Service Function Chaining (SFC)

Release draft (534a1d1)

OPNFV

February 25, 2016

CHAPTER

ONE

INTRODUCTION

Note: This is the working documentation for the SFC project.

The OPNFV Service Function Chaining (SFC) project aims to provide the ability to define an ordered list of a network services (e.g. firewalls, NAT, QoS). These service are then “stitched” together in the network to create a service chain. This project provides the infrastructure to install the upstream ODL SFC implementation project in an NFV environment.

CHAPTER TWO

DEFINITIONS

Definitions of most terms used here are provided in the IETF SFC Architecture RFC. Additional terms specific to the OPNFV SFC project are defined below.

CHAPTER

THREE

ABBREVIATIONS

Table 3.1: Abbreviations

Abbreviation	Term
NS	Network Service
NFVO	Network Function Virtualization Orchestrator
NF	Network Function
NSH	Network Services Header (Service chaining encapsulation)
ODL	OpenDaylight SDN Controller
RSP	Rendered Service Path
SDN	Software Defined Networking
SF	Service Function
SFC	Service Function Chain(ing)
SFF	Service Function Forwarder
SFP	Service Function Path
VNF	Virtual Network Function
VNFM	Virtual Network Function Manager
VNF-FG	Virtual Network Function Forwarding Graph
VIM	Virtual Infrastructure Manager

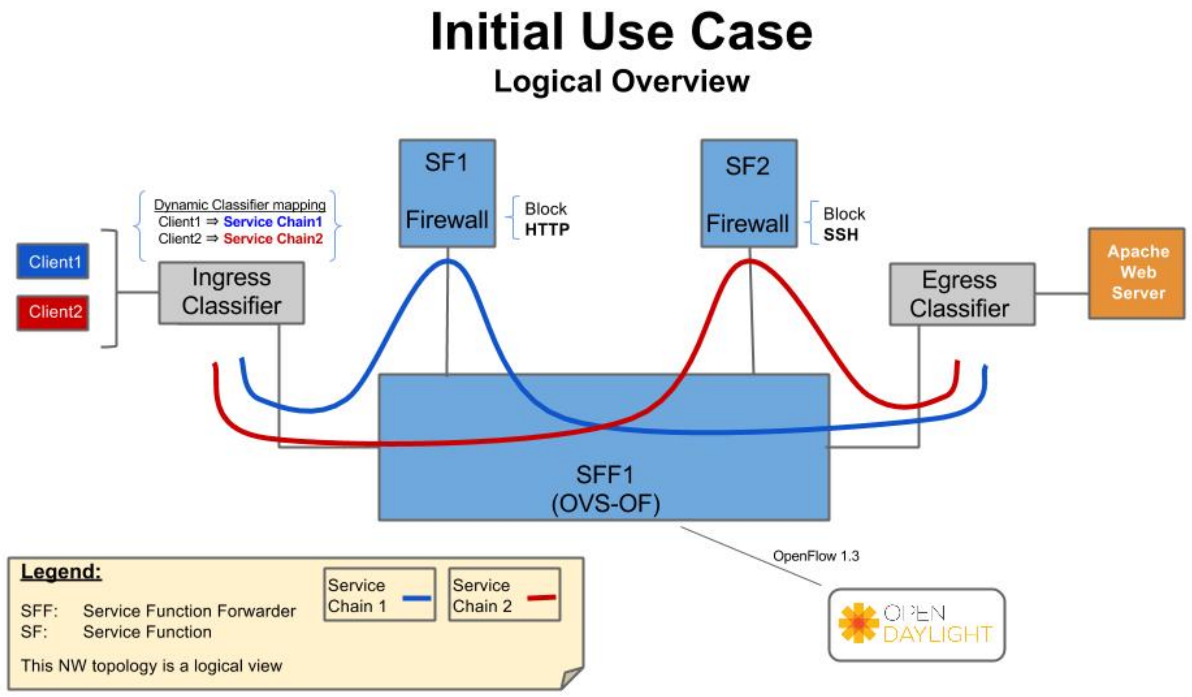
CHAPTER

FOUR

USE CASES

This section outlines the Brahmaputra use cases driving the initial OPNFV SFC implementation.

The use cases targeted in the OPNFV SFC Brahmaputra release focus on creating simple Service Chains using Firewall Service Functions. As can be seen in the following diagram, 2 service chains are created, each through a different Service Function Firewall. Service Chain 1 will block HTTP, while Service Chain 2 will block SSH.



CHAPTER FIVE

ARCHITECTURE

This section describes the architectural approach to incorporating the upstream OpenDaylight (ODL) SFC project into the OPNFV Brahmaputra platform.

5.1 Service Functions

A Service Function (SF) is a Function that provides services to flows traversing a Service Chain. Examples of typical SFs include: Firewall, NAT, QoS, and DPI. In the context of OPNFV, the SF will be a Virtual Network Function. The SFs receive data packets from a Service Function Forwarder.

5.2 Service Function Forwarders

The Service Function Forwarder (SFF) is the core element used in Service Chaining. It is an OpenFlow switch that, in the context of OPNFV, is hosted in an OVS bridge. In OPNFV there will be one SFF per Compute Node that will be hosted in the “br-int” OpenStack OVS bridge.

The responsibility of the SFF is to steer incoming packets to the corresponding Service Function, or to the SFF in the next compute node. The flows in the SFF are programmed by the OpenDaylight SFC SDN Controller.

5.3 Service Chains

Service Chains are defined in the OpenDaylight SFC Controller using the following constructs:

SFC A Service Function Chain (SFC) is an ordered list of abstract SF types.

SFP A Service Function Path (SFP) references an SFC, and optionally provides concrete information about the SFC, like concrete SF instances. If SF instances are not supplied, then the RSP will choose them.

RSP A Rendered Service Path (RSP) is the actual Service Chain. An RSP references an SFP, and effectively merges the information from the SFP and SFC to create the Service Chain. If concrete SF details were not provided in the SFP, then SF selection algorithms are used to choose one. When the RSP is created, the OpenFlows will be programmed and written to the SFF(s).

5.4 Service Chaining Encapsulation

Service Chaining Encapsulation encapsulates traffic sent through the Service Chaining domain to facilitate easier steering of packets through Service Chains. If no Service Chaining Encapsulation is used, then packets much be classified at every hop of the chain, which would be slow and would not scale well.

In ODL SFC, Network Service Headers (NSH) is used for Service Chaining encapsulation. NSH is an IETF specification that uses 2 main header fields to facilitate packet steering, namely:

NSP (NSH Path) The NSP is the Service Path ID.

NSI (NSH Index) The NSI is the Hop in the Service Chain. The NSI starts at 255 and is decremented by every SF. If the NSI reaches 0, then the packet is dropped which avoids loop detections.

NSH also has metadata fields, but that's beyond the scope of this architecture.

In ODL SFC, NSH packets are encapsulated in VXLAN-GPE.

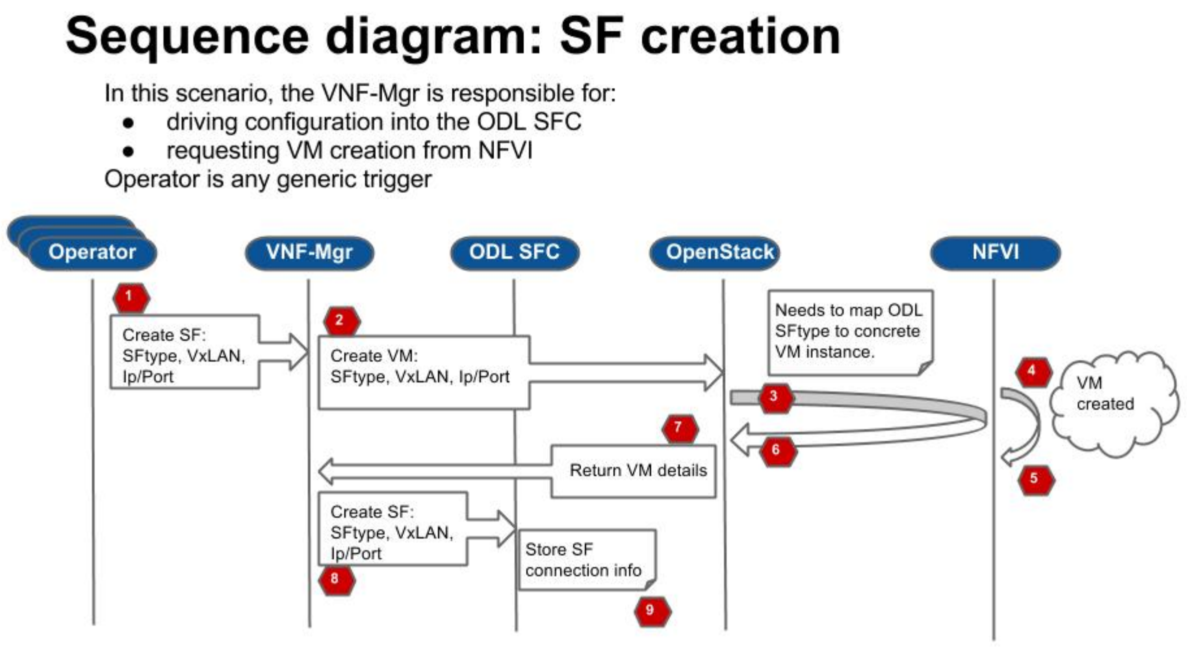
5.5 Classifiers

A classifier is the entry point into Service Chaining. The role of the classifier is to map incoming traffic to Service Chains. In ODL SFC, this mapping is performed by matching the packets and encapsulating the packets in a VXLAN-GPE NSH tunnel.

The packet matching is specific to the classifier implementation, but can be as simple as an ACL, or can be more complex by using PCRF information or DPI.

5.6 VNF Manager

In OPNFV SFC, a VNF Manager is needed to spin-up VMs for Service Functions. It has been decided to use the OpenStack Tacker VNF Mgr to spin-up and manage the life cylcle of the SFs. Tacker will receive the ODL SFC configuration, manage the SF VMs, and forward the configuration to ODL SFC. The following sequence diagram details the interactions with the VNF Mgr:



Sequence diagram: SF creation

In this scenario, the VNF-Mgr is responsible for:

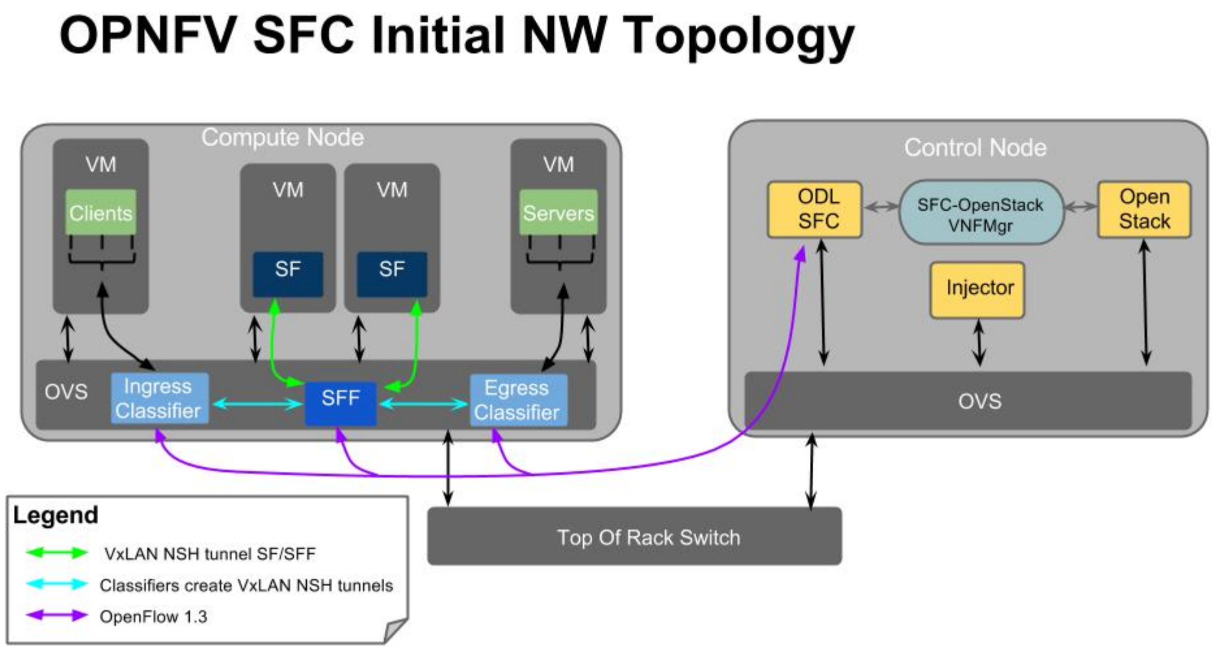
- driving configuration into the ODL SFC
- requesting VM creation from NFVI

Operator is any generic trigger

Operator
VNF-Mgr
ODL SFC
OpenStack
NFVI
1
Create SF: SFtype, VxLAN, Ip/Port
2
Create VM: SFtype, VxLAN, Ip/Port
Needs to map ODL SFtype to concrete VM instance.
3
4
VM created
5
6
7
Return VM details
Create SF: SFtype, VxLAN, Ip/Port
8
Store SF connection info
9

5.7 OPNFV SFC Network Topology

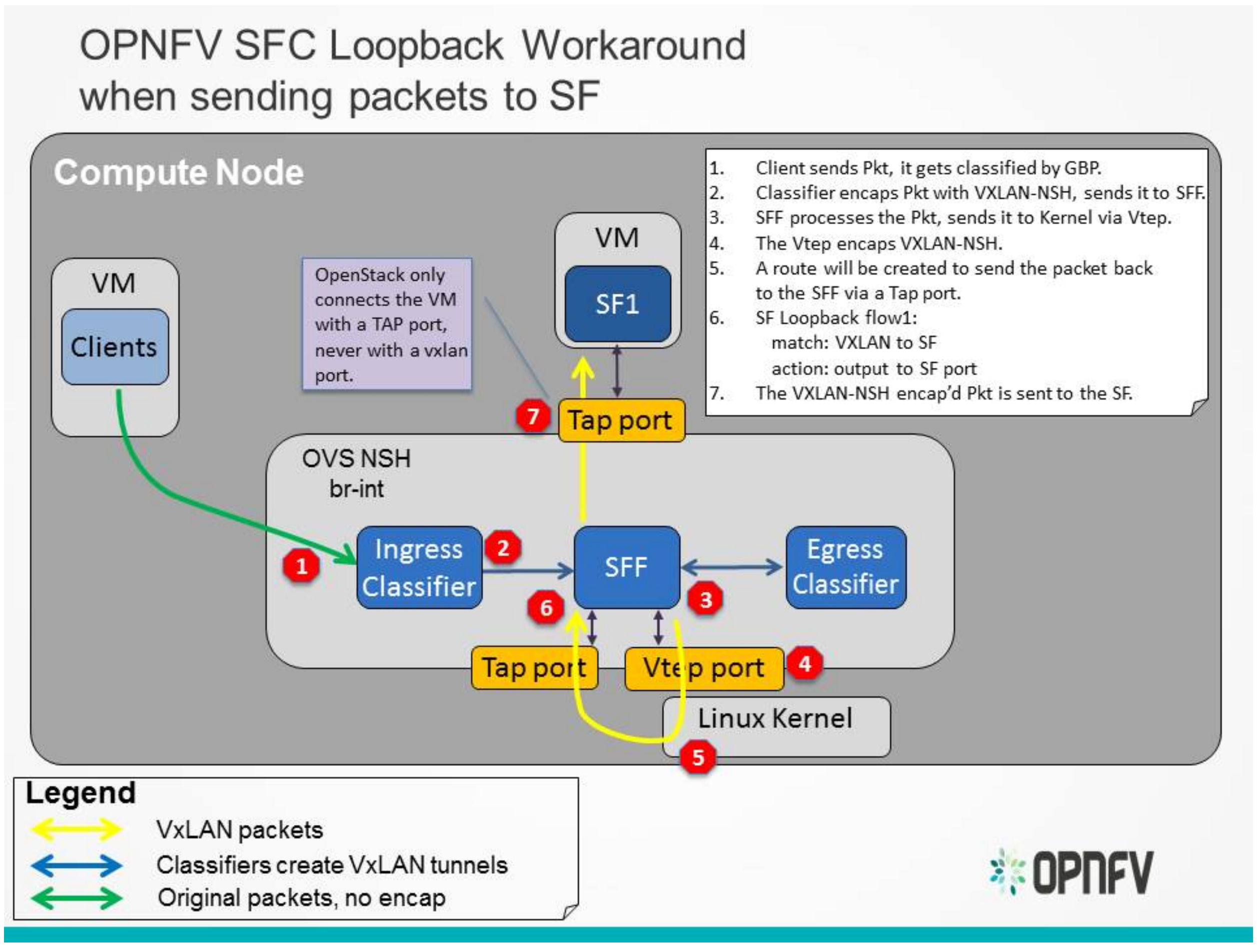
The following image details the Network Topology used in OPNFV Brahmaputra SFC:



5.8 OVS NSH patch workaround

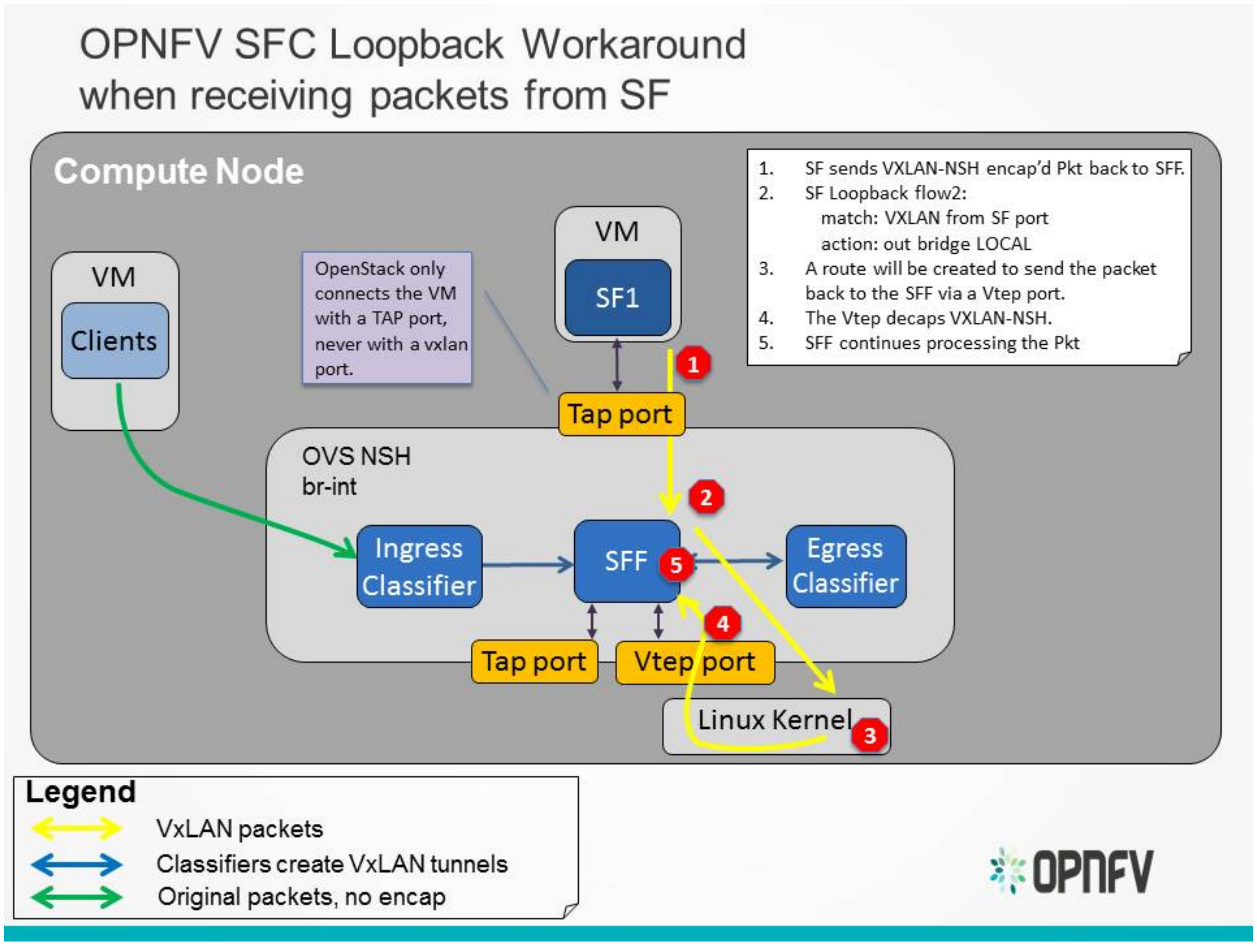
When using NSH with VXLAN tunnels, its important that the VXLAN tunnel is terminated in the SF VM. This allows the SF to see the NSH header, allowing it to decrement the NSI and also to use the NSH metadata. When using VXLAN with OpenStack, the tunnels are not terminated in the VM, but in the “br-int” OVS bridge. There is work ongoing in the upstream OVS community to implemement NSH encapsulation. To get around the way OpenStack handles VXLAN tunnels, the OVS work will also include the ability to encapsulate/decapsulate VXLAN tunnels from OpenFlow rules, instead of relying on the Vtep ports. The ongoing upstream OVS work will probably not be finished by the time OPNFV Brahmaputra is released, so a work-around has been created. This work-around will use a private branch of OVS that has a preliminary version of NSH implemented.

The following diagram illustrates how packets will be sent to an SF, when the SFF has processed the packet and wants to send it to the SF:


OPNFV SFC Loopback Workaround
when sending packets to SF
Compute Node
VM
Clients
OpenStack only connects the VM with a TAP port, never with a vxlan port.
VM
SF1
Tap port
OVS NSH
br-int
Ingress Classifier
SFF
Egress Classifier
Tap port
Vtep port
Linux Kernel
1
2
3
4
5
6
7
Legend
VxLAN packets
Classifiers create VxLAN tunnels
Original packets, no encap
OPNFV

1. Client sends Pkt, it gets classified by GBP.
2. Classifier encaps Pkt with VXLAN-NSH, sends it to SFF.
3. SFF processes the Pkt, sends it to Kernel via Vtep.
4. The Vtep encaps VXLAN-NSH.
5. A route will be created to send the packet back to the SFF via a Tap port.
6. SF Loopback flow1:
   match: VXLAN to SF
   action: output to SF port
7. The VXLAN-NSH encap'd Pkt is sent to the SF.

The following diagram illustrates how packets will sent from an SF to an SFF, once the SF has processed a packet:



OPNFV SFC Loopback Workaround when receiving packets from SF

1. SF sends VXLAN-NSH encap'd Pkt back to SFF.
2. SF Loopback flow2:
   match: VXLAN from SF port
   action: out bridge LOCAL
3. A route will be created to send the packet back to the SFF via a Vtep port.
4. The Vtep decaps VXLAN-NSH.
5. SFF continues processing the Pkt

OpenStack only connects the VM with a TAP port, never with a vxlan port.

CHAPTER

SIX

REQUIREMENTS

This section defines requirements for the initial OPNFV SFC implementation, including those requirements driving upstream project enhancements.

6.1 Minimal Viable Requirement

Deploy a complete SFC solution by integrating OpenDaylight SFC with OpenStack in an OPNFV environment.

6.2 Detailed Requirements

These are the Brahmaputra specific requirements:

1 Placement of SFs on only one Compute node will be supported.

2 The supported Service Chaining encapsulation will be NSH VXLAN-GPE.

3 The version of OVS used must support NSH.

4 The SF VM life cycle will be managed by the Tacker VNF Mgr.

5 The supported classifiers will be either ODL Netvirt or ODL GBP.

6 ODL will be the OpenStack Neutron backend and will handle all networking on the compute nodes.

6.3 Long Term Requirements

These requirements are out of the scope of the Brahmaputra release.

1 Placing SFs on multiple Compute nodes.

2 Dynamic movement of SFs across multiple Compute nodes.

3 Load Balancing across multiple SFs