Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: August 1, 2016

S. Homma
K. Naito
NTT
D. R. Lopez
Telefonica I+D
M. Stiemerling
NEC/H-DA
D. Dolson
Sandvine
A. Gorbunov
Nokia
N. Leymann
Deutsche Telekom AG
P. Bottorff
D. Fedyk
HP Enterprise
January 29, 2016

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

Table of Contents

1. 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.

```
*Distribution model of SP information*

      +----------------+
      | Control Entity |
      +----------------+
           ^ |     indication of routing configuration
           | |           based on packet identifiable information
           | +---------------+-------------------------------+--------->
           | |               |                               |
           | |               |                               |
           | v               v                               v
       +--------+        +-------+        +------+       +-------+
------>|   CF   |------> |  FWD  |------> | SF#1 |------>|  FWD  |----->
       +--------+        +-------+        +------+       +-------+

//////////////////////////////////////////////////////////////////////
*Forwarding Tables*

Locate:     [CF]             [FWD]                            [FWD]

Table:   192.168.1.1       192.168.1.1                     192.168.1.1
          ->FWD#1           ->SF#1                          ->SF#2
         10.0.1.1          10.0.1.1                        10.0.1.1
          ->FWD#1           ->FWD#2                         ->SF#2
         ...               ...                             ...

//////////////////////////////////////////////////////////////////////
*Condition of Packet*

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

         +-------+         +-------+        +-------+      +-------+
Packet:  |  PDU  |         |  PDU  |        |  PDU  |      |  PDU  |
         +-------+         +-------+        +-------+      +-------+
```

Figure 1: Forwarding Based on Flow Identifiable Information

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 |
        +----------------+
            ^ |
            | |     indication of
            | |       stacking headers
            | v
         +--------+       +-------+       +------+       +------+
-------->|   CF   |------>| SF#1  |------>| SF#2 |------>| SF#3 |------>
         +--------+       +-------+       +------+       +------+

//////////////////////////////////////////////////////////////////////
*Forwarding Tables*

Locate:       [CF]

Table:    192.168.1.1           __/__/__/__/__/__/__/__/__/__/__/__/__/
           ->Stack #1,2,3       __/ Packets are forwarded to SFs by __/
          10.0.1.1              __/ the outermost header.          __/
           ->Stack #1,3         __/__/__/__/__/__/__/__/__/__/__/__/__/
          ...

//////////////////////////////////////////////////////////////////////
*Condition of Packet*

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

           +--------+
Header:    |To SF#1 |
           +--------+       +--------+
           |To SF#2 |       |To SF#2 |
           +--------+       +--------+     +--------+
           |To SF#3 |       |To SF#3 |     |To SF#3 |
           +--------+       +--------+     +--------+
               :                :              :              :
           +--------+       +--------+     +--------+      +--------+
Packet:    |  PDU   |       |  PDU   |     |  PDU   |      |  PDU   |
           +--------+       +--------+     +--------+      +--------+
```

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
        +--------+        +-------+       +------+       +-------+
  ----->|   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
          ->Stack ID#2
         ...                ...                          ...

  ////////////////////////////////////////////////////////////////////
  *Condition of Packet*

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

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

Figure 3: Forwarding Based on Service Chain Identifiers

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

o 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

o 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

o 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*

           Port A        Port B   Port C        Port D
              |           |          |           |
              |           |          |           |
     .------. |  +[SF1]+  | .------. |  +[SF2]+  | .------.
    /        \v |     |  v/        \v |     |  v/        \
 --#   FWD1   #--+     +--#   FWD2   #--+     +--#   FWD3   #--
    \        /  ^          \        /  ^          \        /
     '------'   |           '------'   |           '------'
                |                      |
. . . . . . . . | . . . . . . . . . . . | . . . . . . . . . . . . .
                |                      |
*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 |   * 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:
  SP#1
    [ CF ]----------->[ SF#1 ]---->[ SF#2 ]----->[ SF#3 ]------->

  Service Chain#2:
  SP#2
    [ CF ]----------->[ SF#4 ]------------------>[ SF#5 ]------->
      :
  Service Chain#n:
  SP#n
    [ CF ]----------->[ SF#6 ]---->[ SF#7 ]----->[ SF#8 ]------->
```

Figure 7: SF Dedicated Model

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

                                          SC:Service Chain  SP:Service Path
 ///////////////////////////////////////////////////////////////////////////
 *Packet Flow*

 Service Chain#1:
 SP#1
   [ CF ]---->[FWD]-->[SF#1]-->[FWD]-->[SF#2_1]-->[FWD]-->[SF#3]--->

 Service Chain#2:
 SP#2
   [ CF ]---->[FWD]-->[SF#1]-->[FWD]-->[SF#2_2]-->[FWD]-->[SF#3]--->
     :
 Service Chain#n:
 SP#n
   [ CF ]---->[FWD]-->[SF#4]---------------------->[FWD]-->[SF#5]--->
```

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

 SP#2
   [ CF ]---->[FWD]-->[SF#1]-->[FWD]-->[SF#2_2]-->[FWD]-->[SF#3]--->
     :
 Service Chain#n:
 SP#m
   [ 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 |
->|     |================*                *=====================>
  |     |    +-----+  |  # (in DC#1)       #  |   +-----+
  |     |             |  V                 #  |
  |Edge |             |+---+            +---+|   Top domain
  | CF  |     * * * * *||CF | * * * * * *|FWD|| * * * * *
  |     |     *        |+---+            +-+-+|         *
  |     |     *        | | |              | | |    Sub  *
  |     |     *       +-o-o--------------o-o-+   domain*
  |     |     *   SC#1.2 | |SC#1.1        ^ ^       #1  *
  |     |     *    +-----+ |              | |           *
  |     |     *    |       V              | |           *
  |     |     *    |      +---+  +------+  | |           *
  |     |     *    |      |FWD|->|SF#1_1|--+ |           *
  |     |     *    |      +---+  +------+    |           *
  |     |     *    V                         |           *
  |     |     *  +---+  +------+  +---+  +------+       *
  |     |     *  |FWD|->|SF#1_2|->|FWD|->|SF#2_1|       *
  |     |     *  +---+  +------+  +---+  +------+       *
    .         * * * * * * * * * * * * * * * * * * * * * *
    .
  |     |    +-----+   +---------------------+     +-----+
  |     |SC#n| FWD |   | IBN#q               |     | FWD |
  |     |=====================================================>
  |     |    +-----+   |     (in DC#m)       |     +-----+
  +----+               +---------------------+
                   (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|
          --->|CF|--/---->'   |          \ +--+
              +--+ /  SC#1    |           \
                   |          |            |
                   |          |     .------>|--->
                   |         /    /        |
                   \         |   /         /
              +--+  \        |  /         /  +--+
              |CF|---\       V /        /---|CF|
              +--+    '------+---------'    +--+
                             |
                        +----+-------+
                        |Sub-domain#2|
                        |   in DC2   |
                        +------------+
```

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 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 methoed 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.

o Common features of method3

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.

o 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.

o 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.

o 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.

o Common features of method3

This method enables FWDs to save resources for managing forwarding tables and allows all SPs 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 SPs 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.

o 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 virtuarizing 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

virtuarizing technique progresses significantly and network operaters can install SFs more freely.

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.

```
                          +--------------+
     . . . . . . . . . . |Control Entity| . . . . .
     .         .         +--------------+         .
     .         .                                  .
* *  . * * * * . * * * * * * * * * * * * * * *  . * * * * * * * * *
*    .         .                                  .                *
*    .         .                                  .                *
*    .      .-----.        .-----------.       .-----.             *
* +----+   /  DC#1 \      /     WAN     \     /  DC#2 \             *
* |    |=====================================================> SP#1 *
* | CF |=====================================================> SP#2 *
*   :                            :                              :   *
* |    |=====================================================> SP#n *
* +----+   \       /      \             /     \       /             *
*           '_____'        '___________'       '_____'              *
*                                                                   *
*                           SC Domain                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

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.
                 +-------------------------------------+
                 |                                     |
                 v                                     v
            +--------+                            +--------+
            |  CE#1  |                            |  CE#2  |
            +--------+                            +--------+
                 .                                     .
    * * * * * *  .  * * * * * *            * * * * * *  .  * * * * * *
    *            .            *            *            .            *
    *     .-------------.     *            *     .------------.      *
    *    /     DC#1      \    *  .------.  *    /     DC#2     \     *
    * +----+          +-----+ * /  WAN   \ * +-----+            |    *
    * |    |=========>|     | * |        | * | CF/ |==========> SP#1 *
    * | CF |=========>| FWD |===============>| FWD |==========> SP#2 *
    *  :       :        :    * |        | *    :         :       :  *
    * |    |=========>|     | * \        / * |     |==========> SP#n *
    * +----+          +-----+ *  '------'  * +-----+            |    *
    *    \              /    *            *     \             /     *
    *     '-------------'     *            *      '-----------'      *
    *       SC Domain#1       *            *      SC Domain#2        *
    * * * * * * * * * * * * * *            * * * * * * * * * * * * * *
```

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.).

```
   +----------+
   |Top-level | . . . . . . . . . . . . . . . . . . . . . . .
   |Control   |                                              .
   |Entity    |          +------------+     +--------+  .
   +----------+          |sub-domain#1|. . .|  CE#1  |  .
        .                +-----+------+     +--------+  .
        .                      |                         .
        .               .------+---------.   +---+      .
        .      +---+   /                  \--|CF |. . . .
        . . . .|CF |--/                    \ |FWD|      .
        .      |FWD| /                     \+---+      .
        .      +---+ |                       |          .
        .            |                       |          .
        .            |                       |          .
        .      +---+ \                      /           .
        .      |CF |  \                    /  +---+     .
        . . . .|FWD|---\                  /---|CF | . . .
               +---+    '------+---------'    |FWD|
                               |              +---+
          +--------+     +------------+
          |  CE#2  |. . .|sub-domain#2|
          +--------+     +-----+------+
```

Figure 14: Multiple Control Entities in Hierarchical Service Chaining

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]
Dolson, D., Homma, S., Lopez, D., Boucadair, M., and D. Liu, "Hierarchical Service Function Chaining", draft-dolson-sfc-hierarchical-04 (work in progress), December 2015.

[I-D.fedyk-sfc-mac-chain]
Bottorff, P., don.fedyk@hpe.com, d., and H. Assarpour, "Ethernet MAC Chaining", draft-fedyk-sfc-mac-chain-01 (work in progress), January 2016.

[I-D.ietf-sfc-dc-use-cases]
Surendra, S., Tufail, M., Majee, S., Captari, C., and S. Homma, "Service Function Chaining Use Cases In Data Centers", draft-ietf-sfc-dc-use-cases-04 (work in progress), January 2016.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-02 (work in progress), January 2016.

[I-D.ietf-sfc-use-case-mobility]
Haeffner, W., Napper, J., Stiemerling, M., Lopez, D., and J. Uttaro, "Service Function Chaining Use Cases in Mobile Networks", draft-ietf-sfc-use-case-mobility-05 (work in progress), October 2015.

[I-D.song-sfc-legacy-sf-mapping]
Song, H., You, J., Yong, L., Jiang, Y., Dunbar, L., Bouthors, N., and D. Dolson, "SFC Header Mapping for Legacy SF", draft-song-sfc-legacy-sf-mapping-06 (work in progress), August 2015.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <http://www.rfc-editor.org/info/rfc3022>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <http://www.rfc-editor.org/info/rfc6146>.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <http://www.rfc-editor.org/info/rfc6296>.

[RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <http://www.rfc-editor.org/info/rfc7498>.

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

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

Martin Stiemerling
NEC Laboratories Europe / Hochschule Darmstadt
Kurfuerstenanlage 36
Heidelberg 69115
Germany

URI: ietf.stiemerling.org

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

Email: ddolson@sandvine.com

Alexey Gorbunov
Nokia
6000 Connection Drive
Irving, Texas 75039
USA

Phone: +1 214 516 11 41
Email: Alexey.gorbunov82@gmail.com

Nicolai Leymann
DT
Winterfeldtstrasse 21-27
Berlin 10781
Germany

Phone: +49 (0)30 835392761
Email: n.leymann@telekom.de

Paul Bottorff
Hewlett Packard Enterprise
8000 Foothills Blvd.
Roseville, CA
USA

Email: paul.bottorff@hpe.com

Don Fedyk
Hewlett Packard Enterprise
153 Taylor Street
Littleton, MA
USA

Email: don.fedyk@hpe.com

Service Function Chaining
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2017

S. Kumar
J. Guichard
P. Quinn
Cisco Systems, Inc.
J. Halpern
Ericsson
S. Majee
F5 Networks

October 20, 2016

Service Function Simple Offloads
draft-kumar-sfc-offloads-03

Abstract

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

This document describes the rationale, techniques and necessary protocol extensions to achieve such optimization, with focus on one such technique termed "simple offloads".

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

While there are several possible means to achieve this, one of the most flexible, directly connected to functional semantics, is based on allowing service functions themselves to evaluate a particular flow and reflect the result of this evaluation back to the SFC infrastructure.

This document outlines the "simple offloads" mechanism that avoids steering traffic to service functions on flow boundary, on request from the service functions, while still ensuring compliance to the instantiated policy that mandates the SFC.

1.1. Requirements Language

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

2. Definition Of Terms

This document uses the following terms. Additional terms are defined in [RFC7498], [I-D.ietf-sfc-architecture] and [I-D.ietf-sfc-nsh].

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

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

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

3. Service Function Path Reduction

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

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

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

- o Bypass: Mechanism that alters the SFC. Described in this draft for completeness.

- o Simple Offload: Mechanism that alters the SFP alone, does not affect the SFC. This is the primary focus of this draft.

3.1. Bypass

Many service functions do not deliver service to certain types of traffic. For instance, typical WAN optimization service functions are geared towards optimizing TCP traffic and add no value to non-TCP traffic. Non-TCP traffic thus can bypass such a service function. Even in the case of TCP, a WAN optimization SF may not be able to service the traffic if the corresponding TCP flow is not seen by it

from inception. In such a situation a WAN optimization SF can avoid the overhead of processing such a flow or reserving resources for it, if it had the ability to request such flows not be steered to it. In other words such service functions need the ability to request they be bypassed for a specified flow from a certain time in the life of that flow.

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

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

- It adds to the overhead of classification at that classifier as all SF classification requirements have to be met by the classifier.

- It leads to conflicts in classification requirements between the classifier and the SFs. Classification needs of different SFs in the same SFC may vary. A classifier thus cannot classify traffic based on the classification of one of the SFs in the chain. For instance, even though a flow is uninteresting to one SF on an SFC, it may be interesting to another SF in the same SFC.

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

3.2. Simple Offload

Service delivery by a class of service functions involves inspecting the initial portion of the traffic and determining whether traffic should be permitted or dropped. In some service functions, such an inspection may be limited to just the five tuple, in some others it may involve protocol headers, and in yet others it may involve inspection of the byte stream or application content based on the policy specified. Firewall service functions fall into such a class, for example. In all such instances, servicing involves determining whether to permit the traffic to proceed onwards or to deny the traffic from proceeding onwards and drop the traffic. In some cases, dropping of the traffic may be accompanied with the generation of a

response to the originator of traffic or to the destination or both. Once the service function determines the result - permit or deny (or drop), it simply applies the same result to the residual packets of the flow by caching the result in the flow state.

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

As with PERMIT and DENY actions, there are others which are simple enough to be supported. Some are listed here for illustration.

Unidirectional Offload: Client-Server communication, typical of HTTP request-response transactions, imposes higher cost on SFs in one direction. Reponses often carry more bytes, sometimes orders of mangnitude more, as compared to requests. SFs could avoid the cost of moving the bits in the response direction to which it may add no value, once the policy is satisfied, if the response flow can be offloaded. Hence Offloads must be requestable on a unidirectional flow boundary.

TCP Control Exception Offload: Most SFs maintain flow state and would like to know when a flow terminates, so SFs can cleanup the flow state and associated resources. Such SFs need to receive the TCP control packets, the ones with control flags [RFC0793] set, on the flow even when the flow itself is offloaded, in order to perform such activity. Hence Offloads must be predicatable to offload all but the TCP control packets of a flow.

Time Limited Offload: SF policy may dictate flows be limited to certain period of time among other reasons to optimize SF load. SFs can request a flow be offloaded for a specific time duration after which, all traffic on that flow gets redirected to the SF as was done before the offload was initiated. Hence Offloads must be requestable on a time limit.

Volume Limited Offload: As with time limited offlaods, SF policy may dictate flows be limited to certain volume of data. SFs can request a flow be offloaded until a specified number of bytes traverse the flow. Hence Offloads must be requestable on a volume limit.

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

3.2.1. Stateful SFF

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

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

3.2.2. Packet Re-ordering

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

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

3.2.3. Race Conditions

The tuple that make up an end-to-end flow or connection, such as a five tuple TCP connection, may be reused in a very short span of time when very high performing end points are involved. A very remote manifestation of this behavior may involve the wrong incarnation of a flow at the SFF receiving the flow offload request from a SF.

Implementations of simple offloads must thus be aware of such a possibility and include appropriate measures to address it. It is important to note that a SFF must maintain correctness and hence it is acceptable to not honor a simple offloads request to resolve such an occurrence. After all SFs exist with right security posture to protect against malicious traffic.

A simple and widely used method to serialize reuse of tuples is to use an incarnation number in addition to the five-tuple. The steering SFF can pass an opaque cookie, which in its simplest form could be the incarnation number, that is preserved by the SF and passed along with the simple offload request. SFF can thus correctly identify the right incarnation of the flow. SYN detection at the SFF to take corrective action is another option. The SFF implementations may employ any technique deemed appropriate.

3.2.4. Policy Implications

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

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

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

3.2.5. Capabilities Exchange

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

Capabilities exchange is outside the scope of this document.

4. Methods For SFP Reduction

There are a number of different models that may be used to facilitate SFP shortening.

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

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

b. Control-plane: Messages are required between the SF and the service controller as well as between the SFF and the service controller. Service controller messaging is out of scope of this document and it is assumed to be service controller specific, which may include open or standard interfaces.

4.1. SFP In-band Offload

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

By performing such hop-by-hop offloads, SFP can be reduced from its original length, steering traffic to only the SFFs and the SFs that really need to see the traffic.

Figure 1 to Figure 3 show an example of SF and SFF performing offload operations, with PERMIT action, and the effect thereafter on the SFP.

```
                 SFID(1)            SFID(2)            SFID(3)
               +------+           +------+           +------+
           ....| SF1  |....   ....| SF2  |....   ....| SF3  |....
           .   +------+   .   .   +------+   .   .   +------+   .
           .      |       .   .      |       .   .      |       .
           .      |       .   .      |       .   .      |       .
           .      |       .   .      |       .   .      |       .
           .      |       .   .      |       .   .      |       .
           .      |       .   .      |       .   .      |       .
    +----+ .   +------+   .   .   +------+   .   .   +------+   .
    | CF |------| SFF1 |-----------| SFF2 |-----------| SFF3 |------ Net
    +----+ .   +------+   .   .   +------+   .   .   +------+   .
           .               .   .               .   .               .
     SFP1 ...              .....               .....              ... >
```

SFC1 = {SF1, SF2, SF3}
SFC1 -> SFP1

Where,
SFC1 is a service function chain
SF1, SF2 and SF3 are three service functions
SFP1 is the servcie function path for SFC1
CF is the classifier starting SFP1 based on policy

Note: Network forwarders are omitted from the figure for simplicity

Figure 1: SFC1 with corresponding SFP1

```
                                 O
                                 f
                 SFID(1)         f +- SFID(2)            SFID(3)
                +------+         l | +------+           +------+
            ....| SF1  |....     o | | SF2  |       ....| SF3  |....
            .   +------+   .     a | +------+       .   +------+   .
            .      |       .     d |    |           .      |       .
            .      |       .       |    |           .      |       .
            .      |       .       |    |           .      |       .
            .      |       .       v    |           .      |       .
            .      |       .            |           .      |       .
    +----+  .   +------+   .       +------+       .   +------+   .
    | CF |------| SFF1 |-----------| SFF2 |-----------|  SFF3 |----- Net
    +----+  .   +------+   .       +------+       .   +------+   .
            .              .                      .              .
     SFP1 ...              ......................               ... >
```

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

```
            O                                       O
            f                                       f
            f +- SFID(1)            SFID(2)         f +- SFID(3)
            l | +------+           +------+        l | +------+
            o | | SF1  |           | SF2  |        o | | SF3  |
            a | +------+           +------+        a | +------+
            d |    |                   |           d |    |
              |    |                   |             |    |
              |    |                   |             |    |
              v    |                   |             v    |
                   |                   |                  |
    +----+      +------+           +------+           +------+
    | CF |------| SFF1 |-----------| SFF2 |-----------| SFF3 |----- Net
    +----+      +------+           +------+           +------+
     SFP1 ............................................................ >
```

Figure 3: SFP1 after SFID(1) and SFID(3) perform Offloads

4.1.1. Progression Of SFP Reduction

SFP reduction happens one SFF at a time: by collapsing the SFF-to-SF hops into the SFF or the SFC infrastructure.

Figure 1 to Figure 3 show one sequence of offload events that lead to a shorter SFP.

Corresponding transformation of the actual forwarding path is captured by the states below.

Stage-1: Prior to any offloads, service function path SFP1 (corresponding to SFC1) has the following actual forwarding path as shown in Figure 1:

```
CF ->
SFF1 -> SF1 -> SFF1 ->
SFF2 -> SF2 -> SFF2 ->
SFF3 -> SF3 -> SFF3 ->
```

Stage-2: After SF2 performs a simple offload, service function path SFP1 changes to the one represented below, as also shown in Figure 2:

```
CF ->
SFF1 -> SF1 -> SFF1 ->
SFF2 ->
SFF3 -> SF3 -> SFF3 ->
```

Stage-3: After SF1 and SF3 both perform simple offloads, service function path SFP1 changes to the one represented below, as also show in Figure 3:

```
CF ->
SFF1 ->
SFF2 ->
SFF3 ->
```

When all the SFs in a SFP perform offloads the forwarding path is reduced to pass through just the SFFs.

4.2. Service Controller Offload

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

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

5. Simple Offload Data-plane Signaling

Since Offload and action are signaled at the time of returning the traffic to SFF, post servicing the traffic, such signaling can be integrated into the SFC service header of the packet.

Figure 4 and Figure 5 show the bits necessary to achieve the signaling using the SFC encapsulation as described in [I-D.ietf-sfc-nsh]. In particular, for NSH MD-Type1 header format, the offload bits are communicated via the flags field in the very first byte of the fixed context headers. For NSH MD-Type2 header format, the offload bits are communicated via a new standard TLV - Simple Offload TLV. The standard TLV is requested to be allocated from the TLV Class, "Standard Class", from the IANA.

By integrating the signaling with the packets, the simple offloads scale with the traffic in the data plane.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|D| F |X|         Context Header 1                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|B|U|T|D|R|R|R|R| Context Header 2                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Context Header 3                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Context Header 4                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

X : Extend flags into first byte of "Context Header 2"
B : Bidirectional Offload
U : Unidirectional Offload
T : TCP-control Exception Offload
D : Drop Offload

Figure 4: NSH Type-1 Offload Bits shown for DC Allocation

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     STANDARD CLASS            | SimpleOffload |0|0|0|   0x2   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|B|U|T|D|S|V|R|R|R|R|R|R|         Offload-data                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

B : Bidirectional Offload
U : Unidirectional Offload
T : TCP-control Exception Offload
D : Drop Offload
S : Time Limited Offload
V : Volume Limited Offload

Figure 5: NSH Type-2 Offload Bits

5.1. Offload Flags Definition

Offload Control Flags:

B, Bidirectional Offload: SF requests both flows in the connection, described by the payload, be offloaded, by setting B=1. B=0 otherwise.

U, Unidirectional Offload: SF requests only the current flow in the connection, described by the payload, be offloaded, by setting U=1. U=0 otherwise.

One and only one of 'B' and 'U' MUST be specified to indicate offload. In the event a NSH encapsulated packet is received with both 'B' and 'U' offload flags set to 1, 'B' MUST take precedence.

Offload Function Flags:

B|U, Permit Offload: When either B=1 or U=1, the implicit function is to PERMIT or allow all packets on the flow(s) to traverse along the SFP, unless over-ridden by other functional flags.

D, Drop Offload: Setting D=1, requests packets on the offloaded flow(s) be dropped; D MUST be set to 0 otherwise. D=1 modifies the default PERMIT behavior of 'B' and 'U' flags.

T, TCP-control Exception Offload: Setting T=1 requests TCP control packets to be exempted from Offload behavior. TCP control packets MUST continue to be forwarded to the SF while the rest of the packets must be allowed to bypass the SF contingent upon the

application of other offload flags. T MUST be set to 0 otherwise.

S, Time Limited Offload: Setting S=1 requests the flow(s) to be offloaded for the duration specified, in seconds, in offload-data field. After that duration, offload behavior must be cancelled and affected flow(s) MUST be redirected to the SF. S MUST be set to 0 otherwise.

V, Volume Limited Offload: Setting V=1 requests the flow(s) to be offloaded until the volume of data specified, in Kilo Bytes, in offload-data field has traversed the flow(s). After that volume of data has traversed, offload behavior must be cancelled and affected flow(s) MUST be redirected to the SF. V MUST be set to 0 otherwise.

6. Acknowledgements

The authors would like to thank Abhjit Patra, Nagaraj Bagepalli, Kent Leung, Erik Nordmark, Diego Lopez for their comments, thoughtful questions and suggestions, review, etc.

7. IANA Considerations

7.1. Standard Class Registry

IANA is requested to allocate a "STANDARD" class from the TLV Class registry. Allocation of the registry values under this class shall follow the "IETF Review" policy defined in RFC 5226 [RFC5226].

7.1.1. Simple Offloads TLV

IANA is requested to allocate TLV type with value 0x1 from the STANDARD TLV class registry. The format of the "Simple Offloads" TLV is as defined in this draft.

TLV#	Name	Description	Reference
1	Simple Offloads	SF Flow Offload to SFF	This document

Table 1: Standard Class Registry

8. Security Considerations

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

9. References

9.1. Normative References

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-11 (work in progress), July 2015.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-10 (work in progress), September 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

9.2. Informative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <http://www.rfc-editor.org/info/rfc7498>.

Authors' Addresses

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

Email: smkumar@cisco.com

Jim Guichard
Cisco Systems, Inc.

Email: jguichar@cisco.com

Paul Quinn
Cisco Systems, Inc.

Email: paulq@cisco.com

Joel Halpern
Ericsson

Email: joel.halpern@ericsson.com

Sumandra Majee
F5 Networks
90 Rio Robles
San Jose, CA
US

Email: S.Majee@F5.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 28, 2015

P. Quinn
J. Guichard
S. Kumar
M. Smith
Cisco Systems, Inc.
W. Henderickx
Alcatel-Lucent
T. Nadeau
Brocade
P. Agarwal

R. Manur
Broadcom
A. Chauhan
Citrix
J. Halpern
Ericsson
S. Majee
F5
U. Elzur
Intel
D. Melman
Marvell
P. Garg
Microsoft
B. McConnell
Rackspace
C. Wright
Red Hat Inc.
K. Glavin
Riverbed
C. Zhang
L. Fourie
Huawei US R&D
R. Parker
Affirmed Networks
M. Zarny
Goldman Sachs
February 24, 2015

Network Service Header
draft-quinn-sfc-nsh-07.txt

Abstract

This draft describes a Network Service Header (NSH) inserted onto encapsulated packets or frames to realize service function paths.

NSH also provides a mechanism for metadata exchange along the instantiated service path.

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].

Status of this Memo

Copyright Notice

Table of Contents

2. 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, campus, and so forth.

The current service function deployment models are 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 requires an agile service insertion model that supports elastic service delivery; the movement of service functions and application workloads in the network and the ability to easily bind service policy to granular information such as per-subscriber state are necessary.

The approach taken by NSH is composed of the following elements:

1. Service path identification

2. Transport independent per-packet/frame service metadata.

3. Optional variable TLV metadata.

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

An NSH aware control plane is outside the scope of this document.

The SFC Architecture document [SFC-arch] 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.

2.1. Definition of Terms

Classification: Locally instantiated policy and customer/network/ service profile matching of traffic flows for identification of appropriate outbound forwarding actions.

SFC Network Forwarder (NF): SFC network forwarders provide network connectivity for service functions forwarders and service functions. SFC network forwarders participate in the network overlay used for service function chaining as well as in the SFC encapsulation.

Service Function Forwarder (SFF): A service function forwarder is responsible for delivering traffic received from the NF to one or more connected service functions, and from service functions to the NF.

Service Function (SF): A function that is responsible for specific treatment of received packets. A service function can act at the network layer or other OSI layers. A service function can be a virtual instance or be embedded in a physical network element. One of multiple service functions can be embedded in the same network element. Multiple instances of the service function can be enabled in the same administrative domain.

Service Node (SN): Physical or virtual element that hosts one or more service functions and has one or more network locators associated with it for reachability and service delivery.

Service Function Chain (SFC): A service function chain defines an ordered set of service functions that must be applied to packets and/or frames selected as a result of classification. The implied order may not be a linear progression as the architecture allows for nodes that copy to more than one branch. The term service chain is often used as shorthand for service function chain.

Service Function Path (SFP): The instantiation of a SFC in the network. Packets follow a service function path from a classifier through the requisite service functions

Network Node/Element: Device that forwards packets or frames based on outer header information. In most cases is not aware of the presence of NSH.

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

Network Service Header: Data plane header added to frames/packets. The header contains information required for service chaining, as well as metadata added and consumed by network nodes and service elements.

Service Classifier: Function that performs classification and imposes an NSH. Creates a service path. Non-initial (i.e. subsequent) classification can occur as needed and can alter, or create a new service path.

Service Hop: NSH aware node, akin to an IP hop but in the service overlay.

Service Path Segment: A segment of a service path overlay.

NSH Proxy: Acts as a gateway: removes and inserts NSH on behalf of a service function that is not NSH aware.

2.2. Problem Space

Network Service Header (NSH) addresses several limitations associated with service function deployments today.

1. Topological Dependencies: Network service deployments are often coupled to network topology. Such dependency imposes constraints on the service delivery, potentially inhibiting the network operator from optimally utilizing service resources, and reduces the flexibility. This limits scale, capacity, and redundancy across network resources.

2. Service Chain Construction: Service function chains today are most typically built through manual configuration processes. These are slow and error prone. With the advent of newer service deployment models the control/management planes provide not only connectivity state, but will also be increasingly utilized for the creation of network services. Such a control/management planes could be centralized, or be distributed.

3. Application of Service Policy: Service functions rely on topology information such as VLANs or packet (re) classification to determine service policy selection, i.e. the service function specific action taken. Topology information is increasingly less viable due to scaling, tenancy and complexity reasons. The topological information is often stale, providing the operator with inaccurate placement that can result in suboptimal resource utilization. Furthermore topology-centric information often does not convey adequate information to the service functions, forcing functions to individually perform more granular classification.

4. Per-Service (re)Classification: Classification occurs at each service function independent from previously applied service functions. More importantly, the classification functionality often differs per service function and service functions may not

leverage the results from other service functions.

5. Common Header Format: Various proprietary methods are used to share metadata and create service paths. An open header provides a common format for all network and service devices.

6. Limited End-to-End Service Visibility: Troubleshooting service related issues is a complex process that involve both network-specific and service-specific expertise. This is especially the case when service function chains span multiple DCs, or across administrative boundaries. Furthermore, the physical and virtual environments (network and service) can be highly divergent in terms of topology and that topological variance adds to these challenges.

7. Transport Dependence: Service functions can and will be deployed in networks with a range of transports requiring service functions to support and participate in many transports (and associated control planes) or for a transport gateway function to be present.

Please see the Service Function Chaining Problem Statement [SFC-PS] for a more detailed analysis of service function deployment problem areas.

3. Network Service Header

A Network Service Header (NSH) contains metadata and service path information that are added to a packet or frame and used to create a service plane. The packets and the NSH are then encapsulated in an outer header for transport.

The service header is added by a service classification function - a device or application - that determines which packets require servicing, and correspondingly which service path to follow to apply the appropriate service.

3.1. Network Service Header Format

An NSH is composed of a 4-byte base header, a 4-byte service path header and context headers, as shown in Figure 1 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 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Base Header                                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Service Path Header                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                Context Headers                                ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Network Service Header

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

Service Path Header: provide path identification and location within a path.

Context headers: carry opaque metadata and variable length encoded information.

3.2. NSH Base Header

```
 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|C|R|R|R|R|R|R|   Length  |    MD Type    | Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: NSH Base Header

Base Header Field Descriptions

Version: The version field is used to ensure backward compatibility going forward with future NSH updates.

O bit: Indicates that this packet is an operations and management (OAM) packet. SFF and SFs nodes MUST examine the payload and take appropriate action (e.g. return status information).

OAM message specifics and handling details are outside the scope of this document.

C bit: Indicates that a critical metadata TLV is present (see Section 3.4.2). This bit acts as an indication for hardware implementers to decide how to handle the presence of a critical TLV without necessarily needing to parse all TLVs present. The C bit MUST be set to 1 if one or more critical TLVs are present.

All other flag fields are reserved.

Length: total length, in 4-byte words, of the NSH header, including optional variable TLVs.

MD Type: indicates the format of NSH beyond the base header and the type of metadata being carried. This typing is used to describe the use for the metadata. A new registry will be requested from IANA for the MD Type.

NSH defines two MD types:

0x1 which indicates that the format of the header includes fixed length context headers.

0x2 which does not mandate any headers beyond the base header and service path header, and may contain optional variable length context information.

The format of the base header is invariant, and not described by MD Type.

NSH implementations MUST support MD-Type 0x1, and SHOULD support MD-Type 0x2.

Next Protocol: indicates the protocol type of the original packet. A new IANA registry will be created for protocol type.

This draft defines the following Next Protocol values:

0x1 : IPv4
0x2 : IPv6
0x3 : Ethernet

3.3. Service Path Header

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Service Path ID                      | Service Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Service path ID (SPI): 24 bits
Service index (SI): 8 bits

Figure 3: NSH Service Path Header

Service Path Identifier (SPI): identifies a service path. Participating nodes MUST use this identifier for path selection. An administrator can use the service path value for reporting and troubleshooting packets along a specific path.

Service Index (SI): provides location within the service path. Service index MUST be decremented by service functions or proxy nodes after performing required services. MAY be used in conjunction with service path for path selection. Service Index is also valuable when troubleshooting/reporting service paths. In addition to location within a path, SI can be used for loop detection.

3.4. NSH MD-type 1

When the base header specifies MD Type 1, NSH defines four 4-byte mandatory context headers, as per Figure 4. These headers must be present and the format is opaque as depicted in Figure 5.

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

Figure 4: NSH MD-type=0x1

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Context data                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Context Header

3.4.1. Mandatory Context Header Allocation Guidelines

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Network Platform Context                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Network Shared Context                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Service Platform Context                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Service Shared Context                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Context Data Significance

Figure 6, above, and the following examples of context header allocation are guidelines that illustrate how various forms of information can be carried and exchanged via NSH.

Network platform context: provides platform-specific metadata shared between network nodes. Examples include (but are not limited to) ingress port information, forwarding context and encapsulation type.

Network shared context: metadata relevant to any network node such as the result of edge classification. For example, application information, identity information or tenancy information can be shared using this context header.

Service platform context: provides service platform specific metadata shared between service functions. This context header is analogous to the network platform context, enabling service platforms to exchange platform-centric information such as an identifier used for load balancing decisions.

Service shared context: metadata relevant to, and shared, between service functions. As with the shared network context, classification information such as application type can be conveyed using this context.

The data center[dcalloc] and mobility[moballoc] context header allocation drafts provide guidelines for the semantics of NSH fixed context headers in each respective environment.

3.5. NSH MD-type 2

When the base header specifies MD Type 2, NSH defines variable length only context headers. There may be zero or more of these headers as per the length field.

```
 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|C|R|R|R|R|R|R|   Length  |  MD-type=0x2  | Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Service Path ID                      | Service Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~           Optional Variable Length Context Headers            ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: NSH MD-type=0x2

3.5.1. Optional Variable Length Metadata

NSH MD Type 2 MAY contain optional variable length context headers. The format of these headers is as described below.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          TLV Class            |      Type     |R|R|R|   Len   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Variable Metadata                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Variable Context Headers

TLV Class: describes the scope of the "Type" field. In some cases, the TLV Class will identify a specific vendor, in others, the TLV Class will identify specific standards body allocated types.

Type: the specific type of information being carried, within the scope of a given TLV Class. Value allocation is the responsibility of the TLV Class owner.

The most significant bit of the Type field indicates whether the TLV is mandatory for the receiver to understand/process. This effectively allocates Type values 0 to 127 for non-critical options and Type values 128 to 255 for critical options. Figure 7 below illustrates the placement of the Critical bit within the Type field.

```
+-+-+-+-+-+-+-+-+
|C|     Type    |
+-+-+-+-+-+-+-+-+
```

Figure 9: Critical Bit Placement Within the TLV Type Field

Encoding the criticality of the TLV within the Type field is consistent with IPv6 option types.

If a receiver receives an encapsulated packet containing a TLV with the Critical bit set in the Type field and it does not understand how to process the Type, it MUST drop the packet. Transit devices MUST NOT drop packets based on the setting of this bit.

Reserved bits: three reserved bit are present for future use. The

reserved bits MUST be zero.

Length: Length of the variable metadata, in 4-byte words.

4. NSH Actions

Service header aware nodes - service classifiers, SFF, SF and NSH proxies, have several possible header related actions:

1. Insert or remove service header: These actions can occur at the start and end respectively of a service path. Packets are classified, and if determined to require servicing, a service header imposed. The last node in a service path, an SFF, removes the NSH. A service classifier MUST insert an NSH. At the end of a service function chain, the last node operating on the service header MUST remove it.

   A service function can re-classify data as required and that re-classification might result in a new service path. In this case, the SF acts as a logical classifier as well. When the logical classifier performs re-classification that results in a change of service path, it MUST remove the existing NSH and MUST impose a new NSH with the base header reflecting the new path.

2. Select service path: The base header provides service chain information and is used by SFFs to determine correct service path selection. SFFs MUST use the base header for selecting the next service in the service path.

3. Update a service header: NSH aware service functions MUST decrement the service index. A service index = 0 indicates that a packet MUST be dropped by the SFF performing NSH-based forwarding.

   Service functions MAY update context headers if new/updated context is available.

   If an NSH proxy (see Section 7) is in use (acting on behalf of a non-NSH-aware service function for NSH actions), then the proxy MUST update service index and MAY update contexts. When an NSH proxy receives an NSH-encapsulated packet, it removes the NSH before forwarding it to an NSH unaware SF. When it receives a packet back from an NSH unaware SF, it re-encapsulates it with the NSH, decrementing the service index.

4. Service policy selection: Service function instances derive policy selection from the service header. Context shared in the service header can provide a range of service-relevant information such as traffic classification. Service functions SHOULD use NSH to select local service policy.

Figure 10 maps each of the four actions above to the components in the SFC architecture that can perform it.

Component	Insert or remove service header			Select service path	Update a service header		Service Policy Selection
	Insert	Remove	Remove and Insert		Dec. Service Index	Update Context Header	
Service Classification Function	+					+	
Service Function Forwarder(SFF)		+		+		+	
Service Function (SF)					+	+	+
NSH Proxy	+	+			+	+	

Figure 10: NSH Action and Role Mapping

5. NSH Encapsulation

Once NSH is added to a packet, an outer 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 as is.

The service header is independent of the encapsulation used and is encapsulated in existing transports. The presence of NSH is indicated via protocol type or other indicator in the outer encapsulation.

See Section 11 for NSH encapsulation examples.

6. NSH Usage

The NSH creates a dedicated service plane, that addresses many of the limitations highlighted in Section 2.2. More specifically, NSH enables:

1. Topological Independence: Service forwarding occurs within the service plane, via a network overlay, the underlying network topology does not require modification. Service functions have one or more network locators (e.g. IP address) to receive/send data within the service plane, the NSH contains an identifier that is used to uniquely identify a service path and the services within that path.

2. Service Chaining: NSH contains path identification information needed to realize a service path. Furthermore, 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. Metadata Sharing: NSH provides a mechanism to carry shared metadata between network devices and service function, and between service functions. The semantics of the shared metadata is communicated via a control plane to participating nodes. Examples of metadata include classification information used for policy enforcement and network context for forwarding post service delivery.

4. Transport Agnostic: NSH is transport independent and is carried in an overlay, over existing underlays. If an existing overlay topology provides the required service path connectivity, that existing overlay may be used.

7. NSH Proxy Nodes

In order to support NSH-unaware service functions, an NSH proxy is used. The proxy node removes the NSH header and delivers the original packet/frame via a local attachment circuit to the service function. Examples of a local attachment circuit include, but are not limited to: VLANs, IP in IP, GRE, VXLAN. When complete, the service function returns the packet to the NSH proxy via the same or different attachment circuit.

NSH is re-imposed on packets returned to the proxy from the non-NSH-aware service.

Typically, an SFF will act as an NSH-proxy when required.

An NSH proxy MUST perform NSH actions as described in Section 4.

8. Fragmentation Considerations

Work in progress

9. Service Path Forwarding with NSH

9.1. SFFs and Overlay Selection

As described above, 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 provide a level of indirection between the service path/topology and the network transport. 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 and location of a logical SF (locator and order). The logical SF may be a single SF, or a set of SFs that are equivalent. In the latter case, the SFF provides load distribution amongst the collection of SFs as needed. SI may also serve as a mechanism for loop detection with in a service path since each SF in the path decrements the index; an index of 0 indicates that a loop occurred and packet must be discarded.

This indirection -- path ID 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.). As mentioned above, an existing overlay topology may be used provided it offers the requisite connectivity.

The mapping of SPI to transport occurs on an SFF. The SFF consults the SPI/ID values to determine the appropriate overlay transport protocol (several may be used within a given network) and next hop for the requisite SF. Figure 10 below depicts an SPI/SI to network overlay mapping.

SPI	SI	NH	Transport
10	3	1.1.1.1	VXLAN-gpe
10	2	2.2.2.2	nvGRE
245	12	192.168.45.3	VXLAN-gpe
10	9	10.1.2.3	GRE
40	9	10.1.2.3	GRE
50	7	01:23:45:67:89:ab	Ethernet
15	1	Null (end of path)	None

Figure 11: SFF NSH Mapping Example

Additionally, further indirection is possible: the resolution of the required SF function locator may be a localized resolution on an SFF,rather than a service function chain control plane responsibility, as per figures 11 and 12 below.

SPI	SI	NH
10	3	SF2
245	12	SF34
40	9	SF9

Figure 12: NSH to SF Mapping Example

SF	NH	Transport
SF2	10.1.1.1	VXLAN-gpe
SF34	192.168.1.1	UDP
SF9	1.1.1.1	GRE

Figure 13: 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) overlay links to be used (for load distribution, redundancy or policy), see Figure 13. The metric depicted in Figure 13 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 sate, capacity, etc.)

```
+----------------------------------+
| SPI | SI |   NH        |  Metric |
+----------------------------------+
| 10  |  3 | 10.1.1.1    |  1      |
|     |    | 10.1.1.2    |  1      |
|     |    |             |         |
| 20  | 12 | 192.168.1.1 |  1      |
|     |    | 10.2.2.2    |  1      |
|     |    |             |         |
| 30  |  7 | 10.2.2.3    |  10     |
|     |    | 10.3.3.3    |  5      |
+----------------------------------+
(encap type omitted for formatting)
```

Figure 14: NSH Weighted Service Path

9.2. Mapping NSH to Network Overlay

As described above, the mapping of SPI to network topology may result in a single overlay path, or it might result in a more complex topology. Furthermore, the SPIx to overlay mapping occurs at each SFF independently. Any combination of topology selection is possible.

Examples of mapping for a topology:

1. Next SF is located at SFFb with locator 10.1.1.1
   SFFa mapping: SPI=10 --> VXLAN-gpe, dst-ip: 10.1.1.1

2. Next SF is located at SFFc with multiple locator for load distribution purposes:
   SFFb mapping: SPI=10 --> VXLAN-gpe, dst_ip:10.2.2.1, 10.2.2.2, 10.2.2.3, equal cost

3. Next SF is located at SFFd with two path to SFFc, one for redundancy:
   SFFc mapping: SPI=10 --> VXLAN-gpe, dst_ip:10.1.1.1 cost=10, 10.1.1.2, 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 overlay paths as required. For example, the overlay path between service functions forwarders 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 function plane.

9.3. 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 the NSH information (packet capture, IPFIX, etc.). The information can be used for service scheduling and placement decisions, troubleshooting and compliance verification.

9.4. Service Graphs

In some cases, a service path is exactly that -- a linear list of service functions that must be traversed. However, increasingly, the "path" is actually a true directed graph. Furthermore, within a given service topology several directed graphs may exist with packets moving between graphs based on non-initial classification (usually performed by a service function). Note: strictly speaking a path is a form of graph; the intent is to distinguish between a directed graph and a path.

```
                ,---.          ,---.            ,---.
               /     \        /     \          /     \
              (  SF2  )      (  SF7  )        (  SF3  )
        ,------\     +.       \     /          \     /
       ;        |---'  `-.     `---'\            `-+-'
       |        :         :          \             ;
       |         \        |           :           ;
     ,-+-.        `.     ,+--.        :           |
    /     \         '---+     \        \          ;
   (  SF1  )           (  SF6  )        \        /
    \     /             \     +--.       :      /
     `---'               `---'    `-.  ,-+-.  /
                                     `+     +'
                                     (  SF4  )
                                      \     /
                                       `---'
```

Figure 15: Service Graph Example

The SPI/SI combination provides a simple representation of a directed graph, the SPI represents a graph ID; and the SI a node ID. The

service topology formed by SPI/SI support cycles, weighting, and alternate topology selection, all within the service plane. The realization of the network topology occurs as described above: SPI/ID mapping to an appropriate transport and associated next network hops.

NSH-aware services receive the entire header, including the SPI/SI. An SF can now, based on local policy, alter the SPI, which in turn effects both the service graph, and in turn the selection of overlay at the SFF. The figure below depicts the policy associated with the graph in Figure 14 above. Note: this illustrates multiple graphs and their representation; it does not depict the use of metadata within a single service function graph.

```
+-----------------------------------------------------------------------+
|                                                  SPI: 21 Bob: SF7     |
|                  SPI: 20 Bad : SF2 --> SF6 --> SF4                    |
|SPI: 10 SF1 --> SF2 --> SF6                       SPI: 22 Alice: SF3   |
|                  SPI: 30 Good: SF4                                    |
|                             SPI:31 Bob: SF7                           |
|                             SPI:32 Alice: SF3                         |
+-----------------------------------------------------------------------+
```

Figure 16: Service Graphs Using SPI

This example above does not show the mapping of the service topology to the network overlay topology. As discussed in the sections above, the overlay selection occurs as per network policy.

10. Policy Enforcement with NSH

10.1. NSH Metadata and Policy Enforcement

As described in Section 3, NSH provides the ability to carry metadata along a service path. This metadata may be derived from several sources, common examples include:

Network nodes: 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 pathing.

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.

Service functions: Service functions often perform very detailed and valuable classification. In some cases they may terminate, and be able to inspect encrypted traffic. SFs may update, alter or impose metadata information.

Regardless of the source, metadata reflects the "result" of classification. The granularity of classification may vary. For example, a network switch might only be able to classify based on a 5-tuple, whereas, a service function may be able to inspect application information. Regardless of granularity, the classification information can be represented in NSH.

Once the data is added to 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. The following two examples highlight the relationship between metadata and policy:

```
+-------------------------------------------------+
|    ,---.             ,---.              ,---.    |
|   /     \           /     \            /     \   |
|  (  SCL  )-------->(  SF1  )--------->(  SF2  )  |
|   \     /           \     /            \     /   |
|    `---'             `---'              `---'    |
|5-tuple:            Permit             Inspect    |
|Tenant A            Tenant A           AppY       |
|AppY                                               |
+-------------------------------------------------+
```

Figure 17: Metadata and Policy

```
+-------------------------------------------------+
|    ,---.             ,---.              ,---.    |
|   /     \           /     \            /     \   |
|  (  SCL  )-------->(  SF1  )--------->(  SF2  )  |
|   \     /           \     /            \     /   |
|    `-+-'             `---'              `---'    |
|      |              Permit            Deny AppZ  |
|  +---+---+          employees                    |
|  |       |                                       |
|  +-------+                                       |
|  external                                        |
|  system:                                         |
|  Employee                                        |
|  App Z                                           |
+-------------------------------------------------+
```

Figure 18: 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, rather they relied on a antecedent classification for local policy enforcement.

10.2. Updating/Augmenting Metadata

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

1. Metadata Augmentation: Information may be added to NSH's existing metadata, as depicted in Figure 18. For example, if the initial classification returns the tenant information, a secondary classification (perhaps a DPI or SLB) may augment the tenant classification with application information. 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 trafic as "internet" but a security service function determines that the traffic is really "attack". Figure 19 illustrates an example of updating metadata.

```
 +-----------------------------------------------------+
|    ,---.             ,---.              ,---.        |
|   /     \           /     \            /     \       |
|  (  SCL  )-------->(  SF1  )--------->(  SF2  )      |
|   \     /           \     /            \     /       |
|    `-+-'             `---'              `---'        |
|      |              Inspect           Deny           |
|  +---+---+          employees         employee+      |
|  |       |          Class=AppZ        appZ           |
|  +-------+                                           |
|  external                                            |
|  system:                                             |
|  Employee                                            |
|                                                      |
+-----------------------------------------------------+
```

Figure 19: Metadata Augmentation

```
+-----------------------------------------------------+
|    ,---.             ,---.              ,---.        |
|   /     \           /     \            /     \       |
|  (  SCL  )-------->(  SF1  )--------->(  SF2  )      |
|   \     /           \     /            \     /       |
|    `---'             `---'              `---'        |
|5-tuple:             Inspect             Deny         |
|Tenant A             Tenant A            attack       |
|                      --> attack                      |
+-----------------------------------------------------+
```

Figure 20: Metadata Update

10.3. Service Path ID and Metadata

Metadata information may influence the service path selection since the service path identifier can represent the result of classification. A given SPI can represent all or some of the metadata, and be updated based on metadata classification results. This relationship provides the ability to create a dynamic services 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 9.4. Figure 20 illustrates an example of this behavior.

```
+-----------------------------------------------------+
|   ,---.             ,---.              ,---.        |
|  /     \           /     \            /     \       |
| (  SCL  )-------->(  SF1  )--------->(  SF2  )      |
|  \     /           \     /            \     /       |
|   `---'             `---' \              `---'      |
|5-tuple:            Inspect \            Original    |
|Tenant A            Tenant A \           next SF     |
|                     --> DoS  \                       |
|                                \                     |
|                                 ,---.                |
|                                /     \               |
|                               (  SF10 )              |
|                                \     /               |
|                                 `---'                |
|                                  DoS                 |
|                               "Scrubber"             |
+-----------------------------------------------------+
```

Figure 21: Path ID and Metadata

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

11. NSH Encapsulation Examples

11.1. GRE + NSH

```
IPv4 Packet:
+----------+--------------------+--------------------+
|L2 header | L3 header, proto=47|GRE header,PT=0x894F|
+----------+--------------------+--------------------+
--------------+----------------+
NSH, NP=0x1   |original packet |
--------------+----------------+


L2 Frame:
+----------+--------------------+--------------------+
|L2 header | L3 header, proto=47|GRE header,PT=0x894F|
+----------+--------------------+--------------------+
---------------+---------------+
NSH, NP=0x3    |original frame |
---------------+---------------+
```

Figure 22: GRE + NSH

11.2. VXLAN-gpe + NSH

```
IPv4 Packet:
+----------+-----------------------+---------------------+
|L2 header | IP + UDP dst port=4790 |VXLAN-gpe NP=0x4(NSH)|
+----------+-----------------------+---------------------+
--------------+----------------+
NSH, NP=0x1   |original packet |
--------------+----------------+


L2 Frame:
+----------+-----------------------+---------------------+
|L2 header | IP + UDP dst port=4790 |VXLAN-gpe NP=0x4(NSH)|
+----------+-----------------------+---------------------+
---------------+---------------+
NSH,NP=0x3     |original frame |
---------------+---------------+
```

Figure 23: VXLAN-gpe + NSH

11.3. Ethernet + NSH

```
IPv4 Packet:
+-------------------------------+---------------+--------------------+
|Outer Ethernet, ET=0x894F      | NSH, NP = 0x1 | original IP Packet |
+-------------------------------+---------------+--------------------+


L2 Frame:
+-------------------------------+---------------+----------------+
|Outer Ethernet, ET=0x894F      | NSH, NP = 0x3 | original frame |
+-------------------------------+---------------+----------------+
```

Figure 24: Ethernet + NSH

12. Security Considerations

As with many other protocols, NSH data can be spoofed or otherwise modified. In many deployments, NSH will be used in a controlled environment, with trusted devices (e.g. a data center) thus mitigating the risk of unauthorized header manipulation.

NSH is always encapsulated in a transport protocol and therefore, when required, existing security protocols that provide authenticity (e.g. RFC 2119 [RFC6071]) can be used.

Similarly if confidentiality is required, existing encryption protocols can be used in conjunction with encapsulated NSH.

13. Open Items for WG Discussion

1. MD type 1 metadata semantics specifics
2. Bypass bit in NSH.
3. Rendered Service Path ID (RSPID).

14. Contributors

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

Andrew Dolganow
Alcaltel-Lucent
Email: andrew.dolganow@alcatel-lucent.com

Rex Fernando
Cisco Systems
Email: rex@cisco.com

Praveen Muley
Alcaltel-Lucent
Email: praveen.muley@alcatel-lucent.com

Navindra Yadav
Cisco Systems
Email: nyadav@cisco.com

15. Acknowledgments

The authors would like to thank Nagaraj Bagepalli, Abhijit Patra, Ron Parker, 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 Carlos Pignataro and Larry Kreeger for their invaluable ideas and contributions which are reflected throughout this draft.

16. IANA Considerations

16.1. NSH EtherType

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

16.2. 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.

16.2.1. NSH Base Header Reserved Bits

There are ten bits at the beginning of the NSH Base Header. New bits are assigned via Standards Action [RFC5226].

Bits 0-1 - Version
Bit 2 - OAM (O bit)
Bits 2-9 - Reserved

16.2.2. MD Type Registry

IANA is requested to set up a registry of "MD Types". These are 8-bit values. MD Type values 0, 1, 2, 254, and 255 are specified in this document. Registry entries are assigned by using the "IETF Review" policy defined in RFC 5226 [RFC5226].

MD Type	Description	Reference
0	Reserved	This document
1	NSH	This document
2	NSH	This document
3..253	Unassigned	
254	Experiment 1	This document
255	Experiment 2	This document

Table 1

16.2.3. TLV Class Registry

IANA is requested to set up a registry of "TLV Types". These are 16-bit values. Registry entries are assigned by using the "IETF Review" policy defined in RFC 5226 [RFC5226].

16.2.4. 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 and 3 are defined in this draft. New values are assigned via Standards Action [RFC5226].

Next Protocol	Description	Reference
0	Reserved	This document
1	IPv4	This document
2	IPv6	This document
3	Ethernet	This document
4..253	Unassigned	

Table 2

17. References

17.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

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

17.2. Informative References

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6071] Frankel, S. and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", RFC 6071, February 2011.

[SFC-PS] Quinn, P., Ed. and T. Nadeau, Ed., "Service Function Chaining Problem Statement", 2014, <http://datatracker.ietf.org/doc/draft-ietf-sfc-problem-statement/>.

[SFC-arch] Quinn, P., Ed. and J. Halpern, Ed., "Service Function Chaining (SFC) Architecture", 2014, <http://datatracker.ietf.org/doc/draft-quinn-sfc-arch>.

[VXLAN-gpe] Quinn, P., Agarwal, P., Kreeger, L., Lewis, D., Maino, F., Yong, L., Xu, X., Elzur, U., and P. Garg, "Generic Protocol Extension for VXLAN", <https://datatracker.ietf.org/doc/draft-quinn-vxlan-gpe/>.

[dcalloc] Guichard, J., Smith, M., and S. Kumar, "Network Service Header (NSH) Context Header Allocation (Data Center)", 2014, <https://datatracker.ietf.org/doc/draft-guichard-sfc-nsh-dc-allocation/>.

[moballoc] Napper, J. and S. Kumar, "NSH Context Header Allocation -- Mobility", 2014, <https://datatracker.ietf.org/doc/

draft-napper-sfc-nsh-mobility-allocation/>.

Authors' Addresses

Paul Quinn
Cisco Systems, Inc.

Email: paulq@cisco.com

Jim Guichard
Cisco Systems, Inc.

Email: jguichar@cisco.com

Surendra Kumar
Cisco Systems, Inc.

Email: smkumar@cisco.com

Michael Smith
Cisco Systems, Inc.

Email: michsmit@cisco.com

Wim Henderickx
Alcatel-Lucent

Email: wim.henderickx@alcatel-lucent.com

Tom Nadeau
Brocade

Email: tnadeau@lucidvision.com

Puneet Agarwal

Email: puneet@acm.org

Rajeev Manur
Broadcom

Email: rmanur@broadcom.com

Abhishek Chauhan
Citrix

Email: Abhishek.Chauhan@citrix.com

Joel Halpern
Ericsson

Email: joel.halpern@ericsson.com

Sumandra Majee
F5

Email: S.Majee@F5.com

Uri Elzur
Intel

Email: uri.elzur@intel.com

David Melman
Marvell

Email: davidme@marvell.com

Pankaj Garg
Microsoft

Email: Garg.Pankaj@microsoft.com

Brad McConnell
Rackspace

Email: bmcconne@rackspace.com

Chris Wright
Red Hat Inc.

Email: chrisw@redhat.com

Kevin Glavin
Riverbed

Email: kevin.glavin@riverbed.com

Hong (Cathy) Zhang
Huawei US R&D

Email: cathy.h.zhang@huawei.com

Louis Fourie
Huawei US R&D

Email: louis.fourie@huawei.com

Ron Parker
Affirmed Networks

Email: ron_parker@affirmednetworks.com

Myo Zarny
Goldman Sachs

Email: myo.zarny@gs.com

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

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

September 6, 2016

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

Abstract

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

Status of This Memo

Table of Contents

1. Introduction

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

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

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

```
                +----------------+
                |SFC-unaware     |
                |Service Function|
                | (Legacy SF)    |
                +----+----+------+
                     ^    |
                     |    |
                +----+----+------+
                |Switch(optional)|
                +----+----+------+
                     |    |
                  (2)|    |(3)
                     |    |
                +----+----V--------+
           (1)  |      SFC         | (4)
       -------->|      Proxy       +------->
                +------------------+
```

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

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

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

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

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

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

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

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

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

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

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

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

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

2. Terminology

The terminology used in this document is defined below:

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

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

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

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

3. Mechanisms

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

3.1. For Transparent Service Functions

3.1.1. VLAN

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

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

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

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

3.1.2. VXLAN

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

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

3.1.3. Ethernet MAC Address

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

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

3.1.4. 5-tuple

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

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

3.2. For Non-transparent Service Functions

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

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

```
                        +------------------------+
                        |                        |
                        |   SFC Control Plane    |
                        |                        |
                        +---^------------^-------+
                            |            |
                            |C2          |   (1)
                            |            |
                        +---V----+       |C4
                  -----+  SFF   +------  |
                        |        |       |
                        +----+---+       |
                             |       +---V----+
                             +-------+  SFC   |
                               (2)   | Proxy  |
                                     +---^----+
                                         |
                                      (3)|(4)
                                +------V---------+
                                |SFC-unaware     |
                                |Service Function|
                                +----------------+
```

4. Operation Considerations

4.1. Examplar Mechanisms

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

Table 1: Mapping Examples

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

4.2. Challenges to Support Legacy SF

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

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

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

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

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

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

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

4.3. Metadata

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

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

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

5. Security Considerations

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

6. Acknowledgement

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

7. References

7.1. Normative References

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

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

7.2. Informative References

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

Authors' Addresses

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

Email: haibin.song@huawei.com

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

Email: youjianjie@huawei.com

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

Email: lucy.yong@huawei.com

Yuanlong Jiang
Huawei
Bantian, Longgang district
Shenzhen 518129
China

Email: jiangyuanlong@huawei.com

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

Email: ldunbar@huawei.com

Nicolas Bouthors
Qosmos

Email: nicolas.bouthors@qosmos.com

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

Email: ddolson@sandvine.com