

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: June 7, 2019

R. Fernando  
Cisco Systems  
S. Mackie  
Juniper Networks  
D. Rao  
Cisco Systems  
B. Rijsman

M. Napierala  
ATT Labs  
T. Morin  
Orange  
December 4, 2018

Service Function Chaining using Virtual Networks with BGP VPNs  
draft-ietf-bess-service-chaining-06

Abstract

This document describes how service function chains (SFC) can be applied to traffic flows using routing in a virtual (overlay) network to steer traffic between service nodes. Chains can include services running in routers, on physical appliances or in virtual machines. Service chains have applicability at the subscriber edge, business edge and in multi-tenant datacenters. The routing function into SFCs and between service functions within an SFC can be performed by physical devices (routers), be virtualized inside hypervisors, or run as part of a host OS.

A BGP control plane for route distribution is used to create virtual networks implemented using IP MPLS, VXLAN or other suitable encapsulation, where the routes within the virtual networks cause traffic to flow through a sequence of service nodes that apply packet processing functions to the flows.

Two techniques are described: in one the service chain is implemented as a sequence of distinct VPNs between sets of service nodes that apply each service function; in the other, the routes within a VPN are modified through the use of special route targets and modified next-hop resolution to achieve the desired result.

In both techniques, service chains can be created by manual configuration of routes and route targets in routing systems, or through the use of a controller which contains a topological model of the desired service chains.

Internet-Draft

SFC using Virtual Networks with BGP

December 2018

This document also contains discussion of load balancing between network functions, symmetric forward and reverse paths when stateful services are involved, and use of classifiers to direct traffic into a service chain.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 7, 2019.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">4</a>
<a href="#">1.1.</a>	Terminology . . . . .	<a href="#">5</a>
2.	Service Function Chain Architecture Using Virtual Networking	7
<a href="#">2.1.</a>	High Level Architecture . . . . .	<a href="#">8</a>

<a href="#">2.2.</a>	<a href="#">Service Function Chain Logical Model . . . . .</a>	<a href="#">10</a>
<a href="#">2.3.</a>	<a href="#">Service Function Implemented in a Set of SF Instances . .</a>	<a href="#">11</a>
<a href="#">2.4.</a>	<a href="#">SF Instance Connections to VRFs . . . . .</a>	<a href="#">13</a>
<a href="#">2.4.1.</a>	<a href="#">SF Instance in Physical Appliance . . . . .</a>	<a href="#">13</a>
<a href="#">2.4.2.</a>	<a href="#">SF Instance in a Virtualized Environment . . . . .</a>	<a href="#">14</a>

<a href="#">2.5.</a>	<a href="#">Encapsulation Tunneling for Transport . . . . .</a>	<a href="#">15</a>
<a href="#">2.6.</a>	<a href="#">SFC Creation Procedure . . . . .</a>	<a href="#">15</a>
<a href="#">2.6.1.</a>	<a href="#">SFC Provisioning Using Sequential VPNs . . . . .</a>	<a href="#">16</a>
<a href="#">2.6.2.</a>	<a href="#">Modified-Route SFC Creation . . . . .</a>	<a href="#">17</a>
<a href="#">2.6.3.</a>	<a href="#">Common SFC provisioning considerations . . . . .</a>	<a href="#">19</a>
<a href="#">2.7.</a>	<a href="#">Controller Function . . . . .</a>	<a href="#">20</a>
<a href="#">2.8.</a>	<a href="#">Variations on Setting Prefixes in an SFC . . . . .</a>	<a href="#">20</a>
<a href="#">2.8.1.</a>	<a href="#">Using a Default Route . . . . .</a>	<a href="#">20</a>
<a href="#">2.8.2.</a>	<a href="#">Using a Default Route and a Large Prefix . . . . .</a>	<a href="#">21</a>
<a href="#">2.8.3.</a>	<a href="#">Disaggregated Gateway Routers . . . . .</a>	<a href="#">21</a>
<a href="#">2.8.4.</a>	<a href="#">Optimizing VRF usage . . . . .</a>	<a href="#">22</a>
<a href="#">2.8.5.</a>	<a href="#">Dynamic Entry and Exit Signaling . . . . .</a>	<a href="#">22</a>
<a href="#">2.8.6.</a>	<a href="#">Dynamic Re-Advertisements in Intermediate Systems . .</a>	<a href="#">23</a>
<a href="#">2.9.</a>	<a href="#">Layer-2 Virtual Networks and Service Functions . . . . .</a>	<a href="#">23</a>
<a href="#">2.10.</a>	<a href="#">Header Transforming Service Functions . . . . .</a>	<a href="#">23</a>
<a href="#">3.</a>	<a href="#">Load Balancing Along a Service Function Chain . . . . .</a>	<a href="#">24</a>
<a href="#">3.1.</a>	<a href="#">SF Instances Connected to Separate VRFs . . . . .</a>	<a href="#">24</a>
<a href="#">3.2.</a>	<a href="#">SF Instances Connected to the Same VRF . . . . .</a>	<a href="#">25</a>
<a href="#">3.3.</a>	<a href="#">Combination of Egress and Ingress VRF Load Balancing . .</a>	<a href="#">26</a>
<a href="#">3.4.</a>	<a href="#">Forward and Reverse Flow Load Balancing . . . . .</a>	<a href="#">28</a>
<a href="#">3.4.1.</a>	<a href="#">Issues with Equal Cost Multi-Path Routing . . . . .</a>	<a href="#">28</a>
<a href="#">3.4.2.</a>	<a href="#">Modified ECMP with Consistent Hash . . . . .</a>	<a href="#">28</a>
<a href="#">3.4.3.</a>	<a href="#">ECMP with Flow Table . . . . .</a>	<a href="#">29</a>
<a href="#">3.4.4.</a>	<a href="#">Dealing with Different Hash Algorithms in an SFC . .</a>	<a href="#">30</a>
<a href="#">4.</a>	<a href="#">Sharing Service Functions in Different SFCs . . . . .</a>	<a href="#">31</a>
<a href="#">4.1.</a>	<a href="#">Shared SFs in L3 SFCs . . . . .</a>	<a href="#">31</a>
<a href="#">4.2.</a>	<a href="#">Shared SFs in L2 SFCs . . . . .</a>	<a href="#">31</a>
<a href="#">5.</a>	<a href="#">Steering into SFCs Using a Classifier . . . . .</a>	<a href="#">31</a>
<a href="#">6.</a>	<a href="#">External Domain Co-ordination . . . . .</a>	<a href="#">33</a>
<a href="#">7.</a>	<a href="#">Fine-grained steering using BGP Flow-Spec . . . . .</a>	<a href="#">34</a>
<a href="#">8.</a>	<a href="#">Controller Federation . . . . .</a>	<a href="#">34</a>
<a href="#">9.</a>	<a href="#">Coordination Between SF Instances and Controller using BGP .</a>	<a href="#">34</a>
<a href="#">10.</a>	<a href="#">BGP Extended Communities . . . . .</a>	<a href="#">35</a>
<a href="#">10.1.</a>	<a href="#">Route-Target Record . . . . .</a>	<a href="#">35</a>
<a href="#">10.2.</a>	<a href="#">Consistent Hash Sort Order . . . . .</a>	<a href="#">36</a>
<a href="#">10.3.</a>	<a href="#">Load Balance Settings . . . . .</a>	<a href="#">36</a>

<a href="#">11.</a>	Summary and Conclusion . . . . .	<a href="#">37</a>
<a href="#">12.</a>	Security Considerations . . . . .	<a href="#">37</a>
<a href="#">13.</a>	IANA Considerations . . . . .	<a href="#">38</a>
<a href="#">14.</a>	Acknowledgments . . . . .	<a href="#">38</a>
<a href="#">15.</a>	References . . . . .	<a href="#">38</a>
<a href="#">15.1.</a>	Normative References . . . . .	<a href="#">38</a>
<a href="#">15.2.</a>	Informational References . . . . .	<a href="#">39</a>
	Authors' Addresses . . . . .	<a href="#">40</a>

## [1.](#) Introduction

The purpose of networks is to allow computing systems to communicate with each other. Requests are usually made from the client or customer side of a network, and responses are generated by applications residing in a datacenter. Over time, the network between the client and the application has become more complex, and traffic between the client and the application is acted on by intermediate systems that apply network services. Some of these activities, like firewall filtering, subscriber attachment and network address translation are generally carried out in network devices along the traffic path, while others are carried out by dedicated appliances, such as media proxy and deep packet inspection (DPI). Deployment of these in-network services is complex, time-consuming and costly, since they require configuration of devices with vendor-specific operating systems, sometimes with co-processing cards, or deployment of physical devices in the network, which requires cabling and configuration of the devices that they connect to. Additionally, other devices in the network need to be configured to ensure that traffic is correctly steered through the systems that services are running on.

The current mode of operations does not easily allow common operational processes to be applied to the lifecycle of services in the network, or for steering of traffic through them.

The recent emergence of Network Functions Virtualization (NFV) [[NFVE2E](#)] to provide a standard deployment model for network services as software appliances, combined with Software Defined Networking

(SDN) for more dynamic traffic steering can provide foundational elements that will allow network services to be deployed and managed far more efficiently and with more agility than is possible today.

This document describes how the combination of several existing technologies can be used to create chains of functions, while preserving the requirements of scale, performance and reliability for service provider networks. The technologies employed are:

- o Traffic flow between service functions described by routing and network policies rather than by static physical or logical connectivity
- o Packet header encapsulation in order to create virtual private networks using network overlays
- o VRFs on both physical devices and in hypervisors to implement forwarding policies that are specific to each virtual network

- o Optional use of a controller to calculate routes to be installed in routing systems to form a service chain. The controller uses a topological model that stores service function instance connectivity to network devices and intended connectivity between service functions.
- o MPLS or other labeling to facilitate identification of the next interface to send packets to in a service function chain
- o BGP or BGP-style signaling to distribute routes in order to create service function chains
- o Distributed load balancing between service functions performed in the VRFs that service function instance connect to.

Virtualized environments can be supported without necessarily running BGP or MPLS natively. Messaging protocols such as NC/YANG, XMPP or OpenFlow may be used to signal forwarding information. Encapsulation mechanisms such as VXLAN or GRE may be used for overlay transport. The term 'BGP-style', above, refers to this type of signaling.

Traffic can be directed into service function chains using IP routing

at each end of the service function chain, or be directed into the chain by a classifier function that can determine which service chain a traffic flow should pass through based on deep packet inspection (DPI) and/or subscriber identity.

The techniques can support an evolution from services implemented in physical devices attached to physical forwarding systems (routers) to fully virtualized implementations as well as intermediate hybrid implementations.

### [1.1](#). Terminology

This document uses the following acronyms and terms.

Terms	Meaning
-----	-----
AS	Autonomous System
ASBR	Autonomous System Border Router
RR	Route Reflector
RT	Route Target
SDN	Software Defined Network
VM	Virtual Machine
VPN	Virtual Private Network
VRF	VPN Routing and Forwarding table [ <a href="#">RFC4364</a> ]

Table 1

This document follows some of the terminology used in [[RFC7665](#)] and adds some new terminology:

**Network Service:** An externally visible service offered by a network operator; a service may consist of a single service function or a composite built from several service functions executed in one or more pre-determined sequences and delivered by software executing in physical or virtual devices.

**Classification:** Customer/network/service policy used to identify and select traffic flow(s) requiring certain outbound forwarding actions, in particular, to direct specific traffic flows into the ingress of a particular service function chain, or causing branching within a service function chain.

**Virtual Network:** A logical overlay network built using virtual links or packet encapsulation, over an existing network (the underlay).

**Service Function Chain (SFC):** A service function chain defines an ordered set of service functions that must be applied to packets and/or frames selected as a result of classification. An SFC may be either a linear chain or a complex service graph with multiple branches. The term 'Service Chain' is often used in place of 'Service Function Chain'.

**SFC Set:** The pair of SFCs through which the forward and reverse directions of a given classified flow will pass.

**Service Function (SF):** A logical function that is applied to packets. A service function can act at the network layer or other OSI layers. A service function can be embedded in one or more physical network elements, or can be implemented in one or more software instances running on physical or virtual hosts. One or multiple service functions can be embedded in the same network element or run on the same host. Multiple instances of a service function can be enabled in the same administrative domain. We will also refer to 'Service Function' as, simply, 'Service' for simplicity.

A non-exhaustive list of services includes: firewalls, DDOS protection, anti-malware/ant-virus systems, WAN and application acceleration, Deep Packet Inspection (DPI), server load balancers, network address translation, HTTP Header Enrichment functions, video optimization, TCP optimization, etc.

**SF Instance:** An instance of software that implements the packet processing of a service function

**SF Instance Set:** A group of SF instances that, in parallel, implement a service function in an SFC.

**Routing System:** A hardware or software system that performs layer 3 routing and/or forwarding functions. The term includes physical routers as well as hypervisor or Host OS implementations of the forwarding plane of a conventional router.

Gateway: A routing system attached to the source or destination network that peers with the controller, or with the routing system at one end of an SFC. A source network gateway directs traffic from the source network into an SFC, while a destination network gateway distributes traffic towards destinations. The routing systems at each end of an SFC can themselves act as gateways and in a bidirectional SF instance set, gateways can act in both directions VRF: A subsystem within a routing system as defined in [\[RFC4364\]](#) that contains private routing and forwarding tables and has physical and/or logical interfaces associated with it. In the case of hypervisor/Host OS implementations, the term refers only to the forwarding function of a VRF, and this will be referred to as a 'VPN forwarder.'

Ingress VRF: A VRF containing an ingress interface of a SF instance

Egress VRF: A VRF containing an egress interface of a SF instance

Note that in this document the terms 'ingress' and 'egress' are used with respect to SF instances rather than the tunnels that connect SF instances. This is different usage than in VPN literature in general.

Entry VRF: A VRF through which traffic enters the SFC from the source network. This VRF may be used to advertise the destination network's routes to the source network. It could be placed on a gateway router or be collocated with the first ingress VRF.

Exit VRF: A VRF through which traffic exits the SFC into the destination network. This VRF contains the routes from the destination network and could be located on a gateway router. Alternatively, the egress VRF attached to the last SF instance may also function as the exit VRF.

## [2.](#) Service Function Chain Architecture Using Virtual Networking

The techniques described in this document use virtual networks to implement service function chains. Service function chains can be implemented on devices that support existing MPLS VPN and BGP standards [\[RFC4364\]](#), [\[RFC4271\]](#), [\[RFC4760\]](#), as well as other

encapsulations, such as VXLAN [\[RFC7348\]](#). Similarly, equivalent



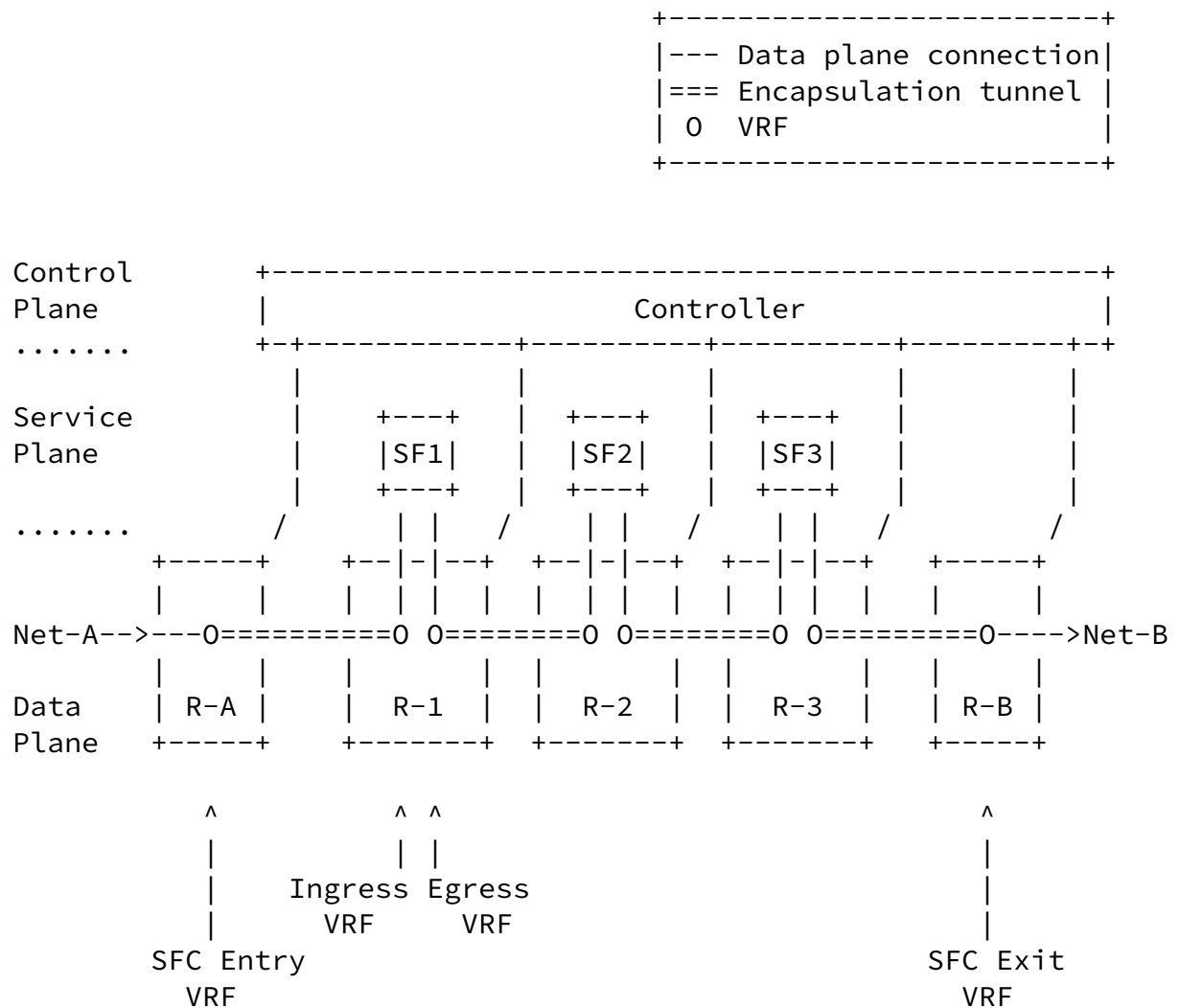
control plane protocols such as BGP-EVPN with type-2 and type-5 route types can also be used where supported [[RFC8365](#)]. The set of techniques described in this document represent one implementation approach to realize the SFC architecture described in [[RFC7665](#)].

The following sections detail the building blocks of the SFC architecture, and outline the processes of route installation and subsequent route exchange to create an SFC.

## [2.1](#). High Level Architecture

Service function chains can be deployed with or without a classifier. Use cases where SFCs may be deployed without a classifier include multi-tenant data centers, private and public cloud and virtual CPE for business services. Classifiers will primarily be used in mobile and wireline subscriber edge use cases. Use of a classifier is discussed in [Section 5](#).

A high-level architecture diagram of an SFC without a classifier, where traffic is routed into and out of the SFC, is shown in Figure 1, below. An optional controller is shown that contains a topological model of the SFC and which configures the network resources to implement the SFC.



High Level SFC Architecture

Figure 1

Traffic from Network-A destined for Network-B will pass through the SFC composed of SF instances, SF1, SF2 and SF3. Routing system R-A contains a VRF (shown as '0' symbol) that is the SFC entry point. This VRF will advertise a route to reach Network-B into Network-A causing any traffic from a source in Network-A with a destination in Network-B to arrive in this VRF. The forwarding table in the VRF in R-A will direct traffic destined for Network-B into an encapsulation tunnel with destination R-1 and a label that identifies the ingress (left) interface of SF1 that R-1 should send the packets out on. The packets are processed by service instance SF-1 and arrive in the egress (right) VRF in R-1. The forwarding entries in the egress VRF direct traffic to the next ingress VRF using encapsulation tunneling. The process is repeated for each service instance in the SFC until

packets arrive at the SFC exit VRF (in R-B). This VRF is peered with Network-B and routes packets towards their destinations in the user

data plane. In this example, routing systems R-A and R-B are gateway routing systems.

In the example, each pair of ingress and egress VRFs are configured in separate routing systems, but such pairs could be collocated in the same routing system, and it is possible for the ingress and egress VRFs for a given SF instance to be in different routing systems. The SFC entry and exit VRFs can be collocated in the same routing system, and the service instances can be local or remote from either or both of the routing systems containing the entry and exit VRFs, and from each other. It is also possible that the ingress and egress VRFs are implemented using alternative mechanisms.

The controller is responsible for configuring the VRFs in each routing system, installing the routes in each of the VRFs to implement the SFC, and, in the case of virtualized services, may instantiate the service instances.

The controller is not responsible for configuring the SFs themselves. It is assumed that there is a separate system that performs that function e.g the VNF Manager functional component in [\[NFVE2E\]](#). Note that coordination may be required between a controller and and VNF manager, for instance to ensure that installed firewall filters in a SF align with the subnets whose packets will pass through it. At this time, there are no standard ways to address this, and custom pair-wise integrations between controllers and VNF managers would be required.

## [2.2.](#) Service Function Chain Logical Model

A service function chain is a set of logically connected service functions through which traffic can flow. Each egress interface of one service function is logically connected to an ingress interface of the next service function.

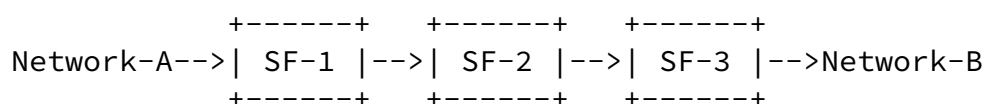


Figure 2

In Figure 2, above, a service function chain has been created that connects Network-A to Network-B, such that traffic from a host in Network-A to a host in Network-B will traverse the service function chain.

As defined in [[RFC7665](#)], a service function chain can be uni-directional or bi-directional. In this document, in order to allow for the possibility that the forward and reverse paths may not be symmetrical, SFCs are defined as uni-directional, and the term 'SFC set' is used to refer to a pair of forward and reverse direction SFCs for some set of routed or classified traffic.

### [2.3](#). Service Function Implemented in a Set of SF Instances

A service function instance is a software system that acts on packets that arrive on an ingress interface of that software system. Service function instances may run on a physical appliance or in a virtual machine. A service function instance may be transparent at layer 2 and/or layer 3, and may support branching across multiple egress interfaces and may support aggregation across ingress interfaces. For simplicity, the examples in this document have a single ingress and a single egress interface.

Each service function in a chain can be implemented by a single service function instance, or by a set of instances in order to provide scale and resilience.



Figure 3

In Figure 3, service function SF-1 is implemented in three service function instances, SFI-11, SFI-12, and SFI-13. Service function SF-2 is implemented in two SF instances. The service function instances are connected to the next service function in the chain using a virtual network, VN-2. Additionally, a virtual network (VN-1) is used to enter the SFC and another (VN-3) is used at the exit.

The logical connection between two service functions is implemented using a virtual network that contains egress interfaces for instances of one service function, and ingress interfaces of instances of the next service function. Traffic is directed across the virtual network between the two sets of service function instances using layer 3 forwarding (e.g. an MPLS VPN) or layer 2 forwarding (e.g. a VXLAN).

The virtual networks could be described as "directed half-mesh", in that the egress interface of each SF instance of one service function can reach any ingress interface of the SF instances of the connected service function.

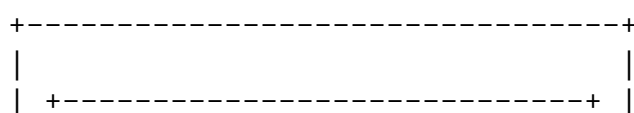
Details on how routing across virtual networks is achieved, and requirements on load balancing across ingress interfaces are discussed in later sections of this document.

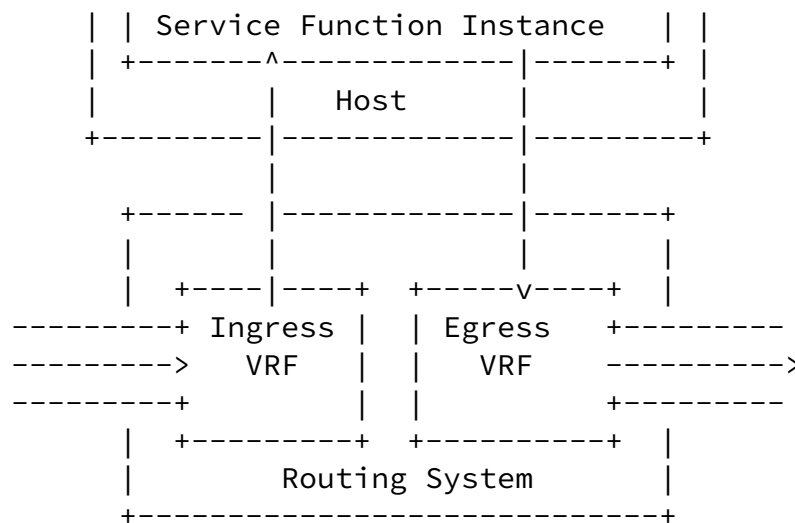
## [2.4.](#) SF Instance Connections to VRFs

SF instances can be deployed as software running on physical appliances, or in virtual machines running on a hypervisor. These two types are described in more detail in the following sections.

### [2.4.1.](#) SF Instance in Physical Appliance

The case of a SF instance running on a physical appliance is shown in Figure 4, below.





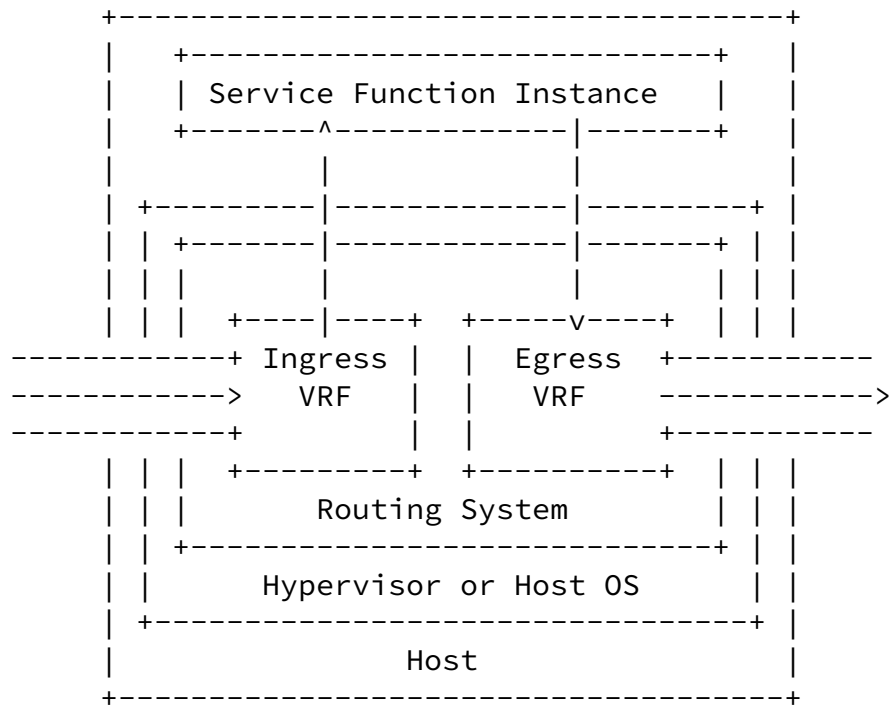
Ingress and Egress VRFs for a Physical Routing System  
and Physical SF Instance

Figure 4

The routing system is a physical device and the service function instance is implemented as software running in a physical appliance (host) connected to it. The connection between the physical device and the routing system may use physical or logical interfaces. Transport between VRFs on different routing systems that are connected to other SF instances in an SFC is via encapsulation tunnels, such as MPLS over GRE, or VXLAN.

#### [2.4.2.](#) SF Instance in a Virtualized Environment

In virtualized environments, a routing system with VRFs that act as VPN forwarders is resident in the hypervisor/Host OS, and is co-resident in the host with one or more SF instances that run in virtual machines. The egress VPN forwarder performs tunnel encapsulation to send packets to other physical or virtual routing systems with attached SF instances to form an SFC. The tunneled packets are sent through the physical interfaces of the host to the other hosts or physical routers. This is illustrated in Figure 5, below.



Ingress and Egress VRFs for a Virtual Routing System  
and Virtualized SF Instance

Figure 5

When more than one instance of an SF is running on a hypervisor, they can be connected to the same VRF for scale out of an SF within an SFC.

The routing mechanisms in the VRFs into and between service function instances, and the encapsulation tunneling between routing systems are identical in the physical and virtual implementations of SFCs and routing systems described in this document. Physical and virtual service functions can be mixed as needed with different combinations of physical and virtual routing systems, within a single service chain.

The SF instances are attached to the routing systems via physical, virtual or logical (e.g, 802.1q) interfaces, and are assumed to perform basic L3 or L2 forwarding.



A single SF instance can be part of multiple service chains. In this case, the SF instance will have dedicated interfaces (typically logical) and forwarding contexts associated with each service chain.

## [2.5.](#) Encapsulation Tunneling for Transport

Encapsulation tunneling is used to transport packets between SF instances in the chain and, when a classifier is not used, from the originating network into the SFC and from the SFC into the destination network.

The tunnels can be MPLS over GRE [[RFC4023](#)], MPLS over UDP [[RFC7510](#)], MPLS over MPLS [[RFC3031](#)], VXLAN [[RFC7348](#)] [[RFC7348](#)], or another suitable encapsulation method.

Tunneling capabilities may be enabled in each routing system as part of a base configuration or may be configured by the controller. Tunnel encapsulations may be programmed by the controller or signaled using BGP. The encapsulation to be used for a given route is signaled in BGP using the procedures described in [[idr-tunnel-encaps](#)], i.e. typically relying on the BGP Tunnel Encapsulation Extended Community.

## [2.6.](#) SFC Creation Procedure

This section describes how service chains are created using two methods:

- o Sequential VPNs - where a conventional VPN is created between each set of SF instances to create the links in the SFC
- o Route Modification - where each routing system modifies advertised routes that it receives, to realize the links in an SFC on the basis of a special service topology RT and a route-policy that describes the service chain logical topology

In both cases the controller, when present, is responsible for creating ingress and egress VRFs, configuring the interfaces connected to SF instances in each VRF, and allocating and configuring import and export RTs for each VRF. Additionally, in the second method, the controller also sends the route-policy containing the service chain logical topology to each routing system. If a controller is not used, these procedures will require to be performed manually or through scripting, for instance.

The source and destination networks' prefixes can be configured in the controller, or may be automatically learned through peering between the controller and each network's gateway. This is further described in [Section 2.8.5](#) and [Section 6](#).

The following sub-sections describe how RT configuration, local route installation and route distribution occur in each of the methods.

It should be noted that depending on the capabilities of the routing systems, a controller can use one or more techniques to realize forwarding along the service chain, ranging from fully centralized to fully distributed. The goal of describing the following two methods is to illustrate the broad approaches and as a base for various optimization options.

Interoperability between a controller implementing one method and a controller implementing a different method is achieved by relying on the techniques described in [section 5](#) and [section 8](#), that describe the use of BGP-style service chaining within domains that are interconnected using standard BGP VPN route exchanges.

#### [2.6.1](#). SFC Provisioning Using Sequential VPNs

The task of the controller in this method of SFC provisioning is to create a set of VPNs that carry traffic to the destination network through instances of each service function in turn. This is achieved by allocating and configuring RTs such that the egress VRFs of one set of SF instances import an RT that is an export RT for the ingress VRFs of the next, logically connected, set of SF instances.

The process of SFC creation is as follows:

1. Controller creates a VRF in each routing system that is connected to a service instance that will be used in the SFC
2. Controller configures each VRF to contain the logical interface that connects to a SF instance.
3. Controller implements route target import and export policies in the VRFs using the same route targets for the egress VRFs of a service function and the ingress VRFs of the next logically connected service function in the SFC.
4. Controller installs a static route in each ingress VRF whose next hop is the interface that a SF instance is connected to. The prefix for the route is the destination network to be reached by passing through the SFC. The following sections describe

variations that can be used.

5. Routing systems advertise the static routes via BGP as VPN routes with next hop being the IP address of the router, with an encapsulation specified and a label that identifies the service instance interface.
6. Routing systems containing VRFs with matching route targets receive the updates.
7. Routes are installed in egress VRFs with matching import targets. The egress VRFs of each SF instance will now contain VPN routes to one or more routers containing ingress VRFs for SF instances of the next service function in the SFC.

Routes to the destination network via the first set of SF instances are advertised into the source network, and the egress VRFs of the last SF instance set have routes into the destination network.

As discussed further in [Section 3](#), egress VRFs can load balance across the multiple next hops advertised from the next set of ingress VRFs.

#### [2.6.2](#). Modified-Route SFC Creation

In this method of SFC configuration, all the VRFs connected to SF instances for a given SFC are configured with same import and export RT, so they form a VPN-connected mesh between the SF instance interfaces. This is termed the 'Service VPN'. A route is configured or learnt in each VRF with destination being the IP address of a connected SF instance via an interface configured in the VRF. The interface may be a physical or logical interface. The routing system that hosts such a VRF advertises a VPN route for each locally connected SF instance, with a forwarding label that enables it to forward incoming traffic from other routing systems to the connected SF instance. The VPN routes may be advertised via an RR or the controller, which sends these updates to all the other routing systems that have VRFs with the service VPN RT. At this point all the VRFs have a route to reach every SF instance. The same virtual IP address may be used for each SF instance in a set, enabling load-balancing among multiple SF instances in the set.

The controller builds a route-policy for the routing systems in the VPN, that describes the logical topology of each service chain that it belongs to. The route-policy contains entries in the form of a tuple for each service chain:

{Service-topology-name, Service-topology-RT, Service-node- sequence}

where Service-node-sequence is simply an ordered list of the service function interface IP addresses that are in the chain.

Every service function chain has a single unique service-topology-RT that is allocated and provisioned on all participating routing systems in the relevant VRFs.

The VRF in the routing system that connects to the destination network (i.e. the exit VRF) is configured to attach the Service-topology-RT to exported routes, and the VRF connected to the source network (i.e. the entry VRF) will import routes using the Service-topology-RT. The controller may also be used to originate the Service-topology-RT attached routes.

The route-policy may be described in a variety of formats and installed on the routing system using a suitable mechanism. For instance, the policy may be defined in YANG and provisioned using Netconf [[RFC6241](#)].

Using Figure 1 for reference, when the gateway R-B advertises a VPN route to Network-B, it attaches the Service-topology-RT. BGP route updates are sent to all the routing systems in the service VPN. The routing systems perform a modified set of actions for next-hop resolution and route installation in the ingress VRFs compared to normal BGP VPN behavior in routing systems, but no changes are required in the operation of the BGP protocol itself. The modification of behavior in the routing systems allows the automatic and constrained flow of traffic through the service chain.

Each routing system in the service VPN will process the VPN route to Network-B via R-B as follows:

1. If the routing system contains VRFs that import the Service-

topology-RT, continue, otherwise ignore the route.

2. The routing system identifies the position and role (ingress/egress) of each of its VRFs in the SFC by comparing the IP address of the route in the VRF to the connected SF instance with those in the Service-node- sequence in the route-policy. Alternatively, the controller may provision the specific service node IP to be used as the next-hop in each VRF, in the route-policy for the VRF.
3. The routing system modifies the next-hop of the imported route with the Service-topology-RT, to select the appropriate next-hop as per the route-policy. It ignores the next-hop and label in the received route. It resolves the selected next-hop in the local VRF routing table.

4.
  - a. The imported route to Network-B in the ingress VRF is modified to have a next-hop of the IP address of the logically connected SF instance.
  - b. The imported route to Network-B in the egress VRF is modified to have a next hop of the IP address of the next SF instance in the SFC.
5. The egress VRFs for the last service function install the VPN route via the gateway R-B unmodified.

Note that the modified routes are not re-advertised into the VPN by the various intermediate routing systems in the SFC.

### [2.6.3.](#) Common SFC provisioning considerations

In both the methods, for physical routers, the creation and configuration of VRFs, interfaces and local static routes can be performed programmatically using Netconf; and BGP route distribution can use a route reflector (which may be part of the controller). In the virtualized case, where a VPN forwarder is present, creation and configuration of VRFs, interfaces and installation of routes may instead be performed using a single protocol like XMPP, NC/YANG or an equivalent programmatic interface.

Also in the virtualized case, the actual forwarding table entries to be installed in the ingress and egress VRFs may be calculated by the controller based on its internal knowledge of the required SFC topology and the connectivity of SF instances to routing systems. In this case, the routes may be directly installed in the forwarders using the programmatic interface and no BGP route advertisement is necessary, except when coordination with external domains ([Section 6](#)) or federation between controller domains is employed ([Section 8](#)). Note however that this is just one typical model for a virtual forwarding based system. In general, physical and virtual routing systems can be treated exactly the same if they have the same capabilities.

In both the methods, the SF instance may also need to be set up appropriately to forward traffic between its input and output interfaces, either via static, dynamic or policy-based routing. If the service function is a transparent L2 service, then the static route installed in the ingress VRF will have a next-hop of the IP address of the routing system interface that the service instance is attached to on its other interface.

## [2.7.](#) Controller Function

The purpose of the controller is to manage instantiation of SFCs in networks and datacenters. When an SFC is to be instantiated, a model of the desired topology (service functions, number of instances, connectivity) is built in the controller either via an API or GUI. The controller then selects resources in the infrastructure that will support the SFC and configures them. This can involve instantiation of SF instances to implement each service function, the instantiation of VRFs that will form virtual networks between SF instances, and installation of routes to cause traffic to flow into and between SF instances. It can also include provisioning the necessary static, dynamic or policy based forwarding on the service function instance to enable it to forward traffic.

For simplicity, in this document, the controller is assumed to contain all the required features for management of SFCs. In actual implementations, these features may be distributed among multiple inter-connected systems. E.g. An overarching orchestrator might

manage the overall SFC model, sending instructions to a separate virtual machine manager to instantiate service function instances, and to a virtual network manager to set up the service chain connections between them.

The controller can also perform necessary BGP signaling and route distribution actions as described throughout this document.

## [2.8.](#) Variations on Setting Prefixes in an SFC

The SFC Creation section above described the basic procedures for a couple of SFC creation methods. This section describes some techniques that can extend and provide optimizations on top of the basic procedures.

### [2.8.1.](#) Using a Default Route

In the methods described above, it can be noted that only the gateway routing systems need the specific network prefixes to steer traffic in and out of the SFC. The intermediate systems can direct traffic in the ingress and egress VRFs by using only a default route. Hence, it is possible to avoid installing the network prefixes in the intermediate systems. This can be done by splitting the SFC into two sections - one linking the entry and exit VRFs and the other including the intermediate systems. For instance, this may be achieved by using two different Service-topology-RTs in the second method.

### [2.8.2.](#) Using a Default Route and a Large Prefix

In the configuration methods described above, the network prefixes for each network (Network-A and Network-B in the example above) connected to the SFC are used in the routes that direct traffic through the SFC. This creates an operational linkage between the implementation of the SFC and the insertion of the SFC into a network.

For instance, subscriber network prefixes will normally be segmented across subscriber attachment points such as broadband or mobile gateways. This means that each SFC would have to be configured with

the subscriber network prefixes whose traffic it is handling.

In a variation of the SFC configuration method described above, the prefixes used in each direction can be such that they include all possible addresses at each side of the SFC. For example, in Figure 1, the prefix for Network-A could include all subscriber IP addresses and the prefix for Network-B could be the default route, 0/0.

Using this technique, the same routes can be installed in all instances of an SFC that serve different groups of subscribers in different geographic locations.

The routes forwarding traffic into a SF instance and to the next SF instance are installed when an SFC is initially built, and each time a SF instance is connected into the SFC, but there is no requirement for VRFs to be reconfigured when traffic from different networks pass through the service chain, so long as their prefix is included in the prefixes in the VRFs along the SFC.

In this variation, it is assumed that no subscriber-originated traffic will enter the SFC destined for an IP address also in the subscriber network address range. This will not be a restriction in many cases.

### [2.8.3.](#) Disaggregated Gateway Routers

As a slight variation of the above, a network prefix may be disaggregated and spread out among various gateway routers, for instance, in the case of virtual machines in a data-center. In order to reduce the scaling requirements on the routing systems along the SFC, the SFC can again be split into two sections as described above. In addition, the last egress VRF may act as the exit VRF and install the destination network's disaggregated routes. If the destination network's prefixes can be aggregated, for instance into a subnet

prefix, then the aggregate prefix may be advertised and installed in the entry VRF.

### [2.8.4.](#) Optimizing VRF usage



It may be desirable to avoid using distinct ingress and egress VRFs for the service instances in order to make more efficient use of VRF resources, especially on physical routing systems. The ingress VRF and egress VRF may be treated as conceptual entities and the forwarding realized using one or more options described in this section, combined with the methods described earlier.

For instance, the next-hop forwarding label described earlier serves the purpose of directing traffic received from other routing systems directly towards an attached service instance. On the other hand, if the encapsulation mechanism or the device in use requires an IP lookup for incoming packets from other routing systems, then the specific network prefixes may be installed in the intermediate service VRFs to direct traffic towards the attached service instances.

Similarly, a per-interface policy-based-routing rule applied to an access interface can serve to direct traffic coming in from attached service instances towards the next SF set.

#### [2.8.5.](#) Dynamic Entry and Exit Signaling

When either of the methods of the previous sections are employed, the prefixes of the attached networks at each end of an SFC can be signaled into the corresponding VRFs dynamically. This requires that a BGP session is configured either from the network device at each end of the SFC into each network or from the controller.

If dynamic signaling is performed, and a bidirectional SFC set is configured, and the gateways to the networks connected via the SFC exchange routes, steps must be taken to ensure that routes to both networks do not get advertised from both ends of the SFC set by re-origination. This can be achieved if a new BGP Extended Community [[RFC4360](#)] is implemented to control re-origination. When a route is re- originated, the RTs of the re-originated routes are appended to the new RT-Record Extended Community, and if the RT for the route already exists in the Extended Community, the route is not re-originated (see [Section 10.1](#)).

#### [2.8.6.](#) Dynamic Re-Advertisements in Intermediate Systems

The intermediate routing systems attached to the service instances may also use the dynamic signaling technique from the previous section to re-advertise received routes up the chain. In this case, the ingress and egress VRFs are combined into one; and a local route-policy ensures the re-advertised routes are associated with labels that direct incoming traffic directly to the attached service instances on that routing system.

#### [2.9.](#) Layer-2 Virtual Networks and Service Functions

There are SFs that operate at layer-2, in a transparent mode, and forward traffic based on the MAC DA. When such a SF is present in the SFC, the procedures at the routing system are modified slightly. The L3 routes are the same as the L3 SF case, but now an L2 header has to be supplied for each packet passing through an SF. A convenient destination MAC address to use is the one that each VRF uses for the default gateway of its network. This can be the same for all VRFs in routing systems in the domain of a controller. The VRF at the egress of the last SF will rewrite the gateway MAC address with the MAC address of the actual destination before encapsulating and forwarding.

A SFC may also be set up between end systems or network segments within the same Layer-2 bridged network. In this case, applying the procedures described earlier, the segments or groups of end systems are placed in distinct Layer-2 virtual networks, which are then inter-connected via a sequence of intermediate Layer-2 virtual networks that form the links in the SFC. Each virtual network maps to a pair of ingress and egress MAC VRFs on the routing systems to which the SF instances are attached. The routing systems at the ends of the SFC will advertise the locally learnt or installed MAC entries using BGP-EVPN type-2 routes, which will get installed in the MAC VRFs at the other end. The intermediate systems may use default MAC routes installed in the ingress and egress MAC VRFs, or the other variations described earlier in this document.

#### [2.10.](#) Header Transforming Service Functions

If a service function performs an action that changes the source address in the packet header (e.g., NAT), the routes that were installed as described above may not support reverse flow traffic.

The solution to this is for the controller modify the routes in the reverse direction to direct traffic into instances of the transforming service function. The original routes with a source prefix (Network-A in Figure 2) are replaced with a route that has a

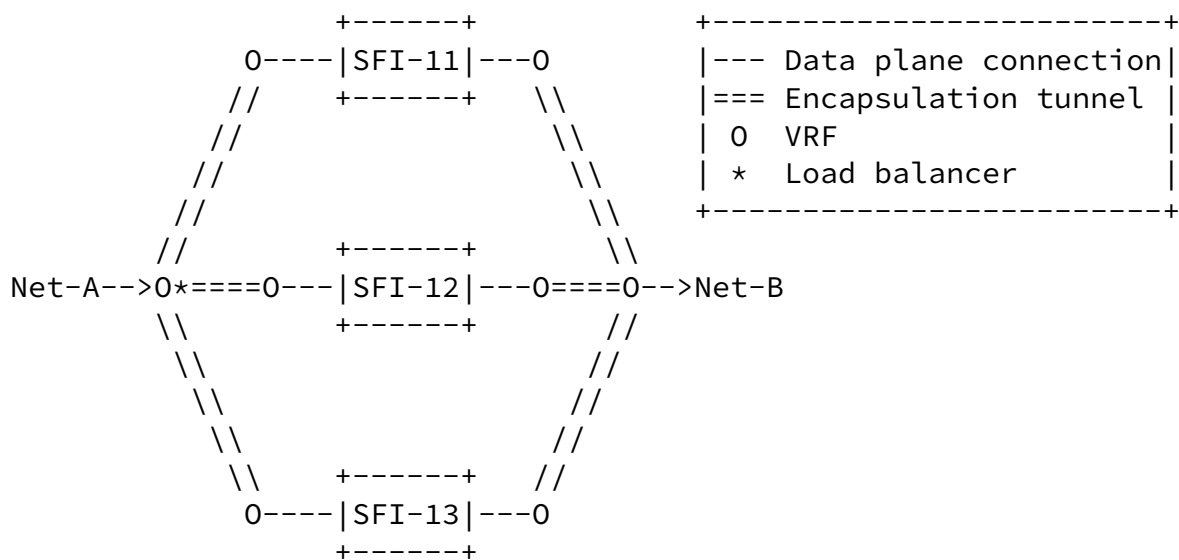
prefix that includes all the possible addresses that the source address could be mapped to. In the case of network address translation, this would correspond to the NAT pool.

### [3.](#) Load Balancing Along a Service Function Chain

One of the key concepts driving NFV [[NFVE2E](#)] is the idea that each service function along an SFC can be separately scaled by changing the number of service function instances that implement it. This requires that load balancing be performed before entry into each service function. In this architecture, load balancing is performed in either or both of egress and ingress VRFs depending on the type of load balancing being performed, and if more than one service instance is connected to the same ingress VRF.

#### [3.1.](#) SF Instances Connected to Separate VRFs

If SF instances implementing a service in an SFC are each connected to separate VRFs (e.g. instances are connected to different routers or are running on different hosts), load balancing is performed in the egress VRFs of the previous service, or in the VRF that is the entry to the SFC. The controller distributes BGP multi-path routes to the egress VRFs. The destination prefix of each route is the ultimate destination network, or its representative aggregate or default. The next-hops in the ECMP set are BGP next-hops of the service instances attached to ingress VRFs of the next service in the SFC. The load balancing corresponds to BGP Multipath, which requires that the route distinguishers for each route are distinct in order to recognize that distinct paths should be used. Hence, each VRF in a distributed, SFC environment should have a unique route distinguisher.



Egress VRF Load Balancing across SF Instances  
Connected to Different VRFs

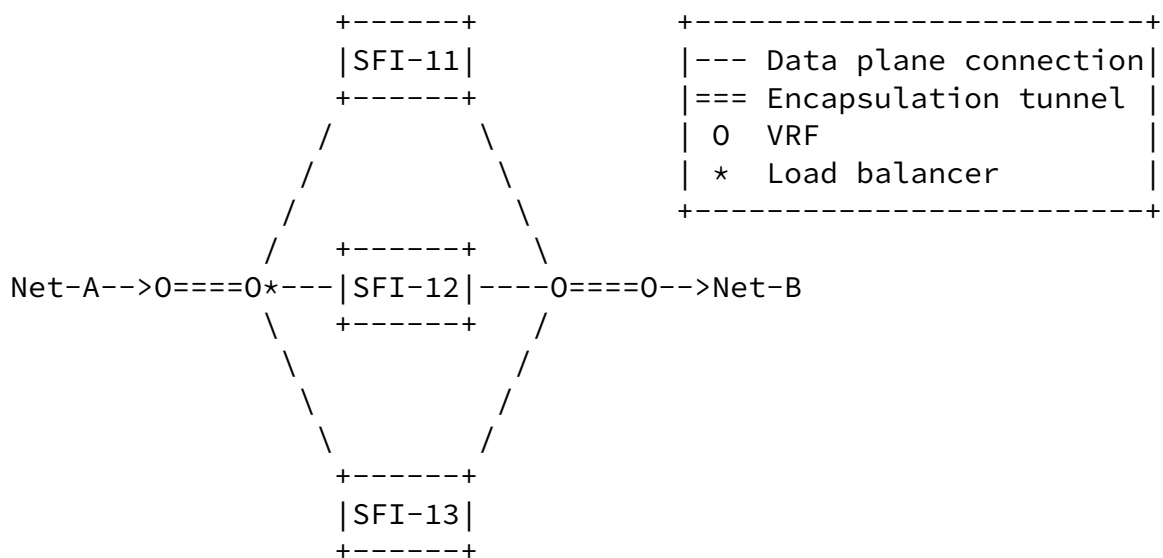
Figure 6

In the diagram, above, a service function is implemented in three service instances each connected to separate VRFs. Traffic from Network-A arrives at VRF at the start of the SFC, and is load balanced across the service instances using a set of ECMP routes with next hops being the addresses of the routing systems containing the ingress VRFs and with labels that identify the ingress interfaces of the service instances.

In the case that the bandwidth of the links between the load balancer and the ingress VRFs are unequal, or that the bandwidth capacity of the service function instances are unequal, this can be signalled in the routes for each ingress VRF using the extended community described in [[draft-ietf-idr-link-bandwidth](#)] and procedures from [[draft-malhotra-bess-evpn-unequal-lb](#)] could be followed.

### 3.2. SF Instances Connected to the Same VRF

When SF instances implementing a service in an SFC are connected to the same ingress VRF, load balancing is performed in the ingress VRF across the service instances connected to it. The controller will install routes in the ingress VRF to the destination network with the interfaces connected to each service instance as next hops. The ingress VRF will then use ECMP to load balance across the service instances.



Ingress VRF Load Balancing across SF Instances  
Connected to the Same VRF

Figure 7

In the diagram, above, a service is implemented by three service instances that are connected to the same ingress and egress VRFs. The ingress VRF load balances across the ingress interfaces using ECMP, and the egress traffic is aggregated in the egress VRF.

If forwarding labels that identify each SFI ingress interface are used, and if the routes to each SF instance are advertised with



## Load Balancing across SF Instances

Figure 8

In Figure 8, above, an SFC is composed of two services implemented by three service instances and two service instances, respectively. The service instances SFI-11 and SFI-12 are connected to the same ingress and egress VRFs, and all the other service instances are connected to separate VRFs.

Traffic entering the SFC from Network-A is load balanced across the ingress VRFs of the first service function by the chain entry VRF, and then load balanced again across the ingress interfaces of SFI-11 and SFI-12 by the shared ingress VRF. Note that use of standard ECMP will lead to an uneven distribution of traffic between the three service instances (25% to SFI-11, 25% to SFI-12, and 50% to SFI-13). This issue can be mitigated through the use of BGP link bandwidth extended community [[draft-ietf-idr-link-bandwidth](#)] and use of procedures described in [[draft-malhotra-bess-evpn-unequal-lb](#)]. As described in the previous section, if a next-hop forwarding label is used, another way to mitigate this effect would be to advertise routes to each SF instance connected to a VRF with a different route distinguisher.

After traffic passes through the first set of service instances, it is load balanced in each of the egress VRFs of the first set of

service instances across the ingress VRFs of the next set of service instances.

### [3.4.](#) Forward and Reverse Flow Load Balancing

This section discusses requirements in load balancing for forward and reverse paths when stateful service functions are deployed.

#### [3.4.1.](#) Issues with Equal Cost Multi-Path Routing

As discussed in the previous sections, load balancing in the forward SFC in the above example can automatically occur with standard BGP, if multiple equal cost routes to Network-B are installed into all the ingress VRFs, and each route directs traffic through a different

service function instance in the next set. The multiple BGP routes in the routing table will translate to Equal Cost Multi-Path in the forwarding table. The hash used in the load balancing algorithm (per packet, per flow or per prefix) is implementation specific.

If a service function is stateful, it is required that forward flows and reverse flows always pass through the same service function instance. Standard ECMP does not provide this capability, since the hash calculation will see different input data for the same flow in the forward and reverse directions (since the source and destination fields are reversed).

Additionally, if the number of SF instances changes, either increasing to expand capacity, or decreases (planned, or due to a SF instance failure), the hash table in ECMP is recalculated, and most flows will be directed to a different SF instance and user sessions will be disrupted.

There are a number of ways to satisfy the requirements of symmetric forward/reverse paths for flows and minimal disruption when SF instances are added to or removed from a set. Two techniques that can be employed are described in the following sections.

#### [3.4.2.](#) Modified ECMP with Consistent Hash

Symmetric forwarding into each side of an SF instance set can be achieved with a small modification to ECMP if the packet headers are preserved after passing through the SF instance set and assuming that the same hash function, same hash salt and same ordering association of hash buckets to ECMP routes is used in both directions. Each packet's 5-tuple data is used to calculate which hash bucket, and therefore which service instance, that the packet will be sent to, but the source and destination IP address and port information are swapped in the calculation in the reverse direction. This method

only requires that the list of available service function instances is consistently maintained in load balance tables in all the routing systems rather than maintaining flow tables. This requirement can be met by the use of a distinct VPN route for each instance.

In the SFC architecture described in this document, when SF instances are added or removed, the controller is required to install (or



remove) routes to the SF instances. The controller could configure the load balancing function in VRFs that connect to each added (or removed) SF instance as part of the same network transaction as route updates to ensure that the load balancer configuration is synchronized with the set of SF instances.

The consistent ordering among ECMP routes in the routing systems could be achieved through configuration of the routing systems by the controller using, for instance, Netconf; or when the routes are signaled using BGP by the controller or a routing system, the order for a given instance can be sent in a new 'Consistent Hash Sort Order' BGP Extended Community (defined in [Section 10.2](#)).

The effect of rehashing when SF instances are added or removed can be minimized, or even eliminated using variations of the technique of consistent hashing [[consistent-hash](#)]. Details are outside the scope of this document.

#### [3.4.3](#). ECMP with Flow Table

A second refinement that can ensure forward/reverse flow consistency, and also provides stability when the number of SF instances changes ('flow-stickiness'), is the use of dynamically configured IP flow tables in the VRFs. In this technique, flow tables are used to ensure that existing flows are unaffected if the number of ECMP routes changes, and that forward and reverse traffic passes through the same SF instance in each set of SF instances implementing a service function.

The flow tables are set up as follows:

1. User traffic with a new 5-tuple enters an egress VRF from a connected SF instance.
2. The VRF calculates the ECMP hash across available routes (i.e., ECMP group) to the ingress interfaces of the SF instances in the next SF instance set. The consistent hash technique described in [section 3.4.2](#) must be used here and in subsequent steps.
3. The VRF creates a new flow entry for the 5-tuple of the new traffic with the next-hop being the chosen downstream ECMP group

member (determined in the step 2. above). All subsequent packets for the same flow will be forwarded using flow lookup and, hence, will use the same next-hop.

4. The encapsulated packet arrives in the routing system that hosts the ingress VRF for the selected SF instance.
5. The ingress VRF of the next service instance determines if the packet came from a routing system that is in an ECMP group in the reverse direction(i.e., from this ingress VRF back to the previous set of SF instances).
6. If an ECMP group is found, the ingress VRF creates a flow entry for the reversed 5-tuple with next-hop of the tunnel on which traffic arrived. This is for the traffic in the reverse direction.
7. If multiple SF instances are connected to the ingress VRF, the ECMP consistent hash is used to choose which one to send the traffic into.
8. A forward flow table entry is created for the traffic's 5-tuple with next hop of the interface of the SF instance chosen in the previous step.
9. The packet is sent into the selected SF instance.

The above method ensures that forward and reverse flows pass through the same SF instances, and that if the number of ECMP routes changes when SF instances are added or removed, all existing flows will continue to flow through the same SF instances, but new flows will use the new ECMP hash. The only flows affected will be those that were passing through an SF instance that was removed, and those will be spread among the remaining SF instances using the updated ECMP hash.

If the consistent hash algorithm is used in both directions, then only the forwarding flow entries would be required, and would be built independently in each direction. If distinct VPN routes with next-hop forwarding labels are used, then only the flow table in step 3 is sufficient to provide flow stickiness.

#### 3.4.4. Dealing with Different Hash Algorithms in an SFC

In some cases, there will be two or more hash algorithms in forwarders along an SFC. E.g. when a physical router is at the entry and exit of the chain, and virtual forwarders are used within the chain. Forward and reverse flows will mostly not pass through the

same SF instances of the first SF, and the SFC will not operate as intended if the first SF is stateful. It may be impractical, or prohibitively expensive to implement the flow table-based methods described above to achieve flow stability and symmetry. This issue can be mitigated by ensuring that the first SF is not stateful, or by placing a null SF between the physical router and the first actual SF in the SFC. This ensures that the hash method on both sides of stateful service instances is the same, and the SFC will operate with flow stability and symmetry if the methods described above are employed.

#### [4.](#) Sharing Service Functions in Different SFCs

##### [4.1.](#) Shared SFs in L3 SFCs

Sharing SFs among multiple SFCs requires that packets emerging from an SF can be mapped to the correct next SF. This can be achieved by configuring an SF to accept packets from multiple subnets on each interface, configuring addresses from each of these subnets on SFs and by using next-table policies to direct traffic between subnet-specific VRFs to and from SF interfaces. However, in mobility and wireline applications, which are the most common ones where sharing is desired, classification is on the basis of subscriber id and traffic type, so discrimination on the basis of subnets is too coarse-grained. Using host routes along SFC paths could achieve the desired result of SF sharing, but will not scale appropriately.

##### [4.2.](#) Shared SFs in L2 SFCs

Layer 2 transparent SFs can be shared among multiple service chains by using a different VLAN for each chain as packets pass through each SF. Forwarding into different VLANs can be accomplished by using different labels in the encapsulation of packets arriving at an ingress VRF. Egress VRFs can have next hops into different SFs based on the VLAN of each egressing packet. The service chains sharing an SF might have different networks from each other at each end, or might be selected on the basis of 5-tuple filtering from one network.

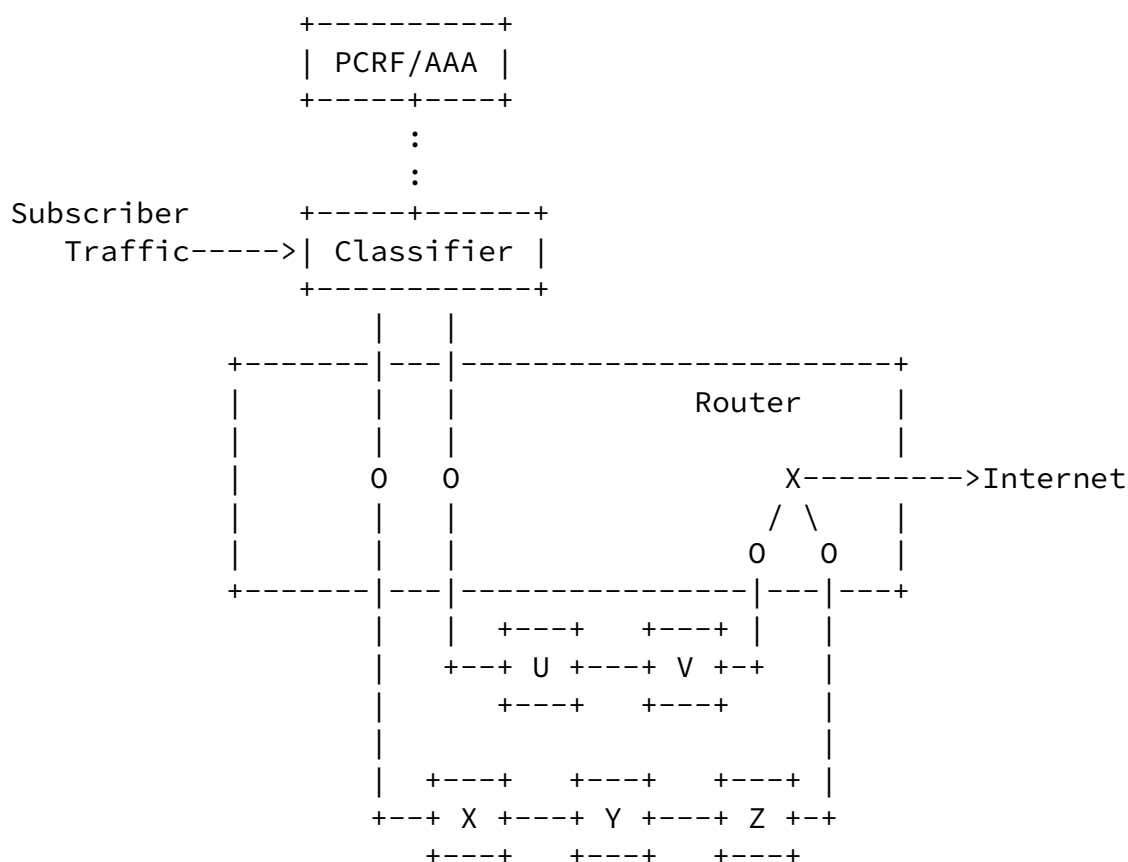
#### [5.](#) Steering into SFCs Using a Classifier

In many applications of SFCs, a classifier will be used to direct traffic into SFCs. The classifier inspects the first or first few packets in a flow to determine which SFC the flow should be sent

into. The decision criteria can be based on just the IP 5-tuple of the header (i.e filter-based forwarding), or could involve analysis of the payload of packets using deep packet inspection. Integration with a subscriber management system such as PCRF or AAA may be

required in order to identify which SFC to send traffic to based on subscriber policy.

An example logical architecture is shown in Figure 9, below where a classifier is external to a physical router that is hosting the VRFs that form the ends of two SFC sets. In the case of filter-based forwarding, classification could occur in a VRF on the router.



Subscriber/Application-Aware Steering with a Classifier

Figure 9

In the diagram, the classifier receives subscriber traffic and sends the traffic out of one of two logical interfaces, depending on classification criteria. The logical interfaces of the classifier are connected to VRFs in a router that are entries to two SFCs (shown as 0 in the diagram).

In this scenario, the entry VRF for each chain does not advertise the destination network prefixes and the modified method of setting prefixes, described in [Section 2.8.2](#) can be employed. Also, the exit VRF for each SFC does not peer with a gateway or proxy node in the destination network and packets are forwarded using IP lookup in the main routing table or in a VRF that the exit traffic from the SFCs is

directed into (shown as X in the diagram). A flow table may be required to ensure that reverse traffic is sent into the correct SFC.

An alternative would be where the classifier is itself a distributed, virtualized service function, but with multiple egress interfaces. In that case, each virtual classifier instance could be attached to a set of VRFs that connect to different SFCs. Each chain

entry VRF would load balance across the first SF instance set in its SFC. The reverse flow table mechanism described in [Section 3.4.3](#) could be employed to ensure that flows return to the originating classifier instance which may maintain subscriber context and perform charging and accounting.

## [6.](#) External Domain Co-ordination

It is likely that SFCs will be managed as a separate administrative domain from the networks that they receive traffic from, and send traffic to. If the connected networks use BGP for route distribution, the controller in the SFC domain can join the network domains by creating BGP peering sessions with routing systems or route reflectors in those network domains to exchange VPN routes, or with local border routers that peer with the external domains. While a controller can modify route targets for the VRFs within its SFC domain, it is likely to not have any control over the external networks with which it is peering. Hence, the design does not assume that the RTs of external network domains can be modified by the controller. It may however learn those RTs and use them in its modified route advertisements.

In order to steer traffic from external network domains into an SFC, the controller will advertise a destination network's prefixes into the peering source network domain with a BGP next-hop and label associated with the SFC entry point that may be on a routing system attached to the first SF instance. This advertisement may be over regular MP-BGP/VPN peering which assumes existing standard VPN routing/forwarding behavior on the network domain's routers (PEs/ASBRs). The controller can learn routes to networks in external domains at the egress of an SFC and advertise routes to those network into other external domains using the first ingress routing instance as the next hop thus allowing dynamic steering through re-origination of routes.

An operational benefit of this approach is that the SFC topology within a domain need not be exposed to other domains. Additionally, using non-specific routes inside an SFC, as described in [Section 2.8.1](#), means that new networks can be attached to a SFC without needing to configure prefixes inside the chain.

The controller will typically remove the destination network's RTs and replace them with the RTs of the source network while advertising the modified routes. Alternatively, an external domain may be provisioned with an additional export-only RT and an import-only RT that the controller can use.

## [7.](#) Fine-grained steering using BGP Flow-Spec

When steering traffic from an external network domain into an SFC based on attributes of the packet flow, BGP Flow-spec can be used as a signaling option.

In this case, the controller can advertise one or more flow-spec routes into the entry VRF with the appropriate Service-topology-RT for the SFC. Alternatively, it can use the procedures described in [\[RFC5575\]](#) or [\[flowspec-redirect-ip\]](#) on the gateway router to redirect traffic towards the first SF.

If it is desired to steer specific flows from a network domain's existing routers, the controller can advertise the above flow-spec routes to the network domain's border routers or route reflectors.

## [8.](#) Controller Federation

When SFCs are distributed geographically, or in very large-scale environments, there may be multiple SFC controllers present and they may variously employ both of the SFC creation methods described in [Section 2.6](#). If there is a requirement for SFCs to span controller domains there may be a requirement to exchange information between controllers. Again, a BGP session between controllers can be used to exchange route information as described in the previous sections and allow such domain spanning SFCs to be created.

## [9.](#) Coordination Between SF Instances and Controller using BGP

In many cases, the configuration of SF instance determines its network behavior. E.g. when NAT pools are set up, or when an SSL gateway is configured with a set of enterprise IP addresses to use. In these cases, the addresses that will be used by the SFs need to be known in the networks connecting to them in order that traffic can be properly routed. When SFCs are involved, this means that the controller has to be notified when such configuration changes are made in SF instances. Sometimes, the changes will be made by end-customers and it is desirable the controller adjust the SFC routing configuration automatically when the change is made, and without customers needing to notify the service provider via a portal, for instance, or requiring development of integration modules linking the SF instances and the controller.

One option for automatic notification for SFs that support BGP is for the connected forwarding system (physical or virtual SFF) to also support BGP, and for SF instances to be configured to peer with the SFF. When changes are made to the configuration of a SF instance, that for example, the SF will accept packets from a particular network prefix on one of its interfaces, the SF instance will send a BGP route update to the SFF it is connected to and which it has a BGP session with. The controller can then adjust the routes along SFCs to ensure that packets with destinations in the new prefix reach the reconfigured SF instance.

BGP could also be used to signal from the controller to a SF instance that certain traffic should be sent out from a particular interface. This could be used to direct suspect traffic to a security scrubbing center, for example.

Note that the SFF need not support a BGP stack itself; it can proxy BGP messages to the controller which will support such a stack.

## [10.](#) BGP Extended Communities

### [10.1.](#) Route-Target Record

Route-Target Record (RT-Record) is defined as a transitive BGP Extended Community, that contains an Route-Target value representing one of the RTs that the route has been attached with previously, and which may no longer be attached to the route on subsequent re-advertisements (see [Section 2.8.5](#)).

A Sub-Type code 0x13 is assigned in the three BGP Extended Community types - Two-Octet AS-Specific 0x00, IPv4-Address-Specific 0x01 and Four-Octet AS-Specific 0x02. A Sub-Type code 0x0013 is also assigned in the BGP Transitive IPv6 Address-Specific Extended Community.

The Extended Community is encoded as follows:

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x00,0x01,0x02| Sub-Type=0x13 |   Route-Target Value           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Route-Target Value contd.      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Type field of the BGP Route-Target Extended Community is copied into the Type field of the RT Record Extended Community.

The Value field (Global Administrator and Local Administrator) of the Route-Target Extended Community is copied into the Route-Target Value field of the RT Record Extended Community.

When comparing a RT-Record to a Route-Target, only the Type and the Route-Target value fields are used in the comparison. The sub-type field is masked out.



When a speaker re-originates a route that contains one or more RTs, it must add each of these RTs as RT Record extended communities in the re-originated route.

A speaker must not re-originate a route with an RT, if this RT is already present as an RT Record extended community.

## [10.2.](#) Consistent Hash Sort Order

Consistent Hash Sort Order is an optional transitive Opaque BGP Extended Community of type 0x14, defined as follows:

Type Field : The value of the high-order octet is 0x03 (transitive opaque). The value of the low-order octet is assigned as 0x14 by IANA from the Transitive Opaque Extended Community Sub-Types registry.

Value Field : The value field contains a Sort Order sub-field that indicates the relative order of this route among the ECMP set for the prefix, to be sorted in increasing order. It is a 32-bit unsigned integer. The field is encoded as shown below:

```
+-----+
| Sort Order (4 octets)      |
+-----+
| Reserved (2 octets)       |
+-----+
```

## [10.3.](#) Load Balance Settings

Consistent Hash Sort Order is an optional transitive Opaque BGP Extended Community of type 0x14, defined as follows:

Type Field : The value of the high-order octet is 0x03 (transitive opaque). The value of the low-order octet is assigned as 0xaa by IANA from the Transitive Opaque Extended Community Sub-Types registry.

Value Field : The value field contains flags that indicate which

values in an IP packet's 5-tuple should be used as inputs to the ECMP hash algorithm. The field is encoded as shown below:

```

* 0                               1                               2                               3
* 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
* +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
* | Type  0x03      | Sub-Type 0xaa | s d c p P R R R | R R R R R R R R |
* +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
* | Reserved      | B R R R R R R R | Reserved      | Reserved      |
* +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
*
* Type: 0x03 Opaque
* SubType: 0xAA LoadBalance attribute information (TBA)
* s: Use l3_source_address ECMP Load-balancing
* d: Use l3_destination_address ECMP Load-balancing
* c: Use l4_protocol ECMP Load-balancing
* p: Use l4_source_port ECMP Load-balancing
* P: Use l4_destination_port ECMP Load-balancing
* B: Use source_bias (instead of ECMP load-balancing)
* R: Reserved

```

## 11. Summary and Conclusion

The architecture for service function chains described in this document uses virtual networks implemented as overlays in order to create service function chains. The virtual networks use standards-based encapsulation tunneling, such as MPLS over GRE/UDP or VXLAN, to transport packets into an SFC and between service function instances without routing in the user address space. Two methods of installing routes to form service chains are described.

In environments with physical routers, a controller may operate in tandem with existing BGP route reflectors, and would contain the SFC topology model, and the ability to install the local static interface routes to SF instances. In a virtualized environment, the controller can emulate route reflection internally and simply install required routes directly without advertisements occurring.

## 12. Security Considerations

The security considerations for SFCs are broadly similar to those concerning the data, control and management planes of any device placed in a network. Details are out of scope for this document.

### 13. IANA Considerations

The new BGP Extended Communities in are assigned types as defined above in the IANA registry for extended communities.

### 14. Acknowledgments

The authors would like to thank D. Daino, D.R. Lopez, D. Bernier, W. Haeffner, A. Farrel, L. Fang, and N. So, for their contributions to the earlier drafts. The authors would also like to thank the following individuals for their review and feedback on the original proposals: E. Rosen, J. Guchard, P. Quinn, P. Bosch, D. Ward, A. Ganesan, N. Seth, G. Pildush and N. Bitar. The authors also thank Wim Henderickx for his useful suggestions on several aspects of the draft.

### 15. References

#### 15.1. Normative References

- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", [RFC 3031](#), DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC4023] Worster, T., Rekhter, Y., and E. Rosen, Ed., "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", [RFC 4023](#), DOI 10.17487/RFC4023, March 2005, <<https://www.rfc-editor.org/info/rfc4023>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", [RFC 4271](#), DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", [RFC 4360](#), DOI 10.17487/RFC4360, February 2006, <<https://www.rfc-editor.org/info/rfc4360>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", [RFC 4364](#), DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", [RFC 4760](#), DOI 10.17487/RFC4760, January 2007,

- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", [RFC 5575](#), DOI 10.17487/RFC5575, August 2009, <<https://www.rfc-editor.org/info/rfc5575>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", [RFC 7348](#), DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", [RFC 7510](#), DOI 10.17487/RFC7510, April 2015, <<https://www.rfc-editor.org/info/rfc7510>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", [RFC 7665](#), DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC8365] Sajassi, A., Ed., Drake, J., Ed., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)", [RFC 8365](#), DOI 10.17487/RFC8365, March 2018, <<https://www.rfc-editor.org/info/rfc8365>>.

## 15.2. Informational References

### [consistent-hash]

Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., and D. Lewin, "'Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot

Spots on the World Wide Web", 1997, <Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing. ACM Press New York, NY, USA. pp. 654--663>.

[[draft-ietf-idr-link-bandwidth](#)]

Mohapatra, P. and R. Fernando, "BGP Link Bandwidth Extended Community", March 2018.

Fernando, et al.

Expires June 7, 2019

[Page 39]

---

Internet-Draft

SFC using Virtual Networks with BGP

December 2018

[[draft-malhotra-bess-evpn-unequal-lb](#)]

Malhotra, N., Sajassi, A., Rabadan, J., Drake, J., Lingala, A., and S. Thoria, "Weighted Multi-Path Procedures for EVPN All-Active Multi-Homing", June 2018.

[flowspec-redirect-ip]

Uttaro, J., Haas, J., Texier, M., Karch, A., Sreekanth, A., Ray, S., Simpson, A., and W. Henderickx, "BGP Flow-Spec Redirect to IP Action", February 2015.

[idr-tunnel-encaps]

Rosen, E., Patel, K., and G. van de Velde, "The BGP Tunnel Encapsulation Attribute", February 2018.

[NFVE2E] ETSI, "Network Functions Virtualisation (NFV): Architectural Framework", 2013.

#### Authors' Addresses

Rex Fernando  
Cisco Systems  
170 W. Tasman Drive  
San Jose, CA 95134

Email: [rex@cisco.com](mailto:rex@cisco.com)

Stuart Mackie  
Juniper Networks  
1133 Innovation Way  
Sunnyvale, CA 94089

Email: wsmackie@juniper.net

Dhananjaya Rao  
Cisco Systems  
170 W. Tasman Drive  
San Jose, CA 95134

Email: dhr Rao@cisco.com

Bruno Rijsman

Email: brunorijsman@gmail.com

Fernando, et al.

Expires June 7, 2019

[Page 40]

---

Internet-Draft

SFC using Virtual Networks with BGP

December 2018

Maria Napierala  
ATT Labs  
200 Laurel Avenue  
Middletown, NJ 07748

Email: mnapierala@att.com

Thomas Morin  
Orange  
2, Avenue Pierre Marzin  
Lannion, France 22307

Email: thomas.morin@orange.com

