Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: April 17, 2015

S. Mackie
B. Risjman
Juniper Networks
M. Napierala
AT&T
D. Daino
Telecom Italia
D.R. Lopez
Telefonica I+D
D. Bernier
Bell Canada
W. Haeffner
Vodafone

October 17, 2014

Service Function Chains Using Virtual Networking
draft-mackie-sfc-using-virtual-networking-01.txt

Status of this Memo

Copyright Notice

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.

The architecture uses a controller to calculate and install routes to implement an SFC, based on a topological model of the chain and knowledge of the network addresses that should pass through the chain. An advantage of the approach is that SFCs can be implemented without alteration to today's BGP standard, and without change to the current operation of routers.

Service chains need to support load balancing between network functions, and symmetric forward and reverse paths are required when stateful services are involved. This document shows how these requirements can be met by using VRFs at the ingress and egress of each service instance and by performing load balancing after the egress of each service as part of the routing function.

Table of Contents

1. Introduction

The purpose of networks is to allow computing systems to communicate with each other. Traditionally, requests are 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 itself 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 virtual networks

- o 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.

When BGP signaling and MPLS labeling are used, service function chains can be introduced to today’s networks, using existing network equipment and avoiding the need to introduce new network protocols, modify existing protocols or develop device enhancements.

Virtualized environments can be supported without running BGP or MPLS natively in data centers by encapsulating BGP messages in a more generally deployed protocol, such as XMPP, and by using GRE or VXLAN encapsulation for transport.

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.

The architecture 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 follows some of the terminology used in [draft-quinn-sfc-arch] 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: Locally applied customer/network/service policy used to identify and select traffic flow(s) requiring appropriate 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 via 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.

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

Service Function (SF): A function that is responsible for specific treatment of received 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.

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

2. Service Function Chain Architecture Using Virtual Networking

The architecture described in this document uses virtual networks managed with a controller to implement service function chains. Service function chains can be implemented on devices that support today's MPLS VPN and BGP standards [RFC4364, RFC4271, RFC4760], without requiring additional feature development, but other encapsulations, such as VXLAN [draft-mahalingam-vxlan], could be used, and use of other control plane protocols is possible.

The following sections detail the building blocks of the architecture, and outlines the process of route installation by the controller to create an SFC. A detailed, worked example of SFC instantiation that includes routing table and packet flow details is contained in Appendix A.

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 .

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.

```
                                         +-------------------------+
                                         |--- Data plane connection|
                                         |=== Encapsulation tunnel |
                                         | O  VRF                  |
                                         +-------------------------+

    Control       +------------------------------------------------+
    Plane         |                   Controller                   |
    .......       +-+------------+----------+----------+---------+-+
                    |            |          |          |         |
    Service         |    +---+   |  +---+   |  +---+   |         |
    Plane           |    |SF1|   |  |SF2|   |  |SF3|   |         |
                    |    +---+   |  +---+   |  +---+   |         |
    .......        /      | |   /    | |   /    | |   /         /
            +-----+    +--|-|--+  +--|-|--+  +--|-|--+   +-----+
            |     |    |  | |  |  |  | |  |  |  | |  |   |     |
    Net-A-->---O==========O O========O O========O O=========O---->Net-B
            |     |    |       |  |       |  |       |   |     |
    Data    | R-A |    |  R-1  |  |  R-2  |  |  R-3  |   | R-B |
    Plane   +-----+    +-------+  +-------+  +-------+   +-----+

               ^          ^ ^                               ^
               |          | |                               |
               |    Ingress Egress                          |
               |      VRF    VRF                            |
            SFC Entry                                    SFC Exit
               VRF                                         VRF
```

Figure 1- High level SFC Architecture

The diagram shows an SFC that traffic from Network-A destined for Network-B will pass through. Routing system R-A contains a VRF (shown as “O” symbol) that is the SFC entry point. This VRF will advertise a route to 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 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.

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.

2.2. Summary of Operation

The controller installs forwarding entries or distributes routes to each ingress VRF to direct traffic into the ingress interfaces of the service instances that will form an SFC. Overlay networking is used to transport traffic between the egress VRF of one service instance and the ingress VRF of the next.

Traffic may be directed into an SFC via routing as shown in the diagram, or a classifier may be used to select an SFC entry point based on filter criteria (see Section).

The controller, which functions as an extended BGP route reflector, exchanges routes between routing systems, which can be physical devices (conventional router) or be running as the forwarding function of a hypervisor or Host OS (VPN forwarder). The controller connects service instances of different services to each other by creating VPNs based on route targets in advertised routes. The controller directs traffic into SF instance ingress interfaces by installing particular routes based on the controller's knowledge of connectivity of service instances to VRFs, and the desired SFC topology. In most cases, a controller will configure forward and reverse SFCs at the same time, and this will result in routes being installed in the VRFs on both sides of each SF instance in a symmetrical SFC set.

As will be described in detail in this document, the architecture allows the logical connection between two service functions to be described by a virtual private network, and to be implemented using well-known MPLS L3 VPN technology. The forwarding into service function instances, and between service function instances, is performed by VRFs, which provide convenient containers within which to specify traffic forwarding polices, such as load balancing, and QoS shaping and policing. The controller maintains a global SFC topological model, and the VRFs are configured only with next hops to locally connected service instances, and to other routing systems that are connected to instances of the next service in the SFC.

This architecture does not rely on any new metadata header to be carried with the user packets for steering in an SFC [draft-boucadair-sfc-arch, draft-quinn-sfc-nsh], although metadata for use inside service functions would be supported [draft-rijsman-sfc-metadata-considerations].

2.3. 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.

```
                      +------+   +------+   +------+
         Network-A-->| SF-1 |-->| SF-2 |-->| SF-3 |-->Network-B
                      +------+   +------+   +------+
```

Figure 2- A Chain of Service Functions

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 [draft-boucadair-sfc-arch], a service function chain is uni-directional, while in [draft-quinn-sfc-arch] SFCs can be unidirectional 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.

2.4. 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 3, and may support branching across multiple egress interfaces and/or 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.

```
   +-------------------------------------------------------------------+
   | Logical Service Functions Connected in a Chain                    |
   |                                                                   |
   |                +--------+           +--------+                    |
   |       Net-A--->|  SF-1  |----------->|  SF-2  |--->Net-B         |
   |                +--------+           +--------+                    |
   |                                                                   |
   +-------------------------------------------------------------------+
   | Service Function Instances Connected by Virtual Networks          |
   |       ......                ......                               |
   |      :      :   +------+   :      :                              |
   |      :      :-->|SFI-11|-->:      :               ......         |
   |      :      :   +------+   :      :   +------+   :      :        |
   |      :      :              :      :-->|SFI-21|-->:      :        |
   |      :      :   +------+   :      :   +------+   :      :        |
   |   A->: VN-1 :-->|SFI-12|-->: VN-2 :              : VN-3 :-->B    |
   |      :      :   +------+   :      :   +------+   :      :        |
   |      :      :              :      :-->|SFI-22|-->:      :        |
   |      :      :   +------+   :      :   +------+   :      :        |
   |      :      :-->|SFI-13|-->:      :               ''''''         |
   |      :      :   +------+   :      :                              |
   |       ''''''                ''''''                               |
   +-------------------------------------------------------------------+
```

Figure 3- Service Functions Are Composed of SF Instances Connected Via Virtual Networks

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 (MPLS VPN).

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.

2.5. SF Instance Connections to VRFs

SF instances can be deployed as software running on physical appliances, or in virtual machines running on a hypervisor.

2.5.1. SF Instance in Physical Appliance

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

```
              +---------------------------------+
              |                                 |
              | +-----------------------------+ |
              | | Service Function Instance   | |
              | +-------^-------------|-------+ |
              |         |   Host      |         |
              +---------|-------------|---------+
                        |             |
                +------ |-------------|-------+
                |       |             |       |
                |  +----|----+  +-----v----+  |
          ---------+ Ingress |  | Egress   +---------
          --------->   VRF   |  |   VRF    ---------->
          ---------+         |  |          +---------
                |  +---------+  +----------+  |
                |        Routing System       |
                +-----------------------------+
```

Figure 4- Ingress and Egress VRFs for a Physical Routing System and Physical SF Instance

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. 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.

2.5.2. 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 coresident 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.

```
          +-------------------------------------+
          |   +-----------------------------+   |
          |   | Service Function Instance   |   |
          |   +-------^-------------|-------+   |
          |           |             |           |
          | +---------|-------------|---------+ |
          | | +-------|-------------|-------+ | |
          | | |       |             |       | | |
          | | |  +----|----+  +-----v----+  | | |
     ------------+ Ingress |  |  Egress  +-----------
     ------------>   VRF   |  |   VRF    ------------>
     ------------+         |  |          +-----------
          | | |  +---------+  +----------+  | | |
          | | |       Routing System        | | |
          | | +-----------------------------+ | |
          | |           Hypervisor            | |
          | +---------------------------------+ |
          |                Host                 |
          +-------------------------------------+
```

Figure 5- Ingress and Egress VRFs for a Virtual Routing System and Virtualized SF Instance

When more than one SF instance is running on a hypervisor, they can be connected to the same VRF for scale out of an SF within an SFC, or to different VRFs if the SF instances implement different service functions in the same SFC or when they belong to different SFCs.

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 implementation of SFCs described in this document. Physical and virtual service functions can be mixed as needed with different combinations of physical and virtual routing systems.

2.6. 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. Tunneling is enabled between all systems that have VRFs that share route targets.

The tunnels can be MPLS over GRE [RFC4023], MPLS over UDP [draft-ietf-mpls-in-udp], MPLS over MPLS [RFC3031], VXLAN [draft-mahalingam-vxlan], or another suitable encapsulation method.

Tunneling may be enabled in each routing system as part of a base configuration, or may be configured by the controller.

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.

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.

2.7.1. Controller for SFC with Physical Routers

In physical devices, a configuration interface (e.g. Netconf [RFC6241]) is used to create and configure VRFs, while BGP signaling is used to advertise route updates.

When physical routers that support BGP are used to connect SF instances in an SFC, the controller acts as a conventional BGP route-reflector with BGP sessions to each routing system. The controller has the additional capability to install routes whose next hop is a local interface name. These routes are needed to enable traffic to flow into the SF instances. These routes cannot be installed using BGP, and here it is assumed that Netconf will be used to make these changes. The routes are calculated based on the topological model of the desired SFC which is mapped onto specific SF instances whose connectivity to VRFs is also known by the controller.

2.7.2. SFC Implementation Procedure with Physical Routers

The following is a summary of the steps involved in creating an SFC using physical routers that fully support BGP. The service function instances are assumed to be already connected to logical interfaces on the router. A linear service chain is assumed in this example with each service function implemented in SF instances each with two interfaces.

A request to create a new SFC is made using an API or GUI. The request will include the functions in the chain, their order along the chain, and indication of the scale requirement for each function. The request will also identify the networks connected to each end of the SFC. The number of instances of each function may be calculated by the controller based on capacity or performance requirements, or may be specified in the creation request.

The service instances to be used in the SFC may be specified in the creation request, or may be selected by the controller from a set of available service instances.

The process of SFC creation is as follows:

1. Controller creates a VRF (via Netconf) in each router that is connected to a service instance that will be used in the SFC

2. Controller configures each VRF (via Netconf) to contain the logical interface that connects to a SF instance.

3. Controller implements route target import and export policies in the VRFs (via Netconf) using the same route targets for ingress and egress VRFs of connected services in the SFC.

4. Controller installs (via Netconf) 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.

5. Routing systems advertise the static routes back to the controller (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. Controller sends route updates (via BGP) to all routers containing VRFs with matching route targets.

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.

Traffic entering a VRF at one end of the SFC will be directed sequentially through a set of SF instances that implement each service function in the SFC.

If multiple SF instances are deployed for a given service function, the VRFs at the egress of the previous service function will load balance across the ingress interfaces of the SF instances of the next service function. Load balancing is described in Section .

The forward and reverse direction SFCs in an SFC set may be configured at the same time in the above procedure. For a symmetric SFC set, each VRF will be an ingress VRF in one direction, and an egress VRF in the other direction.

A detailed example is described in Appendix A.

2.7.3. Controller for SFC with Virtualized Routing

The process of VRF creation and route exchange is different for virtualized routing systems that implement only a VPN forwarding function and do not support a full BGP implementation.

For example, in [draft-ietf-l3vpn-end-system], the actions of BGP and Netconf are performed using XMPP in a virtualized environment. The content of the XMPP messages from the controller corresponds to configuration commands in Netconf and to route update messages in BGP when physical devices are used.

The controller of [draft-ietf-l3vpn-end-system] has a global model of all the routing systems under its control, including VRFs, interfaces, and route targets. The controller calculates all required routes based on dynamically assigned route targets and on import/export policies, and sends instructions to hypervisors/Host OS via XMPP to program the forwarding plane. The only route updates sent from a hypervisor/Host OS to the controller occur when a virtual machine is created, destroyed, or failed.

In [draft-ietf-l3vpn-end-system] the controller is able to peer via BGP with routers and route reflectors for route exchange with physical networks.

2.7.4. SFC Implementation Procedure with Virtual Routers

The following is a summary of the steps involved in creating an SFC using this architecture when VPN forwarders are used in hypervisors or Host OS’s, and the controller implements internal BGP emulation.

A request to create a new SFC is made using an API or GUI. The request will include the functions in the chain, their order along the chain, and indication of the scale requirement for each function. The request will also identify the networks connected to each end of the SFC. The number instances of each function may be calculated by the controller based on capacity or performance requirements or specified by the in the creation request.

The service instances to be used in the SFC may be specified in the creation request, or may be created, as needed, by the controller on suitable hosts from a set of available images.

The process of SFC creation is as follows

1. The controller creates service function instances as virtual machines running on hypervisors. The placement of each SF instance on a specific server can be according to an algorithm that takes into account geography, server type, data center environment (e.g. power supply, rack location) and other criteria, but details are out of scope for this document.

2. Controller creates an internal model of VRFs that each SF instance will connect to.

3. Controller adds the hypervisor interfaces that SF instances are connected to into each VRF in its internal model.

4. Controller instantiates the route target import and export policies in the VRFs in its internal model.

5. Controller, in its internal model, 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 network that will ultimately be reached by passing through the SF instance.

6. Controller calculates the effect of route reflection for the static routes in its model, which results in the controller installing routes in modeled VRFs with matching route targets. These routes have next hops pointing to the VRFs containing static routes. These new routes will direct traffic from the egress of one SF instance to the ingress of the next when installed into the actual VRFs in the infrastructure. In a hybrid physical/virtual SFC the controller would also perform actual route reflection with peered routers that host ingress and egress VRFs of SFCs.

7. Controller creates actual VRFs in the virtual routing systems in hypervisors/Host OS where SF instances reside.

8. The static and reflected routes that were calculated in the controller are sent to the VRFs and installed in their forwarding tables. The egress VRF for each SF instance will now contain VPN routes to one or more systems with ingress VRFs of SF instances of the next service function in the SFC.

Traffic entering a VRF at one end of the SFC will be directed sequentially through a set of SF instances that implement each service function in the SFC.

If multiple SF instances are deployed for a given service function, the VRFs at the egress of the previous service function will load balance across the ingress interfaces of the SF instances of the next service function. Load balancing is described in Section .

2.8. A Variation on Setting Prefixes in an SFC

In the configuration method 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 describe 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 2, Network-A can be a prefix that includes all subscriber IP addresses and 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.

2.9. 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.

An example of this technique, combined with the prefix variation of the previous section, is provided in Appendix

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. 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 must have a unique route distinguisher.

```
                      +------+             +-------------------------+
                 O----|SFI-11|---O         |--- Data plane connection|
                //    +------+   \\        |=== Encapsulation tunnel |
               //                 \\       | O  VRF                  |
              //                   \\      | *  Load balancer        |
             //                     \\     +-------------------------+
            //        +------+       \\
   Net-A-->O*====O----|SFI-12|---O====O-->Net-B
            \\        +------+       //
             \\                     //
              \\                   //
               \\                 //
                \\    +------+   //
                 O----|SFI-13|---O
                      +------+
```

Figure 6 – Load Balancing across SF Instances Connected to Different VRFs

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.

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.

```
                      +------+             +-------------------------+
                      |SFI-11|             |--- Data plane connection|
                      +------+             |=== Encapsulation tunnel |
                     /        \            | O  VRF                  |
                    /          \           | *  Load balancer        |
                   /            \          +-------------------------+
                  /   +------+   \
   Net-A-->O====O*----|SFI-12|----O====O-->Net-B
                  \   +------+   /
                   \            /
                    \          /
                     \        /
                      +------+
                      |SFI-13|
                      +------+
```

Figure 7 – Load Balancing across SF Instances Connected to the Same VRF

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.

3.3. Combination of Egress and Ingress VRF Load Balancing

In Figure 8, below, an example SFC is shown where load balancing is performed in both ingress and egress VRFs.

```
                                        +-------------------------+
                                        |--- Data plane connection|
                       +------+         |=== Encapsulation tunnel |
                       |SFI-11|         | O  VRF                  |
                       +------+         | *  Load balancer        |
                      /        \        +-------------------------+
                     /          \
                    /  +------+  \          +------+
                  O*---|SFI-12|---O*====O---|SFI-21|---O
                 //    +------+    \\  //   +------+   \\
                //                  \\//                \\
               //                    \\                  \\
              //                    //\\                  \\
             //        +------+    //  \\   +------+       \\
    Net-A-->O*====O----|SFI-13|---O*====O---|SFI-22|---O====O-->Net-B
                       +------+             +------+
           ^      ^               ^     ^              ^    ^
           |      |               |     |              |    |
           |    Ingress         Egress  |              |    |
           |                          Ingress       Egress |
        SFC Entry                                        SFC Exit
```

Figure 8 – Load Balancing across SF Instances

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].

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. 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 a SF instance set. In this case, each packet’s 5-tuple data can be used in a hashing function, provided the source and destination IP address and port information are swapped in the reverse calculation and that the same or no hash salt is used for both directions. This method only requires that the list of available service function instances is consistently maintained in all the load balancers, rather than maintaining a distributed flow table.

In the SFC architecture described in this document, when SF instances are added or removed, the controller is required to configure (or remove) static 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 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.

3. The VRF creates a new flow entry for the 5-tuple 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 reverse flow entry for the 5-tuple with next-hop of the tunnel on which traffic arrived.

7. The packet is sent into the SF instance connected to the ingress VRF.

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.

4. 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 include the IP 5-tuple of the header, and/or analysis of the payload of packets using deep packet inspection. Integration with a subscriber management system such as PCRF or AAA will usually 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.

```
                   +----------+
                   | PCRF/AAA |
                   +-----+----+
                         :
                         :
   Subscriber      +-----+------+
      Traffic----->| Classifier |
                   +------------+
                       |   |
               +-------|---|------------------------+
               |       |   |             Router     |
               |       |   |                        |
               |       O   O                  X--------->Internet
               |       |   |                 / \    |
               |       |   |                O   O   |
               +-------|---|----------------|---|---+
                       |   |  +---+   +---+ |   |
                       |   +--+ U +---+ V +-+   |
                       |      +---+   +---+     |
                       |                        |
                       |  +---+   +---+   +---+ |
                       +--+ X +---+ Y +---+ Z +-+
                          +---+   +---+   +---+
```

Figure 9- Subscriber/Application-Aware Steering with a Classifier

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 O in the diagram).

In this scenario, 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).

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 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.

5. Controller Federation

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 the network domains.

When SFCs are distributed geographically, or in very large-scale environments, there may be multiple SFC controllers present. 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 and allow such domain spanning SFCs to be created.

6. 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 or VXLAN, to transport packets into an SFC and between service function instances without routing in the user address space. The controller contains a topological model of the SFC that includes the connections from SF instances to routing systems and the required connectivity between service functions. VRFs with common route targets are used to define the virtual networks that connect sets of SF instances to form the SFC. The SF instances are linked to VRFs by installing static routes that direct traffic through the SF instances. BGP route-reflection is used to distribute these routes and form service function chains.

The architecture can be implemented on today’s routers without modification to existing signaling protocols, and without modification to the operation of the routers themselves. There is no requirement for a service chain header to be added to packets, and no requirement for any service chain topology, other than next hops, to be sent to routing systems.

In environments with physical routers, the 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 refection internally and simply install required routes directly without advertisements occurring.

7. 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.

8. IANA Considerations

There are no IANA considerations.

9. References

9.1. Normative References

None

9.2. Informative References

[NFVE2E] "Network Functions Virtualisation: End to End Architecture, http://docbox.etsi.org/ISG/NFV/70-DRAFT/0010/NFV-0010v016.zip".

[RFC2328] J. Moy, “OSPF Version 2”, RFC 2328, April, 1998.

[draft-quinn-sfc-arch]
Quinn, P. and A. Beliveau, "Service Function Chaining (SFC) Architecture", draft-quinn-sfc-arch-04 (work in progress), January 2014.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

[draft-mahalingam-vxlan] M. Mahalingam, et al. "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks.", draft-mahalingam-dutt-dcops-vxlan-08, February 3, 2014.

[draft-bouadair-sfc-arch] Boucadair, M., and Jacquenet, C., "Service Function Chaining: Framework & Architecture", draft-boucadair-sfc-framework-02, February 2014.

[draft-quinn-sfc-arch]
P. Quinn, at al, "Service Function Chaining (SFC) Architecture", draft-quinn-sfc-arch-04, January 28, 2014.

[draft-ietf-l3vpn-end-system]
P. Marques et al., “BGP-signaled end-system IP/VPNs”, draft-ietf-l3vpn-end-system, October 21, 2013.

[draft-quinn-sfc-nsh]
Quinn, P., et al, "Network Service Header", draft-quinn-sfc-nsh-01, February 2014.

[draft-niu-sfc-mechanism]
Niu, L., Li, H., and Jiang, Y., "A Service Function Chaining Header and its Mechanism", draft-niu-sfc-mechanism-00, January 2014.

[draft-rijsman-sfc-metadata-considerations]
B. Rijsman, et al. “Metadata Considerations”, draft-rijsman-sfc-metadata-considerations-00, February 12, 2014

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC4023] Worster, T., Rekhter, Y., and E. Rosen, "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, March 2005.

[draft-ietf-mpls-in-udp]
Building, K., Sheth, N., Yong, L., Pignataro, C., and F. Yongbing, "Encapsulating MPLS in UDP", draft-ietf-mpls-in-udp-03 (work in progress), September 2013.

[RFC3031] Rosen. E, Viswanathan A, Callon R, “Multiprotocol Label Switching Architecture”, January 2001.

[draft-ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture, work in progress.

[consistent-hash]
Karger, D.; Lehman, E.; Leighton, T.; Panigrahy, R.; Levine, M.; Lewin, D. (1997). "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web". 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]
P. Mohapatra, R. Fernando, “BGP Link Bandwidth Extended Community”, draft-ietf-idr-link-bandwidth, work in progress.

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

The authors would like to thank Nischal Seth and Galina Pildush of Juniper Networks, and Nabil Bitar of Verizon for their review and comments.

Appendix A.SFC Implementation in a Worked Example

A simple service function chain with just one instance of one service function in it will be used in order to describe, in detail, the actions of the control and forwarding planes in this architecture. The example will then be generalized to show how more complex SFCs with multiple service functions, each with multiple SF instances, can be implemented.

It is assumed, in the example, that the routing systems in the network fully support BGP route signaling (receiving advertisements and sending updates), and that Netconf can be used for routing system configuration. Netconf (or other configuration protocol) is required because the current operation of BGP does not support installation of local routes using an interface identifier as a next hop.

As described in Section 7.3, a controller in a virtualized environment can calculate required routes without receiving route advertisements, and the resulting installed routes will be the same as described below.

A.1.   Desciption of Example SFC Topology

The example SFC is shown in Figure 10, below. Two service functions are connected serially, and each is implemented in one service instance. Traffic flowing between Network-A and Network-B should pass through SFI-1 and SFI-2.

```
                            +------------+
                            | Controller |
                            +------------+
                                  |
                             BGP/Netconf
                                  |
                                  v
                  +-----+                     +-----+
                +-|SFI-1|-+                 +-|SFI-2|-+
                | +-----+ |                 | +-----+ |
           +----|---------|----+       +----|---------|-----+
           |    |         |    |       |    |         |     |
           | +------+  +------++-------+ +------+  +------+ |
           | |VRF-11|  |VRF-12|--------->|VRF-21|  |VRF-22| |
           | +------+  +------++-------+ +------+  +------+ |
           |   |^|  R-2        |       |       R-3   |||    |
           +---|||-------------+       +-------------|||----+
               |||                                   |||
           +---|||---+                           +---|||---+
           |   |||   |                           |   |v|   |
           | +-----+ |                           | +-----+ |
    NetA---->|VRF-A| |                           | |VRF-B|----->NetB
           | +-----+ |                           | +-----+ |
           |   R-1   |                           |   R-4   |
           +---------+                           +---------+
```
Figure 10– Service Chain with Two Service Functions

The SFC in the diagram is one half of an SFC set that allows traffic in both directions to pass through the service functions. In the example below, the routes for both directions will be described.

A.1.1. Connectivity in the Example

Routing systems R-1, R-2, R-3 and R-4 are in the same autonomous system and have IP connectivity with each other. The routing systems may be directly connected, or be connected via an Ethernet or IP transport network. The routing systems may be physical or virtual.

VRFs VRF-A, VRF-11, VRF-12, VRF-21, VRF-22 and VRF-B are created (using, for instance, Netconf) on routing systems R-1, R-2, R-3 and R-4.

Interface IF-NetA in system R-1 is present in VRF-A, and connects to Network-A. Similarly, IF-NetB in system R-4 is present in VRF-B, and connects to Network-B.

Service function instance SFI-1 is connected via interfaces on R-2 named IF-11 and IF-12, and which are present in VRF-11 and VRF-12, respectively.

Service function instance SFI-2 is similarly connected to VRF-21 and VRF-22 in routing system R-3.

VRF-A is the entry point into the service function chain and VRF-B is the exit point for traffic flowing from Network-A to Network-B.

The service function instances, SFI-1 and SFI-2 may be directly connected to routing systems R-2 and R-3, or may be connected via a L2 switching network.

In the example, encapsulation tunneling takes place between VRF-A and VRF-11, between VRF-12 and VRF-21, and between VRF-22 and VRF-B.

A.2. Forwarding Control

The forwarding of a packet from Network-A to a destination in Network-B through the service function chain is achieved by the installation by the controller of local routes in the ingress routing systems followed by route reflection to connected egress routing systems.

The combination of the specially installed routes together with conventional route reflection enables the required flow through the virtual networks and through the service function instances.

A.2.1. Initial Local Routes

The initial state is that each router contains routes to the other routers in its main IP routing table, and routes to Network-A and Network-B will be in the tables of the connected VRFs, VRF-A and VRF-B. The routes to Network-A and Network-B may be statically configured, or signaled via a routing protocol from systems in these networks or from gateways to these networks. The routes between the routers in the SFC will have been signaled by some interior gateway protocol, such as OSPF [RFC2328].

The following table shows the import and export policies that are configured when the SF instances are connected.

Routing System	VRF	Import Target	Export Target
R-1	VRF-A	RTA	RTA
R-2	VRF-11	RTA	RTA
R-2	VRF-12	RT12	RT12
R-3	VRF-21	RT12	RT12
R-3	VRF-22	RTB	RTB
R-4	VRF-B	RTB	RTB

These import and export policies allow route exchange between VRF-A and VRF-11 and between VRF-22 and VRF-B.

The following table shows the relevant local routes in the routing systems R-1, R-2, R-3 and R-4.

Routing System	VRF	Prefix	Next Hop
R-1	VRF-A	Network-A	IF-NetA
R-1	Global	R-2	IF-R-1-2
R-2	Global	R-1	IF-R-2-1
R-2	Global	R-3	IF-R-2-3
R-3	Global	R-2	IF-R-3-2
R-3	Global	R-4	IF-R-3-4
R-4	Global	R-3	IF-R-4-3
R-4	VRF-B	Network-B	IF-NetB

The term “Global” in the above table means that route is installed in the main routing table of the routing system.

A naming convention is used for the interfaces that connect routers, so, for instance, interface IF-R-1-2 is the interface on R-1 that connects to R-2.

A.2.2. Route Advertisements to Build Virtual Networks

In the standard way, each routing system sends advertisements for each of its local routes to the controller (acting as a route reflector here). The L3 VPN advertisements for the network in Figure 1 are shown in the following table.

Routing System	Advertised Prefix	Label	Route Distinguisher	Route Target
R-1	Network-A	Lbl-NetA	RDA	RTA
R-4	Network-B	Lbl-NetB	RDB	RTB

The labels are locally significant on each routing system, and are used to identify the interface that incoming labeled packets will be sent out on. [RFC4364].

Using standard route reflection, route updates are sent to each routing system, and routes are installed in VRFs with matching import route targets.

The following additional routes would be installed in the VRFs on R-2 and R-3:

Routing System	VRF	Prefix	BGP Next Hop
R-2	VRF-11	Network-A	push Lbl-NetA, encap GRE, R-1
R-3	VRF-22	Network-B	push Lbl-NetB, encap GRE, R-4

The first new route entry states that for packets with the prefix of Network-A received in VRF-11 in routing system R-2, the next hop action is to push the advertised label (Lbl-NetA), encapsulate with GRE, and send the GRE packet to router R-1. The destination R-1 will be resolved to interface IF-R-2-1 in the forwarding table. The second entry is similar, but for Network-B packets arriving in VRF-22.

At this point, virtual networks corresponding to route targets RTA and RTB exist, and VRF VRF-11 can reach Network-A, and VRF-12 can reach Network-B, but Network-A is not reachable from Network-B, or vice versa.

A.3. Enabling Forwarding into SF Instances by the Controller

In this architecture, the controller has a topological model of each SFC. The model contains the service function instances that implement the SFC together with the desired connectivity to VRFs via specific interfaces in the routing systems. The controller uses that information to determine which local routes to install, in order to have traffic from networks on one side of an SFC reach networks on the other side by passing through the SFC.

The controller uses Netconf to install the routes, since the next hop is an interface name, not an IP address.

A.4. Local Route Installation

A local route is installed in each VRF that connects to a SF instance. The local routes have the prefix of a connected network and a next hop that identifies a local interface that is connected to a SF instance.

In the example SFC, the controller sends the following local routes to routing system R-2:

Routing System	Prefix	Direct Next Hop	Route Dist	Route Target
R-2	Network-B	Local IF-11	RD11	RTA
R-2	Network-A	Local IF-12	RD12	RT12
R-3	Network-B	Local IF-21	RD21	RT12
R-3	Network-A	Local IF-22	RD22	RTB

Today, BGP does not support installation of routes containing an interface identifier as a next hop, and these would need to be installed via Netconf or other means, such as I2RS or XMPP [draft-ietf-i2rs-architecture, draft-ietf-l3vpn-end-system].

The following new routes will exist on R-2 and R-3:

Routing System	VRF	Prefix	Next Hop
R-2	VRF-11	Network-B	Local IF-11
R-2	VRF-12	Network-A	Local IF-12
R-3	VRF-21	Network-B	Local IF-21
R-3	VRF-22	Network-A	Local IF-22

Routing systems R-2 and R-3 will advertise new routes back to the controller since the interfaces are MPLS enabled and present in the VRFs connected to SF instances:

Routing System	Advertised Prefix	Label	Route Dist	Route Target
R-2	Network-B	Lbl-IF-11	RD11	RTA
R-2	Network-A	Lbl-IF-12	RD12	RT12
R-3	Network-B	Lbl-IF-21	RD21	RT12
R-3	Network-A	Lbl-IF-22	RD22	RTB

Having received these advertisements from R-2 and R-3, the controller, acting as route reflector, will send the following route updates:

Prefix	BGP Next Hop	Label	Route Dist	Route Target
Network-B	R-2	Lbl-IF-11	RD11	RTA
Network-A	R-2	Lbl-IF-12	RD12	RT12
Network-B	R-3	Lbl-IF-21	RD21	RT12
Network-A	R-3	Lbl-IF-22	RD22	RTB

These would be resolved in the various routing systems as follows:

Routing System	VRF	Prefix	Next Hop
R-1	VRF-A	Network-B	push Lbl-IF-11, encap GRE, R-2
R-2	VRF-12	Network-B	push Lbl-IF-21, encap GRE, R-3
R-3	VRF-21	Network-A	push Lbl-IF-12, encap GRE, R-2
R-4	VRF-B	Network-A	push Lbl-IF-22, encap GRE, R-3

The routes installed in each router will be as follows:

Routing System	VRF	Prefix	Next Hop
R-1	VRF-A	Network-A	IF-NetA
R-1	VRF-A	Network-B	push Lbl-IF-11, encap GRE, R-2
R-1	Global	R-2	IF-R-1-2
R-2	VRF-11	Network-B	local IF-12
R-2	VRF-11	Network-A	push Lbl-NetA, encap GRE, R-1
R-2	Global	R-1	IF-R-1-2
R-2	VRF-12	Network-A	local IF-21
R-2	VRF-12	Network-B	push Lbl-IF-21, encap GRE, R-3
R-2	Global	R-3	IF-R-2-3
R-3	VRF-21	Network-B	local IF-12
R-3	VRF-21	Network-A	push Lbl-IF-12, encap GRE, R-2
R-3	Global	R-2	IF-R-3-2
R-3	VRF-22	Network-A	local IF-21
R-3	VRF-22	Network-B	push Lbl-NetB, encap GRE, R-4
R-3	Global	R-4	IF-R-3-4
R-4	VRF-B	Network-A	push Lbl-IF-22, encap GRE, R-3
R-4	Global	R-3	IF-R-4-3
R-4	VRF-B	Network-B	IF-NetB

The MPLS tables will be as follows:

Routing System	Label	Action	Direct Next Hop
R-1	Lbl-NetA	pop	IF-NetA
R-2	Lbl-IF-11	pop	IF-11
R-2	Lbl-IF-12	pop	IF-12
R-3	Lbl-IF-21	pop	IF-21
R-3	Lbl-IF-22	pop	IF-22
R-4	Lbl-NetB	pop	IF-NetB

A connection is established between Network-A and Network-B with packets flowing through SF instances SFI-1 and SFI-2 as a result of adding the local routes that point to the interfaces in the routing systems that the SF instance is connected to is that.

A.5. Detailed Packet Flow

This section describes the details of how a packet is forwarded from Network-A to Network-B through the example SFC. The table, below, shows the routes for one direction only.

Routing System	VRF	Prefix	Next Hop
R-1	VRF-A	Network-B	push Lbl-IF-11, encap GRE, R-2
R-1	Global	R-2	IF-R-1-2
R-2	VRF-11	Network-B	local IF-12
R-2	VRF-12	Network-B	push Lbl-IF-21, encap GRE, R-3
R-2	Global	R-3	IF-R-2-3
R-3	VRF-21	Network-B	local IF-21
R-3	VRF-22	Network-B	push Lbl-NetB, encap GRE, R-4
R-3	Global	R-4	IF-R-3-4
R-4	VRF-B	Network-B	IF-NetB

The MPLS table entries for this direction are:

Routing System	Label	Action	Direct Next Hop
R-2	Lbl-IF-11	pop	IF-11
R-3	Lbl-IF-21	pop	IF-21
R-4	Lbl-NetB	pop	IF-NetB

The detailed packet forwarding, labeling and encapsulation for a packet from a host HostA in Network-A to HostB in Network-B would be as follows:

1. System R-1 is advertising a route for the prefix corresponding to Network-B into Network-A, or a gateway in Network-A has a route configured such that packets originating in Network-A with destination in Network-B will be sent to R-1.

2. An IP packet arrives at interface IF-NetA on system R-1 with a header containing HostB as the destination IP address and HostA as the source IP address.

3. Interface IF-NetA on R-1 is configured in VRF VRF-A, so the packets arrive in VRF-A, whose forwarding table directs it to push a label Lbl-IF-11 onto the packet and use GRE encapsulation to send the packet to R-2.

4. A GRE header is added and the packet now has the following structure:

```
+-----------------------------------------------------+
| Delivery Header (Source IP-R1, Destination=IP-R2) |
+-----------------------------------------------------+
| GRE Header (Ethertype=0x8847 (MPLS Unicast))      |
+-----------------------------------------------------+
+ MPLS Label (Lbl-IF-11)                            |
+-----------------------------------------------------+
| User IP Header (Source=HostA, Destination=HostB)  |
+-----------------------------------------------------+
| Payload                                           |
+-----------------------------------------------------+
```

   where IP-R1 and IP-R2 are the addresses used for the connection between R-1 and R-2. These could be loopback addresses, interface addresses or server addresses, depending on the implementation.

5. The encapsulated packet arrives at R-2. The outer header is stripped off together with the GRE header. The label is looked up in the MPLS table where the action is to pop the label and to send to IF-11.

6. The packet exits IF-11 on R-2 and arrives at the ingress interface of the SF instance SFI-1. The packet passes through SFI-1, where the internal logic of SFI-1 processes the packet information. Since, in this case, SFI-1 is a transparent service function, the packet emerges with the header unchanged (assuming the logic did not determine that the packet should be dropped).

7. The packet is sent from the egress interface of SFI-1 to interface IF-12 on system R-2, which is configured in VRF-12.

8. The forwarding table in VRF-12 directs that a label Lbl-IF-21 be pushed onto the packet header and GRE encapsulation is used to send the packet to R-3.

9. A GRE encapsulation header is added to the original IP header with source IP of IP-R2 and destination address IP-R3. The encapsulated packet is sent to R-3.

10. The packet arrives at R-3, where the GRE header is removed, and the label Lbl-IF-21 is looked up in the MPLS table where the action is to pop the label and send out of interface IF-21.

11. The packet exits IF-21 on R-3 and arrives at the ingress interface of the SF instance SFI-2. The packet passes through SFI-2, where the internal logic of SFI processes the packet information.

12. The packet is sent from the egress interface of SFI-2 to interface IF-22 on system R-3, which is configured in VRF-22.

13. The forwarding table in VRF-22 directs that a label Lbl-NetB be pushed onto the packet header and GRE encapsulation is used to send the packet to R-4.

14. A GRE encapsulation header is added to the original IP header with source IP of IP-R3 and destination address IP-R4. The encapsulated packet is sent to R-4.

15. The packet arrives at R-4, where the GRE header is removed, and the label LBL-NetB is looked up in the MPLS table where the action is to pop the label and send to IF-NetB.

16. The packet is then sent with the original IP header out of interface IF-NetB on R-4 towards the destination HostB in Network-B.

Traffic flowing the reverse direction from HostB to HostA will follow the same path, but in reverse.

A.6. Extended SFCs

Following the pattern described above, when an SFC has more than two service functions, a different route target is used for the VRFs that connect each pair of SF instances. Routes pointing to the connected SF instance interface are installed in each VRF, and route refection causes the installation of corresponding routes for traffic entering VRFs in the same virtual network from the egress of a SF instance. The routes installed in R-2 and R-3 are essentially replicated for each connection between SF instances to form the SFC.

A.7. SFCs Using Routes with Expanded Prefixes

In a variation to the method described above, the routes in each direction can be set to prefixes that include all possible traffic.

In the example SFC, Network-A can be a prefix that includes all subscriber IP addresses and Network-B could set to the default route, 0/0.

With this variation, routes forwarding traffic into a SF instance and to the next SF instance are installed each time a SF instance is connected, but there is no requirement for VRFs to be reconfigured when traffic from new networks pass through the service chain.

If a classifier is used to direct traffic into SFCs, the same SFC configuration can be used in all instances of an SFC in a network, thus simplifying testing and deployment.

A.8. SFCs with Packet Transforming Service Functions

If a service function performs an action that changes the source address in the packet header, 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 that direct traffic into instances of the transforming service function. The original routes with a source prefix (Network-A in the example) 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.

The following table shows the routes that would be installed in the example SFC network if the second service function transformed the packet headers, and technique described in Section was employed. The subscriber address pool is Sub-Pool, and NAT-Pool is the public address space that the second service function uses.

Routing System	VRF	Prefix	Next Hop
R-1	VRF-A	Sub-Pool	IF-NetA
R-1	VRF-A	0/0	push Lbl-IF-11, encap GRE, R-2
R-1	Global	R-2	IF-R-1-2
R-2	VRF-11	0/0	local IF-12
R-2	VRF-11	Sub-Pool	push Lbl-NetA, encap GRE, R-1
R-2	Global	R-1	IF-R-1-2
R-2	VRF-12	Sub-Pool	local IF-21
R-2	VRF-12	0/0	push Lbl-IF-21, encap GRE, R-3
R-2	Global	R-3	IF-R-2-3
R-3	VRF-21	0/0	local IF-12
R-3	VRF-21	Sub-Pool	push Lbl-IF-12, encap GRE, R-2
R-3	Global	R-2	IF-R-3-2
R-3	VRF-22	NAT-Pool	local IF-21
R-3	VRF-22	0/0	push Lbl-NetB, encap GRE, R-4
R-3	Global	R-4	IF-R-3-4
R-4	VRF-B	NAT-Pool	push Lbl-IF-22, encap GRE, R-3
R-4	Global	R-3	IF-R-4-3
R-4	VRF-B	0/0	IF-NetB

Authors’ Addresses

Stuart Mackie
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
USA

Email: wsmackie@juniper.net

Bruno Rijsman
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
USA

Email: brijsman@juniper.net

Maria Napierala
AT&T Labs
200 Laurel Avenue
Middletown, NJ 07748

Email: mnapierala@att.com

Diego Daino
Telecom Italia
Via Guglielmo Reiss Romoli
274 – 10148 Turin
Italy

Email: diego.daino@telecomitalia.it

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Email: diego@tid.es

Daniel Bernier
Bell Canada
1 Carrefour Alexander-Graham Bell
Building A-7,Verdun
Quebec, H3E 3B3
Canada

Email: daniel.bernier@bell.ca

Walter Haeffner
Vodafone D2 GmbH
Ferdinand-Braun-Platz 1
Dusseldorf 40549
DE

Email: walter.haeffner@vodafone.com