Hierarchical Service Function Chaining (hSFC)
draft-dolson-sfc-hierarchical-06

Abstract

Hierarchical Service Function Chaining (hSFC) is a network architecture allowing an organization to compartmentalize a large-scale network into multiple domains of administration.

The goals of hSFC are to make a large-scale network easier to reason about, simpler to control and to able support independent functional groups within large operators.

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 December 29, 2016.
Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction .................. 3
2. Hierarchical Service Function Chaining (hSFC) ............ 4
   2.1. Top Level .................. 4
   2.2. Lower Levels ............... 5
3. Internal Boundary Node (IBN) ................. 7
   3.1. IBN Path Configuration ............ 7
       3.1.1. Flow-Stateful IBN ............ 7
       3.1.2. Encoding Upper-Level Paths in Metadata ......... 9
       3.1.3. Using Unique Paths per Upper-Level Path ......... 9
       3.1.4. Nesting Upper-Level NSH within Lower-Level NSH ...... 10
       3.1.5. Stateful / Metadata Hybrid .......... 11
   3.2. Gluing Levels Together ............. 12
   3.3. Decrementing Service Index .......... 12
4. Sub-domain Classifier ................. 13
5. Control Plane Elements ................ 13
6. Extension for Adopting to NSH-Unaware Service Functions ..... 14
   6.1. Purpose ........................ 15
   6.2. Requirements for IBN .......... 16
7. Acknowledgements .................. 16
8. IANA Considerations .................. 17
9. Security Considerations ............... 17
10. References ........................ 17
    10.1. Normative References .......... 17
    10.2. Informative References .......... 18
Appendix A. Examples of Hierarchical Service Function Chaining 18
    A.1. Reducing the Number of Service Function Paths ......... 18
    A.2. Managing a Distributed Data-Center Network ............ 20
Authors’ Addresses .................. 22
1. Introduction

Service Function Chaining (SFC) is a technique for prescribing differentiated traffic forwarding policies within an SFC-enabled domain. SFC is described in detail in the SFC architecture document [RFC7665], and is not repeated here.

In this document we consider the difficult problem of implementing SFC across a large, geographically dispersed network comprised of millions of hosts and thousands of network forwarding elements, involving multiple operational teams (with varying functional responsibilities). We expect asymmetrical routing is inherent in the network, while recognizing that some Service Functions (SFs) require bidirectional traffic for transport-layer sessions (e.g., NATs, firewalls). We assume that some Service Function Paths (SFPs) need to be selected on the basis of application-specific data visible to the network, with transport-layer coordinate (typically, 5-tuple) stickiness to specific SF instances.

Note: in this document, the notion of the "path" of a packet is the series of SF instances traversed by a packet. The means of delivering packets between SFs (the forwarding mechanisms enforced in the underlying network) is not relevant to the discussion.

Difficult problems are often made easier by decomposing them in a hierarchical (nested) manner. So instead of considering an omniscient SFC Control Plane that can manage (create, withdraw, supervise, etc.) complete SFPs from one end of the network to the other, we decompose the network into smaller sub-domains. Each sub-domain may support a subset of the network applications or a subset of the users. The criteria for determining decomposition into SFC-enabled sub-domains are beyond the scope of this document.

Note that decomposing a network into multiple SFC-enabled domains should permit end-to-end visibility of SFs and SFPs. Decomposition should also be implemented with special care to ease monitoring and troubleshooting of the network and services as a whole.

An example of simplifying a network by using multiple SF domains is further discussed in [I-D.ietf-sfc-dc-use-cases].

We assume the SFC-aware nodes use NSH [I-D.ietf-sfc-nsh] or a similar labeling mechanism.

The "domains" discussed in this document are assumed to be under control of a single organization, such that there is a strong trust relationship between the domains. The intention of creating multiple domains is to improve the ability to operate a network. It is
outside of the scope of the document to consider domains operated by different organizations.

2. Hierarchical Service Function Chaining (hSFC)

A hierarchy has multiple levels. The top-most level encompasses the entire network domain to be managed, and lower levels encompass portions of the network.

2.1. Top Level

Considering the example depicted in Figure 1, a top-level network domain includes SFC data plane components distributed over a wide area, including:

- Classifiers (CFs),
- Service Function Forwarders (SFFs) and
- Sub-domains.

For the sake of clarity, components of the underlay network are not shown; an underlay network is assumed to provide connectivity between SFC data plane components.

Top-level SFPs carry packets from classifiers through a series of SFFs and sub-domains, with the operations within sub-domains being opaque to the higher levels.

We expect the system to include a top-level control-plane having responsibility for configuring forwarding and classification (see [I-D.ietf-sfc-control-plane]). The top-level Service Chaining control-plane manages end-to-end service chains and associated service function paths from network edge points to sub-domains and configuring top-level classifiers at a coarse level (e.g., based on source or destination host) to forward traffic along paths that will transit appropriate sub-domains. Figure 1 shows one possible service chain passing from edge, through two sub-domains, to network egress. The top-level control plane does not configure classification or forwarding within the sub-domains.

At this network-wide level, the number of SFPs required is a linear function of the number of ways in which a packet is required to traverse different sub-domains and egress the network. Note that the various paths which may be taken within a sub-domain are not represented by distinct network-wide SFPs; specific policies at the ingress nodes of each sub-domain bind flows to sub-domain paths.
Packets are classified at the edge of the network to select the paths by which sub-domains are to be traversed. At the ingress of each sub-domain, paths are reclassified to select the paths by which SFs in the sub-domain are to be traversed. At the egress of each sub-domain, packets are returned to the top-level paths. Contrast this with an approach requiring the top-level classifier to select paths to specify all of the SFs in each sub-domain.

It should be assumed that some SFs require bidirectional symmetry of paths (see more in Section 4). Therefore the classifiers at the top level must be configured with policies ensuring outgoing packets take the reverse path of incoming packets through sub-domains.

One path is shown from edge classifier to SFF1 to Sub-domain#1 (residing in data-center1) to SFF1 to SFF2 (residing in data-center 2) to Sub-domain#2 to SFF2 to network egress.

Figure 1: Network-wide view of top level of hierarchy

2.2. Lower Levels

Each of the sub-domains in Figure 1 is an SFC-enabled domain.

Unlike the top level, data packets entering the sub-domain are already SFC-encapsulated. Figure 2 shows a sub-domain interfaced with a higher-level domain by means of an Internal Boundary Node.
(IBN). It is the purpose of the IBN to apply classification rules and direct the packets to the selected local SFPs terminating at an egress IBN. The egress IBN finally restores packets to the original SFC shim and hands them off to SFFs.

Each sub-domain intersects a subset of the total paths that are possible in the higher-level domain. An IBN is concerned with higher-level paths, but only those traversing its sub-domain. A top-level control element may configure the IBN as an SF (i.e., the IBN plays the SF role in the top-level domain).

Each sub-domain is likely to have a control-plane that can operate independently of the top-level control-plane. The sub-domain control-plane configures the classification and forwarding rules in the sub-domain. The classification rules reside in the IBN, where SFC encapsulation of the top-level domain is converted to/from SFC encapsulation of the lower-level domain.

---

**Figure 2: Sub-domain within a higher-level domain**

---

If desired, the pattern can be applied recursively. For example, SF#1.1 in Figure 2 could be a sub-domain of the sub-domain.

3. Internal Boundary Node (IBN)

A network element termed "Internal Boundary Node" (IBN) bridges packets between domains. It behaves as an SF to the higher level, and looks like a classifier and end-of-chain to the lower level.

To achieve the benefits of hierarchy, the IBN should be applying more granular traffic classification rules at the lower level than the traffic passed to it. This means that the number of SFPs within the lower level is greater than the number of SFPs arriving to the IBN.

The IBN is also the termination of lower-level SFPs. This is because the packets exiting lower-level SF paths must be returned to the higher-level SF paths and forwarded to the next hop in the higher-level domain.

3.1. IBN Path Configuration

An operator of a lower-level domain may be aware of which high-level paths transit their domain, or they may wish to accept any paths.

When packets enter the sub-domain, the Service Path Identifier (SPI) and Service Index (SI) are re-marked according to the path selected by the classifier.

After exiting a path in the sub-domain, packets can be restored to an original upper-level SFP by these methods:

1. Saving SPI and SI in transport-layer flow state,
2. Pushing SPI and SI into metadata,
3. Using unique lower-level paths per upper-level path coordinates,
4. Nesting NSH headers, encapsulating the higher-level NSH headers within the lower-level NSH headers,
5. Saving upper-level by a flow ID and placing an hSFC flow ID into metadata,

3.1.1. Flow-Stateful IBN

An IBN can be flow-aware, returning packets to the correct higher-level SFP on the basis of the transport-layer coordinates (typically, a 5-tuple) of packets exiting the lower-level SFPs.
When packets are received by the IBN on a higher-level path, the encapsulated packets are parsed for IP and transport-layer (TCP, UDP, etc.) coordinates. State is created, indexed by these coordinates ((source-IP, destination-IP, source-port, destination-port and transport protocol) typically). The state contains at least critical fields of the encapsulating SFC header (or perhaps the entire header).

The simplest approach has the packets return to the same IBN at the end of the chain that classified the packet at the start of the chain. This is because the required transport-coordinates state is rapidly changing and most efficiently kept locally. If the packet is returned to a different IBN for egress, transport-coordinates state must be synchronized between the IBNs.

When a packet returns to the IBN at the end of a chain, the SFC header is removed, the packet is parsed for IP and transport-layer coordinates, and state is retrieved from them. The state contains the information required to forward the packet within the higher-level service chain.

State cannot be created by packets arriving from the lower-level chain; when state cannot be found for such packets, they must be dropped.

This stateful approach is limited to use with SFs that retain the transport coordinates of the packet. This approach cannot be used with SFs that modify those coordinates (e.g., NATs) or otherwise create packets for new coordinates other than those received (e.g., as an HTTP cache might do to retrieve content on behalf of the original flow). In both cases, the fundamental problem is the inability to forward packets when state cannot be found for the packet transport-layer coordinates.

In the stateful approach, there are issues caused by having state, such as how long the state should be maintained (it must time out eventually), as well as whether the state needs to be replicated to other devices to create a highly available network.

It is valid to consider the state to be disposable after failure, since it can be re-created by each new packet arriving from the higher-level domain. For example, if an IBN loses all flow state, the state is re-created by an end-point retransmitting a TCP packet.

If an SFC domain handles multiple network regions (e.g., multiple private networks), the coordinates may be augmented with additional parameters, perhaps using some metadata to identify the network region.
In this stateful approach, it is not necessary for the sub-domain’s control-plane to modify paths when higher-level paths are changed. The complexity of the higher-level domain does not cause complexity in the lower-level domain.

Since it doesn’t depend on NSH in the lower domain, this flow-stateful approach can be applied to translation methods of converting NSH to other forwarding techniques. (Refer to Section 6.)

3.1.2. Encoding Upper-Level Paths in Metadata

An IBN can push the upper-level Service Path Identifier (SPI) and Service Index (SI) (or encoding thereof) into a metadata field of the lower-level encapsulation (e.g., placing upper-level path information into a metadata field of NSH). When packets exit the lower-level path, the upper-level SPI and SI can be restored from the metadata retrieved from the packet.

This approach requires the SFs in the path to be capable of forwarding the metadata and appropriately attaching metadata to any packets injected for a flow.

Using new metadata may inflate packet size when variable-length metadata (type 2 from NSH [I-D.ietf-sfc-nsh]) is used.

It is conceivable that the MD-type 1 Mandatory Context Header fields of NSH [I-D.ietf-sfc-nsh] are not all relevant to the lower-level domain. In this case, one of the metadata slots of the Mandatory Context Header could be repurposed within the lower-level domain, and restored when leaving.

In this metadata approach, it is not necessary for the sub-domain’s control element to modify paths when higher-level paths are changed. The complexity of the higher-level domain does not cause complexity in the lower-level domain.

3.1.3. Using Unique Paths per Upper-Level Path

In this approach, paths within the sub-domain are constrained so that a SPI (of the sub-domain) unambiguously indicates the egress SPI and SI (of the upper domain). This allows the original path information to be restored at sub-domain egress from a look-up table using the sub-domain SPI.

Whenever the upper-level domain provisions a path via the lower-level domain, the lower-level domain controller must provision corresponding paths to traverse the lower-level domain.
A down-side of this approach is that the number of paths in the lower-level domain is multiplied by the number of paths in the higher-level domain that traverse the lower-level domain. I.e., a sub-path must be created for each combination of upper SPI/SI and lower chain.

3.1.4. Nesting Upper-Level NSH within Lower-Level NSH

In this approach, when packets arrive at the IBN in the top-level domain, the classifier in the IBN determines the path for the lower-level domain and pushes the new NSH header in front of the original NSH header.

As shown in Figure 3 the Lower-NSH Header used to forward packets in the lower-level domain precedes the Upper-NSH Header from the top-level domain.

```
+------------------+
| Overlay Header   |
+------------------+
| Lower-NSH Header |
+------------------+
| Upper-NSH Header |
+------------------+
| Original Packet  |
+------------------+
```

Figure 3: Encapsulation of NSH within NSH

The traffic with the above stack of two-layer-NSH header is to be forwarded according to the Lower-NSH header in the lower-level SFC domain. The Upper-NSH header is preserved in the packets but not used for forwarding. At the last SFF of the chain of the lower-level domain (which resides in the IBN), the Lower-NSH header is removed from the packet, and then the packet is forwarded by the IBN to an SFF of the upper-level domain, which will be forwarded according to the Upper-NSH header.

With such encapsulation, Upper-NSH information is carried along the extent of the lower-level chain without modification.

A benefit of this approach is that it does not require state in the IBN or configuration to encode fields in meta-data.

However, the down-side is it does require SFs in the lower-level domain to be able to parse multiple layers of NSH. If the SF injects
packets, it must also be able to deal with adding appropriate multiple layers of headers to injected packets.

### 3.1.5. Stateful / Metadata Hybrid

The basic idea of this approach is for the IBN to save upper domain encapsulation information such that it can be retrieved by a unique identifier, termed an "hSFC Flow ID". An example ID is shown in Table 1.

<table>
<thead>
<tr>
<th>hSFC Flow ID</th>
<th>SPI</th>
<th>SI</th>
<th>Context1</th>
<th>Context2</th>
<th>Context3</th>
<th>Context4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>254</td>
<td>100</td>
<td>2112</td>
<td>12345</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 1: Example Mapping of an hSFC Flow ID to Upper-Level Header

The ID is placed in the metadata in NSH headers of the packet in the lower domain, as shown in Figure 4. When packets exit the lower domain, the IBN uses the ID to retrieve the appropriate NSH encapsulation for returning the packet to the upper domain.

```
Ver 0 | C | R | R | R | R | R | R | R | Length | MD-type=0x1 | Next Protocol
+----------------------------------|
Service Path Identifier | Service Index|
+----------------------------------|
hSFC Flow ID
+----------------------------------|
Mandatory Context Header
+----------------------------------|
Mandatory Context Header
+----------------------------------|
Mandatory Context Header
```

Figure 4: Storing hSFC Flow ID in lower-level metadata

Advantages of this approach include:

- Does not require state based on 5-tuple, so it works with functions that change the IP addresses or ports of a packet such as NATs,
o Does not require all domains to have the same metadata scheme,

o Can be used to restore any upper-domain information, not just service path,

o The lower domain only requires a single item of metadata regardless of the number of items of metadata used in the upper domain. (For MD-Type 1, this leaves 3 slots for use in the lower domain.)

o No special functionality is required of the SF, other than the usual ability to preserve metadata and to apply metadata to injected packets.

Disadvantages include those of other stateful approaches, including state timeout and replication mentioned in Section 3.1.1.

There may be a large number of unique NSH encapsulations to be stored, given that the hSFC Flow ID must represent all of the bits in the upper-level encapsulation. This might consume a lot of memory or create out-of-memory situations in which IDs cannot be created or old IDs are discarded while still in use.

3.2. Gluing Levels Together

The SPI or metadata on a packet received by the IBN may be used as input to reclassification and path selection within the lower-level domain.

In some cases the meanings of the various path IDs and metadata must be coordinated between domains.

One approach is to use well-known identifier values in metadata, communicated by some organizational registry.

Another approach is to use well-known labels for chain identifiers or metadata, as an indirecton to the actual identifiers. The actual identifiers can be assigned by control-plane systems. For example, a sub-domain classifier could have a policy, "if pathID=classA then chain packet to path 1234"; the higher-level controller would be expected to configure the concrete higher-level pathID for classA.

3.3. Decrementing Service Index

Because the IBN acts as a Service Function to the higher-level domain, it must decrement the Service Index in the NSH headers of the higher-level path.
4. Sub-domain Classifier

Within the sub-domain (referring to Figure 2), after the IBN removes higher-level encapsulation from incoming packets, it sends the packets to the classifier, which selects the encapsulation for the packet within the sub-domain.

One of the goals of the hierarchical approach is to make it easy to have transport-flow-aware service chaining with bidirectional paths. For example, it is desired that for each TCP flow, the client-to-server packets traverse the same SFs as the server-to-client packets, but in the opposite sequence. We call this bidirectional symmetry. If bidirectional symmetry is required, it is the responsibility of the control-plane to be aware of symmetric paths and configure the classifier to chain the traffic in a symmetric manner.

Another goal of the hierarchical approach is to simplify the mechanisms of scaling in and scaling out service functions. All of the complexities of load-balancing among multiple SFs can be handled within a sub-domain, under control of the classifier, allowing the higher-level domain to be oblivious to the existence of multiple SF instances.

Considering the requirements of bidirectional symmetry and load-balancing, it is useful to have all packets entering a sub-domain to be received by the same classifier or a coordinated cluster of classifiers. There are both stateful and stateless approaches to ensuring bidirectional symmetry.

5. Control Plane Elements

Although control protocols have not yet been standardized, from the point of view of hierarchical service function chaining we have these expectations:

- Each control-plane instance manages a single level of hierarchy of a single domain.
- Each control-plane is agnostic about other levels of hierarchy. This aspect allows humans to reason about the system within a single domain and allows control-plane algorithms to use only domain-local inputs. Top-level control does not need visibility to sub-domain policies, nor does sub-domain control need visibility to higher-level policies.
Sub-domain control-planes are agnostic about control-planes of other sub-domains. This allows both humans and machines to manipulate sub-domain policy without considering policies of other domains.

Recall that the IBN acts as an SF in the higher-level domain (receiving SF instructions from the higher-level control-plane) and as a classifier in the lower-level domain (receiving classification rules from the sub-domain control-plane). In this view, it is the IBN that glues the layers together.

The above expectations are not intended to prohibit network-wide control. A control hierarchy can be envisaged to distribute information and instructions to multiple domains and sub-domains. Control hierarchy is outside the scope of this document.

6. Extension for Adopting to NSH-Unaware Service Functions

The hierarchical approach can be used for dividing networks into NSH-aware and NSH-unaware domains by converting NSH encapsulation to other forwarding techniques (e.g., 5-tuple-based routing with OpenFlow), as shown in Figure 5.
SF#1 and SF#5 are NSH-aware and SF#2, SF#3 and SF#4 are NSH-unaware. In the NSH-unaware domain, packets are conveyed in a format supported by SFs which are deployed there.

Figure 5: Dividing NSH-aware and NSH-unaware domains

6.1. Purpose

This approach is expected to facilitate service chaining in networks in which NSH-aware and NSH-unaware SFs coexist. Some examples of such situations are:

- In a period of transition from legacy SFs to NSH-aware SFs and
- Supporting multi-tenancy.
6.2. Requirements for IBN

In this usage, an IBN classifier is required to have an NSH conversion table for applying packets to appropriate lower-level paths and returning packets to the correct higher-level paths. For example, the following methods would be used for saving/restoring upper-level path information:

- Saving SPI and SI in transport-layer flow state (refer to Section 3.1.1) and
- Using unique lower-level paths per upper-level NSH coordinates (refer to Section 3.1.3).

Especially, the use of unique paths approach would be good for translating NSH to a different forwarding technique in the lower level. A single path in the upper level may be branched to multiple paths in the lower level such that any lower-level path is only used by one upper-level path. This allows unambiguous restoration to the upper-level path.

In addition, an IBN might be required to convert metadata contained in NSH to the format appropriate to the packet in the lower-level path. For example, some legacy SFs identify subscriber based on information of network topology, such as VID, and IBN would be required to create VLAN to packets from metadata if subscriber identifier is conveyed as metadata in higher-level domains.

Other fundamental functions required as IBN (e.g., maintaining metadata of upper level or decrementing Service Index) are same as normal usage.

7. Acknowledgements

The concept of Hierarchical Service Path Domains was introduced in [I-D.homma-sfc-forwarding-methods-analysis] as a means to improve scalability of service chaining in large networks.

The concept of nested NSH headers was introduced in [I-D.ao-sfc-for-dc-interconnect] as a means of creating hierarchical SFC in a data center.

The authors would like to thank the following individuals for providing valuable feedback:

Ron Parker
Christian Jacquenet
8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

Hierarchical service function chaining makes use of service chaining architecture, and hence inherits the security considerations described in the architecture document.

Furthermore, hierarchical service function chaining inherits security considerations of the data-plane protocols (e.g., NSH) and control-plane protocols used to realize the solution.

The systems described in this document bear responsibility for forwarding internet traffic. In some cases the systems are responsible for maintaining separation of traffic in private networks.

This document describes systems within different domains of administration that must have consistent configurations in order to properly forward traffic and to maintain private network separation. Any protocol designed to distribute the configurations must be secure from tampering.

All of the systems and protocols must be secure from modification by untrusted agents.

Security considerations related to the control plane are discussed in [I-D.ietf-sfc-control-plane].

10. References

10.1. Normative References

[I-D.ietf-sfc-control-plane]

[I-D.ietf-sfc-nsh]
The advantage of hierarchical service function chaining compared with normal or flat service function chaining is that it can reduce the management complexity significantly. This section discusses examples that show those advantages.

A.1. Reducing the Number of Service Function Paths

In this case, hierarchical service function chaining is used to simplify service function chaining management by reducing the number of Service Function Paths.

As shown in Figure 6, there are two domains, each with different concerns: a Security Domain that selects Service Functions based on network conditions and an Optimization Domain that selects Service Functions based on traffic protocol.

In this example there are five security functions deployed in the Security Domain. The Security Domain operator wants to enforce the five different security policies, and the Optimization Domain operator wants to apply different optimizations (either cache or video optimization) to each of these two types of traffic. If we use
flat SFC (normal branching), 10 SFPs are needed in each domain. In contrast, if we use hierarchical SFC, only 5 SFPs in Security Domain and 2 SFPs in Optimization Domain will be required, as shown in Figure 7.

In the flat model, the number of SFPs is the product of the number of functions in all of the domains. In the hSFC model, the number of SFPs is the sum of the number of functions. For example, adding a "bypass" path in the Optimization Domain would cause the flat model to require 15 paths (5 more), but cause the hSFC model to require one more path in the Optimization Domain.


Figure 6: Flat SFC (normal branching)
Figure 7: Simplified path management with Hierarchical SFC

A.2. Managing a Distributed Data-Center Network

Hierarchical service function chaining can be used to simplify inter-data-center SFC management. In the example of Figure 8, shown below, there is a central data center (Central DC) and multiple local data centers (Local DC#1, #2, #3) that are deployed in a geographically distributed manner. All of the data centers are under a single administrative domain.

The central DC may have some service functions that the local DC needs, such that the local DC needs to chain traffic via the central DC. This could be because:

- Some service functions are deployed as dedicated hardware appliances, and there is a desire to lower the cost (both CAPEX and OPEX) of deploying such service functions in all data centers.

- Some service functions are being trialed, introduced or otherwise handle a relatively small amount of traffic. It may be cheaper to manage these service functions in a single central data center and steer packets to the central data center than to manage these service functions in all data centers.
For large data center operators, one local DC may have tens of thousands of servers and hundreds of thousands of virtual machines. SFC can be used to manage user traffic. For example, SFC can be used to classify user traffic based on service type, DDoS state etc.

In such large scale data center, using flat SFC is very complex, requiring a super-controller to configure all data centers. For example, any changes to Service Functions or Service Function Paths in the central DC (e.g., deploying a new SF) would require updates to all of the Service Function Paths in the local DCs accordingly. Furthermore, requirements for symmetric paths add additional complexity when flat SFC is used in this scenario.

Conversely, if using hierarchical SFC, each data center can be managed independently to significantly reduce management complexity. Service Function Paths between data centers can represent abstract notions without regard to details within data centers. Independent controllers can be used for the top level (getting packets to pass the correct data centers) and local levels (getting packets to specific SF instances).
Authors’ Addresses

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

Phone: +1 519 880 2400
Email: ddolson@sandvine.com

Shunsuke Homma
NTT, Corp.
3-9-11, Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Email: homma.shunsuke@lab.ntt.co.jp

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

Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Dapeng Liu
Alibaba Group
Beijing 100022
China

Email: max.ldp@alibaba-inc.com
Ting Ao  
ZTE Corporation  
No.889,Bibo Rd.,Zhangjiang Hi-tech Park  
Shanghai 201203  
China  
Phone: +86-21-688976442  
Email: ao.ting@zte.com.cn

Vu Anh Vu  
Soongsil University  
369 Sangdo-ro  
Seoul, Dongjak-gu 06978  
Korea  
Email: vuva@dcn.ssu.ac.kr
Analysis on Forwarding Methods for Service Chaining
draft-homma-sfc-forwarding-methods-analysis-05

Abstract

This document presents the results of analyzing packet forwarding methods and path selection patterns for achieving Service Chaining. In Service Chaining, data packets need to be forwarded to the appropriate service functions deployed in networks based on service provided for the packets, and distribution of the service-oriented route information and steering data packets following the route information would be required.

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 August 1, 2016.
Table of Contents

1. Introduction ............................................. 3
2. Definition of Terms ........................................ 4
3. Classification of Forwarding Methods and SP Selection
   Patterns .................................................. 5
   3.1. Forwarding Methods .................................... 5
       3.1.1. Method 1: Forwarding Based on Flow Identifiable
              Information ..................................... 6
       3.1.2. Method 2: Forwarding with Stacked Headers ......... 7
       3.1.3. Method 3: Forwarding Based on Service Chain
              Identifiers ....................................... 9
   3.2. Service Path Selection Patterns ......................... 12
       3.2.1. Pattern 1: Static Selection of End-to-End Service
              Path ........................................... 13
       3.2.2. Pattern 2: Dynamic Selection of Segmented Service
              Path ........................................... 16
4. Consideration on Forwarding Methods and Paths Selection
   Patterns .................................................. 21
   4.1. Analysis of Forwarding Methods .......................... 22
       4.1.1. Analysis of Method 1: Using Flow Identifiable
              Information ..................................... 22
       4.1.2. Analysis of Method 2: Stacking Headers ............ 23
       4.1.3. Analysis of Method 3: Using Service Chain Identifier
              .............................................. 24
   4.2. Analysis of Service Path Selection Patterns ............... 27
       4.2.1. Analysis of Pattern 1: Static SP Selection ....... 27
       4.2.2. Analysis of Pattern 2: Dynamic SP Selection ....... 28
   4.3. Example of selecting Methods and Patterns ................. 32
       4.3.1. Example#1: Enterprise Datacenter Network .......... 32
       4.3.2. Example#2: Current Mobile Service Providers Network.
              .............................................. 32
       4.3.3. Example#3: Fixed and Mobile Converged Service
              Providers Network .................................. 34
5. Acknowledgements ........................................... 35
Introduction

Some IETF working groups of and other Standards Developing Organizations are now discussing use cases of a technology that provides service-oriented traffic forwarding schemes to convey packets to the various service functions, deployed in networks, for providing network services. In this document, we define such technology as Service Chaining. (This draft does not focus only on "Service Function Chaining (SFC)" architecture, and thus, use the term "Service Chaining." SFC is one of approaches to realize Service Chaining.) There are several methods to achieve Service Chaining, and the applicable method will vary depending on the service requirements of individual networks.

This draft assumes that Service Chaining is achieved by the following steps:

a. A traffic classification function identifies the service that is associated to each incoming packets by inspecting the key information such as IP address or 5-tuple.

b. The forwarding path used by packets for reaching the appropriate service functions, is established according to the services provided for the packets. The path might be established in advance.

c. Forwarding functions forward the packets to the next destination along the path established in step b.

d. A service function operates on received packets. Once the invocation of a service function is completed, the packet is forwarded to the next forwarding function.

e. Steps c and d are repeated until each packet has been transferred to all required service functions.

f. After a packet has been transferred to all required Service Functions, it is forwarded to its original destination.

There are several forwarding methods for Service Chaining, and they can be classified into certain categories in terms of distribution of information for setting the paths and decision of the paths. The methods used to distribute the information for path setting and the
patterns used to decide the paths will affect the mechanism of Service Chaining in terms of scalability and service flexibility.

The applicable methods vary depending on network requirements, and thus, classifying and determining forwarding methods will be important in designing the architecture of Service Function Chaining (SFC). This document provides the results of analysis of different forwarding methods for Service Chaining.

OAM, security, and redundancy are outside the scope of this draft.

2. Definition of Terms

Term "Classification", "Classifier" referred to [RFC7665]. Term "Service Function", "Service Node" referred to [I-D.ietf-sfc-dc-use-cases].

Service Chaining: A technology that enables data packets to invoke a set of service functions.

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 (CF): An element that performs classification.

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). A Service Function can be a virtual element or be embedded in a physical network element. One of multiple Service Functions can be embedded in the same network element. Multiple occurrences of the Service Function can be enabled 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 Service Functions includes: firewalls, WAN and application acceleration, Deep Packet Inspection (DPI), LI (Lawful Intercept) module, server load balancers, NAT44 [RFC3022], NAT64 [RFC6146], NPTv6 [RFC6296], HOST_ID injection, HTTP Header Enrichment functions, TCP optimizer, etc.

Forwarder (FWD): The entity, responsible for forwarding data packets according to the ordered set of service functions that need to be invoked. A forwarder maintains one or more forwarding tables,
which contain entries that asset the forwarder in its forwarding
decision-making process.

Control Entity (CE): One or a set of control entities responsible
for managing service topology and indicating forwarding
configurations to forwarders.

Service Chain (SC): A service chain defines an ordered list of
service functions that must be applied to packets selected as a
result of classification. The implied order may not be a linear
progression as the architecture allows for nodes that copy to more
than one branch.

Service Path (SP): The forwarding path followed by packets that are
associated to a given service chain. Packets follow a service
path through the requisite service functions that need to be
invoked, as per the service chain instructions. Service path
shows a specific path that traverses several service function
instances. For example, SC is written as SF#1 -> SF#2 -> SF#3
(This shows an ordered list of SFs), and SP is written as
SF#1_1(1_1 means instance 1 of SF1) -> SF#2_1 -> SF#3_1.

Segmented Service Path: A Segmented Service Path is an actual path
established between FWDs. A service path might be composed of
some segmented service paths.

Service Chaining Domain (SC Domain): The domain managed by one or a
set of CEs.

Service Path Information (SP Information): The information used to
forward packets to the appropriate SFs according to the service
that needs to be provided. Examples of SP information include
routing configuration for forwarders, headers for forwarding
packets to required SFs, and service/flow identifiable tags.

3. Classification of Forwarding Methods and SP Selection Patterns

3.1. Forwarding Methods

In Service Chaining, data packets are transferred to service
functions, which might be located outside the regular computed path
to the original destination. Therefore, a routing mechanism that is
different from general L2/L3 switching/forwarding might be required.
The forwarding mechanism can be classified into three methods in
terms of distribution of SP information and packet forwarding.
3.1.1. Method 1: Forwarding Based on Flow Identifiable Information

The mechanism of method 1 is shown in Figure 1. In this method, forwarding configuration information is based on flow identifiable information, such as 5-tuple (e.g. dst IP, src IP, dst port, src port, tcp) are indicated to the CF and each FWD. There might be an CE to handle this. The flow identifiable information can be constructed with some fields of L2, L3 or L4 or combination thereof. The information can be configured either before packets arrive, or at the time packets arrive at CF and FWD. Each FWD identifies the packets with flow identifiable information and forwards the packets to the SFs according to the configuration. This method does not require the modification of any field in the original packet header.
3.1.2. Method 2: Forwarding with Stacked Headers

The mechanism of method 2 is shown in Figure 2. In this method, the CF classifies packets and stacks headers in which actual network address is included, e.g., MPLS, GRE headers or IPv6 Segment Routes, onto the packets based on the classification. The packet is transferred to the destination according to the outermost header, and a SF or FWD, as the destination, removes the outermost header after receiving the packet. The processes are repeated until all stacked...
headers are removed. This method does not require any forwarding entries for forwarding packets based on the service information.

*Distribution model of SP information*

```
+----------------+
| Control Entity |
+----------------+
     ^
     v
-----+-----+-----+-----+
   CF | SF#1 | SF#2 | SF#3 |
-----+-----+-----+-----+
```

*Forwarding Tables*

Locate: [CF]

Table: 192.168.1.1

- Stack #1,2,3
- Stack #1,3

...  

*Condition of Packet*

Locate: [CF] [SF#1] [SF#2] [SF#3]

Header:  

Packet:  

Figure 2: Forwarding with Stacked Multiple Headers
3.1.3. Method 3: Forwarding Based on Service Chain Identifiers

The mechanism of this method is shown in Figure 3. In this method, the corresponding service chain identifier is mapped to each packet by a CF based on the classification. The forwarding configuration based on the identifiers is sent to each FWD. Each FWD identifies the SP assigned to the received packet from the identifier, and forwards the packet to the next hop. After a packet has traversed all SFs, the identifier is removed and the packet is transported to the original destination.
*Distribution model of SP information*

```
+----------------+  +----------------+  +----------------+  +----------------+
| Control Entity |  | indication of attached tag |  | and routing configuration based on tags |
+----------------+  +----------------+  +----------------+  +----------------+
     ^                      ^                      ^                      ^
     v                      v                      v                      v
-----| CF |-----| FWD |-----| SF#1 |-----| FWD |-----|
       |     |     |     |     |     |     |     |
```

*Forwarding Tables*

Locate:  [CF]          [FWD]                          [FWD]

Table:  192.168.1.1        IF ID#1,3                   IF ID#1,2,5
       ->Stack ID#1       ->SF#1                       ->SF#2
       10.0.1.1           10.0.1.1                     10.0.1.1
       ->Stack ID#2       ...                         ...
       ...                ...                         ...

*Condition of Packet*

Locate:  [CF]          [FWD]                          [SF#1]           [FWD]

SC-ID:  +-------+        +-------+      +-------+       +-------+
Packet: | ID#1  |        | ID#1  |      | ID#1  |       | ID#1  |
         +-------+        +-------+      +-------+       +-------+
         +-------+        +-------+      +-------+       +-------+

Figure 3: Forwarding Based on Service Chain Identifiers

Then, there are mainly three approaches to map service chain identifiers to packets as follows.

- **Tagging an extra header:**

  In this approach, an extra header which has a service chain identifier is attached on each packet. This document defines such headers as service identifiable tags. Some existing tags, such as
VLAN-tag or MPLS-tag, or dedicated headers, such as NSH, could be used as service identifiable tags. As an example, SFC[RFC7665] is categorized into this approach. An example of packet format in tagging approach with NSH is shown in Figure 4. In this example, a service chain identifier is included in NSH.

```
+----------+-------+--------+-------------˜˜--+
|   NSH    | Ether | IPv6   |IPv6 Payload |
| \SC-ID   | Header| Header |             |
+----------+-------+--------+-------------˜˜--+
|---      Ethernet Packet       ---|
```

Figure 4: Packet Format in Tagging Approach

- Inserting into an optional field:

In this approach, a service chain identifier is inserted into an optional field inside a packet frame, such as IPv6 extension header. An example of an IPv6 packet with a service chain identifier inserted as an extension header is shown in Figure 5.

```
+-------+--------+----------+-------------˜˜--+
| Ether |IPv6    |IP Opt.Fld|IPv6 Payload |
| Header|Base Hdr| \SC-ID   |             |
+-------+--------+----------+-------------˜˜--+
|--  IPv6 Header  --|
|---         Ethernet Packet               ---|
```

Figure 5: Packet Format in Inserting Approach

- Overloading on a destination or source address:

In this approach, service chain identifier is overloaded on a destination or a source address such as MAC or IP address. In other words, the addresses are used for both showing the destination or source in network and identifying service chain which each packet belongs to. An address is required for each hop in a service chain, and FWDs switch the address to new one for the next hop by referring the address of the received packet. An
example of using destination address overloading is shown in Figure 6. In this example, SFs are used as L2 transparent mode, and service chain identifiers are overloaded on destination MAC addresses. FWD2 refers the destination MAC address which shows the address for Port B, and changes it to the address for Port D for sending the packet to the next hop in the service chain. When using non-transparent SFs in the overloading approach, the identifier is carried from the FWD to the SF in the source address (SA) and is carried from the SF to the FWD in the destination address (DA). More detailed processes of the overloading approach using MAC addresses is described in Ethernet MAC Chaining [I-D.fedyk-sfc-mac-chain].

*Network Structure*

```
<table>
<thead>
<tr>
<th>Port A</th>
<th>Port B</th>
<th>Port C</th>
<th>Port D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+[SF1]+</td>
<td></td>
<td>+[SF2]+</td>
</tr>
<tr>
<td>/</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>--# FWD1 #--</td>
<td>++# FWD2 #--</td>
<td>++# FWD3 #--</td>
<td></td>
</tr>
</tbody>
</table>
| \_____/  ^   \_____/  ^   \_____/  ^   \_____/  ^   \_____/  ^   \_____/  ^

*Packet Frame*

```

```
+--------+-------˜-+        +--------+-------˜-+
|MAC-DA:B| Payload |        |MAC-DA:D| Payload |
+--------+-------˜-+        +--------+-------˜-+

---------- Flow Direction ---------->
```

Figure 6: Overview of DA Overloading Approach

3.2. Service Path Selection Patterns

Since SC contains only logical information (e.g., a set of services that are associated with flows and their sequences), the actual instances, which are called SPs, are needed in order for the forwarding process to work. In this process, an instance of SP is created at certain points during a packet’s delivery. Therefore, to forward packets, the SC needs to be turned into an SP, which indicates specific FWDs (or switches, routers) and SFs that the
packets will be forwarded to. From the perspective of points translating SC to SP, the methods that establish SPs from end-to-end are classified into two patterns.

3.2.1. Pattern 1: Static Selection of End-to-End Service Path

The translation point is a CF; that is, the SP is statically pre-established as an end-to-end path and a CF forwards packets along the appropriate path based on the result of the classification. Each FWD on the SP has a forwarding table to uniquely determine the next destination of packets, and each FWD statically forwards the received packets toward the next destination based on the table. FWD requires only a function to receive indications of forwarding configurations from the CE. Pattern 1 can be achieved in the following models.

3.2.1.1. SF Dedicated Model: Network Slicing Model

In this model, an SF instance (or a set of SF instances) is used by only one single SP; in other words, a set of SF instances is prepared for each SP. This model also enables operators to establish SPs without any FWDs by slicing network physically or virtually and deploying a set of SFs required for service providing in each sliced network. A CF assigns packets to the network in which the appropriate SF set is installed inline, and the packets traverse the SFs by being forwarded along the pre-configured route. The overview of network slicing model is shown in Figure 7.
**Path Structure**

```
* network#1                                 *
+----+     *      +------+     +------+     +------+   *
|    |SC#1 *      | SF#1 |     | SF#2 |     | SF#3 |   * SP#1
|    |=======================================================>
|    |     *      +------+     +------+     +------+   *
|    |     * * * * * * * * * * * * * * * * * * * * * * *
|    |     * network#2                                 *
|    |     +--------+                                 +-*
|    |SC#2 *      | SF#4 |     | SF#5 |     | SF#6 |   * SP#2
|    |=======================================================>
|    |     *      +------+     +------+     +------+   *
|    |     * * * * * * * * * * * * * * * * * * * * * * *
|    |     * CF *      +------+     +------+     +------+   *
|    |     * * * * * * * * * * * * * * * * * * * * * * *
|    |     * network#n                                 *
|    |     +--------+                                 +-*
|    |SC#n *      | SF#6 |     | SF#7 |     | SF#8 |   * SP#n
|    |=======================================================>
|    |     *      +------+     +------+     +------+   *
|    |     * * * * * * * * * * * * * * * * * * * * * * *
```

SC: Service Chain  SP: Service Path

/////////////////////////////////////////////////////////////////////

*How packets traverse*

Service Chain#1:
SF#1
```
[ CF ]----------->[ SF#1 ]----->[ SF#2 ]----->[ SF#3 ]------->
```

Service Chain#2:
SF#2
```
[ CF ]----------->[ SF#4 ]------------------->[ SF#5 ]------->
```

Service Chain#n:
SF#n
```
[ CF ]----------->[ SF#6 ]----->[ SF#7 ]----->[ SF#8 ]------->
```

Figure 7: SF Dedicated Model

Homma, et al. Expires August 1, 2016 [Page 14]
3.2.1.2. SF Shared Model

In this model, an SF is shared by multiple SPs. Several SPs are mixed at shared SFs, and thus this method requires FWDs for forwarding each packet to the corresponding next hop by identifying the SP which each packet belongs to. The overview of the SF shared model is shown Figure 8.

*Path Structure*

```
+----+     +---+   +----+   +---+   +------+
|    |SC#1 |FWD|   |SF#1|   |FWD|   |SF#3| SP#1
+----+     +---+   +----+   +---+   +------+
     |SC#2 | #   | #   | #   | SP#2
+----+   #   +----+   #   +----+
     | FWD| # |SF#2_1| # |SF#3| SP#1
     | #   +---+   #   +---+   +---+
     |    |SC#n |FWD|   |SF#4| |FWD| |SF#5| SP#n
+----+   +----+   +----+   +----+
```

SC: Service Chain  SP: Service Path

Figure 8: SF Shared Model
3.2.2. Pattern 2: Dynamic Selection of Segmented Service Path

The mechanism of this pattern is shown in Figure 9. The translation points are CFs and some FWDs. The SP is established by a series of multiple paths, which are sectioned by CFs and FWDs. The resulting path is referred to as a segmented path in this draft. CFs or FWDs that select the next segmented path might require notification of forwarding configuration information from the CE. Moreover, some FWDs require functions to select the destination of packets from various alternatives and to retrieve the information for selecting the next path. For example, each FWD obtains metric information or load conditions of servers and selects an optimal segmented path based on the information. The CE might support the selection mechanism and may notify CFs or FWDs of it.
*Path Structure*

```
+----+     +---+   +----+   +---+   +------+   +---+   +----+
|    |SC#1 |FWD|   |SF#1|   |FWD|   |SF#2_1|   |FWD|   |SF#3| SP#1
|    |========================*=====================================>
|    |     |   |   |    |   |   +------+   |   |   |    | SP#2
|    |     |   |   |    |   | #======================>
|    |     |   |            | #==============# |   |
|    |     |   |            | #==============# |   |
|    +----+   | # |   |SF#2_2| # |   |   +----+
|    |         |   |            | #==============# |
|    |         +---+            +---+   +------+   +---+
|    | CF |     +---+            +---+   +------+   +---+
|    | .   |     +---+            +---+   +------+   +---+
|    | .   |     +---+            +---+   +------+   +---+
|    | .   |     +---+            +---+   +------+   +---+
|    | SC#n |FWD|   |SF#4|                      |FWD|   |SF#5| SP#m
|    |........................................................................>
|    +----+     +---+   +----+                      +---+   +----+
```

SC: Service Chain  SP: Service Path

```
///////////////////////////////////////////////////////////////////////
```

*How packets traverse*

Service Chain#1:

SP1

```
[ CF ]---->[FWD]-->[SF#1]-->[FWD]-->[SF#2_1]-->[FWD]-->[SF#3]--->
```

SP2

```
[ CF ]---->[FWD]-->[SF#1]-->[FWD]-->[SF#2_2]-->[FWD]-->[SF#3]--->
```

Service Chain#n:

SPm

```
[ CF ]---->[FWD]-->[SF#4]---------------------->[FWD]-->[SF#5]--->
```

Figure 9: Dynamic Selection of Segmented Service Path

In addition, this pattern supports the establishment of hierarchical domains discussed below:

3.2.2.1. Hierarchical Service Path Domains

Complex problems often become manageable with a hierarchical approach. This pattern allows network-wide orchestration of Service Chaining to be relatively simple, while hiding the complexities of fine-grained policy-based path selection within sub-domains. Each
sub-domain can be independently administered and orchestrated. This architecture is described in [I-D.dolson-sfc-hierarchical].

Figure 10 shows two levels of hierarchy in a service provider’s network. At the top level in the hierarchy, Service Chaining components are:

1. Edge-classifiers (Edge CF) that reside near the edge of a service provider’s domain.

2. SF sub-domains that reside in data centers.

3. Internal Boundary Nodes (IBNs) that reside in data centers, linking together the levels of the hierarchy. To the higher level, sub-domains are viewed as a SF. To the lower level, this is a classifier and FWD.
*How packets traverse*

```
|       | SC#1   | FWD     | IBN#1               | FWD     |
+-------+--------+---------+---------------------+---------+
|       |        |         |                     |         |
|       |        |         |                     |         |
|       |        |         |                     |         |
|       |        |         |                     |         |
|       |        |         |                     |         |
| Details of sub-domain #q not shown |
```

Figure 10: Service Chain Hierarchy in Service Provider Networks

The components within an SF sub-domain are opaque at the top level; each IBN acts as a single SF node in the top-level domain. A service path in the top-level domain may visit multiple sub-domains.

At the lower level in the hierarchy, each sub-domain contains an independently administrated Service Chaining network, generally comprised of multiple instances of multiple types of hosts, most likely (but not necessarily) within the same data center. There is no need for knowledge of the "big picture" at the level of the SF-sub-domain except as required to forward packets to the other SFs that are the next hop of each chain.

Note that different encapsulation methods can be used at each layer in the hierarchy, provided the SF domain-Proxy can translate between them. For example, MPLS could be used to deliver packets from
network edge to the SF clusters within data centers, and NSH [I-D.ietf-sfc-nsh] could be used within the data center.

Details of Top Level of Hierarchy

In this pattern, referring to Figure 11, network-wide Service Chaining orchestration is only concerned with creating service paths from network edge points to sub-domains within data centers and configuring classifiers at a coarse level to get the correct hosts’ traffic onto paths that will arrive at appropriate sub-domains. The figure shows one possible service chain passing from edge, through two sub-domains, to network egress.

This top level of orchestration may attach metadata to provide context from the network edge into the data center.

```
+-------------+
| Sub-domain#1 |
| in DC1       |
+-------------+

---+-----+---
| CF | SC#1 |
|    |      |
+-----+----+

---+---+---+---
| CF | V   | CF |
|    |   /  |
+-----+-----+

Figure 11: Network-wide view of Top Level of Hierarchy
```

The orchestration at this top level must ensure bidirectional path symmetry so that inbound packets traverse sub-domains in the reverse order as outbound packets.

Because classifiers must have rules to handle any traffic passing through the network, we believe that a useful approach to classification will be to assign traffic to service function paths on the basis of coarse classification like subscriber tier, tenant or...
VRF identifier. These classification rules could be relatively static, changing in response to provisioning but not in response to traffic.

In some networks, it might be possible to create a rule per residential subscriber, resulting in rule updates when subscribers are assigned IP addresses. However, with judicious allocation of IP blocks, entire classes of subscribers could be classified with IP-prefix rules. Similarly, in a mobile network path selection could be based on the APN (Access Point Name) identifier.

Hence, there are methods of globally managing very large networks by choosing a suitable classification granularity.

Details of Lower Levels of Hierarchy

Within each SF sub-domain, there are:

1. An IBN to receive incoming data packets on any of the configured service chains and load-balance (if necessary) traffic to classifiers,

2. Classifier(s) to select internal service chain to use, potentially based on stateful flow analysis, DPI, etc.

3. Service components comprised of FWD and SF.

Local Service Chaining orchestration is concerned with providing viable paths to various functions, providing failure recovery, NFV elasticity, etc.

Classification within each sub-domain can be concerned with determining the local service paths for individual transport-layer flows based on ports, DPI and meta-data provided by the higher-level chain.

For any classifier that is transport-layer-stateful, it is most efficient for the same classifier instance to handle traffic in both directions of a bidirectional connection. State tracking may require that service function paths begin and terminates at the same node with the flow state, where the same classifier instance can be used for both directions of traffic.

4. Consideration on Forwarding Methods and Paths Selection Patterns

This chapter presents the results of analyzing the forwarding methods and architecture patterns in chapter 3.
4.1. Analysis of Forwarding Methods

4.1.1. Analysis of Method 1: Using Flow Identifiable Information

Data Plane Aspects

This method can achieve Service Chaining without changing packet format, such as attaching any header on packets, so it may not imply any overhead or be subject to MTU restrictions. Furthermore, this method does not require additional functions for SFs to apply or handle any header because data packets are transported unaltered. Therefore, it will be easier to use legacy SFs for network operators.

On the other hand, it is difficult to forward a packet to the same FWDs several times because flow identifiable information is not basically changed in the forwarding processes. For example, distinction of incoming ports will be required for FWD to resolve the next hop appropriately when a packet traverses it several times.

Control Plane Aspects

This method might be achieved by using existing control mechanisms. For example, openflow, which is able to provide flexible forwarding control, would be available for creating SPs.

However, this method might require FWDs to configure forwarding entries for each flow to each FWD. For example, if there are 10,000 flows to be handled at a CF/FWD, the forwarding table for each CF/FWD uses 10,000 flow entries at most. Therefore, it might not be feasible for large-scale networks such as carrier networks that handle a SC per user (which means that individual users will be associated with different policies), because some large carriers have over a million users and even more flows. In addition, control signaling would increase because forwarding configuration for each flow to each FWD is required. Moreover, it may be hard to use this method if some SFs modify header fields of a packet or frame, for example, NAT/NAPT, in a chain. For example, if a NAT changes the IP address of packets dynamically, the FWDs that follow need to renew their forwarding tables. This method also have restriction about forwarding based on high-layer information, such as application information in packet payload. The process of detecting high-layer information is usually heavy compared with L2 or L3 forwarding process, and most existing forwarding functions have capability to refer only under L4 headers. Therefore, it will be difficult to use this method to forward packets along SPs decided by detecting high-layer
information since individual L2-4 packet headers may not retain
enough information. An example of this type of problem is a video
streaming imbedded within a web page. The identifiable
information at the L2-4 level does not allow differentiation
between the video stream and the rest of the frame, and thus the
all traffic on the web page is forwarded following the same SC.

The results of the above analysis suggest that, although this method
is beneficial in terms of impact to existing network, it would not be
scalable. Therefore, this method might be suitable for networks with
a limited number of flows.

Measurements taken in multiple residential service providers’
networks indicate that for each 1Gbps of traffic the sustained rate
of new flows can range from 1,000 flows/s to 30,000 flows/s. From
this, for example, there would be between 10,000 and 300,000 new
flows/s on a 10 Gbps link. Therefore, in some networks at some times
of day, this method using 5-tuple as flow identifiable information
would require sustaining up to 300,000 table updates per second for
each FWD. This incurs a significant amount of control traffic and
computational effort.

4.1.2. Analysis of Method 2: Stacking Headers

Data Plane Aspects

In this method, SP information is attached on each packet as
headers for forwarding, and the number of the headers increases
depending on the number of SFs which the packet will traverse.
This means that the size of each packet increases. Packet sizes
may be restricted by the minimum available MTU of any link in the
network and exceeding the MTU will require to fragment the
original packets. Fragmentation adds a new source of errors and
may require forwarding processes to be more complex. For example,
the whole original packet will be discarded even if one of
fragments of the packet gets lost, or in terms of SF equipment, it
would be very wasteful of CPU if fragmented packets need to be
reassembled at every SF resources, and some equipment has
restricted resources and memory for reassembly. Fragmentation
will also cause an increase in traffic as more packets have to be
processed by the network.

Moreover, this method requires SF to be applied to the headers
because they receive packets with optional headers. Therefore SFs
will be required to be able to recognize the headers, or proxy
functions, which remove the headers before inserting packets into
SFs and re-attach the appropriate headers on the returned packet,
will be required. In addition, when a SF is used by multiple SCs,
it will be challenging for SFs to process packets because header length attached on each packet may vary and SFs are required to have a mechanism to recognize the header length for each packet.

Control Plane Aspects

In this method, none of the FWDs require any specific forwarding tables for Service Chaining or interface to receive forwarding configuration information. In short, FWDs are stateless or eliminated at hops, and this method has advantages of high scale in SPs managements and lower latency. In addition, no CEs will be required to manage the forwarding configuration of FWDs for Service Chaining, and so the control mechanism might become simple compared with other methods.

On the other hand, some relay nodes such as switches or SFs are required to have a function to remove the outermost header from the received packets. FWDs also don’t have to identify flows or services, so cannot change the following SPs. Moreover, CF must grasp all of addresses of relay nodes which packets will traverse, and it will require any CE to manage addresses of relay nodes and a link between CF and the CE. There are already several existing technologies that can be used to achieve this method, such as segment routing.

The results of the above analysis indicate that this method would be appropriate when the number of SFs in a SC is small, and most SFs are deployed in a single domain. On the other hand, it may be unsuitable in cases where there are many SFs in a chain, or packets have to traverse multiple domains.

4.1.3. Analysis of Method 3: Using Service Chain Identifier

Data Plane Aspects

The common features of this method and the individual features of each approach to map service chain identifiers in terms of data plane aspects are described below.

- Common features of method 3

  In this method, a service chain identifier, defined for each SC, is mapped into each packet. FWDs recognize the next hops of received packets from the identifiers independent of any information of original packets. Therefore, SFs which modify original packet format can also be used. In addition, it is
easy to change the following SPs on a route by renewing the identifier.

On the other hand, attachment of an identifier might expand packet size, and it would cause an increase of traffic amount or problems which happens as a result of exceeding MTU (The problems are stated in Section 4.1.2.). However, by adopting a single fixed-length identifier, the problems might be prevented. Or, when overloading the identifier on an existing field, such as MAC address, packet size is not changed and such issues would not occur.

Moreover, forwarding along SPs is provided based on service chain identifiers, and so if there are network nodes which are unaware of the identifiers, such as routers without functions to forward packets based on the identifiers, in a SC domain, some tunnel would be required for passing packets over them.

- Tagging an extra header:

  In this approach, the identifiers are prepended to packets, and so a single mapping mechanism could be used independently of the formats of the target packets.

  Conversely, this approach requires SFs to parse the extra headers (Problems which happens as result of inserting packet with optional headers into SFs are stated in Section 4.1.2). In case that an existing header, which SFs can recognize, is used as a service identifiable tag, this problem might be restricted. For example, some SFs can recognize VLAN- tags, and they would not need any improvement for the SFs if they are used as service identifiable tags. However, using an existing header might have some effects on the original uses.

- Inserting into an optional field:

  In this approach, service chain identifiers are inserted in some field of the original packets, and the packets seem normal formats from SFs. Therefore, any improvement for enabling SFs to handle the identifiers would not be required.

  Meanwhile, identifier insertion or packet forwarding mechanisms would vary depending on the formats of the original packets, because positions where identifiers are inserted are different for each packet format. For example, optional field positions of IPv4 and IPv6 headers are different. Furthermore, especially, the inserting approach, using IPv4 optional fields, might have some problems. For example, some server OS and
applications strip the IPv4 optional field due to security concerns. Therefore, it appears this is a difficult solution for IPv4 networks.

Also, in case that existing field is used for storing the identifier, amount of identifier information might be small compared with tagging an extra header approaches.

- Overloading on a destination or source address:

In this approach, a destination or source address is used for identifying service chain which the packet belongs to in addition to original usage, and so packets size increase caused by attaching additional headers does not occur. Also, any improvement for enabling SFs or any other network equipment to handle the identifiers would not be required, because the packets seem normal formats from them. In other words, this approach can coexist with legacy equipment.

Meanwhile, the addresses for Service Chaining are overwritten on the original address in this approach, and so an additional encapsulation would be required during the Service Chaining process when retaining the original address information. Therefore, for cases when L2 or L3 addresses are used for identifying subscribers, the overloading approach might require the MTU expansion for additional encapsulation. Moreover, when using L2 addresses as service chain identifier and sending packets to another L2 domain across a L3 domain, an extra means such as L3 tunnel is required.

Control Plane Aspects

The common features of this method and the individual features of the overloading approach in terms of control plane aspects are described below.

- Common features of method3

This method enables FWDs to save resources for managing forwarding tables and allows all SFs to be established in advance in most of cases. This prevents an increase of control signals such as openflow or Gx/Sd, and also enables changing the following SFs without changing forwarding configuration of FWDs.
On the other hand, this method requires a new control mechanism based on service chain identifiers, therefore, FWDs, CE and interface between them have to be updated to apply forwarding configuration based on the identifiers.

- Overloading on a destination or source address:

  Overloading approach might be achieved without new control mechanisms or drastic remodeling of existing control entities. For example, MAC chaining can be established by using programmed standard openflow switches.

  On the other hand, in the overloading approach, each SP is composed of a set of unique addresses, and thus FWDs are required to have addresses as many as service chains which pass through them.

The results of the above analysis indicate that this method has many advantages in terms of scalability, and it might be appropriate for use in large-scaled networks in which there are so many SFs and various types of flows. On the other hand, when the identifier handling mechanism is an entirely new architecture such as SFC[RFC7665], renewal or introduction of several equipment such as FWDs and CE will be required.

4.2. Analysis of Service Path Selection Patterns

4.2.1. Analysis of Pattern 1: Static SP Selection

In this pattern, the mechanism of FWDs would be simpler than the one in pattern 2 because FWDs do not require any functions to select paths or retrieve any information for next hop resolution purposes. Moreover, it is not necessary to maintain the state of each flow. Therefore, existing network virtualizing techniques, such as VxLAN or MPLS, can be used to achieve Service Chaining in this pattern. Especially, network slicing model does not require any special forwarding mechanisms.

On the other hand, this pattern has restriction in the management of SPs. When adding new SFs to a SC, removing SFs from a SP, or migrating SFs to other locations, re-establishment of SP would be required. This restriction in network slicing model would be more strict because this model need to establish a new network for adding a SP. For relaxing the restriction, it is desirable to use this pattern together with a means, such as load balancer, which enable to add the same kind SFs into a SP without changing the configuration of the SP. Or the restriction would be relaxed when network...
In addition, this pattern would also have restriction for use in wide area networks which include multiple domains. This pattern requires unified management of FWDs and SFs, in an SC domain, for setting end-to-end paths. Therefore, the management system of SPs, for example, a CE, for wide-area networks that include several segments might be massive and complex. Figure 12 shows the case in which SPs are established across multiple datacenters in pattern 1. In this case, a CE (or a set of CEs) manages multiple datacenters as a single SC domain for establishing SPs across the datacenters.

![Diagram of SP establishment across multiple DCs in Pattern 1](image)

Figure 12: Establishment of SPs across Multiples DCs in Pattern 1

4.2.2. Analysis of Pattern 2: Dynamic SP Selection

In this pattern, SPs are established with a combination of segmented paths, so it enables SPs to be established flexibly (which means, CEs do not need to constantly manage the entire end-to-end SP) based on additional information such as the SF load conditions.

Furthermore, as described in the previous section, in cases where some SPs traverse multiple datacenters across a WAN, SPs could be established with a combination of segmented paths that each datacenter determines independently based on the Service Chain information. Therefore, it might be possible to separate SC domains.
into several small areas for WANs, which would enable a simpler configuration of each CE. Figure 13 shows the case in which SPs are established across multiple datacenters in pattern 2. In Figure 13, each CE manages a single datacenter independently, and the CEs synchronize the Service Chain information for establishing and determining the appropriate segmented SPs in each domain.

However, the (fault) monitoring of the whole SC can become more difficult, as multiple domains are part of the SC. On the other hand, each domain can perform its management as required (and this is probably better as it is more specific). This will require an overarching (fault) monitoring where information from multiple SC domains is collected and aggregated to get a full view of the end-to-end service of the SC.

Moreover, in this pattern, some FWDs may require additional mechanisms to select the next segmented path, and the FWDs must maintain the states of each flow because some SFs require a stateful process, and the FWDs need to insert packets into the same SF instances in the same session.

In case that SC information is conveyed to some components via data plane as any encapsulation, a new protocol such as SFC [RFC7665] will be required.
Synchronization of Service Chain info.

Figure 13: Establishment of SPs Across Multiples DCs in pattern 2

Also, the detailed analysis of the establishment of "Hierarchical Service Path domains" is shown in the following section.

4.2.2.1. Analysis of Hierarchical Service Path domains

The dynamic selection of SPs pattern allows multiple independent domains of administration. (In the example, two levels were shown, but the pattern could be extended to multiple levels.)

This pattern allows even the largest networks to implement SC from the edges of the network by using coarse-grained classification. Classification choices can be made that are feasible within the constraints of the edge classifiers and FWDs. There is no need to maintain flow state or react to traffic at the top level.

This pattern allows control of sub-domains to be delegated to different owners. Each domain is simpler to comprehend than would be the case by dealing with a single flat network. Furthermore, failures and errors are localized (See Figure 14.).
This hierarchical model supports the management of large networks by adhering to these principles:

1. At higher levels of hierarchy, packet classification is coarse, to minimize state and control-plane chatter.

2. At lower levels of hierarchy, packet classification can be more granular because classifiers in the lower levels deal with a subset of the entire network: fewer flows, lower bit-rate and a subset of network policy.

However, in this model, a new component that can proxy between the different domains, termed "Internal Boundary Node (IBN)," will be required. It has some commonality with the legacy SF proxy discussed in [I-D.song-sfc-legacy-sf-mapping].

This model also requires some coordination of path information within the IBN, since the IBN must map packets back and forth between domains. Solving this probably requires sharing metadata dictionaries among controllers and inventing a scheme that provides a level of indirection by naming path identifiers and metadata values.
4.3. Example of selecting Methods and Patterns

In this section, clarifications about the most suitable method and pattern are made for the following example networks based on the results of the above analysis.

4.3.1. Example#1: Enterprise Datacenter Network

The conditions of the target network are as follows:

Network type: Network with a single DC.

Intended service: For providing several network service to traffic of one or several business offices.

Variation of service: A group of adopting network service varies per office.

The number of SFs included in a service chain: Less than 5 (ref. section 3.2.1. Sample north-south service function chains in [I-D.ietf-sfc-dc-use-cases]).

Features of SFs: SFs are set statically, and SFs are exclusively used for each service.

On the basis of the conditions "network type" and "features of SFs", pattern 1 with SF dedicated model would be selected.

As the condition "variation of service" describes, such network requires few flow entries for each FWD, so method 1 would be applicable. Method 1 also does not require SFs to have any additional mechanism to apply any header, thus the impact of implementing this method would be less than other methods.

4.3.2. Example#2: Current Mobile Service Providers Network

The conditions of the target network are as follows:

Network type: Network with a single DC (e.g., (S)Gi-LAN (3GPP, [TS.23.203])).

Intended service: For providing network access service and several network service to traffic of millions customers.

Variation of service: Service varies per user or applications.

The number of SFs included in a service chain: Around 5 (ref. examples of service in [I-D.ietf-sfc-use-case-mobility]).
Features of SFs: Many SFs are hardware equipment and they are deployed statically. Also, many SFs are used for several service. A function to inspect user traffic in detail, such as TDF (3GPP, [TS.23.203]), is located at the ingress of the network, and it might behave as a CF.

On the basis of the conditions "network type" and "features of SFs," pattern 1 with SF shared model would be selected. In such network, classification based on deep packet inspection such as application type inspections is done, and paths branching will not be happen.

As the other conditions describe, the operator must handle millions of flows and the flows traverse multiple SFs, so method 3 would be applicable. Configuring such amounts of flows among large scale network might be too much work for operators.

The examples of concrete service of such network are described as follows:

1. HTTP Modification

   Packet Gateway(P-GW), which is defined in 3GPP (ref. [TS.23.203]), detects traffic to the specific website and that traffic must be sent through a special element to insert additional data to the HTTP header or advertisement to the HTTP traffic, so the destination site can apply specific deals with the operator’s customer (simplify DRM, premium service, etc.). That would require flow entries with mobile source IP, destination IP and port.

2. VoLTE Calls

   VoLTE calls are sent via a special SP. The VoLTE control plane selects all application network elements. But to reach application network elements it fully relies on standard routing and switching mechanisms. With Service Chaining it is possible to select the SP which can provide required QoS. That would require to set flow entries with mobile source IP, destination IP and port.

3. Secure Internet Access

   Some customers’ HTTP traffic is forwarded to one or more security functions to inspect for malware. This case would require flow entries with source IP, destination IP and port.

4. Content Optimizer
Based on the policy rules, a SC/SP with the Content Optimization might be provided. Content optimization primarily affects video and HTTP traffic, and saves valuable radio resources in the specific radio cells during times of congestion. A controller might monitor Key Performance Indicators (KPIs) of the radio network to detect congestion. When congestion is detected, the controller might enforce a content optimization policy for the users on the congested radio cell. Most resource-expensive traffic can be transcoded by a content optimizer to save bandwidth. Selecting traffic for optimization would require to set flow entries with mobile source IP, destination IP and port. Also, content optimization might require changing SCs/SPs assigned to users flows based on the result of KPI monitoring or the time of day.

On the other hand, method 1 might be also selected with pattern 1 with SF dedicated model. For example, the series of the above service might be achieved by static configured flow entries, for example, with incoming port. However, it will require many incoming ports for FWDs when the operator would like to share a SF with multiple SCs, and it will not be scalable.

4.3.3. Example#3: Fixed and Mobile Converged Service Providers Network

The conditions of the target network are as follows:

Network type: Network with multiple DCs (e.g., SFs are deployed at multiple DCs based on their applications).

Intended service: For providing network access service or several network service to traffic of millions customers.

Variation of service: Service varies per user. Also, the service assigned to each flow might vary based on using applications.

The number of SFs included in a service chain: More than 5. (Various services such as enriched security service and value added services would be provided)

Features of SFs: Many SFs are deployed as VNFs (Virtualized Network Functions), and some SFs are shared with multiple SCs. Also, some SFs changes the following SPs dynamically based on the result of the process.

On the basis of the conditions "network type" and "features of SFs," pattern 2 would be selected. Pattern 2 allows hierarchical approach which enables operators to deploy SFs in multiple domains easily based on service requirements. For example, operators can deploy SFs
into several domains based on application types. This concept is introduced in [I-D.ietf-sfc-dc-use-cases].

From the above conditions describe, the operator must handle enormous flows and paths branching, thus method 3 will be appreciable for such network. Especially, security scenario sometimes requires paths branching based on the result of packet inspection such as processes of DPI or traffic analyzer. Some security functions such as web application firewall (WAF) are specialized for each application, and it might be inefficient to insert all traffic into such SFs. Therefore, for inserting only target packets to appropriate security functions, classifying and paths branching based on packet inspection would be required.

5. Acknowledgements

The authors would like to thank Konomi Mochizuki and Lily Guo for their reviews and comments.

6. Contributors

The following people are active contributors to this document and have provided review, content and concepts (listed alphabetically by surname):

Poul Bottorff
Hewlett Packard Networking

Mohamed Boucadair
France Telecom

Nicolas Bouthors
Qosmos

Hiroshi Dempo
NEC

Christian Jacquenet
France Telecom

Ron Parker
Affirmed Networks

Chuong D. Pham
Telstra

Paul Quinn
Cisco Systems
7. IANA Considerations

This memo includes no request to IANA.

8. References

[I-D.dolson-sfc-hierarchical]

[I-D.fedyk-sfc-mac-chain]

[I-D.ietf-sfc-dc-use-cases]

[I-D.ietf-sfc-nsh]

[I-D.ietf-sfc-use-case-mobility]

[I-D.song-sfc-legacy-sf-mapping]


Authors' Addresses

Shunsuke Homma
NTT, Corp.
3-9-11, Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81 422 59 3486
Email: homma.shunsuke@lab.ntt.co.jp

Kengo Naito
NTT, Corp.
3-9-11, Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Email: k.naito@nttv6.jp

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

Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com
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

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction ................................................. 3
   1.1. Scope ................................................. 3
   1.2. Assumptions .......................................... 4
   1.3. Specification of Requirements ......................... 4
   1.4. Definition of Terms ................................... 5
2. Architectural Concepts ....................................... 7
   2.1. Service Function Chains ............................... 7
   2.2. Service Function Chain Symmetry ....................... 8
   2.3. Service Function Paths ............................... 9
      2.3.1. Service Function Chains, Service Function Paths, and
             Rendered Service Path ................................ 10
3. Architecture Principles ...................................... 11
4. Core SFC Architecture Components .......................... 12
   4.1. SFC Encapsulation .................................... 13
   4.2. Service Function (SF) .................................. 14
   4.3. Service Function Forwarder (SFF) ...................... 14
      4.3.1. Transport Derived SFF ............................. 16
   4.4. SFC-Enabled Domain .................................... 16
   4.5. Network Overlay and Network Components ............... 17
   4.6. SFC Proxy ............................................ 17
   4.7. Classification ........................................ 18
   4.8. Re-Classification and Branching ....................... 18
   4.9. Shared Metadata ....................................... 19
5. Additional Architectural Concepts .......................... 19
   5.1. The Role of Policy .................................... 19
   5.2. SFC Control Plane ..................................... 20
   5.3. Resource Control ...................................... 21
   5.4. Infinite Loop Detection and Avoidance ................ 22
   5.5. Load Balancing Considerations ......................... 22
   5.6. MTU and Fragmentation Considerations ................ 23
   5.7. SFC OAM .............................................. 24
   5.8. Resilience and Redundancy ............................ 25
6. Security Considerations ...................................... 25
7. Contributors and Acknowledgments .......................... 28
8. IANA Considerations ......................................... 29
9. Informative References ...................................... 29
Authors’ Addresses ............................................. 30
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.
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 caching 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 independence, 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:

```
+--------------+                  +------------------˜˜˜
|   Service    |       SFC        |  Service  +---+   +---+
|Classification|  Encapsulation   |   Function  |sf1|...|sfn|
+--------------+                  +------------------˜˜˜
```

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:
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.
When traffic is returned from the SF:

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

```
+---+    +---++--->web server
source+-->|sff|+-->|sf1|+--->web server
+---+    +---++--->web server
```

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.

```
+---+ +---+ +---+   +---+ +---+ +---+
|sf2| |sf2| |sf3|   |sf3| |sf4| |sf4|
+---+ +---+ +---+   +---+ +---+ +---+

source+-->|sff|+-->|sff|+--->|sff|+--->|sff|+-->destination

+-----+-----+       +-----+-----+       +-----+-----+
|                   |                   |
+                   +

+----+    +----+     +----+     +----+    +----+
|sf1|    |sf3|     |sf5|     |                  |
+----+    +----+     +----+     +                  +
```

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.
Expects 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 SFP 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 SFPs, 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 operator's 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

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

Halpern & Pignataro Expires January 25, 2016


Authors' Addresses

Joel Halpern (editor)
Ericsson

Email: jmh@joelhalpern.com

Carlos Pignataro (editor)
Cisco Systems, Inc.

Email: cpignata@cisco.com
Abstract

This document describes a Network Service Header (NSH) imposed on packets or frames to realize service function paths. The NSH also provides a mechanism for metadata exchange along the instantiated service paths. The NSH is the SFC encapsulation required to support the Service Function Chaining (SFC) architecture (defined in RFC7665).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on May 7, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect
Table of Contents

1. Introduction .......................... 3
   1.1. Requirements Language .............. 4
   1.2. Applicability ..................... 4
   1.3. Definition of Terms ............... 5
   1.4. Problem Space ...................... 6
   1.5. NSH-based Service Chaining ........ 6
2. Network Service Header .................. 7
   2.1. Network Service Header Format ...... 7
   2.2. NSH Base Header .................... 8
   2.3. Service Path Header .............. 11
   2.4. NSH MD Type 1 ..................... 12
   2.5. NSH MD Type 2 ..................... 12
   2.5.1. Optional Variable Length Metadata 13
3. NSH Actions ................................ 14
4. NSH Transport Encapsulation ............. 16
5. Fragmentation Considerations ............ 17
6. Service Path Forwarding with NSH ......... 17
   6.1. SFFs and Overlay Selection .......... 17
   6.2. Mapping the NSH to Network Topology .. 21
   6.3. Service Plane Visibility ............ 21
   6.4. Service Graphs .................... 22
7. Policy Enforcement with NSH ............. 22
   7.1. NSH Metadata and Policy Enforcement 22
   7.2. Updating/Augmenting Metadata ........ 24
   7.3. Service Path Identifier and Metadata 25
8. Security Considerations ................. 26
   8.1. NSH Security Considerations from Operators’ Environments 27
   8.2. NSH Security Considerations from the SFC Architecture .. 28
   8.2.1. Integrity ....................... 29
   8.2.2. Confidentiality ................. 31
9. Contributors ................................ 32
10. Acknowledgments ........................ 33
11. IANA Considerations .................... 33
   11.1. Network Service Header (NSH) Parameters 34
       11.1.1. NSH Base Header Bits .......... 34
       11.1.2. NSH Version .................. 34
       11.1.3. MD Type Registry .............. 34
       11.1.4. MD Class Registry ............. 35
       11.1.5. New IETF Assigned Optional Variable Length Metadata 35
               Type Registry .................. 36
       11.1.6. NSH Base Header Next Protocol . 36
1. Introduction

Service functions are widely deployed and essential in many networks. These service functions provide a range of features such as security, WAN acceleration, and server load balancing. Service functions may be instantiated at different points in the network infrastructure such as the wide area network, data center, and so forth.

Prior to development of the SFC architecture [RFC7665] and the protocol specified in this document, current service function deployment models have been relatively static and bound to topology for insertion and policy selection. Furthermore, they do not adapt well to elastic service environments enabled by virtualization.

New data center network and cloud architectures require more flexible service function deployment models. Additionally, the transition to virtual platforms demands an agile service insertion model that supports dynamic and elastic service delivery. Specifically, the following functions are necessary:

1. The movement of service functions and application workloads in the network.

2. The ability to easily bind service policy to granular information, such as per-subscriber state.

3. The capability to steer traffic to the requisite service function(s).

The Network Service Header (NSH) specification defines a new data plane protocol, which is an encapsulation for service function chains. The NSH is designed to encapsulate an original packet or frame, and in turn be encapsulated by an outer transport encapsulation (which is used to deliver the NSH to NSH-aware network elements), as shown in Figure 1:
The NSH is composed of the following elements:

2. Indication of location within a Service Function Path.
3. Optional, per packet metadata (fixed length or variable).

[RFC7665] provides an overview of a service chaining architecture that clearly defines the roles of the various elements and the scope of a service function chaining encapsulation. Figure 3 of [RFC7665] depicts the SFC architectural components after classification. The NSH is the SFC encapsulation referenced in [RFC7665].

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].

1.2. Applicability

The NSH is designed to be easy to implement across a range of devices, both physical and virtual, including hardware platforms.

The intended scope of the NSH is for use within a single provider's operational domain. This deployment scope is deliberately constrained, as explained also in [RFC7665], and limited to a single network administrative domain. In this context, a "domain" is a set of network entities within a single administration. For example, a network administrative domain can include a single data center, or an overlay domain using virtual connections and tunnels. A corollary is that a network administrative domain has a well defined perimeter.

An NSH-aware control plane is outside the scope of this document.
1.3. Definition of Terms

Byte: All references to "bytes" in this document refer to 8-bit bytes, or octets.

Classification: Defined in [RFC7665].

Classifier: Defined in [RFC7665].

Metadata: Defined in [RFC7665]. The metadata, or context information shared between classifiers and SFs, and among SFs, is carried on the NSH’s Context Headers. It allows summarizing a classification result in the packet itself, avoiding subsequent re-classifications. Examples of metadata include classification information used for policy enforcement and network context for forwarding post service delivery.

Network Locator: Data plane address, typically IPv4 or IPv6, used to send and receive network traffic.

Network Node/Element: Device that forwards packets or frames based on an outer header (i.e., transport encapsulation) information.

Network Overlay: Logical network built on top of existing network (the underlay). Packets are encapsulated or tunneled to create the overlay network topology.

NSH-aware: NSH-aware means SFC-encapsulation-aware, where the NSH provides the SFC encapsulation. This specification uses NSH-aware as a more specific term from the more generic term SFC-aware [RFC7665].

Service Classifier: Logical entity providing classification function. Since they are logical, classifiers may be co-resident with SFC elements such as SFs or SFFs. Service classifiers perform classification and impose the NSH. The initial classifier imposes the initial NSH and sends the NSH packet to the first SFF in the path. Non-initial (i.e., subsequent) classification can occur as needed and can alter, or create a new service path.

Service Function (SF): Defined in [RFC7665].

Service Function Chain (SFC): Defined in [RFC7665].

Service Function Forwarder (SFF): Defined in [RFC7665].

Service Function Path (SFP): Defined in [RFC7665].
Service Plane: The collection of SFFs and associated SFs creates a
service-plane overlay in which all SFs and SFC Proxies reside
[RFC7665].

SFC Proxy: Defined in [RFC7665].

1.4. Problem Space

The NSH addresses several limitations associated with service
function deployments. [RFC7498] provides a comprehensive review of
those issues.

1.5. NSH-based Service Chaining

The NSH creates a dedicated service plane; more specifically, the NSH enables:

1. Topological Independence: Service forwarding occurs within the
   service plane, so the underlying network topology does not
   require modification. The NSH provides an identifier used to
   select the network overlay for network forwarding.

2. Service Chaining: The NSH enables service chaining per [RFC7665].
   The NSH contains path identification information needed to
   realize a service path. Furthermore, the NSH provides the
   ability to monitor and troubleshoot a service chain, end-to-end
   via service-specific OAM messages. The NSH fields can be used by
   administrators (via, for example, a traffic analyzer) to verify
   (account, ensure correct chaining, provide reports, etc.) the
   path specifics of packets being forwarded along a service path.

3. The NSH provides a mechanism to carry shared metadata between
   participating entities and service functions. The semantics of
   the shared metadata is communicated via a control plane, which is
   outside the scope of this document, to participating nodes.
   [I-D.ietf-sfc-control-plane] provides an example of such in
   Section 3.3. Examples of metadata include classification
   information used for policy enforcement and network context for
   forwarding post service delivery. Sharing the metadata allows
   service functions to share initial and intermediate
   classification results with downstream service functions saving
   re-classification, where enough information was enclosed.

4. The NSH offers a common and standards-based header for service
   chaining to all network and service nodes.

5. Transport Encapsulation Agnostic: The NSH is transport
   encapsulation-independent, meaning it can be transported by a
variety of encapsulation protocols. An appropriate (for a given deployment) encapsulation protocol can be used to carry NSH-encapsulated traffic. This transport encapsulation may form an overlay network and if an existing overlay topology provides the required service path connectivity, that existing overlay may be used.

2. Network Service Header

An NSH is imposed on the original packet/frame. This NSH contains service path information and optionally metadata that are added to a packet or frame and used to create a service plane. Subsequently, an outer transport encapsulation is imposed on the NSH, which is used for network forwarding.

A Service Classifier adds the NSH. The NSH is removed by the last SFF in the service chain or by an SF that consumes the packet.

2.1. Network Service Header Format

The NSH is composed of a 4-byte Base Header, a 4-byte Service Path Header and optional Context Headers, as shown in Figure 2 below.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Base Header                                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Service Path Header                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                Context Header(s)                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Network Service Header

Base header: Provides information about the service header and the payload protocol.

Service Path Header: Provides path identification and location within a service path.

Context header: Carries metadata (i.e., context data) along a service path.
2.2. NSH Base Header

Figure 3 depicts the NSH base header:

```
  0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|O|U|    TTL    |   Length  |U|U|U|U|MD Type| Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: NSH Base Header

Base Header Field Descriptions:

Version: The version field is used to ensure backward compatibility going forward with future NSH specification updates. It MUST be set to 0x0 by the sender, in this first revision of the NSH. If a packet presumed to carry an NSH header is received at an SFF, and the SFF does not understand the version of the protocol as indicated in the base header, the packet MUST be discarded, and the event SHOULD be logged. Given the widespread implementation of existing hardware that uses the first nibble after an MPLS label stack for equal-cost multipath (ECMP) decision processing, this document reserves version 01b. This value MUST NOT be used in future versions of the protocol. Please see [RFC7325] for further discussion of MPLS-related forwarding requirements.

O bit: Setting this bit indicates an Operations, Administration, and Maintenance (OAM, see [RFC6291]) packet. The actual format and processing of SFC OAM packets is outside the scope of this specification (see for example [I-D.ietf-sfc-oam-framework] for one approach).

The O bit MUST be set for OAM packets and MUST NOT be set for non-OAM packets. The O bit MUST NOT be modified along the SFP.

SF/SFF/SFC Proxy/Classifier implementations that do not support SFC OAM procedures SHOULD discard packets with O bit set, but MAY support a configurable parameter to enable forwarding received SFC OAM packets unmodified to the next element in the chain. Forwarding OAM packets unmodified by SFC elements that do not support SFC OAM procedures may be acceptable for a subset of OAM functions, but can result in unexpected outcomes for others; thus, it is recommended to analyze the impact of forwarding an OAM packet for all OAM functions prior to enabling this behavior. The configurable parameter MUST be disabled by default.
TTL: Indicates the maximum SFF hops for an SFP. This field is used for service plane loop detection. The initial TTL value SHOULD be configurable via the control plane; the configured initial value can be specific to one or more SFPs. If no initial value is explicitly provided, the default initial TTL value of 63 MUST be used. Each SFF involved in forwarding an NSH packet MUST decrement the TTL value by 1 prior to NSH forwarding lookup. Decrementing by 1 from an incoming value of 0 shall result in a TTL value of 63. The packet MUST NOT be forwarded if TTL is, after decrement, 0.

This TTL field is the primary loop prevention mechanism. This TTL mechanism represents a robust complement to the Service Index (see Section 2.3), as the TTL is decrement by each SFF. The handling of incoming 0 TTL allows for better, although not perfect, interoperation with pre-standard implementations that do not support this TTL field.

Length: The total length, in 4-byte words, of the NSH including the Base Header, the Service Path Header, the Fixed Length Context Header or Variable Length Context Header(s). The length MUST be 0x6 for MD Type equal to 0x1, and MUST be 0x2 or greater for MD Type equal to 0x2. The length of the NSH header MUST be an integer multiple of 4 bytes, thus variable length metadata is always padded out to a multiple of 4 bytes.

Unassigned bits: All other flag fields, marked U, are unassigned and available for future use, see Section 11.1.1. Unassigned bits MUST be set to zero upon origination, and MUST be ignored and preserved unmodified by other NSH supporting elements. At reception, all elements MUST NOT modify their actions based on these unknown bits.

Metadata (MD) Type: Indicates the format of the NSH beyond the mandatory Base Header and the Service Path Header. MD Type defines the format of the metadata being carried. Please see the IANA Considerations Section 11.1.3.

This document specifies the following four MD Type values:

0x0 - This is a reserved value. Implementations SHOULD silently discard packets with MD Type 0x0.

0x1 - This indicates that the format of the header includes a fixed length Context Header (see Figure 5 below).

0x2 - This does not mandate any headers beyond the Base Header and Service Path Header, but may contain optional variable length Context Header(s). With MD Type 0x2, a Length of 0x2 implies there are no Context Headers. The semantics of the variable length Context
Header(s) are not defined in this document. The format of the optional variable length Context Headers is provided in Section 2.5.1.

0xF - This value is reserved for experimentation and testing, as per [RFC3692]. Implementations not explicitly configured to be part of an experiment SHOULD silently discard packets with MD Type 0xF.

The format of the Base Header and the Service Path Header is invariant, and not affected by MD Type.

The NSH MD Type 1 and MD Type 2 are described in detail in Sections 2.4 and 2.5, respectively. NSH implementations MUST support MD types 0x1 and 0x2 (where the length is 0x2). NSH implementations SHOULD support MD Type 0x2 with length greater than 0x2. Devices that do not support MD Type 0x2 with length greater than 0x2 MUST ignore any optional context headers and process the packet without them; the base header length field can be used to determine the original payload offset if access to the original packet/frame is required. This specification does not disallow the MD Type value from changing along an SFP; however, the specification of the necessary mechanism to allow the MD Type to change along an SFP are outside the scope of this document and would need to be defined for that functionality to be available. Packets with MD Type values not supported by an implementation MUST be silently dropped.

Next Protocol: indicates the protocol type of the encapsulated data. The NSH does not alter the inner payload, and the semantics on the inner protocol remain unchanged due to NSH service function chaining. Please see the IANA Considerations section below, Section 11.1.6.

This document defines the following Next Protocol values:

0x1: IPv4
0x2: IPv6
0x3: Ethernet
0x4: NSH
0x5: MPLS
0xFE: Experiment 1
0xFF: Experiment 2

The functionality of hierarchical NSH using a Next Protocol value of 0x4 NSH is outside the scope of this specification. Packets with Next Protocol values not supported SHOULD be silently dropped by default, although an implementation MAY provide a configuration parameter to forward them. Additionally, an implementation not explicitly configured for a specific experiment [RFC3692] SHOULD silently drop packets with Next Protocol values 0xFE and 0xFF.
2.3. Service Path Header

Figure 4 shows the format of the Service Path Header:

```
+-----------------------------+-----------------------------+
|     Service Path Identifier (SPI)    |     Service Index       |
+-----------------------------+-----------------------------+
```

Service Path Identifier (SPI): 24 bits
Service Index (SI): 8 bits

Figure 4: NSH Service Path Header

The meaning of these fields is as follows:

Service Path Identifier (SPI): Uniquely identifies a service function path. Participating nodes MUST use this identifier for Service Function Path selection (SFP). The initial classifier MUST set the appropriate SPI for a given classification result.

Service Index (SI): Provides location within the SFP. The initial classifier for a given SFP SHOULD set the SI to 255, however the control plane MAY configure the initial value of SI as appropriate (i.e., taking into account the length of the service function path). The Service Index MUST be decremented by a value of 1 by Service Functions or by SFC Proxy nodes after performing required services and the new decremented SI value MUST be used in the egress packet’s NSH. The initial Classifier MUST send the packet to the first SFF in the identified SFP for forwarding along an SFP. If re-classification occurs, and that re-classification results in a new SPI, the (re)classifier is, in effect, the initial classifier for the resultant SPI.

The SI is used in conjunction with Service Path Identifier for Service Function Path Selection and for determining the next SFF/SF in the path. The SI is also valuable when troubleshooting or reporting service paths. While the TTL provides the primary SFF based loop prevention for this mechanism, SI decrement by SF serves as a limited loop prevention mechanism. NSH packets, as described above, are discarded when an SFF decrements the TTL to 0. In addition, an SFF which is not the terminal SFF for a Service Function Path will discard any NSH packet with an SI of 0, as there will be no valid next SF information.
2.4. NSH MD Type 1

When the Base Header specifies MD Type = 0x1, a Fixed Length Context Header (16-bytes) MUST be present immediately following the Service Path Header, as per Figure 5. The value of a Fixed Length Context Header that carries no metadata MUST be set to zero.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|O|U|    TTL    |   Length  |U|U|U|U|MD Type| Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                 Fixed Length Context Header                   |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: NSH MD Type=0x1

This specification does not make any assumptions about the content of the 16 byte Context Header that must be present when the MD Type field is set to 1, and does not describe the structure or meaning of the included metadata.

An SFC-aware SF or SFC Proxy needs to receive the data structure and semantics first in order to process the data placed in the mandatory context field. The data structure and semantics include both the allocation schema and order, and the meaning of the included data. How an SFC-aware SF or SFC Proxy gets the data structure and semantics is outside the scope of this specification.

An SF or SFC Proxy that does not know the format or semantics of the Context Header for an NSH with MD Type 1 MUST discard any packet with such an NSH (i.e., MUST NOT ignore the metadata that it cannot process), and MUST log the event at least once per the SPI for which the event occurs (subject to thresholding).

[I-D.guichard-sfc-nsh-dc-allocation] and [I-D.napper-sfc-nsh-broadband-allocation] provide specific examples of how metadata can be allocated.

2.5. NSH MD Type 2

When the base header specifies MD Type = 0x2, zero or more Variable Length Context Headers MAY be added, immediately following the Service Path Header (see Figure 6). Therefore, Length = 0x2,
indicates that only the Base Header followed by the Service Path Header are present. The optional Variable Length Context Headers MUST be of an integer number of 4-bytes. The base header Length field MUST be used to determine the offset to locate the original packet or frame for SFC nodes that require access to that information.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|O|U|    TTL    |   Length  |U|U|U|U|MD Type| Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Service Path Identifier              | Service Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
Figure 6: NSH MD Type=0x2
```

2.5.1. Optional Variable Length Metadata

The format of the optional variable length Context Headers, is as depicted in Figure 7.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Metadata Class       |      Type     |U|    Length   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Variable Metadata                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      
Figure 7: Variable Context Headers
```

Metadata Class (MD Class): Defines the scope of the ’Type’ field to provide a hierarchical namespace. The IANA Considerations Section 11.1.4 defines how the MD Class values can be allocated to standards bodies, vendors, and others.

Type: Indicates the explicit type of metadata being carried. The definition of the Type is the responsibility of the MD Class owner.

Unassigned bit: One unassigned bit is available for future use. This bit MUST NOT be set, and MUST be ignored on receipt.
Length: Indicates the length of the variable metadata, in bytes. In case the metadata length is not an integer number of 4-byte words, the sender MUST add pad bytes immediately following the last metadata byte to extend the metadata to an integer number of 4-byte words. The receiver MUST round up the length field to the nearest 4-byte word boundary, to locate and process the next field in the packet. The receiver MUST access only those bytes in the metadata indicated by the length field (i.e., actual number of bytes) and MUST ignore the remaining bytes up to the nearest 4-byte word boundary. The Length may be 0 or greater.

A value of 0 denotes a Context Header without a Variable Metadata field.

This specification does not make any assumption about Context Headers that are mandatory-to-implement or those that are mandatory-to-process. These considerations are deployment-specific. However, the control plane is entitled to instruct SFC-aware SFs with the data structure of context header together with its scoping (see e.g., Section 3.3.3 of [I-D.ietf-sfc-control-plane]).

Upon receipt of a packet that belongs to a given SFP, if a mandatory-to-process context header is missing in that packet, the SFC-aware SF MUST NOT process the packet and MUST log an error at least once per the SPI for which the mandatory metadata is missing.

If multiple mandatory-to-process context headers are required for a given SFP, the control plane MAY instruct the SFC-aware SF with the order to consume these Context Headers. If no instructions are provided and the SFC-aware SF will make use of or modify the specific context header, then the SFC-aware SF MUST process these Context Headers in the order they appear in an NSH packet.

If multiple instances of the same metadata are included in an NSH packet, but the definition of that context header does not allow for it, the SFC-aware SF MUST process the first instance and ignore subsequent instances. The SFC-aware SF MAY log or increase a counter for this event.

3. NSH Actions

NSH-aware nodes, which include service classifiers, SFFs, SFs and SFC proxies, may alter the contents of the NSH headers. These nodes have several possible NSH-related actions:

1. Insert or remove the NSH: These actions can occur respectively at the start and end of a service path. Packets are classified, and if determined to require servicing, an NSH will be imposed. A
service classifier MUST insert an NSH at the start of an SFP. An imposed NSH MUST contain both a valid Base Header and Service Path Header. At the end of a service function path, an SFF MUST remove the NSH before forwarding or delivering the un-encapsulated packet. It is therefore the last node operating on the service header.

Multiple logical classifiers may exist within a given service path. Non-initial classifiers may re-classify data and that re-classification MAY result in the selection of a different Service Function Path. When the logical classifier performs re-classification that results in a change of service path, it MUST replace the existing NSH with a new NSH with the Base Header and Service Path Header reflecting the new service path information and MUST set the initial SI. The O bit, the TTL field, as well as unassigned flags, MUST be copied transparently from the old NSH to a new NSH. Metadata MAY be preserved in the new NSH.

2. Select service path: The Service Path Header provides service path information and is used by SFFs to determine correct service path selection. SFFs MUST use the Service Path Header for selecting the next SF or SFF in the service path.

3. Update the NSH: SFs MUST decrement the service index by one. If an SFF receives a packet with an SPI and SI that do not correspond to a valid next hop in a valid Service Function Path, that packet MUST be dropped by the SFF.

Classifiers MAY update Context Headers if new/updated context is available.

If an SFC proxy is in use (acting on behalf of an NSH-unaware service function for NSH actions), then the proxy MUST update Service Index and MAY update contexts. When an SFC proxy receives an NSH-encapsulated packet, it MUST remove the NSH before forwarding it to an NSH-unaware SF. When the SFC Proxy receives a packet back from an NSH-unaware SF, it MUST re-encapsulate it with the correct NSH, and MUST decrement the Service Index by one.

4. Service policy selection: Service Functions derive policy (i.e., service actions such as permit or deny) selection and enforcement from the NSH. Metadata shared in the NSH can provide a range of service-relevant information such as traffic classification.
Figure 8 maps each of the four actions above to the components in the SFC architecture that can perform it.

<table>
<thead>
<tr>
<th>Component</th>
<th>Insert</th>
<th>Remove</th>
<th>Replace</th>
<th>Forward</th>
<th>Update</th>
<th>Service policy sel.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Service Function Forwarder (SFF)</td>
<td></td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service Function (SF)</td>
<td></td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>SFC Proxy</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 8: NSH Action and Role Mapping**

4. **NSH Transport Encapsulation**

Once the NSH is added to a packet, an outer transport encapsulation is used to forward the original packet and the associated metadata to the start of a service chain. The encapsulation serves two purposes:

1. Creates a topologically independent services plane. Packets are forwarded to the required services without changing the underlying network topology.

2. Transit network nodes simply forward the encapsulated packets without modification.

The service header is independent of the transport encapsulation used. Existing transport encapsulations can be used. The presence of an NSH is indicated via a protocol type or another indicator in the outer transport encapsulation.
5. Fragmentation Considerations

The NSH and the associated transport encapsulation header are "added" to the encapsulated packet/frame. This additional information increases the size of the packet.

Within a managed administrative domain, an operator can ensure that the underlay MTU is sufficient to carry SFC traffic without requiring fragmentation. Given that the intended scope of the NSH is within a single provider's operational domain, that approach is sufficient.

However, although explicitly outside the scope of this specification, there might be cases where the underlay MTU is not large enough to carry the NSH traffic. Since the NSH does not provide fragmentation support at the service plane, the transport encapsulation protocol ought to provide the requisite fragmentation handling. For instance, Section 9 of [I-D.ietf-rtgwg-dt-encap] provides exemplary approaches and guidance for those scenarios.

When the transport encapsulation protocol supports fragmentation, and fragmentation procedures needs to be used, such fragmentation is part of the transport encapsulation logic. If, as it is common, fragmentation is performed by the endpoints of the transport encapsulation, then fragmentation procedures are performed at the sending NSH entity as part of the transport encapsulation, and reassembly procedures are performed at the receiving NSH entity during transport de-encapsulation handling logic. In no case would such fragmentation result in duplication of the NSH header.

For example, when the NSH is encapsulated in IP, IP-level fragmentation coupled with Path MTU Discovery (PMTUD) (e.g., [RFC8201]) is used. Since PMTUD relies on ICMP messages, an operator should ensure ICMP packets are not blocked. When, on the other hand, the underlay does not support fragmentation procedures, an error message SHOULD be logged when dropping a packet too big. Lastly, NSH-specific fragmentation and reassembly methods may be defined as well, but these methods are outside the scope of this document, and subject for future work.

6. Service Path Forwarding with NSH

6.1. SFFs and Overlay Selection

As described above, the NSH contains a Service Path Identifier (SPI) and a Service Index (SI). The SPI is, as per its name, an identifier. The SPI alone cannot be used to forward packets along a service path. Rather the SPI provides a level of indirection between the service path/topology and the network transport encapsulation.
Furthermore, there is no requirement, or expectation of an SPI being bound to a pre-determined or static network path.

The Service Index provides an indication of location within a service path. The combination of SPI and SI provides the identification of a logical SF and its order within the service plane, and is used to select the appropriate network locator(s) for overlay forwarding. The logical SF may be a single SF, or a set of eligible SFs that are equivalent. In the latter case, the SFF provides load distribution amongst the collection of SFs as needed.

SI serves as a mechanism for detecting invalid service function paths. In particular, an SI value of zero indicates that forwarding is incorrect and the packet must be discarded.

This indirection -- SPI to overlay -- creates a true service plane. That is, the SFF/SF topology is constructed without impacting the network topology but more importantly, service plane only participants (i.e., most SFs) need not be part of the network overlay topology and its associated infrastructure (e.g., control plane, routing tables, etc.) SFs need to be able to return a packet to an appropriate SFF (i.e., has the requisite NSH information) when service processing is complete. This can be via the overlay or underlay and in some cases require additional configuration on the SF. As mentioned above, an existing overlay topology may be used provided it offers the requisite connectivity.

The mapping of SPI to transport encapsulation occurs on an SFF (as discussed above, the first SFF in the path gets an NSH encapsulated packet from the Classifier). The SFF consults the SPI/ID values to determine the appropriate overlay transport encapsulation protocol (several may be used within a given network) and next hop for the requisite SF. Table 1 below depicts an example of a single next-hop SPI/SI to network overlay network locator mapping.
Additionally, further indirection is possible: the resolution of the required SF network locator may be a localized resolution on an SFF, rather than a service function chain control plane responsibility, as per Table 2 and Table 3 below.

Please note: VXLAN-gpe and GRE in the above table refer to [I-D.ietf-nvo3-vxlan-gpe] and [RFC2784] [RFC7676], respectively.

<table>
<thead>
<tr>
<th>SPI</th>
<th>SI</th>
<th>Next hop(s)</th>
<th>Transport Encapsulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>255</td>
<td>192.0.2.1</td>
<td>VXLAN-gpe</td>
</tr>
<tr>
<td>10</td>
<td>254</td>
<td>198.51.100.10</td>
<td>GRE</td>
</tr>
<tr>
<td>10</td>
<td>251</td>
<td>198.51.100.15</td>
<td>GRE</td>
</tr>
<tr>
<td>40</td>
<td>251</td>
<td>198.51.100.15</td>
<td>GRE</td>
</tr>
<tr>
<td>50</td>
<td>200</td>
<td>01:23:45:67:89:ab</td>
<td>Ethernet</td>
</tr>
<tr>
<td>15</td>
<td>212</td>
<td>Null (end of path)</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 1: SFF NSH Mapping Example

<table>
<thead>
<tr>
<th>SPI</th>
<th>SI</th>
<th>Next hop(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3</td>
<td>SF2</td>
</tr>
<tr>
<td>245</td>
<td>12</td>
<td>SF34</td>
</tr>
<tr>
<td>40</td>
<td>9</td>
<td>SF9</td>
</tr>
</tbody>
</table>

Table 2: NSH to SF Mapping Example
<table>
<thead>
<tr>
<th>SF</th>
<th>Next hop(s)</th>
<th>Transport Encapsulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF2</td>
<td>192.0.2.2</td>
<td>VXLAN-gpe</td>
</tr>
<tr>
<td>SF34</td>
<td>198.51.100.34</td>
<td>UDP</td>
</tr>
<tr>
<td>SF9</td>
<td>2001:db8::1</td>
<td>GRE</td>
</tr>
</tbody>
</table>

Table 3: SF Locator Mapping Example

Since the SPI is a representation of the service path, the lookup may return more than one possible next-hop within a service path for a given SF, essentially a series of weighted (equally or otherwise) paths to be used (for load distribution, redundancy, or policy), see Table 4. The metric depicted in Table 4 is an example to help illustrated weighing SFs. In a real network, the metric will range from a simple preference (similar to routing next-hop), to a true dynamic composite metric based on some service function-centric state (including load, sessions state, capacity, etc.)

<table>
<thead>
<tr>
<th>SPI</th>
<th>SI</th>
<th>NH</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3</td>
<td>203.0.113.1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>203.0.113.2</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>12</td>
<td>192.0.2.1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>203.0.113.4</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>7</td>
<td>192.0.2.10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>198.51.100.1</td>
<td>5</td>
</tr>
</tbody>
</table>

(Encapsulation type omitted for formatting)

Table 4: NSH Weighted Service Path

The information contained in Tables 1-4 may be received from the control plane, but the exact mechanism is outside the scope of this document.
6.2. Mapping the NSH to Network Topology

As described above, the mapping of SPI to network topology may result in a single path, or it might result in a more complex topology. Furthermore, the SPI to overlay mapping occurs at each SFF independently. Any combination of topology selection is possible. Please note, there is no requirement to create a new overlay topology if a suitable one already exists. NSH packets can use any (new or existing) overlay provided the requisite connectivity requirements are satisfied.

Examples of mapping for a topology:

1. Next SF is located at SFFb with locator 2001:db8::1
   SFFa mapping: SPI=10 --> VXLAN-gpe, dst-ip: 2001:db8::1

2. Next SF is located at SFFc with multiple network locators for load distribution purposes:
   SFFb mapping: SPI=10 --> VXLAN-gpe, dst_ip:203.0.113.1, 203.0.113.2, 203.0.113.3, equal cost

3. Next SF is located at SFFd with two paths from SFFc, one for redundancy:
   SFFc mapping: SPI=10 --> VXLAN-gpe, dst_ip:192.0.2.10 cost=10, 203.0.113.10, cost=20

In the above example, each SFF makes an independent decision about the network overlay path and policy for that path. In other words, there is no a priori mandate about how to forward packets in the network (only the order of services that must be traversed).

The network operator retains the ability to engineer the network paths as required. For example, the overlay path between SFFs may utilize traffic engineering, QoS marking, or ECMP, without requiring complex configuration and network protocol support to be extended to the service path explicitly. In other words, the network operates as expected, and evolves as required, as does the service plane.

6.3. Service Plane Visibility

The SPI and SI serve an important function for visibility into the service topology. An operator can determine what service path a packet is "on", and its location within that path simply by viewing NSH information (packet capture, IPFIX, etc.) The information can be used for service scheduling and placement decisions, troubleshooting, and compliance verification.
6.4. Service Graphs

While a given realized service function path is a specific sequence of service functions, the service as seen by a user can actually be a collection of service function paths, with the interconnection provided by classifiers (in-service path, non-initial reclassification). These internal reclassifiers examine the packet at relevant points in the network, and, if needed, SPI and SI are updated (whether this update is a re-write, or the imposition of a new NSH with new values is implementation specific) to reflect the "result" of the classification. These classifiers may, of course, also modify the metadata associated with the packet. [RFC7665], Section 2.1 describes Service Graphs in detail.

7. Policy Enforcement with NSH

7.1. NSH Metadata and Policy Enforcement

As described in Section 2, NSH provides the ability to carry metadata along a service path. This metadata may be derived from several sources. Common examples include:

Network nodes/devices: Information provided by network nodes can indicate network-centric information (such as VRF or tenant) that may be used by service functions or conveyed to another network node post service path egress.

External (to the network) systems: External systems, such as orchestration systems, often contain information that is valuable for service function policy decisions. In most cases, this information cannot be deduced by network nodes. For example, a cloud orchestration platform placing workloads "knows" what application is being instantiated and can communicate this information to all NSH nodes via metadata carried in the context header(s).

Service Functions: A classifier co-resident with Service Functions often perform very detailed and valuable classification.

Regardless of the source, metadata reflects the "result" of classification. The granularity of classification may vary. For example, a network switch, acting as a classifier, might only be able to classify based on a 2-tuple, or based on a 5-tuple, while a service function may be able to inspect application information. Regardless of granularity, the classification information can be represented in the NSH.
Once the data is added to the NSH, it is carried along the service path, NSH-aware SFs receive the metadata, and can use that metadata for local decisions and policy enforcement. Figure 9 and Figure 10 highlight the relationship between metadata and policy:

```plaintext
+-------+        +-------+        +-------+        +-------+        +-------+
|  SFF  )------->(  SFF  |------->|  SFF  |------->|  SFF  |
+-------+        +-------+        +-------+        +-------+        +-------+
^                ^                ^                ^
/                /                /                /
\( Class \)      \( SF1 \)      \( SF2 \)      \( Class \)      \( SF1 \)      \( SF2 \)
\  \ ify /      \  \ ify /      \  \ ify /      \  \ ify /
\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--
S-tuple:          Permit          Inspect         Permit          Deny AppZ
Tenant A          Tenant A         AppY            employees
AppY              employees
```

Figure 9: Metadata and Policy

```plaintext
+------+
| SFF |
+------+
^          ^          ^          ^
/          /          /          /
\( Class \) \( SF1 \) \( SF2 \)
\  \ ify /
\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--
5-tuple:        Permit             Inspect
Tenant A        Tenant A           AppY
```

Figure 10: External Metadata and Policy

In both of the examples above, the service functions perform policy decisions based on the result of the initial classification: the SFs did not need to perform re-classification; instead, they rely on a antecedent classification for local policy enforcement.

Depending on the information carried in the metadata, data privacy impact needs to be considered. For example, if the metadata conveys tenant information, that information may need to be authenticated.
and/or encrypted between the originator and the intended recipients (which may include intended SFs only); one approach to an optional capability to do this is explored in [I-D.reddy-sfc-nsh-encrypt]. The NSH itself does not provide privacy functions, rather it relies on the transport encapsulation/overlay. An operator can select the appropriate set of transport encapsulation protocols to ensure confidentiality (and other security) considerations are met. Metadata privacy and security considerations are a matter for the documents that define metadata format.

7.2. Updating/Augmenting Metadata

Post-initial metadata imposition (typically performed during initial service path determination), the metadata may be augmented or updated:

1. Metadata Augmentation: Information may be added to the NSH’s existing metadata, as depicted in Figure 11. For example, if the initial classification returns the tenant information, a secondary classification (perhaps co-resident with DPI or SLB) may augment the tenant classification with application information, and impose that new information in NSH metadata. The tenant classification is still valid and present, but additional information has been added to it.

2. Metadata Update: Subsequent classifiers may update the initial classification if it is determined to be incorrect or not descriptive enough. For example, the initial classifier adds metadata that describes the traffic as "Internet" but a security service function determines that the traffic is really "attack". Figure 12 illustrates an example of updating metadata.
7.3. Service Path Identifier and Metadata

Metadata information may influence the service path selection since the Service Path Identifier values can represent the result of classification. A given SPI can be defined based on classification results (including metadata classification). The imposition of the SPI and SI results in the packet being placed on the newly specified SFP at the position indicated by the imposed SPI and SI.

This relationship provides the ability to create a dynamic service plane based on complex classification without requiring each node to be capable of such classification, or requiring a coupling to the network topology. This yields service graph functionality as...
described in Section 6.4. Figure 13 illustrates an example of this behavior.

5-tuple: 5-tuple: 5-tuple:
Tenant A Inspect | SFF | Original
--- DoS ++-----++ next SF

"Scrubber"

Figure 13: Path ID and Metadata

Specific algorithms for mapping metadata to an SPI are outside the scope of this document.

8. Security Considerations

NSH security must be considered in the contexts of the SFC architecture and operators’ environments. One important characteristic of NSH is that it is not an end-to-end protocol. As opposed to a protocol that "starts" on a host, and "ends" on a server or another host, NSH is typically imposed by a network device on ingress to the SFC domain and removed at the egress of the SFC domain. As such, and as with any other network-centric protocol (e.g., IP Tunneling, Traffic Engineering, MPLS, or Provider Provisioned Virtual Private Networks) there an underlying trust that the network devices responsible for imposing, removing and acting on NSH information are trusted.

The following sections detail an analysis and present a set of requirements and recommendations in those two areas.
8.1. NSH Security Considerations from Operators’ Environments

Trusted Devices

All classifiers, SFFs and SFSs (hereinafter referred to as "SFC devices") within an operator’s environment are assumed to have been selected, vetted, and actively maintained, therefore trusted by that operator. This assumption differs from the oft held view that devices are untrusted, often referred to as zero trust model. Operators SHOULD regularly monitor (i.e. continuously audit) these devices to help ensure complaint behavior. This trust, therefore, extends into NSH operations: SFC devices are not, themselves, considered as attack vectors. This assumption, and the resultant conclusion is reasonable since this is the very basis of an operator posture; the operator depends on this reality to function. If these devices are not trusted, and indeed compromised, almost the entirety of the operator’s standard-based IP and MPLS protocol suites are vulnerable, and therefore the operation of the entire network is compromised. Although there are well documented monitoring-based methods for detecting compromise, such as include continuous monitoring, audit and log review, these may not be sufficient to contain damage by a completely compromised element.

Methods and best practices to secure devices are also widely documented and outside the scope of this document.

Single Domain Boundary

As per [RFC7665], NSH is designed for use within a single administrative domain. This scoping provides two important characteristics:

i) Clear NSH boundaries

NSH egress devices MUST strip the NSH headers before they send the users’ packets or frames out of the NSH domain.

Means to prevent leaking privacy-related information outside an administrative domain are natively supported by the NSH given that the last SFF of a service path will systematically remove the NSH encapsulation before forwarding a packet exiting the service path.

The second step in such prevention is to filter the transport encapsulation protocol used by NSH at the domain edge. The transport encapsulation protocol MUST be filtered and MUST NOT leave the domain edge.
Depending upon the transport encapsulation protocol used for NSH, this can either be done by completely blocking the transport encapsulation (e.g., if MPLS is the chosen NSH transport encapsulation protocol, it is therefore never allowed to leave the domain) or by examining the carried protocol with the transport encapsulation (e.g., if VxLAN-gpe is used as the NSH transport encapsulation protocol, all domain edges need to filter based on the carried protocol in the VxLAN-gpe.)

The other consequence of this bounding is that ingress packets MUST also be filtered to prevent attackers from sending in NSH packets with service path identification and metadata of their own selection. The same filters as described above for both the NSH at SFC devices and for the transport encapsulation protocol as general edge protections MUST be applied on ingress.

In summary, packets originating outside the SFC-enabled domain MUST be dropped if they contain an NSH. Similarly, packets exiting the SFC-enabled domain MUST be dropped if they contain an NSH.

ii) Mitigation of external threats

As per the trusted SFC devices points raised above, given that NSH is scoped within an operator’s domain, that operator can ensure that the environment, and its transitive properties, comply to that operator’s required security posture. Continuous audits for assurance are recommended with this reliance on a fully trusted environment. The term ‘continuous audits’ describes a method (automated or manual) of checking security control compliance on a regular basis, at some set period of time.

8.2. NSH Security Considerations from the SFC Architecture

The SFC architecture defines functional roles (e.g., SFF), as well as protocol element (e.g. Metadata). This section considers each role and element in the context of threats posed in the areas of integrity and confidentiality. As with routing, the distributed computation model assumes a distributed trust model.

An important consideration is that NSH contains mandatory to mute fields, and further, the SFC architecture describes cases where other fields in NSH change, all on a possible SFP hop-by-hop basis. This means that any cryptographic solution requires complex key distribution and lifecycle operations.
8.2.1. Integrity

SFC devices

SFC devices MAY perform various forms of verification on received NSH packets such as only accepting NSH packets from expected devices, checking that NSH SPI and SI values received from expected devices conform to expected values and so on. Implementation of these additional checks are a local matter and thus out of scope of this document.

NSH Base and Service Path Headers

Attackers who can modify packets within the operator’s network may be able to modify the service function path, path position, and / or the metadata associated with a packet.

One specific concern is an attack in which a malicious modification of the SPI/SI results in an alteration of the path to avoid security devices. The options discussed in this section help thwart that attack, and so does the use of the optional "Proof of Transit" method [I-D.brockners-proof-of-transit].

As stated above, SFC devices are trusted; in the case where an SFC device is compromised, NSH integrity protection would be subject to forging (in many cases) as well.

NSH itself does not mandate protocol-specific integrity protection. However, if an operator deems protection required, several options are viable:

1. SFF/SF NSH verification

Although strictly speaking not integrity protection, some of the techniques mentioned above such as checking expected NSH values are received from expected SFC device(s) can provide a form of verification without incurring the burden of a full-fledged integrity protection deployment.

2. Transport Security

NSH is always encapsulated by an outer transport encapsulation as detailed in Section 4 of this specification, and as depicted in Figure 1. If an operator deems cryptographic integrity protection necessary due to their risk analysis,
then an outer transport encapsulation that provides such protection [RFC6071], such as IPsec, MUST be used.

Although the threat model and recommendations of BCP 72 [RFC3552] Section 5 would normally require cryptographic data origin authentication for the header, this document does not mandate such mechanisms in order to reflect the operational and technical realities of deployment.

Given that NSH is transport independent, as mentioned above, a secure transport, such as IPsec can be used for carry NSH. IPsec can be used either alone, or in conjunction with other transport encapsulation protocols in turn encapsulating NSH.

Operators MUST ensure the selected transport encapsulation protocol can be supported by the transport encapsulation/underlay of all relevant network segments as well as SFFs, SFs and SFC proxies in the service path.

If connectivity between SFC-enabled devices traverses the public Internet, then such connectivity MUST be secured at the transport encapsulation layer. IPsec is an example of such a transport.

3. NSH Variable Header-based Integrity

Lastly, NSH MD-Type 2 provides, via variable length headers, the ability to append cryptographic integrity protection to the NSH packet. The implementation of such a scheme is outside the scope of this document.

NSH metadata

As with the base and service path headers, if an operator deems cryptographic integrity protection needed, then an existing, standard transport protocol MUST be used since the integrity protection applies to entire encapsulated NSH packets. As mentioned above, a risk assessment that deems dataplane traffic subject to tampering will apply not only to NSH but to the transport information and therefore the use of a secure transport is likely needed already to protect the entire stack.

If an MD-Type 2 variable header integrity scheme is in place, then the integrity of the metadata can be ensured via that mechanism as well.
8.2.2. Confidentiality

SFC devices

SFC devices can "see" (and need to use) NSH information.

NSH base and service path headers

SPI and other base/service path information does not typically require confidentiality; however, if an operator does deem confidentiality required, then, as with integrity, an existing transport encapsulation that provides encryption MUST be utilized.

NSH metadata

An attacker with access to the traffic in an operator’s network can potentially observe the metadata NSH carries with packets, potentially discovering privacy sensitive information.

Much of the metadata carried by NSH is not sensitive. It often reflects information that can be derived from the underlying packet or frame. Direct protection of such information is not necessary, as the risks are simply those of carrying the underlying packet or frame.

Implementers and operators MUST be aware that metadata can have privacy implications, and those implications are sometimes hard to predict. Therefore, attached metadata should be limited to that necessary for correct operation of the SFP. Further, [RFC8165] defines metadata considerations that operators can take into account when using NSH.

Protecting NSH metadata information between SFC components can be done using transport encapsulation protocols with suitable security capabilities, along the lines discussed above. If a security analysis deems these protections necessary, then security features in the transport encapsulation protocol (such as IPsec) MUST be used.

One useful element of providing privacy protection for sensitive metadata is described under the "SFC Encapsulation" area of the Security Considerations of [RFC7665]. Operators can and should use indirect identification for metadata deemed to be sensitive (such as personally identifying information) significantly mitigating the risk of a privacy violation. In particular, subscriber identifying information should be handled carefully, and in general SHOULD be obfuscated.
For those situations where obfuscation is either inapplicable or judged to be insufficient, an operator can also encrypt the metadata. An approach to an optional capability to do this was explored in [I-D.reddy-sfc-nsh-encrypt]. For other situations where greater assurance is desired, optional mechanisms such as [I-D.brockners-proof-of-transit] can be used.

9. Contributors

This WG document originated as draft-quinn-sfc-nsh; the following are its co-authors and contributors along with their respective affiliations at the time of WG adoption. The editors of this document would like to thank and recognize them and their contributions. These co-authors and contributors provided invaluable concepts and content for this document’s creation.

- Jim Guichard, Cisco Systems, Inc.
- Surendra Kumar, Cisco Systems, Inc.
- Michael Smith, Cisco Systems, Inc.
- Wim Henderickx, Alcatel-Lucent
- Tom Nadeau, Brocade
- Puneet Agarwal
- Rajeev Manur, Broadcom
- Abhishek Chauhan, Citrix
- Joel Halpern, Ericsson
- Sumandra Majee, F5
- David Melman, Marvell
- Pankaj Garg, Microsoft
- Brad McConnell, Rackspace
- Chris Wright, Red Hat, Inc.
- Kevin Glavin, Riverbed
- Hong (Cathy) Zhang, Huawei US R&D
10. Acknowledgments

The authors would like to thank Sunil Vallamkonda, Nagaraj Bagepalli, Abhijit Patra, Peter Bosch, Darrel Lewis, Pritesh Kothari, Tal Mizrahi and Ken Gray for their detailed review, comments and contributions.

A special thank you goes to David Ward and Tom Edsall for their guidance and feedback.

Additionally the authors would like to thank Larry Kreeger for his invaluable ideas and contributions which are reflected throughout this document.

Loa Andersson provided a thorough review and valuable comments, we thank him for that.

Reinaldo Penno deserves a particular thank you for his architecture and implementation work that helped guide the protocol concepts and design.

The editors also acknowledge comprehensive reviews and respective useful suggestions by Med Boucadair, Adrian Farrel, Juergen Schoenwaelder, Acee Lindem, and Kathleen Moriarty.

Lastly, David Dolson has provides significant review, feedback and suggestions throughout the evolution of this document. His contributions are very much appreciated.

11. IANA Considerations
11.1. Network Service Header (NSH) Parameters

IANA is requested to create a new "Network Service Header (NSH) Parameters" registry. The following sub-sections request new registries within the "Network Service Header (NSH) Parameters" registry.

11.1.1. NSH Base Header Bits

There are five unassigned bits (U bits) in the NSH Base Header, and one assigned bit (O bit). New bits are assigned via Standards Action [RFC8126].

Bit 2 - O (OAM) bit
Bit 3 - Unassigned
Bits 16-19 - Unassigned

11.1.2. NSH Version

IANA is requested to setup a registry of "NSH Version". New values are assigned via Standards Action [RFC8126].

Version 00b: Protocol as defined by this document.
Version 01b: Reserved. This document.
Version 10b: Unassigned.

11.1.3. MD Type Registry

IANA is requested to set up a registry of "MD Types". These are 4-bit values. MD Type values 0x0, 0x1, 0x2, and 0xF are specified in this document, see Table 5. Registry entries are assigned by using the "IETF Review" policy defined in RFC 8126 [RFC8126].
### Table 5: MD Type Values

<table>
<thead>
<tr>
<th>MD Type</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>Reserved</td>
<td>This document</td>
</tr>
<tr>
<td>0x1</td>
<td>NSH MD Type 1</td>
<td>This document</td>
</tr>
<tr>
<td>0x2</td>
<td>NSH MD Type 2</td>
<td>This document</td>
</tr>
<tr>
<td>0x3..0xE</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0xF</td>
<td>Experimentation</td>
<td>This document</td>
</tr>
</tbody>
</table>

### 11.1.4. MD Class Registry

IANA is requested to set up a registry of "MD Class". These are 16-bit values. New allocations are to be made according to the following policies:

- 0x0000 to 0x01ff: IETF Review
- 0x0200 to 0xffff: Expert Review
- 0xfff6 to 0xfffe: Experimental
- 0xffff: Reserved

IANA is requested to assign the values as per Table 6:

### Table 6: MD Class Value

<table>
<thead>
<tr>
<th>MD Class</th>
<th>Meaning</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>IETF Base NSH MD Class</td>
<td>This.I-D</td>
</tr>
</tbody>
</table>

A registry for Types for the MD Class of 0x0000 is defined in Section 11.1.5.

Designated Experts evaluating new allocation requests from the "Expert Review" range should principally consider whether a new MD class is needed compared to adding MD types to an existing class. The Designated Experts should also encourage the existence of an associated and publicly visible registry of MD types although this registry need not be maintained by IANA.
When evaluating a request for an allocation, the Expert should verify that the allocation plan includes considerations to handle privacy and security issues associated with the anticipated individual MD Types allocated within this class. These plans should consider, when appropriate, alternatives such as indirection, encryption, and limited deployment scenarios. Information that can’t be directly derived from viewing the packet contents should be examined for privacy and security implications.

11.1.5. New IETF Assigned Optional Variable Length Metadata Type Registry

The Type values within the IETF Base NSH MD Class, i.e., when the MD Class is set to 0x0000 (see Section 11.1.4), are the Types owned by the IETF. This document requests IANA to create a registry for the Type values for the IETF Base NSH MD Class called the "IETF Assigned Optional Variable Length Metadata Type Registry", as specified in Section 2.5.1.

The type values are assigned via Standards Action [RFC8126].

No initial values are assigned at the creation of the registry.

11.1.6. NSH Base Header Next Protocol

IANA is requested to set up a registry of "Next Protocol". These are 8-bit values. Next Protocol values 0, 1, 2, 3, 4 and 5 are defined in this document (see Table 7. New values are assigned via "Expert Reviews" as per [RFC8126].
<table>
<thead>
<tr>
<th>Next Protocol</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0x1</td>
<td>IPv4</td>
<td>This document</td>
</tr>
<tr>
<td>0x2</td>
<td>IPv6</td>
<td>This document</td>
</tr>
<tr>
<td>0x3</td>
<td>Ethernet</td>
<td>This document</td>
</tr>
<tr>
<td>0x4</td>
<td>NSH</td>
<td>This document</td>
</tr>
<tr>
<td>0x5</td>
<td>MPLS</td>
<td>This document</td>
</tr>
<tr>
<td>0x6..0xFD</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>0xFE</td>
<td>Experiment 1</td>
<td>This document</td>
</tr>
<tr>
<td>0xFF</td>
<td>Experiment 2</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 7: NSH Base Header Next Protocol Values

Expert Review requests MUST include a single code point per request. Designated Experts evaluating new allocation requests from this registry should consider the potential scarcity of code points for an 8-bit value, and check both for duplications as well as availability of documentation. If the actual assignment of the Next Protocol field allocation reaches half of the range, that is when there are 128 unassigned values, IANA needs to alert the IESG. At this point, a new more strict allocation policy SHOULD be considered.

12. NSH-Related Codepoints

12.1. NSH EtherType

An IEEE EtherType, 0x894F, has been allocated for NSH.

13. References

13.1. Normative References


Quinn, et al. Expires May 7, 2018 [Page 37]
13.2. Informative References

[I-D.brockners-proof-of-transit]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C.,
Leddy, J., Youell, S., Mozes, D., and T. Mizrahi, "Proof
of Transit", draft-brockners-proof-of-transit-04 (work in
progress), October 2017.

[I-D.guichard-sfc-nsh-dc-allocation]
Guichard, J., Smith, M., Kumar, S., Majee, S., Agarwal,
P., Glavin, K., Laribi, Y., and T. Mizrahi, "Network
Service Header (NSH) MD Type 1: Context Header Allocation
(Data Center)", draft-guichard-sfc-nsh-dc-allocation-07
(work in progress), August 2017.

[I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol
Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-05 (work
in progress), October 2017.

[I-D.ietf-rtgwg-dt-encap]
Nordmark, E., Tian, A., Gross, J., Hudson, J., Kreeger,
L., Garg, P., Thaler, P., and T. Herbert, "Encapsulation
Considerations", draft-ietf-rtgwg-dt-encap-02 (work in
progress), October 2016.

[I-D.ietf-sfc-control-plane]
Boucadair, M., "Service Function Chaining (SFC) Control
Plane Components & Requirements", draft-ietf-sfc-control-
plane-08 (work in progress), October 2016.

[I-D.ietf-sfc-oam-framework]
Aldrin, S., Pignataro, C., Kumar, N., Akiya, N., Krishnan,
R., and A. Ghanwani, "Service Function Chaining (SFC)
Operation, Administration and Maintenance (OAM)
Framework", draft-ietf-sfc-oam-framework-03 (work in
progress), September 2017.
[I-D.napper-sfc-nsh-broadband-allocation]

[I-D.reddy-sfc-nsh-encrypt]


Authors’ Addresses

Paul Quinn (editor)
Cisco Systems, Inc.

Email: paulq@cisco.com

Uri Elzur (editor)
Intel

Email: uri.elzur@intel.com

Carlos Pignataro (editor)
Cisco Systems, Inc.

Email: cpignata@cisco.com
Abstract

This document defines the use of a structured application information in the service function chaining metadata, and specifies a YANG model for the configuration of the application registry.

The consumers of application information are Service Functions that apply policy and provide application statistics based on the metadata contained in the packet.

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 February 16, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
1. Open Issues

1. Relationship of this YANG module and draft-penno-sfc-yang

2. Any reasons why those attributes are not modeled as boolean: P2P-technology, tunnel-technology, encrypted?

3. The connection between the YANG Model in this document and [I-D.penno-sfc-yang] must be explained

2. Introduction

As described in the Service Function Architecture [RFC7665], Service Functions are provide specific treatment for packets and are part of the end-to-end delivery of services. Many of these network services include application-specific functions, treatments, and optimizations.

The SFC Encapsulation, Network Service Header (NSH), therefore needs to provide with dynamic, flexible, and easily methods to bind service policy to granular traffic information, which includes application information. This is achieved by the ability to carry metadata along
the service function path, which is derived from various sources. (e.g., orchestration systems, DPI Classification, etc.) The consumers of this application information are Service Functions that apply policy and provide application statistics based on this metadata contained in the packet.

This document concerns itself with defining structured application information in the service function chaining metadata.

The "Cisco Systems Export of Application Information in IP Flow Information Export (IPFIX) [RFC6759] specifies an extension to the IPFIX information model [RFC7012] to export application information. This IPFIX information element is registered as the identifier 95 in the IPFIX registry [IANA-IPFIX]. Applications could be identified at different OSI layers, from layer 2 to layer 7. For example, the Link Layer Distribution Protocol [LLDP] can be identified in layer 2, ICMP can be identified in layer 3 [IANA-PROTO], HTTP can be identified in layer 4 [IANA-PORTS], and Webex can be identified in layer 7. However, the layer 7 application registry values are out of scope of [RFC6759]

This document purposes the use of IPFIX [RFC7011] application information to be carried in the NSH MD-Type 1 context metadata [I-D.ietf-sfc-nsh]. Optionally, encoding for NSH MD-Type 2 is provided with the Application ID TLV [I-D.quinn-sfc-nsh-tlv]. The information in the metadata will be provided by an orchestration system or the result of packet processing done by a firewall, Intrusion Protection Service (IPS), Deep Packet Inspection (DPI), amongst others. These are defined as providers of application information.

2.1. Terminology

The reader should be familiar with the terms contained in the following documents:

- Section 1.4 of the "Service Function Chaining (SFC) Architecture" [RFC7665]
- Section 2.1 of the "Network Service Header" [I-D.ietf-sfc-nsh]
- The "Generic Protocol Extension for VXLAN" [I-D.ietf-nvo3-vxlan-gpe]
- Sections 3 and 3.1 of "Cisco Systems Export of Application Information in IP Flow Information Export (IPFIX)" [RFC6759]
2.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), "ro" state data (read-only), "-x" RPC operations, and "-n" notifications.
- Symbols after data node names: "?" means an optional node, "!" a container with presence, and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":")
- Ellipsis ("...") stands for contents of subtrees that are not shown.

3. Application Information Structure

The application information data structure can be seen in Figure 1. It was extracted and adapted from [RFC6759].

```
+---------------+-----------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+
| Class. Eng. ID|    Zero-valued upper-bits ... Selector ID     |
+---------------+-----------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+
```

Figure 1: Application Identification Data Format

Table 1 displays the currently allocated Classification Engine IDs, including their name and value, as well as their corresponding Selector ID default length.
<table>
<thead>
<tr>
<th>Classification Engine ID Value</th>
<th>Classification Engine ID Name</th>
<th>Selector ID default length (in octets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IANA-L3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>PANA-L3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>IANA-L4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>PANA-L4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>USER-Defined</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>PANA-L2</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>PANA-L7</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>ETHERTYPE</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>LLC</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>PANA-L7-PEN</td>
<td>3 (*)</td>
</tr>
</tbody>
</table>

Table 1: Existing Classification Engine IDs

Where:

"PANA = Proprietary Assigned Number Authority". In other words, an enterprise specific version of IANA for internal IDs.

PEN = Private Enterprise Number

(*) There are an extra 4 bytes for the PEN. However, the PEN is not considered part of the Selector ID.

Section 6 of [RFC6759] provides various illustrative examples of the encoding for different applications.

4. Application Information Yang Model

4.1. Module Structure
module: ietf-ipfix-application-information
    +--rw class-id-dictionary
        |   +--rw class-id* [name]
        |       |   +--rw id?        uint8
        |       |   +--rw name       string
        |       |   +--rw description? string
    +--rw application-id-dictionary
        +--rw application-id* [application-name]
        +--rw class-id      -> /class-id-dictionary/class-id/id
        +--rw class-id       -> /class-id-dictionary/class-id/id
        +--rw pen           uint32
        +--rw selector-id   uint32
        +--rw application-name string
        +--rw application-description? string
        +--rw application-category-name? string
        +--rw application-sub-category-name? string
        +--rw application-group-name? string

4.2. Application Information Configuration Module

<CODE BEGINS> file "ietf-ipfix-application-information@2015-04-28.yang"

module ietf-ipfix-application-information {
    yang-version 1;

    namespace "urn:ietf:params:xml:ns:yang:
        + "ietf-ipfix-application-information";

    prefix ipfix-app-info;

    organization
        "IETF SFC (Service Function Chaining) Working Group";

    contact
        "Editor: Christophe Fontaine
            christophe.fontaine@qosmos.com

        Editor: Reinaldo Penno
            rapenno@gmail.com";

    description
        "This module contains a collection of YANG definitions for
         the configuration of application ids.

        Copyright (c) 2015 IETF Trust and the persons identified as
        authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
typedef application-id-ref {
  type leafref {
    path "/ipfix-app-info:application-id-dictionary/"
    + "ipfix-app-info:application-id/ipfix-app-info"
    + ":application-name";
  }
  description "This type is used by data models that need
to reference an application-id";
}

typedef classification-engine-id {
  type enumeration {
    enum "IANA-L3" {
      value 1;
      description
      "IANA-L3";
    }
    enum "PANA-L3" {
      value 2;
      description
      "PANA-L3";
    }
    enum "IANA-L4" {
      value 3;
      description
      "IANA-L4";
    }
    enum "PANA-L4" {
      value 4;
      description
      "PANA-L4";
    }
  }
}
enum "USER-Defined" {
    value 6;
    description "USER-Defined";
}
enum "PANA-L2" {
    value 12;
    description "PANA-L2";
}
enum "PANA-L7" {
    value 13;
    description "PANA-L7";
}
enum "ETHERTYPE" {
    value 18;
    description "ETHERTYPE";
}
enum "LLC" {
    value 19;
    description "LLC";
}
enum "PANA-L7-PEN" {
    value 20;
    description "PANA-L7-PEN";
}

description "The definitions for Classification engine ID names.";
reference "RFC 6759: Cisco Systems Export of Application Information in IP Flow Information Export (IPFIX)";

/*
 * Configuration data nodes
 */

class-id-dictionary {
    description "Dictionary for classification ids";
    class-id {
        key "name";
        unique "id";
        id {
            type uint8;
        }
    }
    id {
        type uint8;
    }
}
description "Classification identifier";
}
leaf name {
  type string;
description "classification Engine name";
}
leaf description {
  type string;
description "Description of the class-id";
}
description "A list of all classification ids";
}
}
container application-id-dictionary {
  description "Dictionary for application ids";
  list application-id {
    key "application-name";
    unique "class-id pen selector-id";
    leaf class-id {
      type leafref {
        path "/ipfix-app-info:class-id-dictionary/
         + "ipfix-app-info:class-id/ipfix-app-info:id";
      }
    }
    mandatory true;
    description "Application Name";
  }
  leaf pen {
    type uint32;
    mandatory true;
    description "Private Enterprise Number, only relevant when used with appropriate class-id. Set to 0 when not used.";
  }
  leaf selector-id {
    type uint32 {
      range "0..16777216";
    }
    mandatory true;
    description "Selector identifier";
  }
  leaf application-name {
    type string;
    mandatory true;
    description "The name of the application";
  }
  leaf application-description {
    type string;
  }
  }
}

description "The description of the application";
}
leaf application-category-name {
  type string;
  description "An attribute that provides a first-level categorization for each Application ID. Examples include browsing, email, file-sharing, gaming, instant messaging, voice-and-video, etc. The category attribute is encoded by the application-category-name Information Element";
}
leaf application-sub-category-name {
  type string;
  description "An attribute that provides a second-level categorization for each Application ID. Examples include backup-systems, client-server, database, routing-protocol, etc. The sub-category attribute is encoded by the application-sub-category-name Information Element";
}
leaf application-group-name {
  type string;
  description "An attribute that groups multiple Application IDs that belong to the same networking application. For example, the ftp-group contains ftp-data (port 20), ftp (port 20), ni-ftp (port 47), sftp (port 115), bftp (port 152), ftp-agent (port 574), ftps-data (port 989). The application-group attribute is encoded by the application-group-name Information Element";
}

description "A list of all applications";
}
5. Service Function Chaining Metadata

When a Deep Packet Inspection (DPI), Firewall or any other Service Function (SF) that can identify applications want to convey this knowledge to other SFs it encoded in the format discussed earlier and add to the context metadata.

As defined in [I-D.ietf-sfc-nsh], there are two formats for the NSH Metadata, or the portion of the NSH header beyond the mandatory Base Hader and Service Path Header: MD-Type 1 and MD-Type 2.

The Application Identification data structure (see Figure 1) can be carried both in MD-Type 1 and MD-Type 2. This document specifies the encoding within NSH MD-Type 1 (see Figure 2), and encoding for NSH MD-Type 2 is provided with the Application ID TLV [I-D.quinn-sfc-nsh-tlv].

The Example in Figure 2 shows the encoding of the SNMP application using MD-Type 1.

```
0                   1                   2                   3
+-----------------+------------------+------------------+------------------+
|       3       |        0      |              161              |
|-----------------|-----------------|-----------------|-----------------|
|                  Network Shared Context
|                  Service Platform Context
|                  Service Shared Context
|                  Service Shared Context
```

Figure 2: Example of Metadata Including the SNMP Application Identification

In this example, the Classification Engine IDs of 3 indicates "IANA-L4", and 161 is the well-known port number for SNMP (with its upper bits zero-valued).

Other Services Functions that need application information associated with a packet or flow can look at this metadata (encoded in either MD-Type 1 or MD-Type 2) and easily find out its value.

6. Relationship to existing YANG Modules

[RFC6728] specifies a data model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) protocols. It is for configuring and monitoring Selection Processes, Caches, Exporting Processes, and
Collecting Processes of IPFIX- and PSAMP-compliant Monitoring Devices using the Network Configuration Protocol (NETCONF). The data model is defined using UML (Unified Modeling Language) class diagrams and formally specified using YANG.

The YANG model is this document allows the configuration of the application id IPFIX information elements (ieId), which in turn, may be used in a template definition (TemplateId).

[I-D.penno-sfc-yang] To be done

7. Expected Usage

Devices or controllers will download the [ETHERTYPE], [IANA-PROTO] and [IANA-PORTS] from the appropriate URIs. However, the configuration of the applications is required for applications not registered in an industry-wide agreed-upon registry. In this case, the Proprietary Assigned Number Authority (PANA) registries (PANA-L2, PANA-L3, PANA-L4, PANA-L7), or the User-Defined registry, must be used to identify new application.

Furthermore, the following attributes are statically assigned per Application ID, and needs to be configured: category, sub-category, application-group.

8. IANA Considerations

TBD

9. Security Considerations

TODO: Update with privacy and security considerations, as requested in Prague IETF93.

10. Acknowledgements

The authors wish to thank Kengo Naito for a thorough review and insightful comments.

11. Changes

12. Informative References

[ETHERTYPE]
"ETHERTYPE", 1984,
[I-D.ietf-nvo3-vxlan-gpe]

[I-D.ietf-sfc-nsh]

[I-D.penno-sfc-yang]

[I-D.quinn-sfc-nsh-tlv]

[IANA-IPFIX]

[IANA-PORTS]

[IANA-PROTO]


Authors' Addresses

Reinaldo Penno
Cisco Systems, Inc.
170 West Tasman Dr
San Jose CA
USA

Email: repenno@cisco.com

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Email: bclaise@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
7200-12 Kit Creek Road
Research Triangle Park, NC 27709-4987
USA

Email: cpignata@cisco.com
Abstract

This document describes requirements for conveying information between Service Function Chaining (SFC) control elements (including management components) and SFC functional elements. Also, this document identifies a set of control interfaces to interact with SFC-aware elements to establish, maintain or recover service function chains. This document does not specify protocols nor extensions to existing protocols.

This document exclusively focuses on SFC deployments that are under the responsibility of a single administrative entity. Inter-domain considerations are out of scope.

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 December 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the
document authors. All rights reserved.

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

Table of Contents

1. Introduction ............................................... 3
   1.1. Scope ................................................. 4
   1.2. Terminology .......................................... 5
   1.3. Assumptions .......................................... 5
2. Generic Considerations ...................................... 6
   2.1. Generic Requirements ................................ 6
   2.2. SFC Control Plane Bootstrapping .................... 6
   2.3. Coherent Setup of an SFC-enabled Domain ........... 7
3. SFC Control Plane Components & Interfaces .................. 8
   3.1. Reference Architecture .............................. 8
   3.2. Centralized vs. Distributed ......................... 9
   3.3. Interface Reference Points ........................... 10
      3.3.1. C1: Interface between SFC Control Plane & SFC
             Classifier ..................................... 10
      3.3.2. C2: Interface between SFC Control Plane & SFF ... 12
      3.3.3. C3: Interface between SFC Control Plane & SFC-aware
              SFs ........................................... 12
      3.3.4. C4: Interface between SFC Control Plane & SFC Proxy . 13
4. Additional Considerations .................................. 14
   4.1. Discovery of the SFC Control Element ............... 14
   4.2. SF Symmetry .......................................... 14
1. Introduction

The dynamic enforcement of a service-derived forwarding policy for packets entering a network that supports advanced Service Functions (SFs) has become a key challenge for operators. Typically, many advanced Service Functions (e.g., Performance Enhancement Proxies ([RFC3135]), NATs [RFC3022][RFC6333][RFC6146], firewalls [I-D.ietf-opsawg-firewalls], etc.) are solicited for the delivery of value-added services, particularly to meet various service objectives such as IP address sharing, avoiding covert channels, detecting and protecting against ever increasing Denial-of-Service (DoS) attacks, etc.

Because of the proliferation of such advanced service functions together with complex service deployment constraints that demand more agile service delivery procedures, operators need to rationalize their service delivery logics and master their complexity while optimising service activation time cycles. The overall problem space is described in [RFC7498]. A more in-depth discussion on use cases
can be found in [I-D.ietf-sfc-use-case-mobility] and [I-D.ietf-sfc-dc-use-cases].

[I-D.ietf-sfc-architecture] presents a model addressing the problematic aspects of existing service deployments, including topological dependence and configuration complexity. It also describes an architecture for the specification, creation, and ongoing maintenance of Service Function Chains (SFC) within a network. That is, how to define an ordered set of Service Functions and ordering constraints that must be applied to packets and/or frames and/or flows selected as a result of classification.

1.1. Scope

While [I-D.ietf-sfc-architecture] focuses on data plane considerations, this document describes requirements for conveying information between SFC control elements (including management components) and SFC functional elements. Also, this document identifies a set of control interfaces to interact with SFC-aware elements to establish, maintain or recover service function chains.

Both distributed and centralized control plane schemes to install SFC-related state and influence forwarding policies are discussed.

This document does not make any assumption on the deployment use cases. In particular, the document implicitly covers fixed, mobile, data center networks and any combination thereof.

This document does not make any assumption about which control protocol to use, whether one or multiple control protocols are required, or whether the same or distinct control protocols will be invoked for each of the control interfaces. It is out of scope of this document to specify a profile for an existing protocol, to define protocol extensions, or to select a protocol.

Considerations related to the chaining of Service Functions that span domains owned by multiple administrative entities are out of scope.

It is out of scope of this document to discuss SF-specific control and policy enforcement schemes; only SFC considerations are elaborated, regardless of the various connectivity services that may be supported in the SFC domain. Likewise, only the control of SFC-aware elements is discussed.

Service catalogue (including guidelines for deriving service function chains) is out of scope.
1.2. Terminology

The reader should be familiar with the terms defined in [RFC7498] and [I-D.ietf-sfc-architecture].

The document makes use of the following terms:

- **SFC data plane functional element**: Refers to SFC-aware Service Function, Service Function Forwarder (SFF), SFC Proxy, or SFC Classifier as defined in the SFC data plane architecture [I-D.ietf-sfc-architecture].

- **SFC Control Element**: A logical entity that instructs one or more SFC data plane functional elements on how to process packets within an SFC-enabled domain.

- **SFC Classification entry**: Refers to an entry maintained by an SFC Classifier that reflects the policies for binding an incoming flow/packet to a given SFC. Actions are associated with matching criteria. For example, packets can be marked with the appropriate SFC-related information to differentiate flows so that subsequent SFFs can forward the flows to a sequence of SFs in a given order. The set of classification entries maintained by a Classifier are referred to as the classification policy table.

- **SFC Forwarding Policy Table**: this table reflects the SFC-specific traffic forwarding policy enforced by SFF components for every relevant incoming packet that is associated to one of the existing SFCs.

  
  [[Note: The question of whether the data plane operates just in terms of SFP IDs or needs SFC IDs, as described in this version of the draft, is still under discussion among the authors.]]

1.3. Assumptions

This document adheres to the assumptions listed in Section 1.2 of [I-D.ietf-sfc-architecture].

This document does not make any assumptions about the co-location of SFC data plane functional elements; this is deployment-specific. This document can accommodate a variety of deployment contexts such as (but not limited to):

- A Service Function Forwarder (SFF) can connect instances of the same or distinct SFs.
- A SF instance can be serviced by one or multiple SFFs.
- One or multiple SFs can be co-located with a SFF.
A boundary 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) can act as an egress node and an ingress node for the same flow.

- Distinct ingress and egress nodes may be crossed by a packet when forwarded in an SFC-enabled domain.
- Distinct ingress nodes may be solicited for each traffic direction (e.g., upstream and downstream).
- An ingress node can embed a Classifier.
- An ingress node may not embed a Classifier, but it can be responsible for dispatching flows among a set of Classifiers.
- The same boundary node may act as an ingress node, an egress node, and also embed a Classifier.
- A Classifier can be hosted in a node that embeds one or more SFs.
- Many network elements within an SFC-enabled domain may behave as ingress/egress nodes.

Furthermore, the following assumptions are made:

- A Control Element can be co-located with a Classifier, SFF or SF.
- One or multiple Control Elements can be deployed in an SFC-enabled domain.
- State synchronization between Control Elements is out of scope.

2. Generic Considerations

2.1. Generic Requirements

For deployments that would require so, SFC forwarding must be allowed even if no control protocols are enabled. Static configuration must be allowed.

A permanent association between an SFC data plane element with a Control Element must not be required; specifically, the SFC-enabled domain must keep on processing incoming packets according to the SFC instructions even during temporary unavailability events of control plane components. SFC implementations that do not meet this requirement will suffer from another flavor of the constrained high availability issue, discussed in Section 2.3 of [RFC7498], supposed to be solved by SFC designs.

2.2. SFC Control Plane Bootstrapping

The interface that is used to feed the SFC control plane with service objectives and guidelines is not part of the SFC control plane itself. Therefore, this document assumes the SFC control plane is provided with a set of information that is required for proper SFC operation with no specific assumption about how this information is
The following information that is likely to be provided to the SFC control plane at bootstrapping includes (non-exhaustive list):

- Locators for Classifiers/SFF/SFs/Proxies, etc.
- SFs serviced by each SFF.
- A list of service function chains, including how they are structured and unambiguously identified.
- Status of each SFC: active/pre-deployment phase/etc. A SFC can be defined at the management level and instantiated in an SFC-enabled domain for pre-deployment purposes (e.g., testing). Actions to activate, modify or withdraw an SFC are triggered by the control plane. Nevertheless, this document does not make any assumption about how an operator instructs the control plane.
- A list of classification guidelines and/or rules to bind flows to SFCs/SFPs.
- Optionally, (traffic/CPU/memory) load balancing objectives at the SFC level or on a per node (e.g., per-SF/SFF/Proxy) basis.
- Security credentials.
- Context information that needs to be shared on a per SFC basis.

Also, the SFC control plane may gather the following information from an SFC-enabled domain at bootstrapping (non-exhaustive list). How this information is collected is left unspecified in this document:

- The list of active SFC-aware SFs (including their locators).
- The list of SFFs and the SFs that are attached to.
- The list of enabled SFC Proxies, and the list of SFC-unaware SFs attached to.
- The list of active SFCs/SFPs as enabled in an SFC-enabled domain.
- The list of Classifiers and their locators, so as to retrieve the classification policy table for each Classifier, in particular.
- The SFC forwarding policy tables maintained by SFFs.

During the bootstrapping phase, a Control Element may detect a conflict between the running configuration in an SFC data plane element and the information maintained by the control plane. Consequently, the control plane undertakes appropriate actions to fix those conflicts. This is typically achieved by invoking one of the interfaces defined in Section 3.3.

2.3. Coherent Setup of an SFC-enabled Domain

Various transport encapsulation schemes and/or variations of SFC header implementations may be supported by one or several nodes of an SFC-enabled domain. For the sake of coherent configuration, the SFC control plane is responsible for instructing all the involved SFC data plane functional elements about the behavior to adopt to select
the transport encapsulation scheme(s), the version of the SFC header to enable, etc.

3. SFC Control Plane Components & Interfaces

3.1. Reference Architecture

The SFC control plane is responsible for the following:

- Build and monitor the service-aware topology. For example, this can be achieved by means of dynamic SF discovery techniques. Those means are out of scope of this document.
- Maintain a repository of service function chains, SFC matching criteria to bind flows to a given service function chain, and mapping between service function chains and SFPs.
- Guarantee the coherency of the configuration and the operation of an SFC-enabled domain.
- Dynamically compute a service-aware forwarding path (distributed model, see Section 3.2)
- Determine a forwarding path in the context of a centralized deployment model (see Section 3.2).
- Update service function chains or adjust SFPs (e.g., for restoration purposes) based on various inputs (e.g., external policy context, path alteration, SF unavailability, SF withdrawal, service decommissioning, etc.).
- Populate SFC forwarding policy tables of involved SFC data plane elements and provides Classifiers with traffic classification rules.

Figure 1 shows the overall SFC control plane architecture, including interface reference points.

This document does not elaborate on the internal decomposition of the SFC Control & Management Plane functional blocks. The components within the SFC Control & Management Planes and their interactions are out of scope.

As discussed in Section 3.2, the SFC control plane can be implemented in a (logically) centralized or distributed fashion.
3.2. Centralized vs. Distributed

The SFC control plane can be (logically) centralized, distributed or a combination thereof. Whether one or multiple SFC Control Elements are enabled is deployment-specific. Nevertheless, the following comments can be made:

SFC management (including SFC monitoring and supervision): is likely to be centralized.

SFC Mapping Rules: i.e., service instructions to bind a flow to a service function chain are likely to be managed by a central SFC Control Element, but the resulting policies can be shared among several Control Elements. Note, these policies can be complemented with local information (e.g., an IPv4 address/IPv6 prefix assigned to a customer) because such information may not be available to the central entity but known only during network attachment phase.

Path computation: can be either distributed or centralized. Distributed path computation means that the selection of the exact
sequence of SF functions that a packet needs to invoke (along with instances and/or SFF locator information) is a result of a distributed path selection algorithm executed by involved nodes. For some traffic engineering proposes, the SFP may be constrained by the control plane; as such, some SFPs can be fully specified (i.e., list all the SFF/SFs that need to be solicited) or partially specified (e.g., exclude some nodes, explicitly select which instance of a given SF needs to be invoked, etc.).

SFC Resiliency (including restoration) refers to mechanisms to ensure high available service function chains. It includes means to detect node/link/path failures. Both centralized and distributed mechanism to ensure SFC resiliency can be envisaged.

Implementing a (logically) centralized path computation engine requires information to be dynamically communicated to the central SFC Control Element, such as the list of available SF instances, SFF locators, load status, SFP availability, etc.

3.3. Interface Reference Points

The following sub-sections describe the interfaces between the SFC Control & Management Planes, as well as various SFC data plane elements

3.3.1. C1: Interface between SFC Control Plane & SFC Classifier

As a reminder, a Classifier is a function that is responsible for classifying traffic based on (pre-defined) rules.

This interface is used to install SFC classification rules in Classifiers. Once classification rules are populated, SFC Classifiers are responsible for binding incoming traffic to service function chains according to these classification rules. Note, the SFC control plane must not make any assumption on how the traffic is to be bound to a given SFC. In other words, classification rules are deployment-specific. For instance, classification can rely on a subset of the information carried in a received packet such as 5-tuple classification, be subscriber-aware, be driven by traffic engineering considerations, or any combination thereof.

The SFC control plane should be responsible for removing invalid (and stale) mappings from the classification tables maintained by the classifiers. Also, local sanity checks mechanisms may be supported locally by the Classifiers, but those are out of scope.
The Classifier may be notified by the control plane about the available SFs (including their locators) or be part of the service function discovery procedure.

Classification rules may be updated, deleted or disabled by the control plane. Criteria that would trigger those operations are deployment-specific.

Given that service function chaining solutions may be applied to very large sets of traffic, any control solution should take scaling issues into consideration as part of the design.

Below are listed some functional objectives for this interface:

- Rationalize the management of classification rules.
- Maintain a global view of instantiated rules in all Classifiers in an SFC-enabled domain.
- Check the consistency of instantiated classification rules within the same Classifier or among multiple Classifier.
- Assess the impact of removing or modifying a classification entry on packets entering an SFC-enabled domain.
- Aggregate classification rules for the sake of performance optimization (mainly reduce lookup delays).
- Adjust classification rules when rules are based on volatile identifiers (e.g., an IPv4 address, IPv6 prefix).
- Allow to rapidly restore SFC states during failure events that occurred at a Classifier (or a Control Element).

The control plane must instruct the Classifier whether it can trust an existing SFC marking of an incoming packet or whether it must be ignored.

For bidirectional packet processing purposes (e.g., full or partial path symmetry), the control plane invokes this interface to configure the appropriate classification entries.

A Classifier can send unsolicited messages through this interface to notify the SFC Control & Management Planes about specific events.

When re-classification is allowed in an SFC-enabled domain, this interface can be used to control Classifiers co-resident with SFC-aware SFs, SFC Proxies, or SFFs to manage re-classification rules.

SFC Classification policy entry should be bound to one single service function chain (or one single SFP); when an incoming packet matches more than one classification entry, tie-breaking criteria should be specified (e.g., priority). Such tie-breaking criteria should be instructed by the control plane.
The identification of instantiated SFCs/SFPs is local to each administrative domain; it is policy-based and deployment-specific.

3.3.2. C2: Interface between SFC Control Plane & SFF

SFFs make traffic forwarding decisions according to the entries maintained in their SFC forwarding policy table. Such table is populated by the SFC control plane through the C2 interface.

This interface is used to instruct a SFF about the SFC-aware SFs that it can service. This interface is also used by the SFF to report the connectivity to their attached (including embedded) SFs. Local means may be enabled between the SFC-aware SFs and SFFs to allow for the dynamic attachment of SFs to a SFF and/or discovery of SFs by a SFF but those means are unspecified in this document.

The C2 interface is also used for collecting states of attributes (e.g., availability, workload, latency), for example, to dynamically adjust Service Function Paths.

3.3.3. C3: Interface between SFC Control Plane & SFC-aware SFs

The SFC control plane uses this interface to interact with SFC-aware SFs.

SFs may need to output some processing results of packets to the SFC control plane. This information can be used by the SFC control plane to update the SFC classification rules and the SFC forwarding policy table entries.

This interface is used to collect such kind of feedback information from SFs. For example, the following information can be exchanged between a SF and the SFC control plane:

- SF execution status: Some SFs may need to send information to the control plane to fine tune SFPs. For example, a threat-detecting SF can periodically send the threat characteristics via this interface, such as high probability of threat with packet of a given size. The control plane can then add an appropriate matching criteria to SFF to steer traffic to a scrubbing center.

- SF load update: When SFs are under stress that yielded the crossing of some performance thresholds, the SFC control plane needs to be notified to adjust SFPs accordingly (especially when the centralized path computation mode is enabled). It is out of scope of this document to specify the exact methods to monitor the performance threshold or stress level of SFs, nevertheless the SFC control plane can invoke those methods for its operations.
The SFC control needs the above status information for various tasks it undertakes, but this information may be acquired directly from SFs or indirectly from other management and control systems in the operational environment.

This interface is also used to instruct an SFC-aware SF about any context information it needs to supply in the context of a given SFC.

Also, this interface informs the SFC-aware SF about the semantics of a context information, which would otherwise have opaque meaning. Several attributes may be associated with a context information such as (but not limited to) the "scope" (e.g., per-packet, per-flow or per host), whether it is "mandatory" or "optional" to process flows bound to a given chain, etc. Note that a context may be mandatory for "chain 1", but optional for "chain 2".

The control plane may indicate, for a given service function chain, an order for consuming a set of contexts supplied in a packet.

A SFC-aware SF can also be instructed about the behavior it should adopt after consuming a context information that was supplied in the SFC header. For example, the context can be maintained or stripped. The SFC-aware SF can be instructed to inject a new context header into the SFC header.

Multiple SFs may be located within the same physical node, and no SFF is enabled in that same node, means to unambiguously forward the traffic to the appropriate SF must be supported.

An SF can be instructed to strip the SFC information for the chains it terminates.

3.3.4. C4: Interface between SFC Control Plane & SFC Proxy

The SFC control plane uses this interface to interact with an SFC Proxy.

The SFC proxy can be instructed about authorized SFC-unaware SFs it can service. A SFC Proxy can be instructed about the behavior it should adopt to process the context information that was supplied in the SFC header on behalf of a SFC-unaware SF, e.g., the context can be maintained or stripped.

The SFC proxy is also instructed about the semantics of a context information, which would otherwise have opaque meaning. Several attributes may be associated with a context information such as (but not limited to) the "scope" (e.g., per-packet, per-flow or per host),
whether it is "mandatory" or "optional" to process flows bound to a
given chain, etc.

The SFC Proxy can also be instructed to add SF some new context
information into the SFC header on behalf of a SFC-unaware SF.

The C4 interface is also used for collecting attribute states (e.g.,
availability, workload, latency), for example, to dynamically adjust
Service Function Paths.

4. Additional Considerations

4.1. Discovery of the SFC Control Element

SFC data plane functional elements need to be provisioned with the
locators of the Control Elements. This can be achieved using a
variety if mechanisms such as static configuration or the activation
of a service discovery mechanism. The exact specification of how
this provisioning is achieved is out of scope.

4.2. SF Symmetry

Some SFs require both directions of a flow to traverse. Some service
function chains require full symmetry. If a SF (e.g., stateful
firewall or NAT) needs both direction of a flow, it is the SF
instantiation that needs both direction of a flow to traverse, not
the abstract SF (which can have many instantiations spread across the
network).

4.3. Pre-deploying SFCs

Enabling service function chains should preserve some deployment
practices adopted by Operators. Particularly, installing a service
function chain (and its associated SFPs) should allow for pre-
deployment testing and validation purposes (that is a restricted and
controlled usage of such service function chain (and associated
SFPs)).

4.4. Withdraw a Service Function (SF)

During the lifetime of a SFC, a given SF can be decommissioned. To
accommodate such context and any other case where a SF is to be
withdrawn, the control plane should instruct the SFC data plane
functional element about the behavior to adopt. Particularly:

1. a first approach would be to update the service function chains
   (and associated SFPs) where that SF is present by removing any
reference to that SF. Doing so avoids to induce service failures for end users.

2. a second approach would be to delete/deactivate any service function chain (and its associated SFPs) that involves that SF but install new service function chains.

4.5. SFC/SFP Operations

Various actions can be executed on a service function chain (and associated SFPs) that is structured by the SFC Control & Management planes. Indeed, a service function chain (and associated SFPs) can be enabled, disabled, its structure modified by adding a new SF hop or remove an SF from the sequence of SFs to be invoked, its classification rules modified, etc.

A modification of a service function chain can trigger control messages with the appropriate SFC-aware nodes accordingly.

4.6. Unsolicited (Notification) Messages

Involved SFC data plane functional element must be instructed to send unsolicited notifications when loops are detected, a problem in the structure of a service function chain is encountered, a long unavailable forwarding path time is observed, etc.

Specific criteria to send unsolicited notifications to a Control Element should be fine tuned by the control plane using the interface defined in Section 3.3.

4.7. SF Liveness Detection

The control plane must allow to detect the liveliness of SFs of an SFC-enabled domain. In particular, it must allow to dynamically detect that a SF instance is out of service and notify the relevant Control Element elements accordingly. The liveness information may be acquired directly from SFs or indirectly from other management and control systems in the operational environment.

Liveness status records for all SF instances, and service function chains (including the SFPs bound to a given chain) are maintained by the SFC Control & Management.

The Classifier may be notified by the control plane or be part of the liveness detection procedure.

The ability of a SFC Control Element to check the liveness of each SF present in service function chain has several advantages, including:
o Enhanced status reporting by the control & management planes (i.e., an operational status for any given service chain derived from liveness state of its SFs).

o Ability to support various resiliency policies (i.e., bypass a node embedding an SF, use alternate node, use alternate chain, drop traffic, etc.)

o Ability to support load balancing capabilities to solicit multiple SF instances that provide equivalent functions.

Because a node embedding a SF can be responsive from a reachability standpoint (e.g., IP level) while the function its provides may be broken (e.g., a NAT module may be down), additional means to assess whether an SF is up and running are required. These means may be service-specific.

4.8. Monitoring & Counters

SFC-specific counters and statistics must be provided using of the interfaces defined in Section 3.3. These data include (but not limited to):

o Number of flows ever and currently assigned to a given service function chain and a given SFP.

o Number of flows, packets, bytes dropped due to policy.

o Number of packets and bytes in/out per service function chain.

o Number of flows, packets, bytes dropped due to unknown service function chain (this is valid in particular for a SF node).

4.9. SFC/SFP Diagnosis

[Note: This section is expected to be removed once the working group adopts a document on OAM.]

The Control & Management planes should allow for the following:

o Assess the status of the serviceability of a SF (i.e., the SF provides the service(s) it is configured for). Obviously, this assessment must not rely only on IP reachability to decide whether a SF is up and running.

o Diagnose the availability of a SFC (including the availability of a particular SFP bound to a given SFC).

o Retrieve the set of service function chains that are enabled within a domain.

o Assess whether an SFC-enabled domain is appropriately configured (including, check the configured chains are matching what should be configured in that domain, and ensure coherent classification rules are installed in and enforced by all the Classifiers of the SFC-enabled domain).
Correlate classification policies with observed forwarding actions (including, assess the output of the classification rule applied on a packet presented to a Classifier of an SFC-enabled domain).

Support the correlation between a service function chain and the actual forwarding path followed by a packet matching that service function chain.

Notify the SFC Control Element whenever some (critical) events occur (for example, a malfunctioning SF instance).

Re-use SF built-in diagnostic procedures specific to each SF.

The SFC control plane must be able to invoke SFC OAM mechanisms, and to determine the results of OAM operations.

4.10. Validity Lifetime

SFC instructions communicated via the various interfaces introduced in Section 3.3 may be associated with validity lifetimes, in which case classification entries will be automatically removed upon the expiry of the validity lifetime without requiring an explicit action from a Control Element.

Lifetimes are used in particular by an SFC data plane element to clear invalid control entries that would be maintained in the system if, for some reason, no appropriate action was undertaken by the control plane to clear such entries.

Both short and long lifetimes may be assigned.

4.11. Considerations Specific to the Centralized Path Computation Model

This section focuses on issues that are specific to the centralized deployment model (Section 3.2).

4.11.1. Service Function Path Adjustment

A SFP is determined by composing SF instances and overlay links among SFFs. Thus, the status of a SFP depends on the states or attributes (e.g., availability, topological location, latency, workload, etc.) of its components. For example, failure of a single SF instance results in failure of the whole SFP. Since these states or attributes of SFP components may vary in time, their changes should be monitored and SFPs should be dynamically adjusted.

Examples of use cases for SFP adjustment are listed below:

SFP fail-over: re-construct a SFP with replacing the failed SF instance with another instance of the same SF.
SFP with better latency experience: re-construct a SFP with a low path stretch considering the changes in topological locations of SF instances and the latency induced by the (overlay) connectivity among SFFs.

Traffic engineered SFC: re-construct SFFs to localize the traffic in the network considering various TE goals such as bypass a node, bypass a link, etc. These techniques may be used for planned maintenance operations on a SFC-enabled domain.

SF/SFC Load balancing: re-construct SFPs to distribute the workload among various SF instances.

For more details about the use cases, refer to [I-D.lee-nfvrg-resource-management-service-chain].

The procedures for SFP adjustment may be handled by the SFC control plane as follows:

- Collect and monitor states and attributes of SF instances and overlay links via the C2 interface (Section 3.3.2) and the C3 interface (Section 3.3.3).
- Evaluate SF instances and overlay links based on the monitoring results.
- Select SF instances to re-determine a SFP according to the evaluation results.
- Replace target SF instances (e.g., in a failure or overladed) with newly selected ones.
- Enforce the updated SFP for upcoming SFC traversal to SFFs via the C1 interface (Section 3.3.1) or the C2 interface (Section 3.3.2).

4.11.2. Head End Initiated SFP Establishment

In some scenarios where a SFC Control Element is not connected to all SFFs in a SFC-enabled domain, the SFC control plane can send the explicit SFF-SF-sequence or SF-sequence to the SFC head-end, e.g., the SFC Classifier via the C1 interface (Section 3.3.1). SFC head-end can use a signaling protocol to establish the SFF-SF-sequence based on the SF-sequence.

4.11.3. (Regional) Restoration of Service Functions

There are situations that it might not be feasible for the Classifier to be notified of the changes of SFF-sequence or SFF-SF-Sequence for a given SFP because of the time taken for the notification and the limited capability of the Classifiers.

If a SF has a large number of instantiations, it scales better if the Classifier doesn’t need to be notified with status of visible instantiations of SFs on a SFP.
It might not be always feasible for the Classifier to be aware of the exact SF instances selected for a given SFP due to too many instances for each SF, notifications not being promptly sent to the Classifier, or other reasons. This is about multiple instances of the same SF attached to one SFF node; those instances can be handled by the SFF via local load balancing schemes.

Regional restoration can take the similar approach as the global restoration: choosing a regional ingress node that can take over the responsibility of installing the new steering policies to the involved SFFs or network nodes. Typically, the regional ingress node should be:

- on the data path of the flow of the given SFC;
- in front of the relevant SFFs or network nodes that are impacted by the change of the SFP;
- capable of encoding the detailed SFP to the Service Chain Header of data packets of the identified flow; and
- capable of removing the detailed SFP encoding in data packets after all the impacted SFFs and network nodes completed the policy installation.

5. Security Considerations

5.1. Secure Communications

The SFC Control Elements and the participating SFC data plane elements must mutually authenticate. SFC data plane elements must ignore instructions received from unauthenticated SFC Control Elements. The credentials details used during authentication can be used by the SFC control plane to decide whether specific authorization may be granted to a Service Function with regards to some specific operations (e.g., authorize a given SF to access specific context information).

In case multiple SFC data plane elements are embedded in the same node, the authentication mechanism may be executed as a whole; not for each instance.

A SFC data plane element must be able to send authenticated unsolicited notifications to a SFC Control Element.

The communication between a Control Element and SFC data plane elements must provide integrity and replay protection.

An SFC Control Element may instruct a Service Function to include specific security token(s) that may be used to decrypt traffic upstream. The security token may be supplied by the SFC control...
plane or by an authorized Service Function (e.g., TLS proxy). The exact details on how authorization is granted to a specific SF, including via a control plane interface, should be specified.

A Service Function must by default discard any action from a SFC Control Element that requires specific right privileges (e.g., access to a legal intercept log, mirror the traffic, etc.).

5.2. Pervasive Monitoring

The authentication mechanism should be immune to pervasive monitoring [RFC7258]. An attacker can intercept traffic by installing classification rules that would lead to redirect all or part of the traffic to an illegitimate network node. Means to protect against attacks that would lead to install, remove, or modify classification rules must be supported.

5.3. Privacy

The SFC control plane must be able to control the information that is leaked outside an SFC-enabled domain. Particularly, the SFC control plane must support means to preserve privacy [RFC6973]. Context headers may indeed reveal privacy information (e.g., IMSI, user name, user profile, location, etc.). Those headers must not be exposed outside the operator’s domain. Also, means to protect context headers from eavesdroppers should be enforced.

5.4. Denial-of-Service (DoS)

In order to protect against denial of service that would be caused by a misbehaving trusted SFC Control Element, SFC data plane elements should rate limit the messages received from an SFC Control Element.

5.5. Illegitimate Discovery of SFs and SFC Control Elements

Means to defend against soliciting illegitimate SFs/SFFs that do not belong to the SFC-enabled domain must be enabled. Such means must be defined in service function discovery and SFC Control Element discovery specification documents.

6. IANA Considerations

This document does not require any IANA actions.
7. Acknowledgements

This document is the result of merging with [I-D.lee-sfc-dynamic-instantiation].

The authors would like to thank Shibi Huang for providing input and LAC Chidung for his review and comments that helped improve this document.

The text about the semantic of a context information is provided by Dave Dolson.

8. References

8.1. Normative References

[I-D.ietf-sfc-architecture]

8.2. Informative References

[I-D.ietf-opsawg-firewalls]

[I-D.ietf-sfc-dc-use-cases]

[I-D.ietf-sfc-use-case-mobility]

[I-D.lee-nfvrg-resource-management-service-chain]
Internet-Draft   Control Plane Components & Requirements       June 2015

[I-D.lee-sfc-dynamic-instantiation]
Lee, S., Pack, S., Shin, M., and E. Paik, "SFC dynamic
instantiation", draft-lee-sfc-dynamic-instantiation-01
(work in progress), October 2014.

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

Shelby, "Performance Enhancing Proxies Intended to

[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.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-
Stack Lite Broadband Deployments Following IPv4
Exhaustion", RFC 6333, August 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.

Attack", BCP 188, RFC 7258, May 2014.

Function Chaining", RFC 7498, April 2015.
Appendix A. RSP-related Considerations

This section records some contributions proposed by L. Dunbar and A. Malis, but have not been discussed yet among authors.

A.1. Encoding the Exact SFF-SF-sequence in Data Packets

Encoding the exact RSP in every packet has the benefit and the issues associated with source routing. This approach may not be optimal when the SP doesn’t change very frequently, as in minutes or hours.

There are contexts that it might not be feasible for the head end Classifier to be notified of the changes of SFF-sequence or SFF-SF-Sequence for a given SFP because of the time taken for the notification and the limited capability of the Classifier nodes.

A.2. Fully Controlled SFF-SF-Sequence for a SFP

This section describes the information that can be exchanged over C2 interface (Section 3.3.2) when the SFC Control Element explicitly passes the steering policies to all SFFs for the SFF-SF-Sequence of a given SFC. In this model, each SFF doesn’t need to signal other SFFs for the SFP.

Suppose the SFC ID for this SFP is "yellow", an example of policy to "sff-a" is depicted in Figure 2 (for illustration proposes)

<table>
<thead>
<tr>
<th>Matching</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFC ID = &quot;yellow&quot; &amp; ingress = sffx-port</td>
<td>next-hop: &quot;sf2&quot; &amp; VID</td>
</tr>
<tr>
<td>SFC ID = &quot;yellow&quot; &amp; ingress = sf2-port</td>
<td>next-hop: &quot;sf3&quot; &amp; VID</td>
</tr>
<tr>
<td>SFC ID = &quot;yellow&quot; &amp; ingress = sf3-port</td>
<td>next-hop: sff-b</td>
</tr>
</tbody>
</table>

Figure 2: Example of Traffic Steering Policy to a SFF node

The SFF nodes may not be directly adjacent to each other. They can be interconnected by tunnels, such as GRE, VxLAN, etc. SFs are attached to a SFF node or SFC Proxy node via Ethernet link or other link types. Therefore, the steering policies to a SFF node for service function chain depends on if the packet comes from previous SFF or comes from a specific SF, i.e., the SFC Forwarding Policy Table entries have to be ingress port specific. There are multiple different steering policies for one flow within one SFF and each set of steering policies is specific for an ingress port.

The semantics of traffic steering rules can be "Match" and "Action", similar to the "route" described in [I-D.ietf-i2rs-rib-info-model]. The "match" and "action" for distinct ports can be different. The
matching criteria for SFF can be more sophisticated. For example, the matching criteria could be any fields in the data packets:

- Ingress port
- Destination MAC address
- Source MAC address
- VLAN_id,
- Destination IP address
- Source IP address
- Source port number
- Destination port number
- DSCP
- Packet size, etc., or any combination thereof.

A SFF node may not support some of the matching criteria listed above. It is important that SFC control plane can retrieve the supported matching criteria by SFF nodes. The "Actions" for traffic steering could be to steer traffic to the attached service function or SF instantiations via a specific port.

The "Actions" to SFC Proxy may include a method to map the SFC Identifier carried in the packet header to a locally significant link identifier, e.g., VLAN-ID, and a method to construct and encapsulate the SFC header back to the packets when they come back from the attached SFs.

This approach does not require using an end-to-end signaling protocol among Classier nodes and SFF nodes. However, there may be problems encountered if SFF nodes are not updated in the proper order or not at the same time. For example, if the SFF "A" and SFF "C" get flow steering policies at slightly different times, some packets might not be directed to some service functions on a chain.

Authors’ Addresses

Hongyu Li
Huawei
Huawei Industrial Base, Bantian, Longgang
Shenzhen
China

EMail: hongyu.li@huawei.com