provides common set of networking utilities
    - Provide enhanced project integration with improved application coexistence

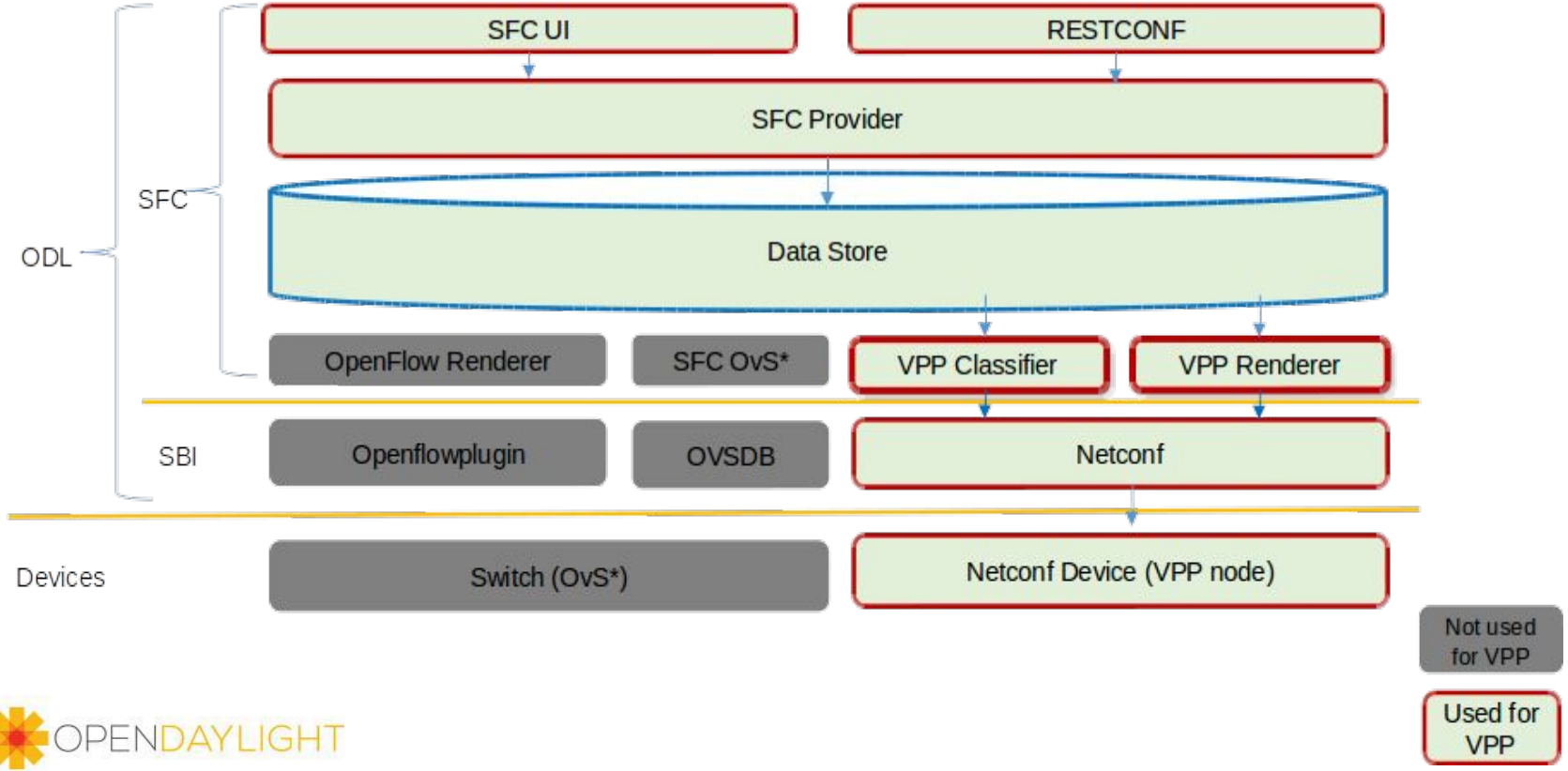
What's left for SFC to work with the Nirvana stack?

- Complete the OPNFV Euphrates integration work
  - Integrate OPNFV SFC with OPNFV Fast Data Stacks
  - Integrate ODL GBP into OPNFV SFC as an additional neutron backend and classifier

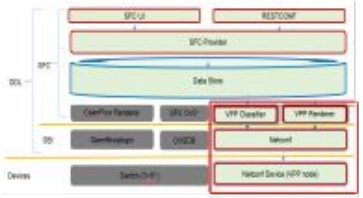
- Add support for additional ODL SFC use cases
  - Advanced Service Function handling
    - Scale in/out, failover, load balancing, bump in the wire
  - BGP and Dynamic routing based Service Chaining
    - https://tools.ietf.org/html/draft-ietf-bess-service-chaining-02#section-2.5

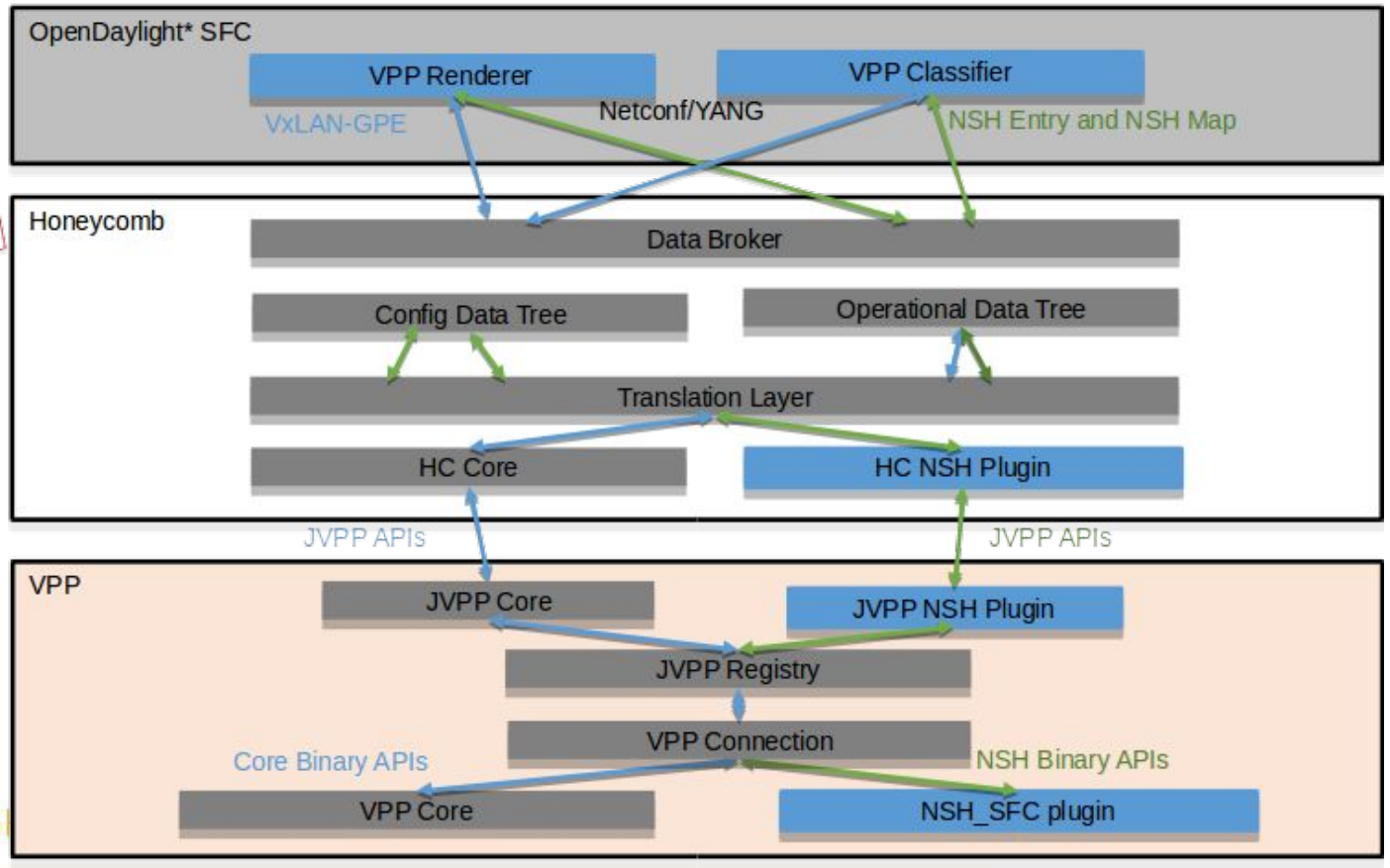
SFC - VPP integration

ODL SFC Integration with VPP



ODL SFC and VPP Integration Architecture



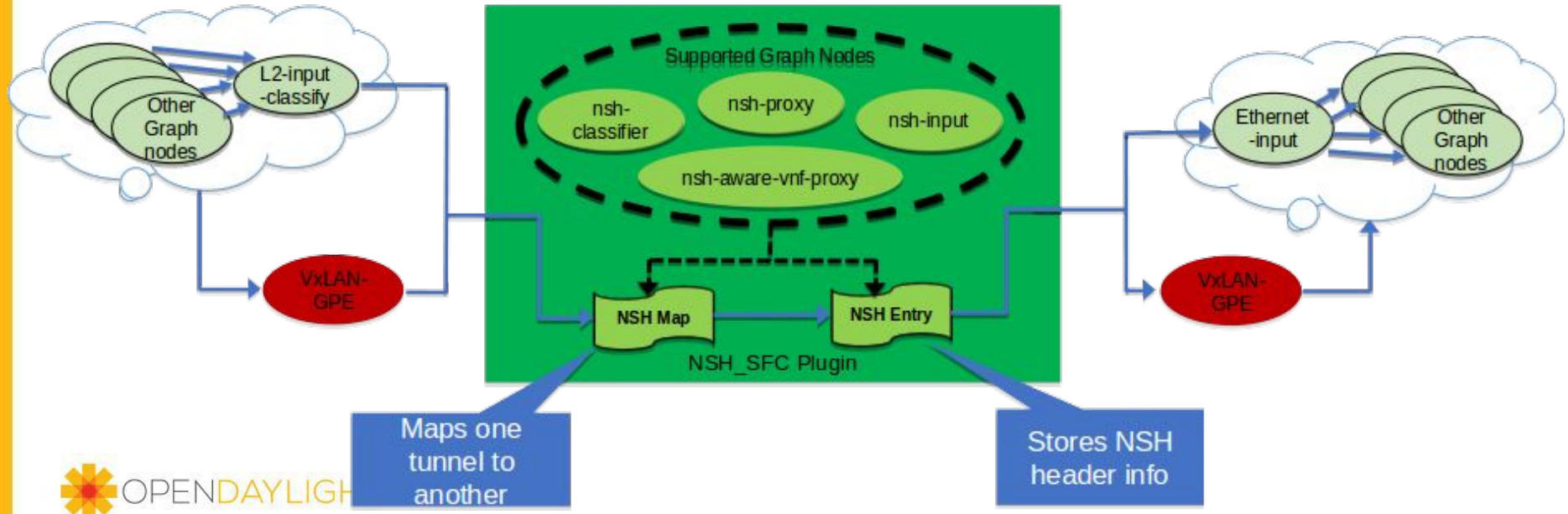


FD.io's NSH_SFC Plugin Internals

- Typical NSH packet processed by VPP and NSH_SFC plugin

Outer MAC header	Outer IP header	UDP header	VxLAN-GPE header	NSH header	Inner MAC header	Inner IP header	L4 header	Payload

- NSH_SFC Plugin



Evolving NSH_SFC Features

16.09

- ✓ Service Function Forwarder
- ✓ Plugin framework in VPP and HC
- ✓ Integration with ODL SFC

17.04

- ✓ NSH-aware SFs by collocating Proxy and SF (e.g. SNAT)
- ✓ Initial MD Type 2 support
- ✓ IOAM over NSH
- ✓ CSIT enabling (functional)

17.01

- ✓ NSH Classifier
- ✓ NSH Proxy
- ✓ Integration test with HC and ODL

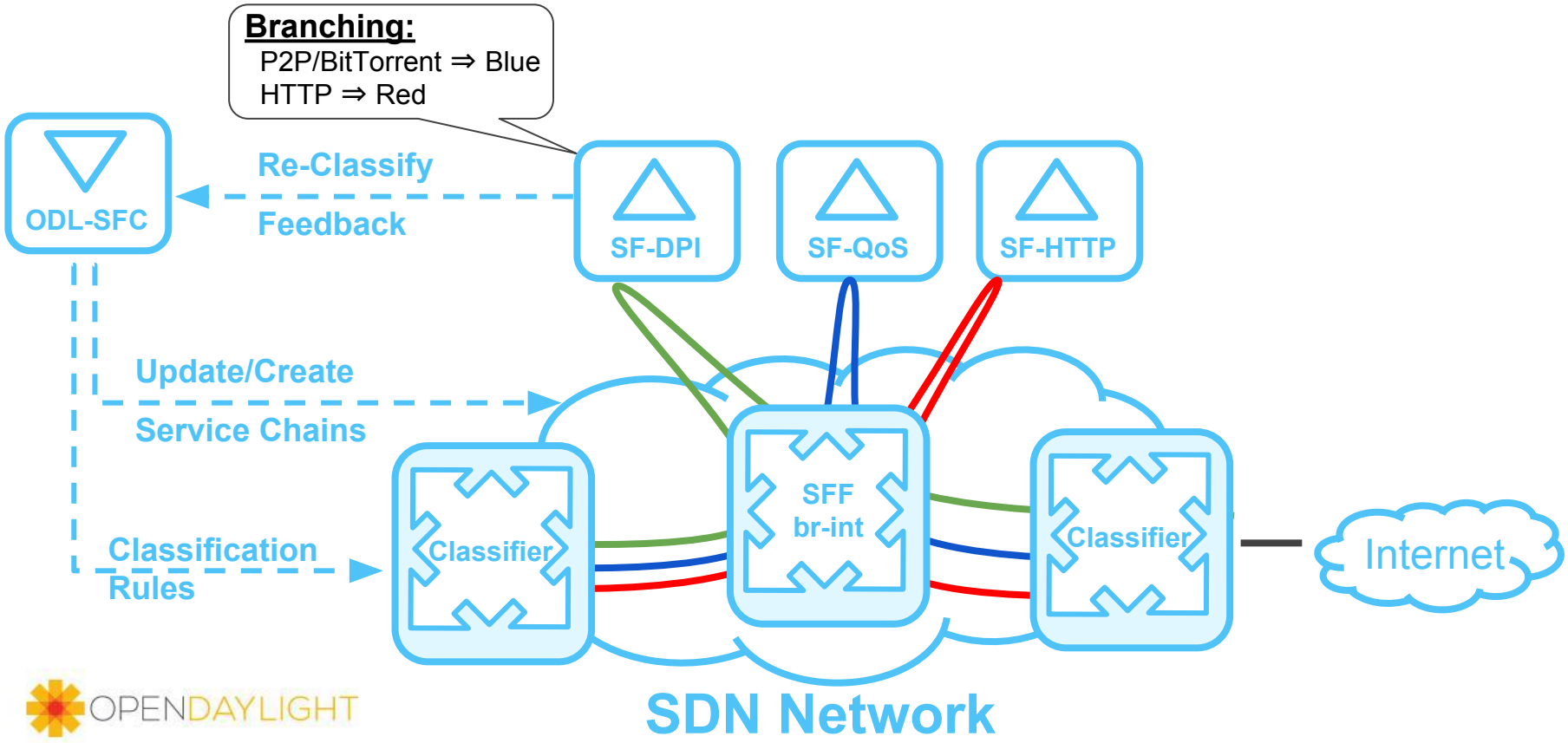
17.07

Under development …….

- ✓ Enable Eth and Geneve as NSH transports
- ✓ Performance optimization
- ✓ Performance automation test
- ✓ Real NSH-aware VNF by enabling NSH_SFC to pass per-packet metadata to real SF (stretch goal)

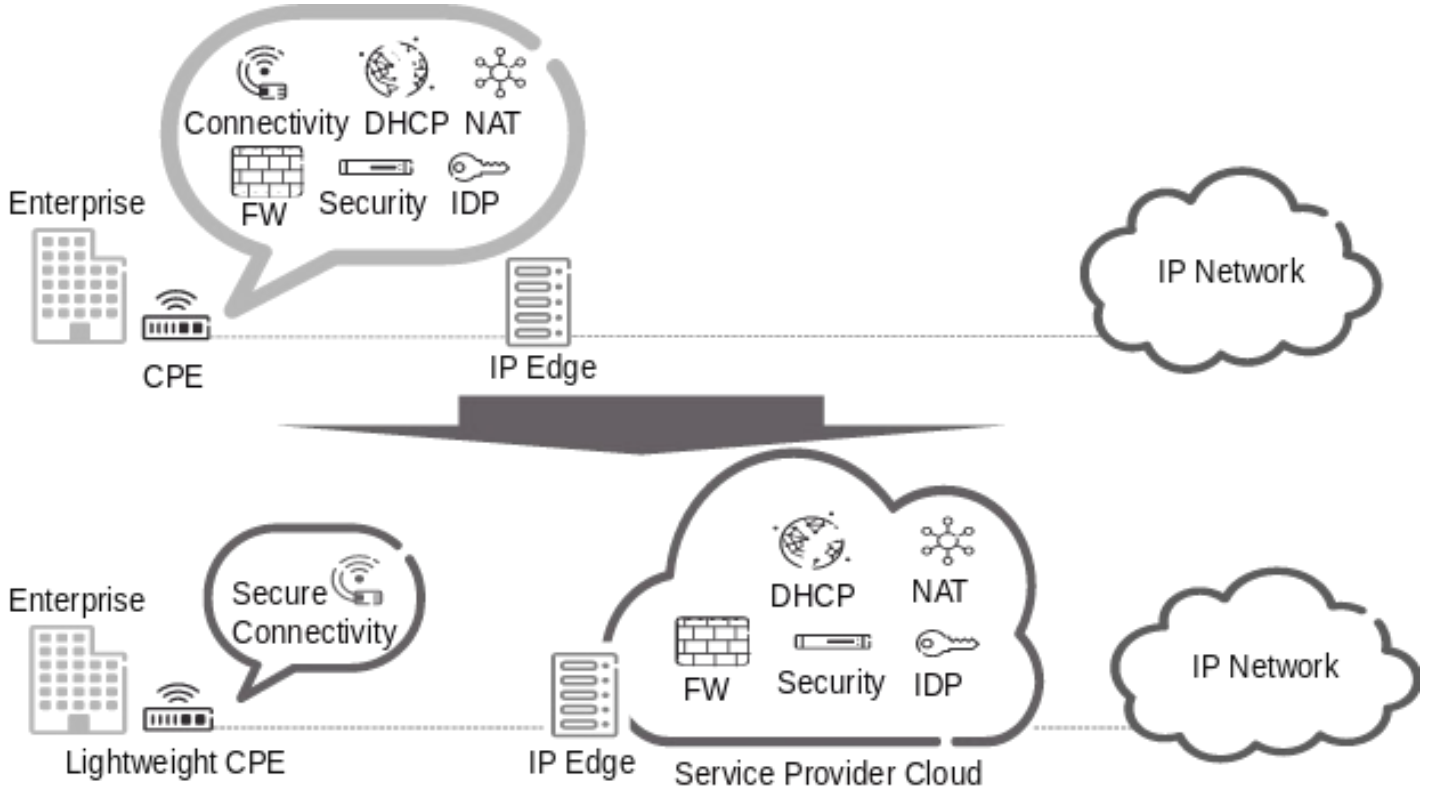
Thank you

Appendix: SFC Use Case: Branching and Reclassification



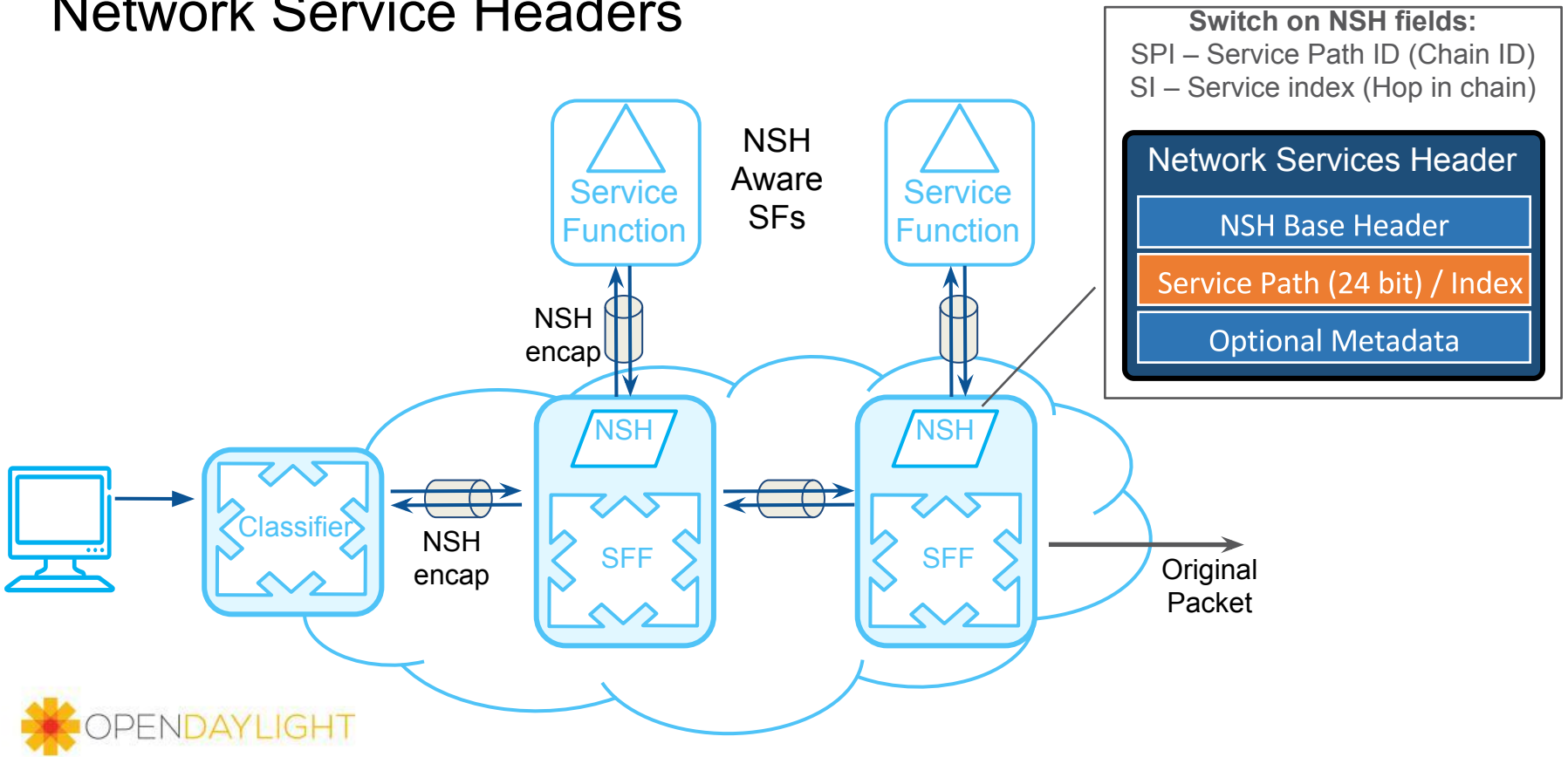
OPENDAYLIGHT

Appendix: vCPE Background

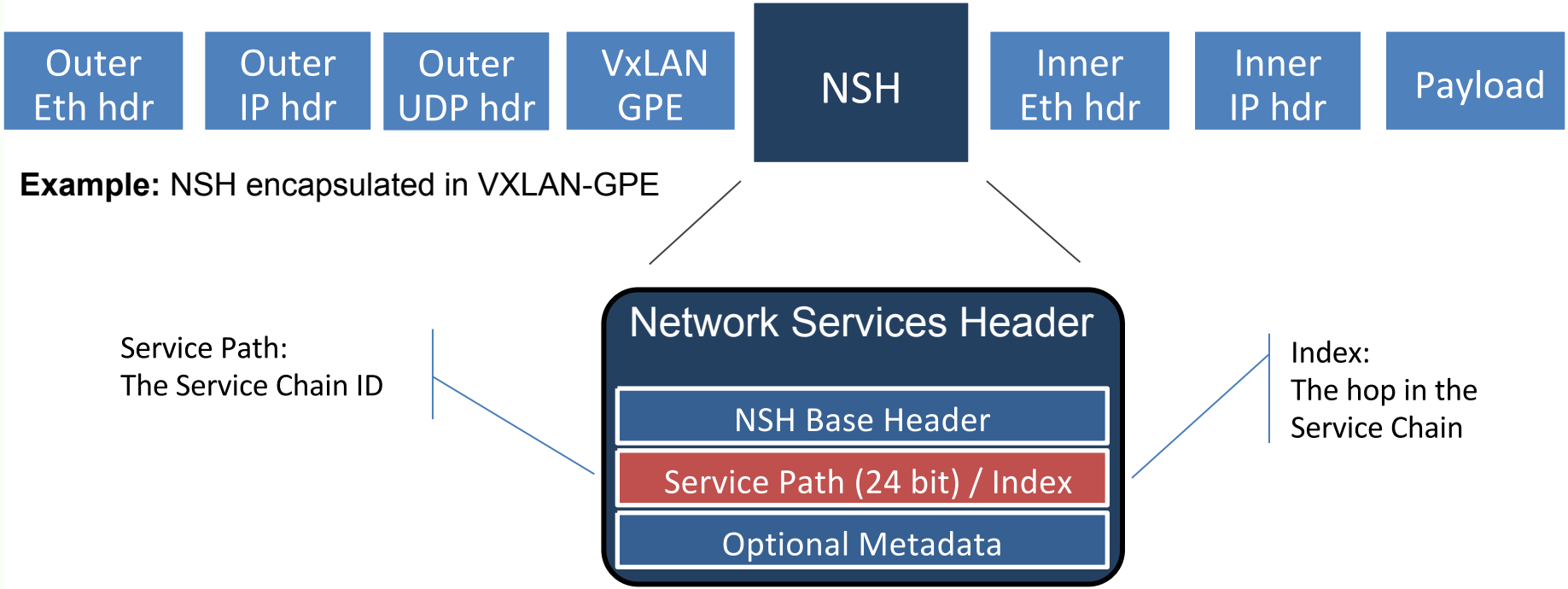


OPENDAYLIGHT

Appendix: Service Chaining Encapsulation
Network Service Headers



Appendix: SFC with NSH



Appendix: SFC classification alternatives

- Mapping Subscriber/Tenant traffic to service chains
  - Match on the subscriber/tenant traffic
    - typically matches on some combination of the packet's 5-tuple
  - Encapsulate packets with NSH Headers that correspond to a particular Service Chain
- Classifiers available in ODL today
  - Group Based Policy
    - Endpoints are grouped together into an EndPoint Group
    - Apply actions between different End Point Groups
    - An action can be to send the packets to SFC
  - Netvirt
    - Primary role is to be an OpenStack Neutron backend, thus configuring all networking for VMs
    - Classification rules can be very easily added to send specific VM traffic to SFC

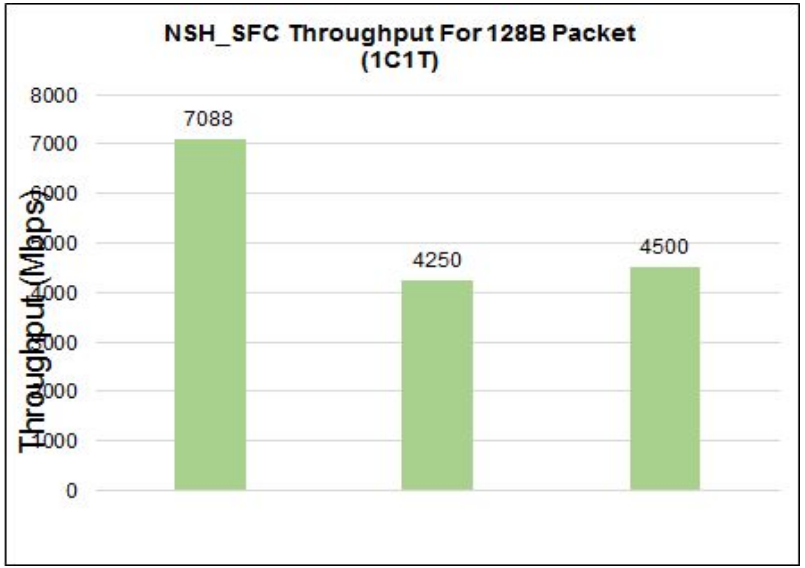
Appendix: Device Under Test

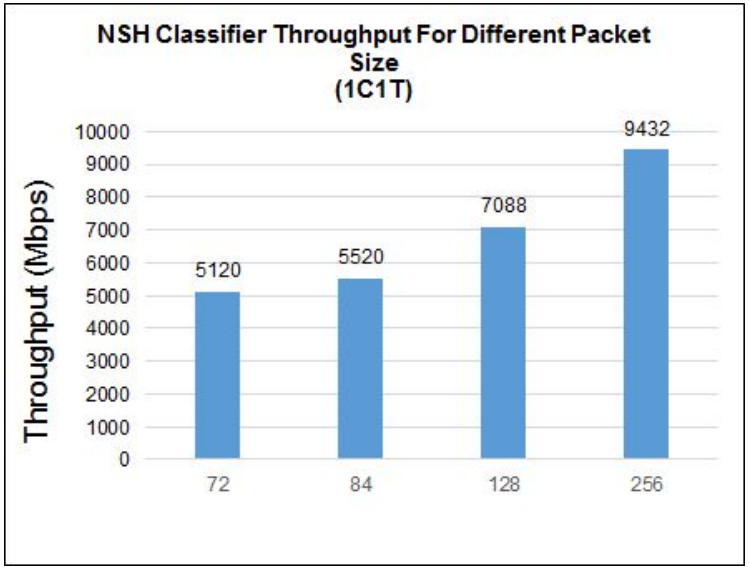
CPU	Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz(Broadwell)
DIMM	2133 MHz, 64GB Total
NIC	2x 82599ES 10-Gigabit SFI/SFP+ Network Connection
PacketGen	Ixia* 10 Gigabit Ethernet Traffic Generator (16 ports)

OS	Ubuntu 16.04 LTS
Kernel	Linux version 4.4.0-21-generic
DPDK	16.11
VPP	17.01
NSH_SFC	17.01
Honeycomb	17.01

Enhanced Intel Speedstep	Enabled
Turbo Boost	Enabled
Processor C3	Disabled
Processor C6	Disabled
Hyper-Threading	Disabled
Intel VT-d	Enabled
CPU Power and Performance Policy	Performance
Memory Freq.	2133 MHz
Total Memory Size	64 GB
Memory RAS and Performance Configuration -> NUMA Optimized	ENABLED
QPI B/W	9.6 GT/s
MLC Streamer	ENABLED
MLC Spatial Prefetcher	ENABLED
DCU Data Prefetcher	ENABLED
DCU Instruction Prefetcher	ENABLED
Direct Cache Access (DCA)	ENABLED

Appendix: NSH_SFC Performance in 17.01 Release

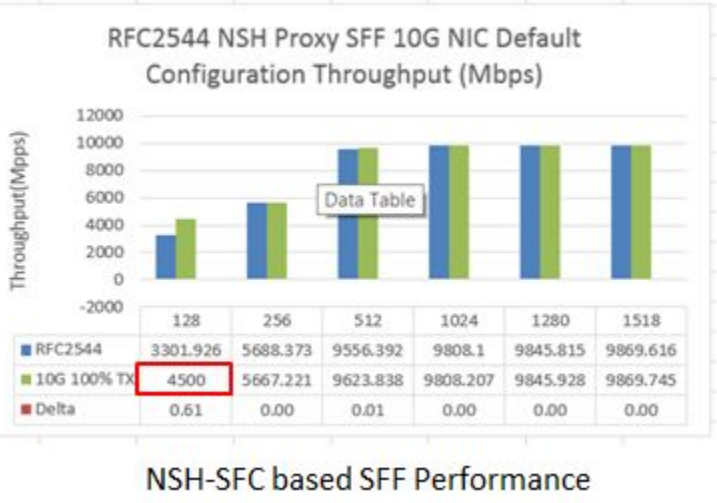


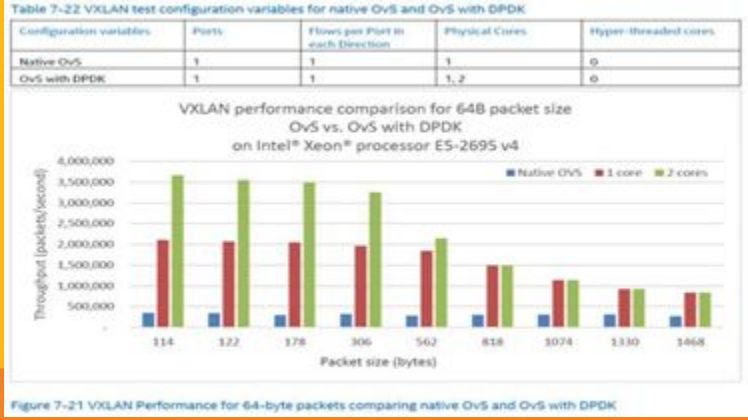


2 to 3 dedicated cores could achieve 10G line rate for small packets for all NSH_SFC use cases

Appendix: NSH_SFC Performance Analysis

VPP Functionality	Test Name	VPP-16.09 [Mbps]	VPP-17.01 [Mbps]	VPP-17.04 [Mbps]	17.01 to 17.04 Relative Change
L2XC	10ge2p1x520: 64B-1t1c-eth-l2xcbase-ndrdisc	9.4	12.7	13.4	6%
L2XC	10ge2p1xl710: 64B-1t1c-eth-l2xcbase-ndrdisc	9.5	12.2	12.4	2%
L2XC dot1ad	10ge2p1x520: 64B-1t1c-dot1ad-l2xcbase-ndrdisc	7.4	8.8	9.3	6%
L2XC dot1q	10ge2p1x520: 64B-1t1c-dot1q-l2xcbase-ndrdisc	7.5	8.8	9.2	5%
L2XC VxLAN	10ge2p1x520: 64B-1t1c-ethip4vxlan-l2xcbase-ndrdisc	5.4	6.5	6.8	5%

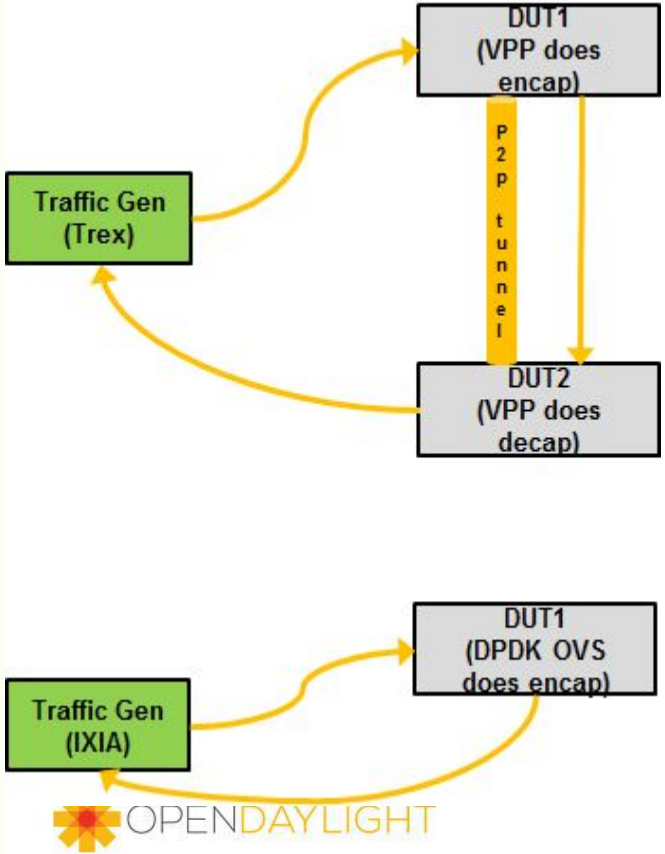




One core performance

- 12.7 drop to 6.5 Mpps caused by VxLAN-GPE encap/decap
- 6.5 Mpps drop to 4.5 Mpps caused by NSH manipulation
- 4.5 Mpps drops to 3.08 Mpps VNF (NSH-aware SNAT) caused by SNAT operation
- VPP's VxLAN performance is about 3x better than DPDK OVS and 13x better than kernel stack

Appendix: VPP and DPDK-netdev OVS VxLAN Test Setup



VPP Functionality	Test Name	VPP-16.09 [Mbps]	VPP-17.01 [Mbps]	VPP-17.04 [Mbps]	17.01 to 17.04 Relative Change
L2XC	10ge2p1x520: 64B-1t1c-eth-l2xcbase-ndrdisc	9.4	12.7	13.4	6%
L2XC	10ge2p1xl710: 64B-1t1c-eth-l2xcbase-ndrdisc	9.5	12.2	12.4	2%
L2XC dot1ad	10ge2p1x520: 64B-1t1c-dot1ad-l2xcbase-ndrdisc	7.4	8.8	9.3	6%
L2XC dot1q	10ge2p1x520: 64B-1t1c-dot1q-l2xcbase-ndrdisc	7.5	8.8	9.2	5%
L2XC VxLAN	10ge2p1x520: 64B-1t1c-ethip4vxlan-l2xcbase-ndrdisc	5.4	**6.5**	6.8	5%