Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 22, 2016

S. Aldrin
Google
R. Krishnan
Dell
N. Akiya
Big Switch
C. Pignataro
Cisco Systems
A. Ghanwani
Dell

July 23, 2015

Service Function Chaining Operation, Administration and Maintenance Framework draft-aldrin-sfc-oam-framework-02

Abstract

This document provides reference framework for Operations, Administration and Maintenance (OAM) for Service Function Chaining (SFC).

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 http://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 January 2016.

Copyright Notice

Table of Contents

Service Function Chaining (SFC) enables the creation of composite services that consist of an ordered set of Service Functions (SF) that are be applied to packets and/or frames selected as a result of classification. SFC is a concept that provides for more than just the application of an ordered set of SFs to selected traffic; rather, it describes a method for deploying SFs in a way that enables dynamic ordering and topological independence of those SFs as well as the exchange of metadata between participating entities. The foundations of SFC are described in the following documents:

- SFC problem statement [I-D.ietf-sfc-problem-statement]
- SFC architecture [I-D.ietf-sfc-archiecture]

The reader is assumed to familiar with the material in these drafts.

This document provides reference framework for Operations, Administration and Maintenance (OAM, [RFC6291]) of SFC. Specifically, this document provides:

- In Section 2, an SFC layering model;
- In Section 3, aspects monitored by SFC OAM;
- In Section 4, functional requirements for SFC OAM;
- In Section 5, a gap analysis for SFC OAM.

1.1. Document Scope

The focus of this document is to provide an architectural framework for SFC OAM, particularly focused on the aspect of the Operations component within OAM. Actual solutions and mechanisms are outside the scope of this document.

2. SFC Layering Model

Multiple layers come into play for implementing the SFC. These include the service layer at which SFC operates and the underlying Network, Transport, Link, etc., layers.

- The service layer, refered to as the "Service Layer" in Figure 1, consists of classifiers and SFs, and uses the

transport network, which could be an overlay network, from a classifier to SF and from one SF to the next.

- The network overlay transport layer, refer to as the "Network", "Transport" and layers below in Figure 1, extends between the various SFs and is mostly transparent to the SFs themselves. It can leverage various overlay network technologies interconnecting SFs and allows establishment of service function paths (SFPs).

- The link layer, refer to as the "Link" in Figure 1, is dependent upon the physical technology used. Ethernet is a popular choice for this layer, but other alternatives are deployed (e.g. POS, DWDM, etc.).

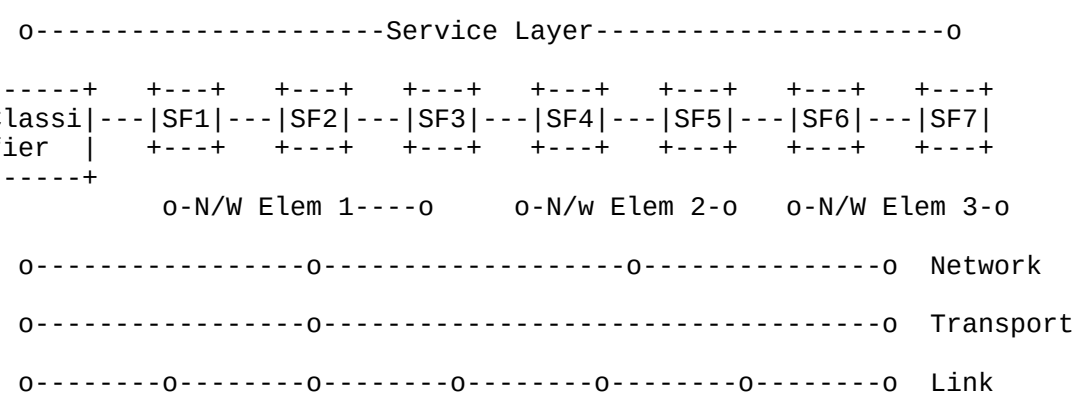


Figure 1: SFC Layering Example

3. Aspects Monitored by SFC OAM?

SFC operates at the service layer. For the purpose of defining the OAM framework, the following aspects of the SFC must be capable of monitored.

1. Service function:

SFs may be SFC-aware or SFC-unaware. An SFC-aware SF is one that understands the SFC encapsulation has the SFF component co-resident with the SF sub-component . An SFC-unware SF is one that does not understand the SFC encapsulation (i.e. a legacy SF) and has to be accessed via an separate SFF and potentially an SFC proxy function.

In both cases, an SF is accessed through an SFF in the SFC architecture. SFC OAM must be able to monitor the SFF associated with a given SF.

2. Service function path:

The SFP comprises a set of SFs that may be ordered or unordered. SFC OAM must be capable of monitoring the SFP and the rendered service path (RSP) that may be used by specific packets.

3. Classifier:

The classifier determines which packets are mapped to an SFP.
SFC OAM must be able to monitor the operation of the classifiers.

The figure below illustrates the various aspects monitored by SFC OAM.

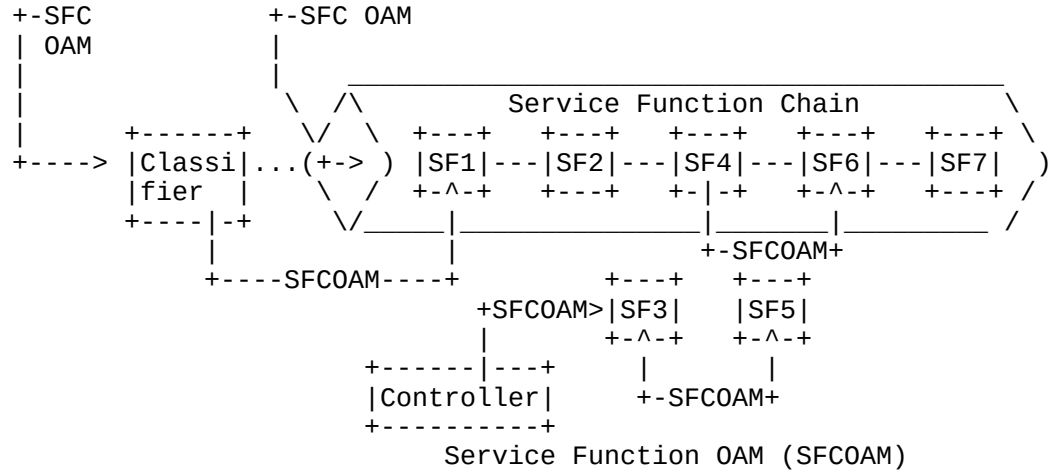


Figure 2: Aspects monitored by SFC OAM

3.1. Operation and Performance of SFs

3.1.1. Monitoring SF Operation

One SFC OAM requirement for the SF component is to allow an SFC aware network device to monitor a specific SF. This is accomplished by monitoring the SFF that the SF is attached to.

A generalized way to monitor the operation of an SF is beyond the scope of SFC OAM, because the functions provided by the SF are not covered by SFC. SFs typically provide their own tools for monitoring.

An optional capability may be provided for an SFF to monitor the operation of its attached SFs and report that on behalf of the SFs.

3.1.2. Service Function Performance Measurement

A second SFC OAM requirement for SF is to allow an SFC aware network device to check the loss and delay to a specific SF, located on the same or different network devices.

3.2. Operation and Performance of SFPs

3.2.1. Monitoring SFP Operation

SFC OAM must be capable of monitoring one or more SFPs or RSPs that are used to realize the SFC and reporting on connectivity and providing fault isolation.

In order to perform service connectivity verification of an SFP, the OAM tools could be initiated from any SFC-aware network device for end-to-end paths, or partial paths terminating on a specific SF, within the SFP. This OAM function is to ensure the SF's chained together has connectivity as it was intended to when SFP was established. Necessary return code(s) should be defined to be sent back in the response to OAM packet, in order to qualify the verification.

When ECMP exists at the service layer on a given SFC (e.g. multiple SFPs, or multiple RSPs), there must be an ability to discover and traverse all available paths.

3.2.2. Service Function Chain Performance Measurement

The ingress of the SFC or an SFC-aware network device must have an ability to perform loss and delay measurements over the SFC as a unit (i.e. end-to-end) or to a specific SF through the SFC.

3.3. Monitoring the Classifier

A classifier defines a flow and maps incoming traffic to a specific SFC, and it is vital that the classifier is correctly defined and functioning. SFC OAM must be able to test the definition of flows and the mapping functionality to expected SFCs.

4. SFC OAM Functions

Section 3 described the various aspects monitored by SFC OAM. This section explores the same from the OAM functionality point of view, which many will be applicable to multiple SFC components.

Various SFC OAM requirements provides the need for various OAM functions at different layers. Many of the OAM functions at different layers are already defined and in existence. In order to support SFC and SF's, these functions have to be enhanced to operate a single SF to multiple SF's in an SFC and also multiple SFC's.

4.1. Connectivity Functions

Connectivity is mainly an on-demand function to verify that the connectivity exists between network elements and that the SFs are operational. Ping is a common tool used to perform this function. OAM messages should be encapsulated with necessary SFC header and with OAM markings when testing the SFC component. OAM messages MAY be encapsulated with necessary SFC

header and with OAM markings when testing the SF component. Some of the OAM functions performed by connectivity functions are as follows:

- Verify the MTU size from a source to the destination SF or through the SFC. This requires the ability for OAM packet to take variable length packet size.
- Verify the packet re-ordering and corruption.
- Verify the policy of an SFC or SF using OAM packet.
- Verification and validating forwarding paths.
- Proactively test alternate or protected paths to ensure reliability of network configurations.

4.2. Continuity Functions

Continuity is a model where OAM messages are sent periodically to validate or verify the reachability to a given SF or through a given SFC. This allows the operator to monitor the network device and to quickly detect failures such as link failures, network failures, SF outages or SFC outages. BFD is one such function which helps in detecting failures quickly. OAM functions supported by continuity check are as follows:

- Ability to provision continuity check to a given SF or through a given SFC.
- Notifying the failure upon failure detection for other OAM functions to take appropriate action.

4.3. Trace Functions

Tracing is an important OAM function that allows the operation to trigger an action (ex: response generation) from every transit device on the tested layer. This function is typically useful to gather information from every transit devices or to isolate the failure point towards an SF or through an SFC. Mechanisms must be provided so that the SFC OAM messages may be sent along the same path that a given data packet would follow. Some of the OAM functions supported by trace functions are:

- Ability to trigger action from every transit device on the tested layer towards an SF or through an SFC, using TTL or other means.
- Ability to trigger every transit device to generate response with OAM code(s) on the tested layer towards an SF or through an SFC, using TTL or other means.
- Ability to discover and traverse ECMP paths within an SFC.

- Ability to skip un-supported SF's while tracing SF's in an SFC.

4.4. Performance Measurement Function

Performance management functions involve measuring of packet loss, delay, delay variance, etc. These measurements could be measured pro-actively and on-demand.

SFC OAM should provide the ability to test the packet loss for an SFC. In an SFC, there are various SF's chained together.

Measuring packet loss is very important function. Using on-demand function, the packet loss could be measured using statistical means. Using OAM packets, the approximation of packet loss for a given SFC could be measured.

Delay within an SFC could be measured from the time it takes for a packet to traverse the SFC from ingress SF to egress SF. As the SFC's are generally unidirectional in nature, measurement of one-way delay is important. In order to measure one-way delay, the clocks have to be synchronized using NTP, GPS, etc.

Delay variance could also be measured by sending OAM packets and measuring the jitter between the packets passing through the SFC.

Some of the OAM functions supported by the performance measurement functions are:

- Ability to measure the packet processing delay of a service function or a service function path along an SFC.

- Ability to measure the packet loss of a service function or a service function path along an SFC.

5. Gap Analysis

This Section identifies various OAM functions available at different levels. It will also identify various gaps within the existing toolset, to perform OAM function on an SFC.

5.1. Existing OAM Functions

There are various OAM tool sets available to perform OAM function and network layer, protocol layers and link layers. These OAM functions could validate some of the network overlay transport. Tools like ping and trace are in existence to perform connectivity check and tracing intermediate hops in a network. These tools support different network types like IP, MPLS, TRILL etc. There is also an effort to extend the tool set to provide connectivity and continuity checks within overlay networks. BFD is another tool which helps in detection of data forwarding failures.

Table 1: OAM Tool GAP Analysis

Layer	Connectivity	Continuity	Trace	Performance
N/W Overlay	Ping	BFD, NVo3	Trace	IPPM
SF	None	None	None	None
SFC	None	None	None	None

5.2. Missing OAM Functions

As shown in Table 1, OAM functions for SFC are not yet standardized. Hence, there are no standards-based tools available to monitor the various components identified in Section 3.

5.3. Required OAM Functions

Primary OAM functions exist for network, transport, link and other layers. Tools like ping, trace, BFD, etc., exist in order to perform these OAM functions. Configuration, orchestration and manageability of SF and SFC could be performed using CLI, Netconf etc.

For configuration, manageability and orchestration, providing data and information models for SFC is very much essential. With virtualized SF and SFC, manageability of these functions has to be done programmatically.

SFC OAM must provide tools that operate through various types of SFs including:

- Transparent SFs: These SFs typically do not make any modifications to the packet. In such cases, the SFF may be able to process OAM messages.

- SFs that modify the packet: These SFs modify packet fields. Certain SFs may modify only the headers corresponding to the network over which it is transported, e.g. the MAC headers or overlay headers. In other cases, the IP header of the application’s packet may be modified, e.g. NAT. In yet other cases, the application session itself may be terminated and a new session initiated, e.g. a load balancer that offers HTTPS termination.

6. Open Issues

- Add more details on performance measurement.

- Call out which OAM functions can be achieved by protocol design vs requiring synthetic traffic.

7. Security Considerations

SFC OAM must provide mechanisms for:

- Preventing usage of OAM channel for DDOS attacks.
- Preventing leakage of OAM packets meant for a given SFC beyond that SFC.
- Preventing leakage of information about an sFC beyond its administrative domain.

7. IANA Considerations

No action is required by IANA for this document.

8. Acknowledgements

TBD

9. Contributing Authors

Pedro A. Aranda Gutierrez
Telefonica I+D
Email: pedroa.aranda@tid.es

Diego Lopez
Telefonica I+D
Email: diego@tid.es

Joel Halpern
Ericsson
Email: joel.halpern@ericsson.com

Sriganesh Kini
Ericsson
Email: sriganesh.kini@ericsson.com

Andy Reid
BT
Email: andy.bd.reid@bt.com

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[I-D.ietf-sfc-problem-statement] Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-10 (work in progress), August 2014.

[I-D.ietf-sfc-architecture]
Halpern J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-09 (work in progress), June 2015.

10.2. Informative References

[RFC6291] Andersson, L., van Helvoort, H., Bonica, R., Romascanu, D., and S. Mansfield, "Guidelines for the Use of the "OAM" Acronym in the IETF", BCP 161, RFC 6291, June 2011.

Authors' Addresses

Sam K. Aldrin
Google
Email: aldrin.ietf@gmail.com

Ram Krishnan
Dell
Email: ramkri123@gmail.com

Nobo Akiya
Big Switch
Email: nobo.akiya.dev@gmail.com

Carlos Pignataro
Cisco Systems
Email: cpignata@cisco.com

Anoop Ghanwani
Dell
Email: anoop@alumni.duke.edu

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 25, 2016

J. Halpern, Ed.
Ericsson
C. Pignataro, Ed.
Cisco

July 24, 2015

Service Function Chaining (SFC) Architecture
draft-ietf-sfc-architecture-11

Abstract

This document describes an architecture for the specification, creation, and ongoing maintenance of Service Function Chains (SFC) in a network. It includes architectural concepts, principles, and components used in the construction of composite services through deployment of SFCs, with a focus on those to be standardized in the IETF. This document does not propose solutions, protocols, or extensions to existing protocols.

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 http://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 January 25, 2016.

Copyright Notice

Table of Contents

1. Introduction

The delivery of end-to-end services often requires various service functions. These include traditional network service functions such as firewalls and traditional IP Network Address Translators (NATs), as well as application-specific functions. The definition and instantiation of an ordered set of service functions and subsequent 'steering' of traffic through them is termed Service Function Chaining (SFC).

This document describes an architecture used for the creation and ongoing maintenance of Service Function Chains (SFC) in a network. It includes architectural concepts, principles, and components, with a focus on those to be standardized in the IETF. Service function chains enable composite services that are constructed from one or more service functions.

An overview of the issues associated with the deployment of end-to-end service function chains, abstract sets of service functions and their ordering constraints that create a composite service and the subsequent "steering" of traffic flows through said service functions, is described in [RFC7498].

The current service function deployment models are relatively static, coupled to network topology and physical resources, greatly reducing or eliminating the ability of an operator to introduce new services or dynamically create service function chains. This architecture presents a model addressing the problematic aspects of existing service deployments, including topological independence and configuration complexity.

1.1. Scope

This document defines the architecture for Service Function Chaining (SFC) as standardized in the IETF. The SFC architecture is predicated on topological independence from the underlying forwarding topology.

In this architecture packets are classified on ingress for handling by the required set of Service Functions (SFs) in the SFC-enabled domain and are then forwarded through that set of functions for processing by each function in turn. Packets may be re-classified as a result of this processing.

The architecture described in this document is independent of the planned usage of the network and deployment context and thus, for example, is applicable to both fixed and mobile networks as well as being useful in many Data Center applications.

The architecture described herein is assumed to be applicable to a single network administrative domain. While it is possible for the architectural principles and components to be applied to inter-domain SFCs, these are left for future study.

1.2. Assumptions

The following assumptions are made:

- There is no standard definition or characterization applicable to all SFs, and thus the architecture considers each SF as an opaque processing element.
- There is no global or standard list of SFs enabled in a given administrative domain. The set of SFs enabled in a given domain is a function of the currently active services which may vary with time and according to the networking environment.
- There is no global or standard SF chaining logic. The ordered set of SFs that needs to be applied to deliver a given service is specific to each administrative entity.
- The chaining of SFs and the criteria to invoke them are specific to each administrative entity that operates an SF-enabled domain.
- Several SF chaining policies can be simultaneously applied within an administrative domain to meet various business requirements.
- The underlay is assumed to provide the necessary connectivity to interconnect the Service Function Forwarders (SFFs, see Section 1.4), but the architecture places no constraints on how that connectivity is realized other than it have the required bandwidth, latency, and jitter to support the SFC.
- No assumption is made on how Forwarding Information Bases (FIBs) and Routing Information Bases (RIBs) of involved nodes are populated.
- How to bind traffic to a given SF chain is policy-based.

1.3. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.4. Definition of Terms

Network Service: An offering provided by an operator that is delivered using one or more service functions. This may also be referred to as a composite service. The term "service" is used to denote a "network service" in the context of this document.

Note: Beyond this document, the term "service" is overloaded with varying definitions. For example, to some a service is an offering composed of several elements within the operator's network, whereas for others a service, or more specifically a network service, is a discrete element such as a "firewall". Traditionally, such services (in the latter sense) host a set of service functions and have a network locator where the service is hosted.

Classification: Locally instantiated matching of traffic flows against policy for subsequent application of the required set of network service functions. The policy may be customer/network/service specific.

Classifier: An element that performs Classification.

Service Function Chain (SFC): A service function chain defines an ordered set of abstract service functions (SFs) and ordering constraints that must be applied to packets and/or frames and/or flows selected as a result of classification. An example of an abstract service function is "a firewall". The implied order may not be a linear progression as the architecture allows for SFCs that copy to more than one branch, and also allows for cases where there is flexibility in the order in which service functions need to be applied. The term service chain is often used as shorthand for service function chain.

Service Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). As a logical component, a Service Function can be realized as a virtual element or be embedded in a physical network element. One or more Service Functions can be embedded in the same network element. Multiple occurrences of the Service Function can exist in the same administrative domain.

One or more Service Functions can be involved in the delivery of added-value services. A non-exhaustive list of abstract Service Functions includes: firewalls, WAN and application acceleration, Deep Packet Inspection (DPI), LI (Lawful Intercept), server load

balancing, NAT44 [RFC3022], NAT64 [RFC6146], NPTv6 [RFC6296], HOST_ID injection, HTTP Header Enrichment functions, TCP optimizer.

An SF may be SFC encapsulation aware, that is it receives and acts on information in the SFC encapsulation, or unaware, in which case data forwarded to the SF does not contain the SFC encapsulation.

Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF. Additionally, a service function forwarder is responsible for delivering traffic to a classifier when needed and supported, transporting traffic to another SFF (in the same or different type of overlay), and terminating the SFP.

Metadata: provides the ability to exchange context information between classifiers and SFs and among SFs.

Service Function Path (SFP): The Service Function Path is a constrained specification of where packets assigned to a certain service function path must go. While it may be so constrained as to identify the exact locations, it can also be less specific. The SFP provides a level of indirection between the fully abstract notion of service chain as a sequence of abstract service functions to be delivered, and the fully specified notion of exactly which SFF/SFs the packet will visit when it actually traverses the network. By allowing the control components to specify this level of indirection, the operator may control the degree of SFF/SF selection authority that is delegated to the network.

SFC Encapsulation: The SFC Encapsulation provides at a minimum SFP identification, and is used by the SFC-aware functions, such as the SFF and SFC-aware SFs. The SFC Encapsulation is not used for network packet forwarding. In addition to SFP identification, the SFC encapsulation carries metadata including data plane context information.

Rendered Service Path (RSP): Within an SFP, packets themselves are of course transmitted from and to specific places in the network, visiting a specific sequence of SFFs and SFs. This sequence of actual visits by a packet to specific SFFs and SFs in the network is known as the Rendered Service Path (RSP). This definition is included here for use by later documents,

such as when solutions may need to discuss the actual sequence of locations the packets visit.

SFC-enabled Domain: A network or region of a network that implements SFC. An SFC-enabled Domain is limited to a single network administrative domain.

SFC Proxy: Removes and inserts SFC Encapsulation on behalf of an SFC-unaware service function. SFC proxies are logical elements.

2. Architectural Concepts

The following sections describe the foundational concepts of service function chaining and the SFC architecture.

Service Function Chaining enables the creation of composite (network) services that consist of an ordered set of Service Functions (SF) that must be applied to packets and/or frames and/or flows selected as a result of classification. Each SF is referenced using an identifier that is unique within an SF-enabled domain.

Service Function Chaining is a concept that provides for more than just the application of an ordered set of SFs to selected traffic; rather, it describes a method for deploying SFs in a way that enables dynamic ordering and topological independence of those SFs as well as the exchange of metadata between participating entities.

2.1. Service Function Chains

In most networks, services are constructed as abstract sequences of SFs that represent SFCs. At a high level, an SFC is an abstracted view of a service that specifies the set of required SFs as well as the order in which they must be executed. Graphs, as illustrated in Figure 1, define an SFC, where each graph node represents the required existence of at least one abstract SF. Such graph nodes (SFs) can be part of zero, one, or many SFCs. A given graph node (SF) can appear one time or multiple times in a given SFC.

SFCs can start from the origination point of the service function graph (i.e., node 1 in Figure 1), or from any subsequent node in the graph. As shown, SFs may therefore become branching nodes in the graph, with those SFs selecting edges that move traffic to one or more branches. The top and middle graphs depict such a case, where a second classification event occurs after node 2, and a new graph is selected (i.e., node 3 instead of node 6). The bottom graph highlights the concept of a cycle, in which a given SF (e.g., node 7 in the depiction) can be visited more than once within a given service chain. An SFC can have more than one terminus.

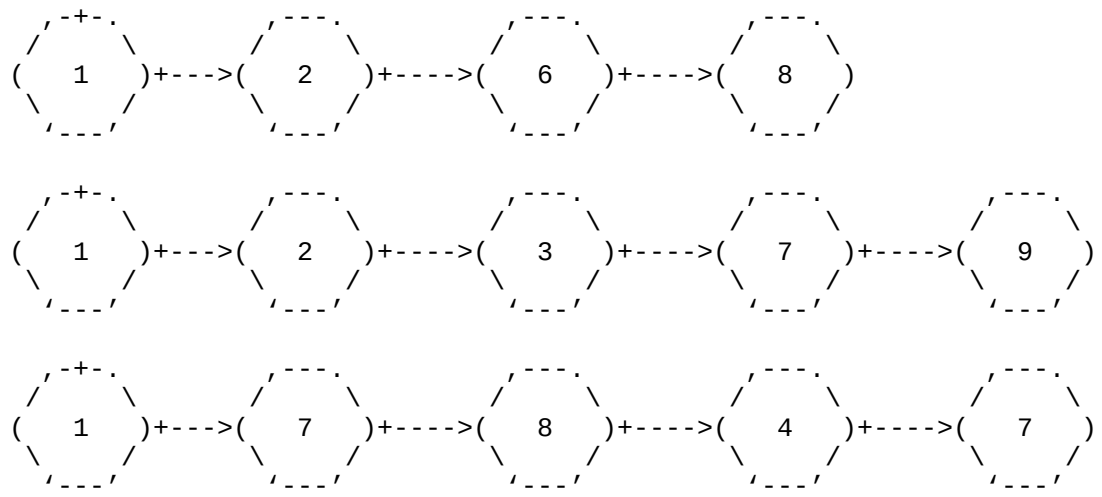


Figure 1: Service Function Chain Graphs

The concepts of classification, re-classification, and branching are covered in subsequent sections of this architecture (see Section 4.7 and Section 4.8).

2.2. Service Function Chain Symmetry

SFCs may be unidirectional or bidirectional. A unidirectional SFC requires that traffic be forwarded through the ordered SFs in one direction (SF1 -> SF2 -> SF3), whereas a bidirectional SFC requires a symmetric path (SF1 -> SF2 -> SF3 and SF3 -> SF2 -> SF1), and in which the SF instances are the same in opposite directions. A hybrid SFC has attributes of both unidirectional and bidirectional SFCs; that is to say some SFs require symmetric traffic, whereas other SFs do not process reverse traffic or are independent of the corresponding forward traffic.

SFCs may contain cycles; that is traffic may need to traverse one or more SFs within an SFC more than once. Solutions will need to ensure suitable disambiguation for such situations.

The architectural allowance that is made for SFPs that delegate choice to the network for which SFs and/or SFFs a packet will visit creates potential issues here. A solution that allows such delegation needs to also describe how the solution ensures that those service chains that require service function chain symmetry can achieve that.

Further, there are state tradeoffs in symmetry. Symmetry may be realized in several ways depending on the SFF and classifier functionality. In some cases, "mirrored" classification (i.e., from Source to Destination and from Destination to Source) policy may be deployed, whereas in others shared state between classifiers may be used to ensure that symmetric flows are correctly identified, then steered along the required SFP. At a high level, there are various common cases. In a non-exhaustive way, there can be for example:

- A single classifier (or a small number of classifiers), in which case both incoming and outgoing flows could be recognized at the same classifier, so the synchronization would be feasible by internal mechanisms internal to the classifier.

- Stateful classifiers where several classifiers may be clustered and share state.

- Fully distributed classifiers, where synchronization needs to be provided through unspecified means.

- A classifier that learns state from the egress packets/flows that is then used to provide state for the return packets/flow.

- Symmetry may also be provided by stateful forwarding logic in the SFF in some implementations.

This is a non-comprehensive list of common cases.

2.3. Service Function Paths

A service function path (SFP) is a mechanism used by service chaining to express the result of applying more granular policy and operational constraints to the abstract requirements of a service chain (SFC). This architecture does not mandate the degree of specificity of the SFP. Architecturally, within the same SFC-enabled domain, some SFPs may be fully specified, selecting exactly which SFF and which SF are to be visited by packets using that SFP, while other SFPs may be quite vague, deferring to the SFF the decisions about the exact sequence of steps to be used to realize the SFC. The specificity may be anywhere in between these extremes.

As an example of such an intermediate specificity, there may be two SFPs associated with a given SFC, where one SFP specifies that any order of SFF and SF may be used as long as it is within data center 1, and where the second SFP allows the same latitude, but only within data center 2.

Thus, the policies and logic of SFP selection or creation (depending upon the solution) produce what may be thought of as a constrained version of the original SFC. Since multiple policies may apply to different traffic that uses the same SFC, it also follows that there may be multiple SFPs may be associated with a single SFC.

The architecture allows for the same SF to be reachable through multiple SFFs. In these cases, some SFPs may constrain which SFF is used to reach which SF, while some SFPs may leave that decision to the SFF itself.

Further, the architecture allows for two or more SFs to be attached to the same SFF, and possibly connected via internal means allowing more effective communication. In these cases, some solutions or deployments may choose to use some form of internal inter-process or inter-VM messaging (communication behind the virtual switching element) that is optimized for such an environment. This must be coordinated with the SFF so that the service function forwarding can properly perform its job. Implementation details of such mechanisms are considered out of scope for this document, and can include a spectrum of methods: for example situations including all next-hops explicitly, others where a list of possible next-hops is provided and the selection is local, or cases with just an identifier, where all resolution is local.

This architecture also allows the same SF to be part of multiple SFPs.

2.3.1. Service Function Chains, Service Function Paths, and Rendered Service Path

As an example of this progressive refinement, consider a service function chain (SFC) which states that packets using this chain should be delivered to a firewall and a caching engine.

A Service Function Path (SFP) could refine this, considering that this architecture does not mandate the degree of specificity an SFP has to have. It might specify that the firewall and caching engine are both to be in a specific Data Center (e.g., in DC1), or it might specify exactly which instance of each firewall and chaching engine is to be used.

The Rendered Service Path (RSP) is the actual sequence of SFFs and SFs that the packets will actually visit. So if the SFP picked the DC, the RSP would be more specific.

3. Architecture Principles

Service function chaining is predicated on several key architectural principles:

1. Topological independence: no changes to the underlay network forwarding topology - implicit, or explicit - are needed to deploy and invoke SFs or SFCs.

2. Plane separation: dynamic realization of SFPs is separated from packet handling operations (e.g., packet forwarding).

3. Classification: traffic that satisfies classification rules is forwarded according to a specific SFP. For example, classification can be as simple as an explicit forwarding entry that forwards all traffic from one address into the SFP. Multiple classification points are possible within an SFC (i.e., forming a service graph) thus enabling changes/updates to the SFC by SFs.

   Classification can occur at varying degrees of granularity; for example, classification can use a 5-tuple, a transport port or set of ports, part of the packet payload, it can be the result of high-level inspections, or it can come from external systems.

4. Shared Metadata: Metadata/context data can be shared amongst SFs and classifiers, between SFs, and between external systems and SFs (e.g., orchestration).

   One use of metadata is to provide and share the result of classification (that occurs within the SFC-enabled domain, or external to it) along an SFP. For example, an external repository might provide user/subscriber information to a service chain classifier. This classifier could in turn impose that information in the SFC encapsulation for delivery to the requisite SFs. The SFs could in turn utilize the user/subscriber information for local policy decisions. Metadata can also share SF output along the SFP.

5. Service definition independence: The SFC architecture does not depend on the details of SFs themselves.

6. Service function chain independence: The creation, modification, or deletion of an SFC has no impact on other SFCs. The same is true for SFPs.

7. Heterogeneous control/policy points: The architecture allows SFs to use independent mechanisms (out of scope for this document) to

populate and resolve local policy and (if needed) local classification criteria.

4. Core SFC Architecture Components

The SFC Architecture is built out of architectural building blocks which are logical components; these logical components are classifiers, service function forwarders (SFF), the service functions themselves (SF), and SFC-proxies. While this architecture describes functionally distinct logical components and promotes transport indepencence, they could be realized and combined in various ways in deployed products, and could be combined with an overlay.

They are interconnected using the SFC Encapsulation. This results in a high level logical architecture of an SFC-enabled Domain which comprises:

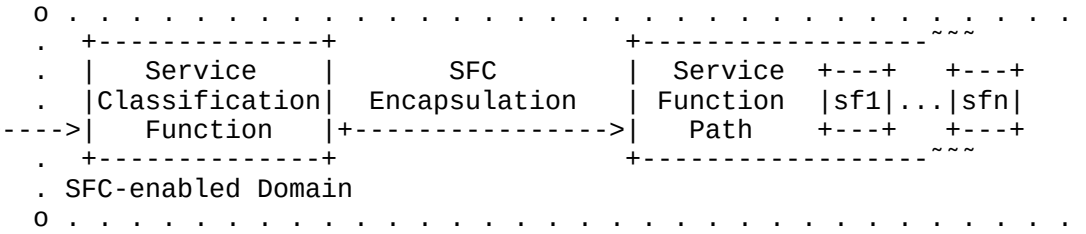


Figure 2: Service Function Chain Architecture

The following sub-sections provide details on each logical component that form the basis of the SFC architecture. A detailed overview of how some of these architectural components interact is provided in Figure 3:

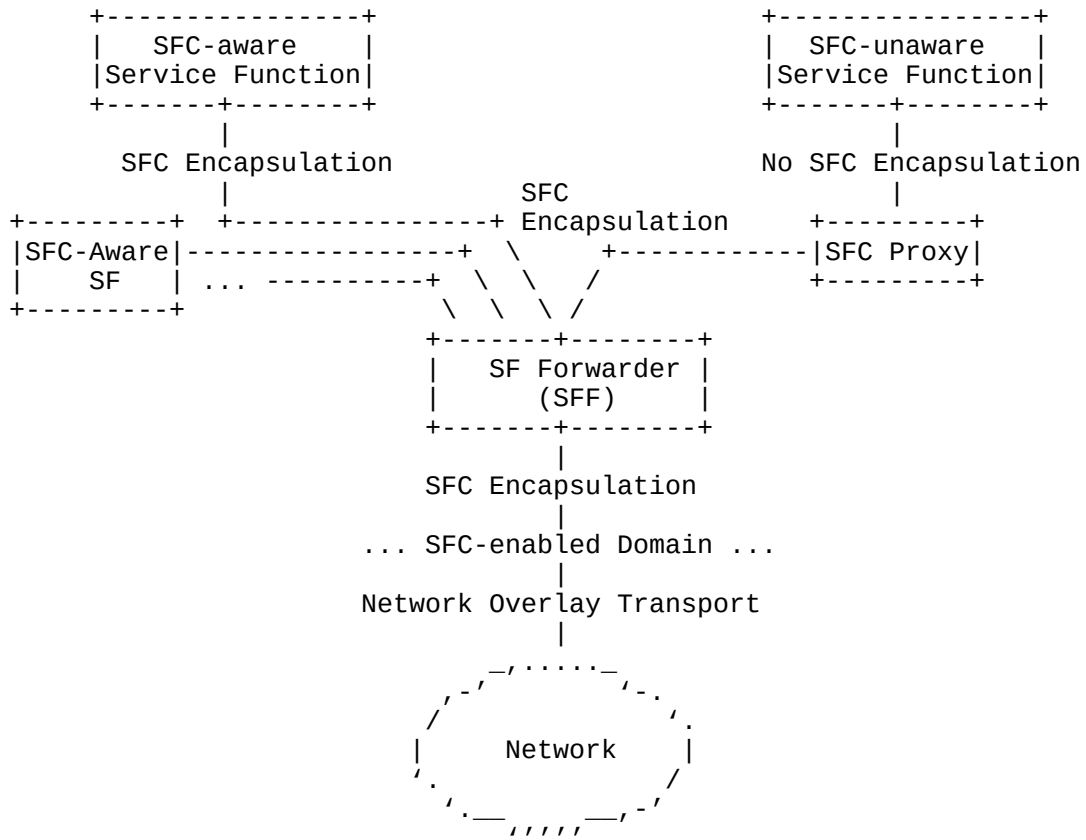


Figure 3: SFC Architecture Components Post Initial Classification

Please note that the depiction in Figure 3 shows packets post initial classification, and therefore including the SFC encapsulation. Although not included in Figure 3, the classifier is an SFC architectural component.

4.1. SFC Encapsulation

The SFC encapsulation enables service function path selection. It also enables the sharing of metadata/context information when such metadata exchange is required.

The SFC encapsulation carries explicit information used to identify the SFP. However, the SFC encapsulation is not a transport encapsulation itself: it is not used to forward packets within the network fabric. If packets need to flow between separate physical platforms, the SFC encapsulation therefore relies on an outer network

transport. Transit forwarders -- such as router and switches -- forward SFC encapsulated packets based on the outer (non-SFC) encapsulation.

One of the key architecture principles of SFC is that the SFC encapsulation remain transport independent. As such any network transport protocol may be used to carry the SFC encapsulated traffic.

4.2. Service Function (SF)

The concept of an SF evolves; rather than being viewed as a bump in the wire, an SF becomes a resource within a specified administrative domain that is available for consumption as part of a composite service. SFs send/receive data to/from one or more SFFs. SFC-aware SFs receive this traffic with the SFC encapsulation.

While the SFC architecture defines the concept and specifies some characteristics of a new encapsulation - the SFC encapsulation - and several logical components for the construction of SFCs, existing SF implementations may not have the capabilities to act upon or fully integrate with the new SFC encapsulation. In order to provide a mechanism for such SFs to participate in the architecture, an SFC proxy function is defined (see Section 4.6). The SFC proxy acts as a gateway between the SFC encapsulation and SFC-unaware SFs. The integration of SFC-unaware service functions is discussed in more detail in the SFC proxy section.

This architecture allows an SF to be part of multiple SFPs and SFCs.

4.3. Service Function Forwarder (SFF)

The SFF is responsible for forwarding packets and/or frames received from the network to one or more SFs associated with a given SFF using information conveyed in the SFC encapsulation. Traffic from SFs eventually returns to the same SFF, which is responsible for injecting traffic back onto the network. Some SFs, such as firewalls, could also consume a packet.

The collection of SFFs and associated SFs creates a service plane overlay in which SFC-aware SFs, as well as SFC-unaware SFs reside. Within this service plane, the SFF component connects different SFs that form a service function path.

SFFs maintain the requisite SFP forwarding information. SFP forwarding information is associated with a service path identifier that is used to uniquely identify an SFP. The service forwarding state enables an SFF to identify which SFs of a given SFP should be applied, and in what order, as traffic flows through the associated

SFP. While there may appear to the SFF to be only one available way to deliver the given SF, there may also be multiple choices allowed by the constraints of the SFP.

If there are multiple choices, the SFF needs to preserve the property that all packets of a given flow are handled the same way, since the SF may well be stateful. Additionally, the SFF may preserve the handling of packets based on other properties on top of a flow, such as a subscriber, session, or application instance identification.

The SFF also has the information to allow it to forward packets to the next SFF after applying local service functions. Again, while there may be only a single choice available, the architecture allows for multiple choices for the next SFF. As with SFs, the solution needs to operate such that the behavior with regard to specific flows (see the Rendered Service Path) is stable. The selection of available SFs and next SFFs may be interwoven when an SFF supports multiple distinct service functions and the same service function is available at multiple SFFs. Solutions need to be clear about what is allowed in these cases.

Even when the SFF supports and utilizes multiple choices, the decision as to whether to use flow-specific mechanisms or coarser grained means to ensure that the behavior of specific flows is stable is a matter for specific solutions and specific implementations.

The SFF component has the following primary responsibilities:

1. SFP forwarding : Traffic arrives at an SFF from the network. The SFF determines the appropriate SF the traffic should be forwarded to via information contained in the SFC encapsulation. Post-SF, the traffic is returned to the SFF, and, if needed, is forwarded to another SF associated with that SFF. If there is another non-local (i.e., different SFF) hop in the SFP, the SFF further encapsulates the traffic in the appropriate network transport protocol and delivers it to the network for delivery to the next SFF along the path. Related to this forwarding responsibility, an SFF should be able to interact with metadata.

2. Terminating SFPs : An SFC is completely executed when traffic has traversed all required SFs in a chain. When traffic arrives at the SFF after the last SF has finished processing it, the final SFF knows from the service forwarding state that the SFC is complete. The SFF removes the SFC encapsulation and delivers the packet back to the network for forwarding.

3. Maintaining flow state: In some cases, the SFF may be stateful. It creates flows and stores flow-centric information. This state

information may be used for a range of SFP-related tasks such as ensuring consistent treatment of all packets in a given flow, ensuring symmetry or for state-aware SFC Proxy functionality (see Section 4.8).

4.3.1. Transport Derived SFF

Service function forwarding, as described above, directly depends upon the use of the service path information contained in the SFC encapsulation. However, existing implementations may not be able to act on the SFC encapsulation. These platforms may opt to use existing transport information if it can be arranged to provide explicit service path information.

This results in the same architectural behavior and meaning for service function forwarding and service function paths. It is the responsibility of the control components to ensure that the transport path executed in such a case is fully aligned with the path identified by the information in the service chaining encapsulation.

4.4. SFC-Enabled Domain

Specific features may need to be enforced at the boundaries of an SFC-enabled domain, for example to avoid leaking SFC information. Using the term node to refer generically to an entity that is performing a set of functions, in this context, an SFC Boundary Node denotes a node that connects one SFC-enabled domain to a node either located in another SFC-enabled domain or in a domain that is SFC-unaware.

An SFC Boundary node can act as egress or ingress. An SFC Egress Node denotes a SFC Boundary Node that handles traffic leaving the SFC-enabled domain the Egress Node belongs to. Such a node is required to remove any information specific to the SFC Domain, typically the SFC Encapsulation. Further, from a privacy perspective, an SFC Egress Node is required to ensure that any sensitive information added as part of SFC gets removed. In this context, information may be sensitive due to network concerns or end-customer concerns. An SFC Ingress Node denotes an SFC Boundary Node that handles traffic entering the SFC-enabled domain. In most solutions and deployments this will need to include a classifier, and will be responsible for adding the SFC encapsulation to the packet.

An SFC Proxy and corresponding SFC-unaware Service Function (see Figure 3) are inside the SFC-enabled domain.

4.5. Network Overlay and Network Components

Underneath the SFF there are components responsible for performing the transport (overlay) forwarding. They do not consult the SFC encapsulation or inner payload for performing this forwarding. They only consult the outer-transport encapsulation for the transport (overlay) forwarding.

4.6. SFC Proxy

In order for the SFC architecture to support SFC-unaware SFs (e.g., legacy service functions) a logical SFC proxy function may be used. This function sits between an SFF and one or more SFs to which the SFF is directing traffic (see Figure 3).

The proxy accepts packets from the SFF on behalf of the SF. It removes the SFC encapsulation, and then uses a local attachment circuit to deliver packets to SFC unaware SFs. It also receives packets back from the SF, reapplies the SFC encapsulation, and returns them to the SFF for processing along the service function path.

Thus, from the point of view of the SFF, the SFC proxy appears to be part of an SFC aware SF.

Communication details between the SFF and the SFC Proxy are the same as those between the SFF and an SFC aware SF. The details of that are not part of this architecture. The details of the communication methods over the local attachment circuit between the SFC proxy and the SFC-unaware SF are dependent upon the specific behaviors and capabilities of that SFC-unaware SF, and thus are also out of scope for this architecture.

Specifically, for traffic received from the SFF intended for the SF the proxy is representing, the SFC proxy:

- Removes the SFC encapsulation from SFC encapsulated packets.
- Identifies the required SF to be applied based on available information including that carried in the SFC encapsulation.
- Selects the appropriate outbound local attachment circuit through which the next SF for this SFP is reachable. This is derived from the identification of the SF carried in the SFC encapsulation, and may include local techniques. Examples of a local attachment circuit include, but are not limited to, VLAN, IP-in-IP, L2TPv3, GRE, VXLAN.

- Forwards the original payload via the selected local attachment circuit to the appropriate SF.

When traffic is returned from the SF:

- Applies the required SFC encapsulation. The determination of the encapsulation details may be inferred by the local attachment circuit through which the packet and/or frame was received, or via packet classification, or other local policy. In some cases, packet ordering or modification by the SF may necessitate additional classification in order to re-apply the correct SFC encapsulation.

- Delivers the packet with the SFC Encapsulation to the SFF, as would happen with packets returned from an SFC-aware SF.

4.7. Classification

Traffic from the network that satisfies classification criteria is directed into an SFP and forwarded to the requisite service function(s). Classification is handled by a service classification function; initial classification occurs at the ingress to the SFC domain. The granularity of the initial classification is determined by the capabilities of the classifier and the requirements of the SFC policy. For instance, classification might be relatively coarse: all packets from this port are subject to SFC policy X and directed into SFP A, or quite granular: all packets matching this 5-tuple are subject to SFC policy Y and directed into SFP B.

As a consequence of the classification decision, the appropriate SFC encapsulation is imposed on the data, and a suitable SFP is selected or created. Classification results in attaching the traffic to a specific SFP.

4.8. Re-Classification and Branching

The SFC architecture supports re-classification (or non-initial classification) as well. As packets traverse an SFP, re-classification may occur - typically performed by a classification function co-resident with a service function. Reclassification may result in the selection of a new SFP, an update of the associated metadata, or both. This is referred to as "branching".

For example, an initial classification results in the selection of SFP A: DPI_1 --> SLB_8. However, when the DPI service function is executed, attack traffic is detected at the application layer. DPI_1 re-classifies the traffic as attack and alters the service path to SFP B, to include a firewall for policy enforcement: dropping the

traffic: DPI_1 --> FW_4. Subsequent to FW_4, surviving traffic would be returned to the original SFF. In this simple example, the DPI service function re-classifies the traffic based on local application layer classification capabilities (that were not available during the initial classification step).

When traffic arrives after being steered through an SFC-unaware SF, the SFC Proxy must perform re-classification of traffic to determine the SFP. The SFC Proxy is concerned with re-attaching information for SFC-unaware SFs, and a stateful SFC Proxy simplifies such classification to a flow lookup.

4.9. Shared Metadata

Sharing metadata allows the network to provide network-derived information to the SFs, SF-to-SF information exchange and the sharing of service-derived information to the network. Some SFCs may not require metadata exchange. SFC infrastructure enables the exchange of this shared data along the SFP. The shared metadata serves several possible roles within the SFC architecture:

- Allows elements that typically operate as ships in the night to exchange information.
- Encodes information about the network and/or data for post-service forwarding.
- Creates an identifier used for policy binding by SFs.

Context information can be derived in several ways:

- External sources
- Network node classification
- Service function classification

5. Additional Architectural Concepts

There are a number of issues which solutions need to address, and which the architecture informs but does not determine. This section lays out some of those concepts.

5.1. The Role of Policy

Much of the behavior of service chains is driven by operator and per-customer policy. This architecture is structured to isolate the policy interactions from the data plane and control logic.

Specifically, it is assumed that the service chaining control plane creates the service paths. The service chaining data plane is used to deliver the classified packets along the service chains to the intended service functions.

Policy, in contrast, interacts with the system in other places. Policies and policy engines may monitor service functions to decide if additional (or fewer) instances of services are needed. When applicable, those decisions may in turn result in interactions that direct the control logic to change the SFP placement or packet classification rules.

Similarly, operator service policy, often managed by operational or business support systems (OSS or BSS), will frequently determine what service functions are available. Operator service policies also determine which sequences of functions are valid and are to be used or made available.

The offering of service chains to customers, and the selection of which service chain a customer wishes to use, are driven by a combination of operator and customer policies using appropriate portals in conjunction with the OSS and BSS tools. These selections then drive the service chaining control logic, which in turn establishes the appropriate packet classification rules.

5.2. SFC Control Plane

The SFC Control Plane is part of the overall SFC architecture, and this section describes its high-level functions. However, the detailed definition of the SFC Control Plane is outside the scope of this document.

The SFC control plane is responsible for constructing SFPs, translating SFCs to forwarding paths and propagating path information to participating nodes to achieve requisite forwarding behavior to construct the service overlay. For instance, an SFC construction may be static; selecting exactly which SFFs and which SFs from those SFFs are to be used, or it may be dynamic, allowing the network to perform some or all of the choices of SFF or SF to use to deliver the selected service chain within the constraints represented by the service path.

In the SFC architecture, SFs are resources; the control plane manages and communicates their capabilities, availability and location in fashions suitable for the transport and SFC operations in use. The control plane is also responsible for the creation of the context (see below). The control plane may be distributed (using new or

existing control plane protocols), or be centralized, or a combination of the two.

The SFC control plane provides the following functionality:

1. An SFC-enabled domain wide view of all available service function resources as well as the network locators through which they are reachable.

2. Uses SFC policy to construct service function chains, and associated service function paths.

3. Selection of specific SFs for a requested SFC, either statically (using specific SFs) or dynamically (using service explicit SFs at the time of delivering traffic to them).

4. Provides requisite SFC data plane information to the SFC architecture components, most notably the SFF.

5. Provide the metadata and usage information classifiers need so that they in turn can provide this metadata for appropriate packets in the data plane.

6. When needed, provide information including policy information to other SFC elements to be able to properly interpret metadata.

5.3. Resource Control

The SFC system may be responsible for managing all resources necessary for the SFC components to function. This includes network constraints used to plan and choose network path(s) between service function forwarders, network communication paths between service function forwarders and their attached service functions, characteristics of the nodes themselves such as memory, number of virtual interfaces, routes, and instantiation, configuration, and deletion of SFs.

The SFC system will also be required to reflect policy decisions about resource control, as expressed by other components in the system.

While all of these aspects are part of the overall system, they are beyond the scope of this architecture.

5.4. Infinite Loop Detection and Avoidance

This SFC architecture is predicated on topological independence from the underlying forwarding topology. Consequently, a service topology is created by Service Function Paths or by the local decisions of the Service Function Forwarders based on the constraints expressed in the SFP. Due to the overlay constraints, the packet-forwarding path may need to visit the same SFF multiple times, and in some less common cases may even need to visit the same SF more than once. The Service Chaining solution needs to permit these limited and policy-compliant loops. At the same time, the solutions must ensure that indefinite and unbounded loops cannot be formed, as such would consume unbounded resources without delivering any value.

In other words, this architecture requires the solution to prevent infinite Service Function Loops, even when Service Functions may be invoked multiple times in the same SFP.

5.5. Load Balancing Considerations

Supporting function elasticity and high-availability should not overly complicate SFC or lead to unnecessary scalability problems.

In the simplest case, where there is only a single function in the SFP (the next hop is either the destination address of the flow or the appropriate next hop to that destination), one could argue that there may be no need for SFC.

In the cases where the classifier is separate from the single function or a function at the terminal address may need sub-prefix (e.g., finer grained address information) or per-subscriber metadata, a single SFP exists (i.e., the metadata changes but the SFP does not), regardless of the number of potential terminal addresses for the flow. This is the case of the simple load balancer. See Figure 4.

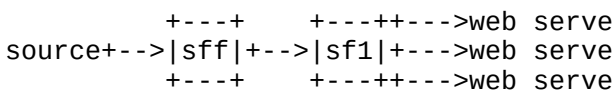


Figure 4: Simple Load Balancing

By extrapolation, in the case where intermediary functions within a chain had similar "elastic" behaviors, we do not need separate chains to account for this behavior - as long as the traffic coalesces to a common next-hop after the point of elasticity.

In Figure 5, we have a chain of five service functions between the traffic source and its destination.

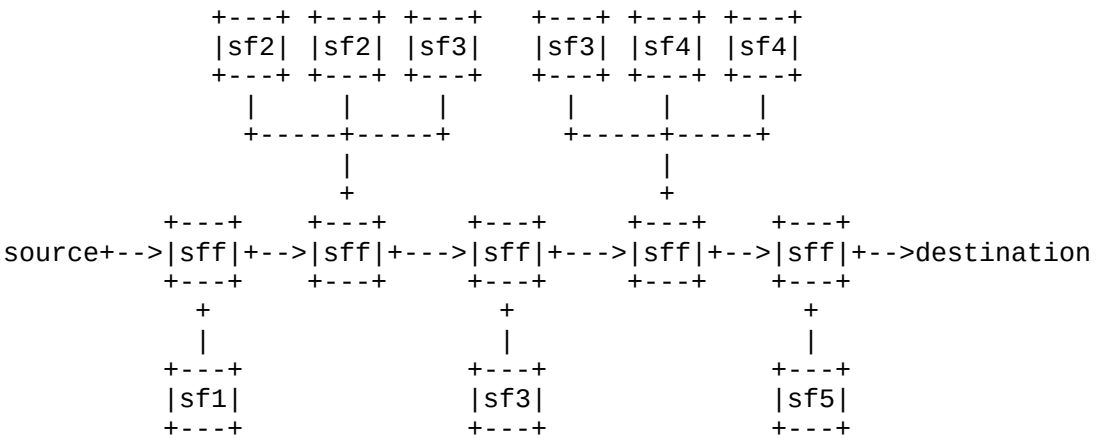


Figure 5: Load Balancing

This would be represented as one service function path: sf1->sf2->sf3->sf4->sf5. The SFF is a logical element, which may be made up of one or multiple components. In this architecture, the SFF may handle load distribution based on policy.

It can also be seen in the above that the same service function may be reachable through multiple SFFs, as discussed earlier. The selection of which SFF to use to reach SF3 may be made by the control logic in defining the SFP, or may be left to the SFFs themselves, depending upon policy, solution, and deployment constraints. In the latter case, it needs to be assured that exactly one SFF takes responsibility to steer traffic through SF3.

5.6. MTU and Fragmentation Considerations

This architecture prescribes additional information being added to packets to identify service function paths and often to represent metadata. It also envisions adding transport information to carry packets along service function paths, at least between service function forwarders. This added information increases the size of the packet to be carried by service chaining. Such additions could potentially increase the packet size beyond the MTU supported on some or all of the media used in the service chaining domain.

Such packet size increases can thus cause operational MTU problems. Requiring fragmentation and reassembly in an SFF would be a major processing increase, and might be impossible with some transports.

Expecting service functions to deal with packets fragmented by the SFC function might be onerous even when such fragmentation was possible. Thus, at the very least, solutions need to pay attention to the size cost of their approach. There may be alternative or additional means available, although any solution needs to consider the tradeoffs.

These considerations apply to any generic architecture that increases the header size. There are also more specific MTU considerations: Effects on Path MTU Discovery (PMTUD) as well as deployment considerations. Deployments within a single administrative control or even a single Data Center complex can afford more flexibility in dealing with larger packets, and deploying existing mitigations that decrease the likelihood of fragmentation or discard.

5.7. SFC OAM

Operations, Administration, and Maintenance (OAM) tools are an integral part of the architecture. These serve various purposes, including fault detection and isolation, and performance management. For example, there are many advantages of SFP liveness detection, including status reporting, support for resiliency operations and policies, and an enhanced ability to balance load.

Service Function Paths create a services topology, and OAM performs various functions within this service layer. Furthermore, SFC OAM follows the same architectural principles of SFC in general. For example, topological independence (including the ability to run OAM over various overlay technologies) and classification-based policy.

We can subdivide the SFC OAM architecture in two parts:

- In-band: OAM packets follow the same path and share fate with user packets, within the service topology. For this, they also follow the architectural principle of consistent policy identifiers, and use the same path IDs as the service chain data packets. Load balancing and SFC encapsulation with packet forwarding are particularly important here.

- Out-of-band: reporting beyond the actual data plane. An additional layer beyond the data-plane OAM allows for additional alerting and measurements.

This architecture prescribes end-to-end SFP OAM functions, which implies SFF understanding of whether an in-band packet is an OAM or user packet. However, service function validation is outside of the scope of this architecture, and application-level OAM is not what this architecture prescribes.

Some of the detailed functions performed by SFC OAM include fault detection and isolation in a Service Function Path or a Service Function, verification that connectivity using SFPs is both effective and directing packets to the intended service functions, service path tracing, diagnostic and fault isolation, alarm reporting, performance measurement, locking and testing of service functions, validation with the control plane (see Section 5.2), and also allow for vendor-specific as well as experimental functions. SFC should leverage, and if needed extend relevant existing OAM mechanisms.

5.8. Resilience and Redundancy

As a practical operational requirement, any service chaining solution needs to be able to respond effectively, and usually very quickly, to failure conditions. These may be failures of connectivity in the network between SFFs, failures of SFFs, or failures of SFs. Per-SF state, as for example stateful-firewall state, is the responsibility of the SF, and not addressed by this architecture.

Multiple techniques are available to address this issue. Solutions can describe both what they require and what they allow to address failure. Solutions can make use of flexible specificity of service function paths, if the SFF can be given enough information in a timely fashion to do this. Solutions can also make use of MAC or IP level redundancy mechanisms such as VRRP. Also, particularly for SF failures, load balancers co-located with the SFF or as part of the service function delivery mechanism can provide such robustness.

Similarly, operational requirements imply resilience in the face of load changes. While mechanisms for managing (e.g., monitoring, instantiating, loading images, providing configuration to service function chaining control, deleting, etc.) virtual machines are out of scope for this architecture, solutions can and are aided by describing how they can make use of scaling mechanisms.

6. Security Considerations

The architecture described here is different from the current model, and moving to the new model could lead to different security arrangements and modeling. In the SFC architecture, a relatively static topologically-dependent deployment model is replaced with the chaining of sets of service functions. This can change the flow of data through the network, and the security and privacy considerations of the protocol and deployment will need to be reevaluated in light of the new model.

Security considerations apply to the realization of this architecture, in particular to the documents that will define

protocols. Such realization ought to provide means to protect against security and privacy attacks in the areas hereby described.

Building from the categorization of [RFC7498], we can largely divide the security considerations in four areas:

Service Overlay: Underneath the Service Function Forwarders, the components that are responsible for performing the transport forwarding consult the outer-transport encapsulation for underlay forwarding. Used transport mechanisms should satisfy the security requirements of the specific SFC deployment. These requirements typically include varying degrees of traffic separation, protection against different attacks (e.g., spoofing, man-in-the-middle, brute-force, or insertion attacks), and can also include authenticity and integrity checking, and/or confidentiality provisions, for both the network overlay transport and traffic it encapsulates.

Boundaries: Specific requirements may need to be enforced at the boundaries of an SFC-enabled domain. These include, for example, to avoid leaking SFC information, and to protect its borders against various forms of attacks. If untrusted parties can inject packets which will be treated as being properly classified for service chaining, there are a large range of attacks which can be mounted against the resulting system. Depending upon deployment details, these likely include spoofing packets from users and creating DDoS and reflection attacks of various kinds. Thus, when a transport mechanisms are selected for use with SFC, they MUST ensure that outside parties can not inject SFC packets which will be accepted for processing into the domain. This border security MUST include any tunnels to other domains. If those tunnels are to be used for SFC without reclassification, then the tunnel MUST include additional techniques to ensure the integrity and validity of such packets.

Classification: Classification is used at the ingress edge of an SFC-enabled domain. Policy for this classification is done using a plurality of methods. Whatever method is used needs to consider a range of security issues. These include appropriate authentication and authorization of classification policy, potential confidentiality issues of that policy, protection against corruption, and proper application of policy with needed segregation of application. This includes proper controls on the policies which drive the application of the SFC Encapsulation and associated metadata to packets. Similar issues need to be addressed if classification is performed within a service chaining domain, i.e., re-classification.

SFC Encapsulation: The SFC Encapsulation provides at a minimum SFP identification, and carries metadata. An operator may consider the SFC Metadata as sensitive. From a privacy perspective, a user may be concerned about the operator revealing data about (and not belonging to) the customer. Therefore, solutions should consider whether there is a risk of sensitive information slipping out of the operators control. Issues of information exposure should also consider flow analysis. Further, when a specific metadata element is defined, it should be carefully considered whether origin authentication is needed for it.

A classifier may have privileged access to information about a packet or inside a packet (see Section 3, bullet 4, and Section 4.9) that is then communicated in the metadata. The threat of leaking this private data needs to be mitigated [RFC6973]. As one example, if private data is represented by an identifier, then a new identifier can be allocated, such that the mapping from the private data to the new identifier is not broadly shared.

Some metadata added to and carried in SFC packets is sensitive for various reasons, including potentially revealing personally identifying information. Realizations of the architecture MUST protect to ensure that such information is handled with suitable care and precautions against inappropriate dissemination of the information. This can have implications to the data plane, the control plane, or both. Data plane protocol definitions for SFC can include suitable provision for protect such information for use when handling sensitive information, with packet or SFP granularity. Equally, the control mechanisms use with SFC can have provisions to determine that such mechanisms are available, and to ensure that they are used when needed. Inability to do so needs to result in error indications to appropriate management systems. In particular, when the control systems know that sensitive information may potentially be added to packets at certain points on certain service chains, the control mechanism MUST verify that appropriate protective treatment of NSH information is available from the point where the information is added to the point where it will be removed. If such mechanisms are unavailable, error notifications SHOULD be generated.

Additionally, SFC OAM Functions need to not negatively affect the security considerations of an SFC-enabled domain.

Finally, all entities (software or hardware) interacting with the service chaining mechanisms need to provide means of security against malformed, poorly configured (deliberate or not) protocol constructs

and loops. These considerations are largely the same as those in any network, particularly an overlay network.

7. Contributors and Acknowledgments

The editors would like to thank Sam Aldrin, Alia Atlas, Nicolas Bouthors, Stewart Bryant, Linda Dunbar, Alla Goldner, Ken Gray, Barry Greene, Anil Gunturu, David Harrington, Shunsuke Homma, Dave Hood, Chris Inacio, Nagendra Kumar, Hongyu Li, Andrew Malis, Guy Meador III, Kengo Naito, Thomas Narten, Ron Parker, Reinaldo Penno, Naiming Shen, Xiaohu Xu, and Lucy Yong for a thorough review and useful comments.

The initial version of this "Service Function Chaining (SFC) Architecture" document is the result of merging two previous documents, and this section lists the aggregate of authors, editors, contributors and acknowledged participants, all who provided important ideas and text that fed into this architecture.

[I-D.boucadair-sfc-framework]:

Authors:

Mohamed Boucadair
Christian Jacquenet
Ron Parker
Diego R. Lopez
Jim Guichard
Carlos Pignataro

Contributors:

Parviz Yegani
Paul Quinn
Linda Dunbar

Acknowledgements:

Many thanks to D. Abgrall, D. Minodier, Y. Le Goff, D. Cheng, R. White, and B. Chatras for their review and comments.

[I-D.quinn-sfc-arch]:

Authors:

Paul Quinn (editor)
Joel Halpern (editor)

Contributors:

Puneet Agarwal
Andre Beliveau
Kevin Glavin
Ken Gray
Jim Guichard
Surendra Kumar
Darrel Lewis
Nic Leymann
Rajeev Manur
Thomas Nadeau
Carlos Pignataro
Michael Smith
Navindra Yadav

Acknowledgements:

The authors would like to thank David Ward, Abhijit Patra, Nagaraj Bagepalli, Darrel Lewis, Ron Parker, Lucy Yong and Christian Jacquenet for their review and comments.

8. IANA Considerations

[RFC Editor: please remove this section prior to publication.]

This document has no IANA actions.

9. Informative References

[I-D.boucadair-sfc-framework]
Boucadair, M., Jacquenet, C., Parker, R., Lopez, D., Guichard, J., and C. Pignataro, "Service Function Chaining: Framework & Architecture", draft-boucadair-sfc-framework-02 (work in progress), February 2014.

[I-D.quinn-sfc-arch]
Quinn, P. and J. Halpern, "Service Function Chaining (SFC) Architecture", draft-quinn-sfc-arch-05 (work in progress), May 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, June 2011.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, July 2013.

[RFC7498] Quinn, P. and T. Nadeau, "Problem Statement for Service Function Chaining", RFC 7498, April 2015.

Authors' Addresses

Joel Halpern (editor)
Ericsson

Email: jmh@joelhalpern.com

Carlos Pignataro (editor)
Cisco Systems, Inc.

Email: cpignata@cisco.com

Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: April 4, 2015

S. Kumar
J. Guichard
P. Quinn
Cisco Systems, Inc.
J. Halpern
Ericsson
Oct 2014

Service Function Path Optimization
draft-kumar-sfc-sfp-optimization-01

Abstract

Service Function Chaining (SFC) enables services to be delivered by selective traffic steering through an ordered set of service functions. Once classified into an SFC, the traffic for a given flow is steered through all the service functions of the SFC for the life of the traffic flow even though this is often not necessary. Steering traffic to service functions only while required and not otherwise, leads to optimal SFCs with improved latencies, reduced resource consumption and better user experience.

This document describes the rationale, techniques and necessary protocol extensions to achieve such optimization.

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 http://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 April 4, 2015.

Copyright Notice

Table of Contents

1. Introduction

Service function chaining involves steering traffic flows through a set of service functions in a specific order. Such an ordered list of service functions is called a Service Function Chain (SFC). The actual forwarding path used to realize an SFC is called the Service Function Path (SFP).

Service functions forming an SFC are hosted at different points in the network, often co-located with different types of service functions to form logical groupings. Applying a SFC thus requires traffic steering by the SFC infrastructure from one service function to the next until all the service functions of the SFC are applied. Service functions know best what type of traffic they can service and how much traffic needs to be delivered to them to achieve complete delivery of service. As a consequence any service function may potentially request, within its policy constraints, traffic no longer be delivered to it or its function be performed by the SFC infrastructure, if such a mechanism is available.

This document outlines mechanisms to not steer traffic to service functions, on request, while still ensuring compliance to the instantiated policy that mandates the SFC.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Definition Of Terms

This document uses the following terms. Additional terms are defined in [I-D.ietf-sfc-problem-statement], [I-D.ietf-sfc-architecture] and [I-D.quinn-sfc-nsh]. Some are reproduced here only for convenience and the reader is advised to consult the referenced documents.

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 a virtual instance or be embedded in a physical network element. One of multiple Service Functions can be embedded in the same network element. Multiple instances of the Service Function can be enabled in the same administrative domain. A non-exhaustive list of Service Functions includes: firewalls, WAN and application acceleration, Deep Packet Inspection (DPI), server load balancers, NAT44 [RFC3022], NAT64 [RFC6146], HOST_ID

injection, HTTP Header Enrichment functions, TCP optimizer, etc.

Service Node (SN): A virtual or physical device that hosts one or more service functions, which can be accessed via the network location associated with it.

Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic along the service path, which includes delivery of traffic to the connected service functions.

Network Forwarder (NF): The entity, typically part of the network infrastructure, responsible for performing the transport function in traffic forwarding.

Service Controller (SC): The entity responsible for managing the service chains, including create/read/update/delete actions as well as programming the service forwarding state in the network - SFP distribution.

Classifier (CF): The entity, responsible for selecting traffic as well as SFP, based on policy, and forwarding the selected traffic on the SFP after adding the necessary encapsulation. Classifier is implicitly an SFF.

Offload: A request or a directive from the SF to alter the SFP so as to remove the requesting SF from the SFP while maintaining the effect of the removed SF on the offloaded flow.

Un-offload: A request or directive to cancel the effect of Offload - leads to altering the SFP so as to insert the requesting SF back into the SFP and steer the flow to it.

3. Service Function Path Optimization

The packet forwarding path of a SFP involves the classifier, one or more SFFs and all the SFs that are part of the SFP. Packets of a flow are forwarded along this path to each of the SFs, for the life of the flow, whether SFs perform the full function in treating the packet or reapply the cached result, from the last application of the function, on the residual packets of the flow. In other words, every packet on the flow incurs the same latency and the end-to-end SFP latency remains more or less constant subject to the nature of the SFs involved. If an SF can be removed from the SFP, for a specific flow, traffic steering to the SF is avoided for that flow; thus leading to a shorter SFP for the flow. When multiple SFs in a SFP are removed, the SFP starts to converge towards the optimum path, which in its best case starts and terminates at the classifier itself

without incurring any latency associated with traversing the SFP.

Although SFs are removed from the SFP, the corresponding SFC is not changed - this is subtle but an important characteristic of this mechanism. In other words, this mechanism does not alter the SFC and still uses the SFP associated with the SFC.

There are two primary approaches to removing an SF from the SFP. Namely,

- Bypass: Mechanism that alters the SFC. Described in this draft for completeness.
- Simple Offload: Mechanism that alters the SFP alone, does not affect the SFC. This is the primary focus of this draft.

3.1. Bypass

Many service functions do not deliver service to certain types of traffic. For instance, typical WAN optimization service functions are geared towards optimizing TCP traffic and add no value to non-TCP traffic. Non-TCP traffic thus can bypass such a service function. Even in the case of TCP, a WAN optimization SF may not be able to service the traffic if the corresponding TCP flow is not seen by it from inception. In such a situation a WAN optimization SF can avoid the overhead of processing such a flow or reserving resources for it, if it had the ability to request such flows not be steered to it. In other words such service functions need the ability to request they be bypassed for a specified flow from a certain time in the life of that flow.

A seemingly simple alternative is to require service functions pre specify the traffic flow types they add value to, such as the one-tuple: IP protocol-type described above. A classifier built to use such data exposed by SFs, may thus enable bypassing such SFs for specific flows by way of selecting a different SFC that does not contain the SF being removed.

Although knowledge of detailed SF profiles helps SFC selection at the classifier starting the SFC, it leads to shortcomings.

- It adds to the overhead of classification at that classifier as all SF classification requirements have to be met by the classifier.
- It leads to conflicts in classification requirements between the classifier and the SFs. Classification needs of different SFs in the same SFC may vary. A classifier thus cannot classify traffic

based on the classification of one of the SFs in the chain. For instance, even though a flow is uninteresting to one SF on an SFC, it may be interesting to another SF in the same SFC.

- o The trigger for bypassing an SF may be dynamic as opposed to the static classification at the classifier - it may originate at the SFs themselves and involve the control and policy planes. The policy and control planes may react to such a trigger by instructing the classifier to select a different SFC for the flow, thereby achieving SF bypass.

3.2. Simple Offload

Service delivery by a class of service functions involves inspecting the initial portion of the traffic and determining whether traffic should be permitted or dropped. In some service functions, such an inspection may be limited to just the five tuple, in some others it may involve protocol headers, and in yet others it may involve inspection of the byte stream or application content based on the policy specified. Firewall service functions fall into such a class, for example. In all such instances, servicing involves determining whether to permit the traffic to proceed onwards or to deny the traffic from proceeding onwards and drop the traffic. In some cases, dropping of the traffic may be accompanied with the generation of a response to the originator of traffic or to the destination or both. Once the service function determines the result - permit or deny (or drop), it simply applies the same result to the residual packets of the flow by caching the result in the flow state.

In essence, the effect of service delivery is a PERMIT or a DENY action on the traffic of a flow. This class of service functions can avoid all the overhead of processing such traffic at the SF, by simply requesting another entity in the SFP, to assume the function of performing the action determined by the service function. Since PERMIT and DENY are very simple actions other entities in the SFP are very likely to be able to perform them on behalf of the requesting SF. A service function can thus offload simple functions to other entities in the SFP.

Since SFF is the one steering traffic to the SFs and hence is on the SFP, is a natural entity to assume the offload function. An SF not interested in traffic being steered to it can simply perform a simple offload by indicating a PERMIT action along with an OFFLOAD request. The SFF responsible for steering the traffic to the SF takes note of the ACTION and offload request. The OFFLOAD directive and the ACTION received from the requesting SF are cached against the SF for that flow. Once cached, residual packets on the flow are serviced by the cached directive and action as if being serviced by the corresponding

SF.

3.2.1. Stateful SFF

SFFs are the closest SFC infrastructure entities to the service functions. SFFs may be state-full and hence can cache the offload and action in both of the unidirectional flows of a connection. As a consequence, action and offload become effective on both the flows simultaneously and remain so until cancelled or the flow terminates.

SFFs may not always honor the offload requests received from SFs. This does not affect the correctness of the SFP in any way. It implies that the SFs can expect traffic to arrive on a flow, which it offloaded, and hence must service them, which may involve requesting an offload again. It is natural to think of an acknowledgement mechanism to provide offload guarantees to the SFs but such a mechanism just adds to the overhead while not providing significant benefit. Offload serves as a best effort mechanism.

3.2.2. Packet Re-ordering

Offload mechanism creates short time-windows where packet re-ordering may occur. While SFs request flows be offloaded to SFFs, packets may still be in flight at various points along the SFP, including some between the SFF and the SF. Once the offload decision is received and committed into the flow entry at the SFF, any packets arriving after and destined to the offloading SF are treated to the offload decision and forwarded along (if it is a PERMIT action). Inflight packets to the offloading SF may arrive at the SFF after one or more packets are already treated to the offload decision and forwarded along.

This is a transitional effect and may not occur in all cases. For instance, if the decision to offload a flow by an SF is based on the first packet of TCP flow, a reasonable time window exists between the offload action being committed into the SFF and arrival of subsequent packet of the same flow at that SFF. Likewise, request/response based protocols such as HTTP may not always be subject to the re-ordering effects.

3.2.3. Policy Implications

Offload mechanism may be controlled by the policy layer. The SFs themselves may have a static policy to utilize the capability offered by the SFC infrastructure. They could also be dynamic and controlled by the specific policy layer under which the SFs operate.

Similarly, the SFC infrastructure, specifically the classifiers and

the SFFs, may be under the SFC infrastructure control plane policy controlling the decision to honor offloads from an SF. This policy in turn may be coarse-grain, at the SF level, and hence static. It can also be fine grain and hence dynamic but it adds to the overhead of policy distribution.

Policy model related to offloads is out of scope of this document.

3.2.4. Capabilities Exchange

Simple offloads can be exposed and negotiated a priori as a capability between the SFFs and the SFs or the corresponding control layers. In the simplest of the implementations, this is provided by the SFC infrastructure and the SFs are statically configured to utilize them without capabilities negotiation, within the constraints of the SF specific policies.

Capabilities exchange is outside the scope of this document.

4. Methods For SFP Optimization

There are a number of different models that may be used to facilitate shortest SFP realization. We present two here.

The shortest SFP methods discussed in the following sections require signaling among the participant components to communicate offload and permit/deny actions. The signaling may be performed in the data-plane or in the control plane.

a. Data-plane: An SFC specific communication channel is needed for SNs to communicate the offload request along with the SF treated packet. [NSH] defines a header specifically for carrying SFP along with metadata and provides such a channel for use with offloads. Necessary bits need to be allocated in NSH to convey the action as well as the offload directive. This signaling may be limited to SN and SFF or may continue from one SFF to another SFF or the classifier. It may also involve signaling directly from the SF to the classifier.

b. Control-plane: Messages are required between the SN and the service controller as well as between the SFF and the control plane. Service controller messaging is out of scope of this document and it is assumed to be service controller specific.

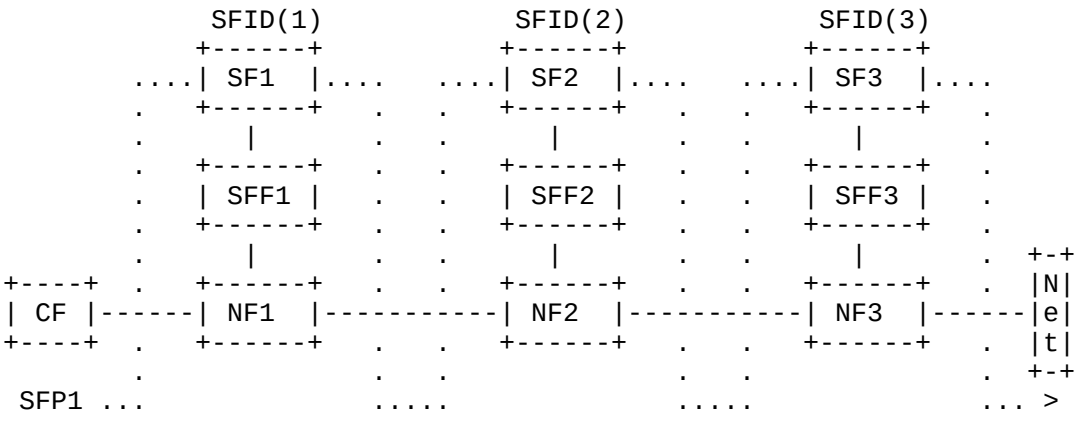
4.1. Hop-by-hop Offload

SNs receive traffic on an overlay from the SFF. SNs service the traffic and turn them back to the SFF on an overlay or forward the traffic on the underlay. In the former case, along with returning the traffic to SFF, they can perform simple offload by signaling OFFLOAD and ACTION to the SFF. SFF caches the OFFLOAD and ACTION while forwarding the serviced packet onwards to the next service hop on the SFP or dropping it. SFF can now enforce the OFFLOAD and ACTION on the residual packets of the flow.

Additionally, SFF may choose to signal the upstream SFFs of the OFFLOAD and ACTION received from an SF. This may continue recursively until the first SFF is reached, which is the classifier itself.

By performing such hop-by-hop offloads, SFP can be reduced to an optimum one, steering traffic to only those SFs that really need to see the traffic.

Figure 1 to Figure 3 show an example of SF and SFF performing an offload operation and the effect thereafter on the SFP.



Service Function Chain, SFC1 = {SF1, SF2, SF3}
where SF1, SF2 and SF3 are three service functions.
Service Function Path SFP1 is the SFP for SFC1.
Classifier CF starts SFP1 based on policy.

Figure 1: SFC1 with corresponding SFP1

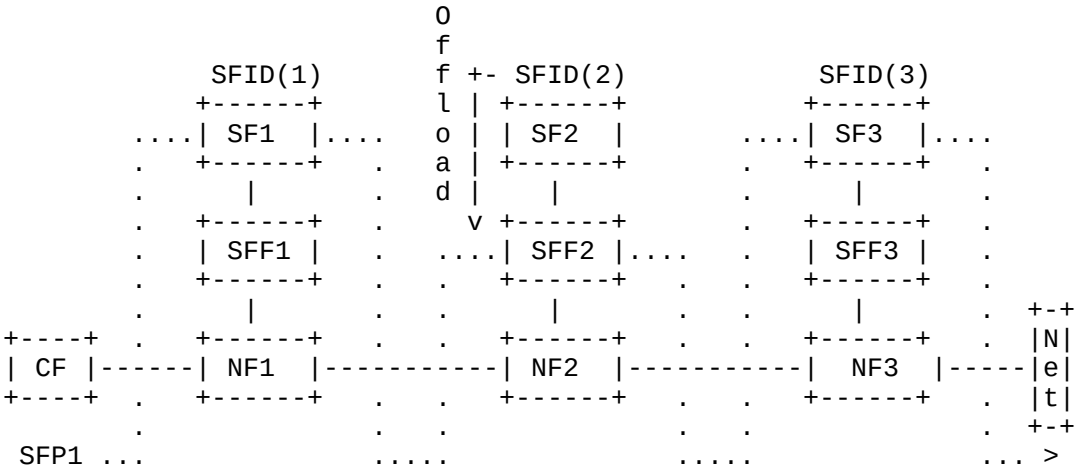


Figure 2: SFP1 after SFID(2) performs an Offload

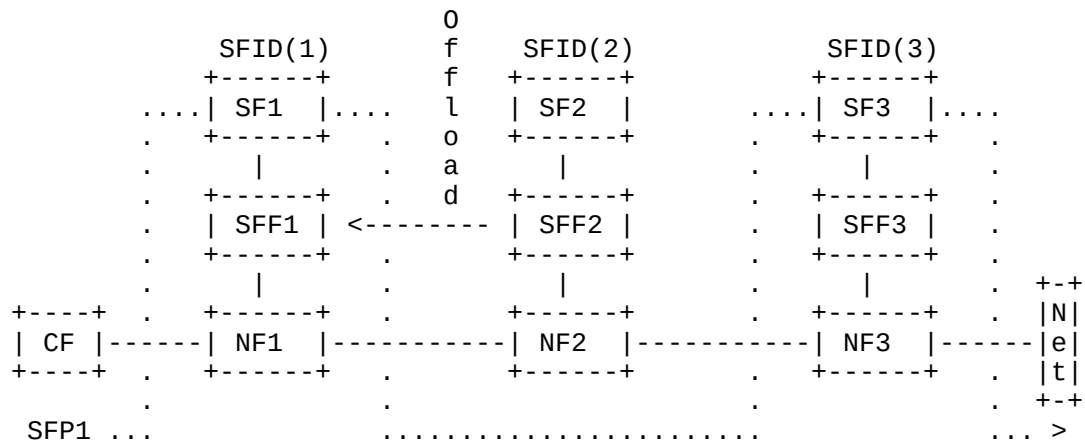


Figure 3: SFP1 after SFF2 propagates the Offload upstream to SFF1

4.1.1. Progression Of SFP Optimization

SFP optimization happens at two levels:

Level-1: Collapsing of the SFF-to-SF hops into the SFF or the SFC infrastructure.

Level-2: Reduction of the (collapsed) SFP within the SFC infrastructure to the shortest possible.

Figure 1 to Figure 3 show one sequence of offload events leading to a shorter SFP. Although not shown in these figures, further offload events ultimately lead to an optimum SFP.

Below steps show one such sequence of offload events that lead to such an optimum SFP.

Stage-1: Prior to any offloads, service function path SFP1 (corresponding to SFC1) has the following actual forwarding path as shown in Figure 1:
CF ->
NF1 -> SFF1 -> SF1 -> SFF1 -> NF1 ->
NF2 -> SFF2 -> SF2 -> SFF2 -> NF2 ->
NF3 -> SFF3 -> SF3 -> SFF3 -> NF3 ->

Stage-2: After SF2 performs a simple offload, which is signaled to upstream SFFs - SFF1, Classifier, SFP1 forwarding path changes to the below as shown in Figure 3:
CF ->
NF1 -> SFF1 -> SF1 -> SFF1 -> NF1 ->
NF3 -> SFF3 -> SF3 -> SFF3 -> NF3 ->

Stage-3: After SF3 performs simple offload, which is signaled to upstream SFFs - SFF2, SFF1, and Classifier, SFP1 forwarding path changes to the below (figure not shown) />:
CF ->
NF1 -> SFF1 -> SF1 -> SFF1 -> NF1 ->

Stage-4: After SF1 performs simple offload, which is signaled to upstream SFFs - Classifier, SFP1 forwarding path changes to the below (figure not shown):
CF ->

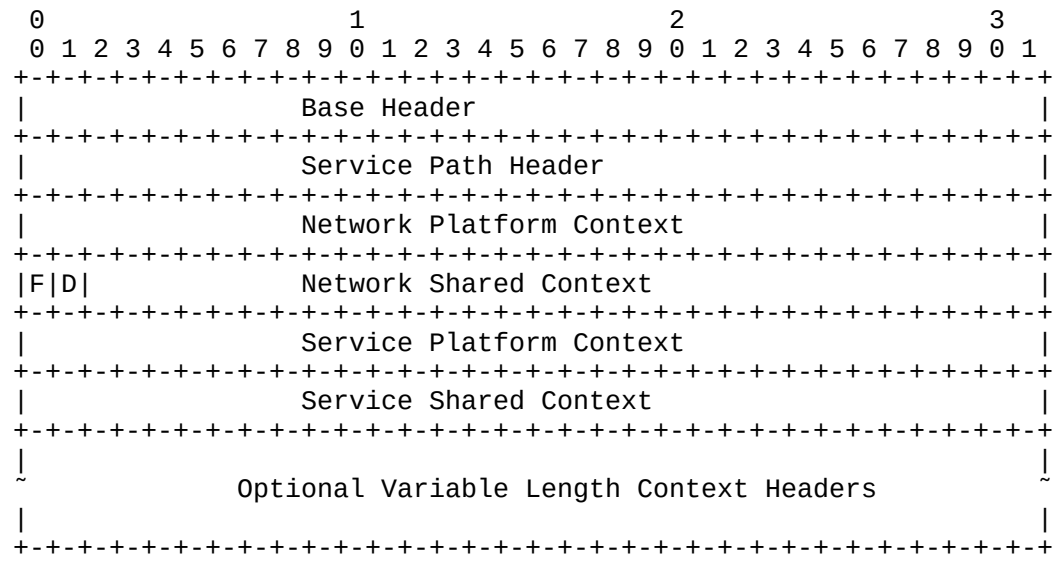
4.2. Service Controller Offload

Each SN signals the service controller of the OFFLOAD and ACTION via control plane messaging for a specific flow. The service controller then signals the appropriate SFFs to offload the requested SFs, there by achieving the hop-by-hop offload behavior.

The service controller has full knowledge of all the SFs of the SFP offloading the flow and hence can determine the optimum SFP within the Service Controller and program the appropriate SFFs to achieve SFP optimization.

5. Offload Data-plane Signaling

Since Offload and action are signaled at the time of returning the traffic to SFF, post servicing the traffic, such signaling can be integrated into the service header of the packet. Figure 4 shows the bits necessary to achieve the signaling using the SFC encapsulation as described in [I-D.quinn-sfc-nsh].



F : Offload bit; indicates offload request when F is set to 1
D : Drop (or Deny) bit, drop when D=1 and permit when D=0;
Drop bit is valid only when F is set to 1

Figure 4: NSH Offload and ACTION Bits

Although SFs can signal SFFs by piggy backing on the serviced packet, SFFs cannot signal in a similar fashion. This is because the traffic is forwarded along the SFP to the next or downstream SFF. The offload signaling has to go to the upstream SFFs, in the opposite direction. SFFs have a choice: perform out-of-band signaling towards the upstream SFFs or wait until traffic arrives on the opposite flow on the reverse SFP, where SFPs are symmetric. SFFs can then piggyback the offload signaling on the reverse traffic towards the upstream SFFs. The actual method employed to signal offload between the SFFs is implementation specific.

6. Acknowledgements

The authors would like to thank Nagaraj Bagepalli for his review comments.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

Security of the offload signaling mechanism is very important. This document does not advocate any additional security mechanisms other the data plane and control plane signaling security mechanisms.

9. References

9.1. Normative References

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-02 (work in progress), September 2014.

[I-D.quinn-sfc-nsh]
Quinn, P., Guichard, J., Fernando, R., Surendra, S., Smith, M., Yadav, N., Agarwal, P., Manur, R., Chauhan, A., Elzur, U., Garg, P., McConnell, B., and C. Wright, "Network Service Header", draft-quinn-sfc-nsh-03 (work in progress), July 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[I-D.ietf-sfc-problem-statement]
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-10 (work in progress), August 2014.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

Authors' Addresses

Surendra Kumar
Cisco Systems, Inc.
170 W. Tasman Dr.
San Jose, CA 95134

Email: smkumar@cisco.com

Jim Guichard
Cisco Systems, Inc.

Email: jguichar@cisco.com

Paul Quinn
Cisco Systems, Inc.

Email: paulq@cisco.com

Joel Halpern
Ericsson

Email: joel.halpern@ericsson.com

SFC working group
Internet-Draft
Intended status: Informational
Expires: March 10, 2017

H. Song
J. You
L. Yong
Y. Jiang
L. Dunbar
Huawei
N. Bouthors
Qosmos
D. Dolson
Sandvine

September 6, 2016

SFC Header Mapping for Legacy SF
draft-song-sfc-legacy-sf-mapping-08

Abstract

A Service Function Chain (SFC) defines a set of abstract Service Functions (SF) and ordering constraints that must be applied to packets and/or frames selected as a result of classification. One assumption of this document is that legacy service functions can participate in service function chains without supporting the SFC header, or even being aware of it. This document provides some of the mechanisms between an SFC proxy and an SFC-unaware service function (herein termed "legacy SF"), to identify the SFC header associated with a packet that is returned from a legacy SF, without an SFC header being explicitly carried in the wired protocol between SFC proxy and legacy SF.

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 http://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 March 10, 2017.

Table of Contents

1. Introduction

A Service Function Chain (SFC) [RFC7665] defines a set of abstract service functions and ordering constraints that must be applied to packets and/or frames selected as a result of classification. One assumption of this document is that some service functions may remain as legacy implementations, i.e. SFC-unaware SFs. The SFC proxy is proposed to act as a gateway between the SFC encapsulation and SFC-unaware SFs. The SFC proxy removes the SFC header and then sends the packet to a legacy SF for processing, but how to associate the

original SFC header with the packet returned from the legacy SF needs to be considered.

This document describes some of the mechanisms between an SFC proxy and a legacy SF, to identify the SFC header associated with a packet that is returned from a legacy SF. The benefit for supporting legacy SF is that SFC-unaware SFs can exist in the SFC-enabled domain. An SFC proxy allows a legacy SF to function in the SFC-enabled domain without modification of the legacy SF.

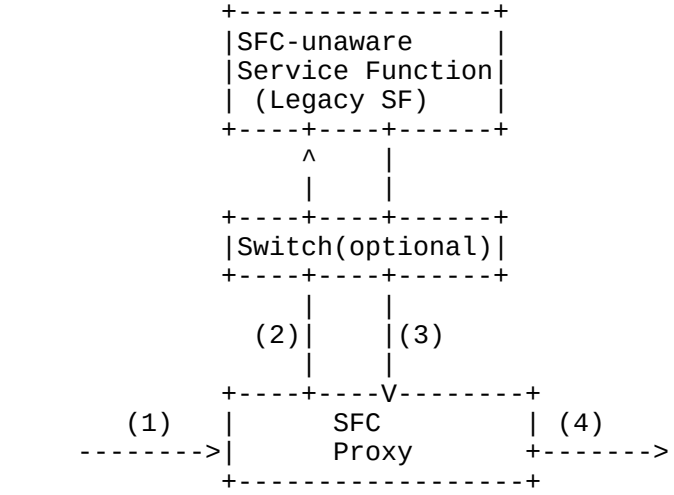


Figure 1: Procedure of a packet processed by a legacy SF

Different classes of legacy SF may have variable support for different types of packets with respect to parsing and semantics (e.g., some classes of legacy SF may accept VLAN-tagged traffic; others may not), usually depending on device configuration. For example, by creation of VLANs, traffic is steered through a firewall.

This document focuses heavily on legacy SFs that are transparent at layer 2. In particular we assume the following conditions apply in the class of legacy SF we are considering proxying:

1. Traffic is forwarded between pairs of interfaces, such that packets received on the "left" are forwarded on the "right" and vice versa.

2. A packet is forwarded between interfaces without modifying the layer 2 header; i.e., neither source MAC nor destination MAC is modified.

3. When supported, VLAN-tagged or Q-in-Q packets are forwarded with the original VLAN tag(s) intact (S-tags and C-tags).

4. Traffic may be discarded by some functions (e.g., by a firewall).

5. Traffic may be injected in either direction by some functions (e.g., extra data coming from a cache, or simply TCP retransmissions). We assume injected traffic relates to a layer 3 or layer 4 flow, and the SF clones layer 2 headers from exemplar packets of the same flow.

6. Traffic may be modified by some functions at layer 3 (e.g., DSCP marking) or higher layers (e.g., HTTP header enrichment or anonymization). Note that modification can be considered a special case of discarding followed by injection.

7. Traffic may be reordered by some functions (e.g., due to queuing/scheduling).

We leave the legacy SFs which modify the original layer 2 packet headers as an open issue for further study.

To support this class of legacy SF, if the payload in the SFC encapsulation is layer 3 traffic, the SFC proxy will extract the layer 3 payload from SFC encapsulation and prepend a new layer 2 header before sending the packet to the SF. However if the payload in the SFC encapsulation is layer 2 traffic, the SFC proxy may extract the layer 2 packet from SFC encapsulation, modify the original source MAC address and use the new source MAC address for mapping to the stored SFC and layer 2 headers when the packets are returned to the SFC proxy. This will not impact the SF processing. The SF will send the traffic back after processing.

As shown in Figure 1, there are four steps. The SFC proxy receives a packet (1) from an SFF, and removes its SFC header, which may optionally contain metadata, and store the SFC header locally, and then (2) sends the de-encapsulated packet to the SF. After the SF processes the packet, the packet will be sent back (3) to the SFC proxy. The SFC proxy retrieves the pre-stored SFC header accordingly, determines the SFC header for the next stage of the path and encapsulates the packet with the next SFC header, returning the packet to an SFF (4).

2. Terminology

The terminology used in this document is defined below:

Legacy SF: A conventional service function that does not support SFC header, i.e., SFC-unaware SF.

Transparent SF: A service function that does not change any bit of the layer 2/3/4 packet header sent to it, but it may drop the packet.

Non-transparent SF: A service function that changes some bits of the layer 2/3/4 packet header sent to it.

SFC Proxy: Removes and inserts SFC encapsulation on behalf of an SFC-unaware service function. SFC proxies are logical elements.

3. Mechanisms

The mapping mechanisms between the SFC proxy and the transparent or non-transparent legacy SFs are discussed in this section. The mechanisms used in this document require that each forwarding entity (i.e., SFC proxy) and its connected service functions are in the same layer 2 network. The detailed definitions of SFC proxy and SFC-unaware SFs is discussed in [RFC7665].

3.1. For Transparent Service Functions

3.1.1. VLAN

If the service function is transparent to packet headers, for example, layer-2-transparent SF, then VLAN can be used for mapping between the SFC proxy and SF. It is assumed that the switch between the SFC proxy and SF delivers traffic for all VLANs, or the SFC proxy and SF may be directly connected.

The SFC proxy removes the SFC header and sends the packet to the SF, with encapsulating a certain VLAN ID that can represent the SFC header. The legacy SF is supposed to accept VLAN-tagged packets and send them back on the same VLAN. It is assumed that the SF is able to process Ethernet packets with VLAN tags and also accept a wide range of VLAN tags. The SFC proxy locally maintains the mapping between VLAN ID/direction and the SFC header.

When receiving the returned packet from the SF, the SFC proxy removes the VLAN part from the packet and retrieves the corresponding SFC header according to the VLAN ID and the direction of packet travel, and then encapsulates SFC header into that packet before sending to

the next service function. Packet direction is required because the SFC header for left-to-right packets is different than the SFC header for right-to-left packets.

3.1.2. VXLAN

If the SFC proxy and SF are already deployed in a nested VLAN network, the VLAN mapping method is not applicable. Then VXLAN [RFC7348] can be used for the mapping, i.e. VNI can be used for the mapping between them. VXLAN is a Layer 2 overlay scheme over a Layer 3 network. It uses MAC Address-in-User Datagram Protocol (MAC-in-UDP) encapsulation. The drawback of this mechanism is that it requires both SFC proxy and SF to support VXLAN.

This approach has similar features and drawbacks of the VLAN scheme, but the number of possible VNIs is larger.

3.1.3. Ethernet MAC Address

The MAC address also can be used to associate an SFC header between the SFC proxy and SF; i.e., each SFC header will be assigned a source MAC address on the SFC proxy. When the SFC proxy receives the returned packet from the SF, it retrieves the packet’s original SFC header by using the source MAC address as a key. And then it encapsulates the packet with that SFC header and sends to the next hop.

An issue with the source-MAC address approach is that there is not symmetry between packets going left-to-right with packets going right-to-left. Such symmetry might be assumed by some legacy SFs. For example, if a layer-2-transparent SF responds to a TCP SYN with a TCP RST, it might do so by reversing the source and destination of the layer 2 header. Such a packet received by the SFC proxy would not result in finding of the correct SFC header. It is assumed that the SF passes the MAC header through without even reversal. A variation that is symmetric assigns a unique source/destination pair for each unique SFC header.

3.1.4. 5-tuple

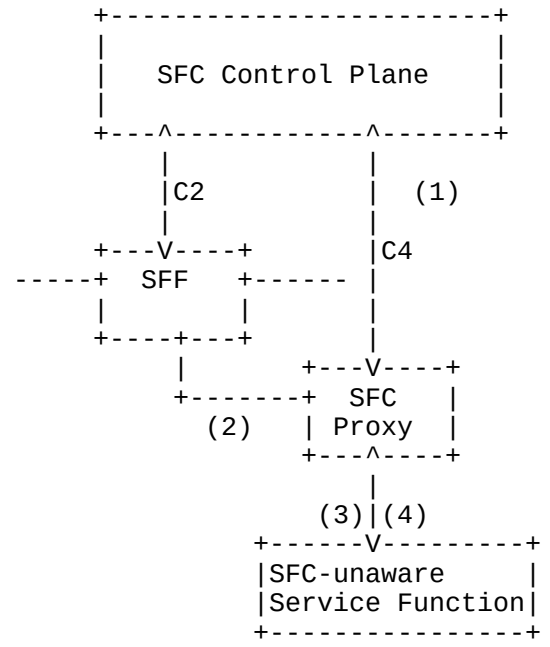
The 5-tuple of a packet carried within SFC encapsulation can be used by the SFC proxy as a key to associate an SFC header when the 5-tuple is not modified by the legacy SF. The SFC proxy maintains a mapping table for the 5-tuple and the SFC header. When the packet returns from the SF instance, the original SFC header for this packet can be retrieved by inquiring the mapping table using 5-tuple as the key. However, this method may not work in multi-tenant scenario, as such uniqueness could be valid only within the scope of a single tenant.

So if the SFC is provided as a multi-tenant service, this method would fail.

3.2. For Non-transparent Service Functions

Non transparent service functions including NAT (Network Address Translation), WOC (WAN Optimization Controller) and etc, are more complicated, as they may change any part of the original packet sent to them. It is better to analyze case by case, to utilize a specific field that the SF does not change for the mapping and retrieving the SFC header. We would like to leave it for open discussion.

The Figure below shows an example procedure that SFC proxy can learn the behavior of the SF changing the packet. In this example, the following method is used for SFC header mapping. The SF needs to report its mapping rules (e.g., 5-tuple mapping rules) to the control plane (e.g., by static configuration), and then the control plane can notify the SFC proxy the mapping information (step 1) via interface C4 [I-D.ietf-sfc-control-plane]. According to the mapping information, the SFC proxy can establish a mapping table for the SFC header, the original header, and the processed header of the packet. After receiving the packet from the SF (step 4), the SFC proxy retrieves the SFC header from the mapping table by using the processed header as a key.



4. Operation Considerations

4.1. Examplar Mechanisms

The following table gives some examplar methods and the conditions to use.

Table 1: Mapping Examples

	Methods	Stored Key-Value	Application Scenario
For Trans-parent SF	VLAN	(Direction, VLAN ID, SFC header) e.g., assign a VLAN ID per bidirectional path-pair	L2 header won't be modified by the SF.
	VXLAN	(Direction, VNI, SFC header) e.g., assign a VNI per bidirectional path-pair	The SF is required to support VXLAN. VNI is not modified by the SF.
	5-tuple	(5-tuple, SFC header) The SFC proxy maintains the mapping table for 5-tuple and the SFC header. Note: an SFC header for each direction of a TCP flow.	5-tuple is not modified by the SF.
For Non-trans-parent SF	Case-by-case	Mapping rules: e.g. 5-tuple -> 5-tuple' SFC Proxy: 5-tuple -> 5-tuple' 5-tuple'-> SFC header	The SFC proxy is configured or is able to obtain the mapping rules of the SF. The SF modifies the 5-tuple based on the mapping rules.

4.2. Challenges to Support Legacy SF

The key problem contemplated in this document is: what packet header should be put on the packets sent to a legacy SF such that packets returned from the legacy SF can be mapped to the original SFC header. We need to consider the relationship between an SFC path and flows

within the path. Should the path act as a qualifier to the flow, or should a flow be allowed to change paths? We assume flows can change path; this means that a given legacy SF cannot handle traffic from more than one routing domain. (Private IP addresses cannot be qualified by the SFC header; different VPNs must use different legacy SFs.)

Because we've assumed that a flow can be on multiple paths, or change paths, or if metadata can vary during the life of a flow, we need to ask to what extent packet accuracy matters. If the SFC header used with a flow is changed from one path to another by the classifier, does it matter if packets retain exactly the original SFC header? If the change is to handle routing updates or fail-over then it would be acceptable to put all packets returning from the legacy SF onto the most recently updated header. If metadata is changed, can that update be applied to all packets of a flow, or does it apply to a specific packet?

In the case that changes to paths and metadata are considered updates to the flow vs. packet properties, the SFC proxy can find the SFC header based on flow (e.g., the 5-tuple of the returning IP packet). If, in contrast, packet accuracy of SFC headers does matter, (e.g., the metadata says something about the specific packet associated with it), then some form of per-packet bookkeeping must be done by the SFC proxy and the 5-tuple cannot be used for the mapping to retrieve the original SFC header.

When packet accuracy does matter, packets injected by the legacy SF pose a fundamental problem. Is there any correct SFC header that can be added? Observation: the same problem exists for a normal (not legacy) SF that wishes to modify or inject a packet.

Because the SFC proxy needs to keep dynamic state by storing packet headers, an expiration time should be used for each mapping entry in the SFC proxy. If the SFC header in that entry has not been witnessed or retrieved after the expiration time, the entry will be deleted from the entry table.

Observation: if metadata is not used, the number distinct SFC headers is known at configuration time, equivalent to the number of paths configured to pass through the SF. The mappings between SFC headers and layer 2 encodings could be configured at this time vs. at run time. However, if metadata is used, a combinatorial explosion of distinct SFC headers may result, which is a problem for any device attempting to store them for later retrieval.

4.3. Metadata

Some classes of SF may need to inject new packets, for example a transparent cache sending content from its disk. The legacy SF usually encapsulates the new packets with the same encapsulation with the related received packets, e.g. with the same 5-tuple, or V-LAN ID. The SFC proxy would associate the new packet with the corresponding SFC header based on the mechanisms discussed in Section 3. However, per-packet metadata should be prohibited for this case.

Some classes of SF may need to inject a packet in the opposite direction of a received packet, for example a firewall responding to a TCP SYN with a RST. If the RST generator is VLAN-type legacy, it may know what VLAN to use; then the SFC proxy would translate VLAN into a reverse SFP and attach a corresponding SFC header insetad of the original SFC header. In this case, the SFC proxy should be configured with the bidirectional SFP, i.e. SFC proxy needs to be designed according to the properties of the SF. Similarly, packet-specific metadata is not recommended to be used.

We leave the metadata model as an open issue that will be documented in other documents. In some cases this information will also assist normal (non-legacy) SFs that wish to modify or inject packets.

5. Security Considerations

When the layer 2 header of the original packet is modified and sent to the SF, if the SF needs to make use of the layer 2 header, it may cause security threats. There may be security issues with state exhaustion on the SFC proxy, e.g., exhausting VLAN IDs, or exhausting 5-tuple state memory.

6. Acknowledgement

The authors would like to thank Ron Parker and Joel Halpern for their valuable comments.

7. References

7.1. Normative References

[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, <http://www.rfc-editor.org/info/rfc7348>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <http://www.rfc-editor.org/info/rfc7665>.

7.2. Informative References

[I-D.ietf-sfc-control-plane] Boucadair, M., "Service Function Chaining (SFC) Control Plane Components & Requirements", draft-ietf-sfc-control-plane-07 (work in progress), August 2016.

Authors' Addresses

Haibin Song
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: haibin.song@huawei.com

Jianjie You
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, 210012
China

Email: youjianjie@huawei.com

Lucy Yong
Huawei
5340 Legacy Drive
Plano, TX 75025
U.S.A.

Email: lucy.yong@huawei.com

Yuanlong Jiang
Huawei
Bantian, Longgang district
Shenzhen 518129
China

Email: jiangyuanlong@huawei.com

Linda Dunbar
Huawei
1700 Alma Drive, Suite 500
Plano, TX 75075
U.S.A.

Email: ldunbar@huawei.com

Nicolas Bouthors
Qosmos

Email: nicolas.bouthors@qosmos.com

David Dolson
Sandvine
408 Albert Street
Waterloo, ON N2L 3V3
Canada

Email: ddolson@sandvine.com