Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 21, 2016

A. Cabellos
UPC-BarcelonaTech
S. Barkai
B. Perlman
Hewlett Packard Enterprise
V. Ermagan
F. Maino
Cisco Systems Inc
A. Rodriguez-Natal
UPC-BarcelonaTech

October 19, 2015

Map-Assisted SFC Proxy using LISP
draft-cabellos-sfc-map-assisted-proxy-00.txt

Abstract

This document specifies a map-assisted SFC proxy. The SFC proxy uses the LISP Mapping System to store the NSH header indexed by 5-tuple, before decapsulating and forwarding the packet to the legacy function. After the function has processed the packet, the SFC proxy retrieves the NSH header from the Mapping System to SFC encapsulate it.

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

1. Introduction

The Locator/ID Separation Protocol (LISP) [RFC6830] is an overlay protocol that creates two namespaces: EIDs (End-point IDentifiers) and RLOCs (Routing LOCators). The LISP Mapping System stores the mappings between both namespaces, LISP provides a standard way for its data-plane elements, called xTRs, to store and retrieve mappings from the Mapping System to make forwarding decisions: Map-Request, Map-Request and Map-Reply. Finally, LISP also offers a flexible syntax for both EIDs and RLOCs by means of LCAFs [I-D.ietf-lisp-lcaf] to define what is an EID and what is an RLOC.

With such architecture in place, the LISP control-plane represents a programmable protocol. The Mapping System is a logically centralized database that stores network state, which is retrieved by data-plane nodes in a standard way to make decisions. Any external control plane can program the LISP Mapping System while any data-plane node can be map-assisted.

This document specifies a map-assisted SFC proxy [I-D.ietf-sfc-architecture]. An SFC acts on behalf the SFC unaware functions on the SFC domain. Basically the SFC Proxy removes the SFC encapsulation, forwards the packet to the SFC unaware function, receives back the packets and reapplies an SFC encapsulation. Specifically this document specifies how to map-assist the encapsulation operation by means of the LISP control-plane.

In short, the SFC Proxy before decapsulating the packet stores (Map-Registers) the NSH header (including Context Headers) [I-D.ietf-sfc-nsh] in the LISP Mapping System indexed by the 5-tuple of the packet {5-tuple->NSH}. After the SFC unaware function has processed the packet, the proxy retrieves (Map-Requests based on the 5-tuple of the packet) the NSH+Context headers to SFC encapsulate the packet.

This has two main benefits; first the SFC proxy is stateless and connectionless. Second, in some cases the legacy function may change the headers of the original packet, the SFC control plane can change the stored mapping {5-tuple->NSH} in the Mapping System accordingly and allow for fast reclassification by the proxy.

2. Overview

2.1. Flow example

This section shows a flow example of map-assisted SFC Proxy processing:

```
                        +------------+     +------------+
                        |LISP Mapping|     |SFC Control |
```

```
                        |   System   |      |   Plane    |
                        |            |      |            |
                        +------------+      +------------+
                              ^
                              |
                        Map-Register
                       {5-tuple->NSH}
                              |
                              |              +----------+
  +----------+        +----+-----+         |   SFC    |
  |   SFF    |------->|SFC Proxy |------->| Unaware  |
  +----------+        +----------+         | Function |
                                             +----------+
```

Figure 1.- SFC Proxy Decapsulation

1. An SFC proxy receives an SFC encapsulated packet as defined in the SFC architecture [I-D.ietf-sfc-architecture].

2. The SFC proxy Map-Registers the SFC encapsulation in the LISP Mapping System (figure 1), this includes the entire NSH header: Base Header, Service Header and Context Headers. The NSH header is indexed by the 5-tuple of the payload. Both the 5-tuple and the NSH header are encoded using two different LISP LCAFs, further details can be found in Section 3.

3. The SFC proxy forwards the packet to the SFC unaware function as specified in the SFC architecture [I-D.ietf-sfc-architecture].

4. The SFC unaware function processes the packet and sends it back to the SFC proxy.

5. Upon reception of the processed packet, the SFC proxy must SFC encapsulate the packet. For this it retrieves the NSH header from the LISP Mapping System using a Map-Request indexed by the 5-tuple of the received packet (figure 2). Once the packet is SFC encapsulated, the SFC proxy forwards it as defined in the SFC architecture [I-D.ietf-sfc-architecture].

```
                        +------------+  Reclassificaton +------------+
                        |LISP Mapping|{New 5-tuple->NSH}|SFC Control |
                        |   System   | <----------------|   Plane    |
                        |            |                  |            |
                        +---------+--+                  +------------+
                          ^       |
                          |  Map-Reply
                          |    {NSH}
                    Map-Request  |
                    {5-tuple?}   |
                          |       v          +----------+
  +----------+        ++---------+         |   SFC    |
  |   SFF    |<-------|SFC Proxy |<-------| Unaware  |
  +----------+        +----------+         | Function |
                                             +----------+
```

Figure 2.- SFC Proxy Encapsulation

2.2. Benefits of Map-Assisted SFC Proxies

The Map-Assisted encapsulation described in step 5 of the previous section brings the following benefits to the SFC architecture:

- The map-assisted SFC proxy is connectionless and stateless, as such it does not need to store state to forward packets from/to SFC unaware functions. Since the required state is stored in the

Mapping System, any other SFC proxy can receive the processed packets and SFC encapsulate them.

- In some scenarios the legacy functions may change the packet header and hence, the SFC proxy must re-classify it. With map-assisted SFC proxies, the SFC control-plane can change the stored state on the Mapping System to accordingly and allow map-assisted stateless reclassification by the SFC-Proxy. This is illustrated in the figure 2 by the "Reclassification" arrow. How the SFC control plane updates information on the LISP Mapping system is out of the scope of this document. In any case, please note that the SFC proxy still operates as described in this document and remains unaware of the reclassification.

3. Encoding of 5-tuple and NSH in LISP messages

This section describes the LCAFs used to encode both the 5-tuple and NSH header (Base, Service Path and Context Headers). The 5-tuple index is encoded in a LISP record as an EID while the NSH header as an RLOC.

3.1. Encoding of 5-tuple Index

The Multiple-tuple EID [I-D.rodrigueznatal-lisp-multi-tuple-eid] is used to encode the 5-tuple EID that indexes the NSH header, specifically using the "Exact Match" mode and EID mask-ken set to 0.

3.2. Encoding of NSH Header

The NSH header (Base Header, Service Path Header and Context Headers) [I-D.ietf-sfc-nsh] is encoded using the JSON Data Model Type LCAF as defined in [I-D.ietf-lisp-lcaf]. The header is encoded in binary format using BSON [BSON] as a single binary field (subtype "Generic binary subtype"):

```
document ::= int32 binary "\x00"
```

A LISP record only transports a single NSH header and all the "Loc" fields are ignored except "Loc-AFI" and "Locator".

4. SFC Proxy Processing

This section specifies the behavior of a map-assisted SFC Proxy, the proxy acts as specified in [I-D.ietf-sfc-architecture] with the following exceptions.

Inbound: For traffic received from the SFF and before removing the SFC encapsulation, the proxy Map-Registers the NSH header (Base, Service and Context) using the 5-tuple and JSON LCAFs defined in Section 3, the 5-tuple is applied to the original payload. After this the SFC Proxy acts as specified in [I-D.ietf-sfc-architecture].

Outbound: For returning traffic from the legacy SF, the SFC Proxy Map-Requests using a 5-tuple lookup LCAF and receives back the entire NSH header encoded using the JSON LCAF. The proxy applies the NSH encapsulation, decrements the Service Index and forwards the traffic as specified in [I-D.ietf-sfc-architecture].

In addition to this please note the following:

- In some scenarios the SFC Control Plane may have changed the {5-tuple->NSH} mapping to account for changes made by the legacy SF to the payload.

- The LISP Mapping System can identify the registering and requesting SFC Proxy using the RLOC of the Map-Register and Map-Request message respectively. This is useful when the inbound and

outbound SFC Proxies are different.

- o This document assumes that the payload is IP (IPv4 or IPv6) and a transport header (TCP or UDP). Further revisions of this document will consider other payloads.

5. Security Considerations

The map-assisted SFC Proxy does not introduce additional security considerations beyond the ones described in [I-D.ietf-sfc-architecture] and [I-D.ietf-lisp-threats].

6. IANA Considerations

This memo includes no request to IANA.

7. References

7.1. Normative References

[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>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <http://www.rfc-editor.org/info/rfc6830>.

7.2. Informative References

[BSON] MongoDB, , ""BSON - Binary JSON Specification "http://bsonspec.org/"", June 2015.

[I-D.ietf-lisp-lcaf] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", draft-ietf-lisp-lcaf-11 (work in progress), September 2015.

[I-D.ietf-lisp-threats] Saucez, D., Iannone, L., and O. Bonaventure, "LISP Threats Analysis", draft-ietf-lisp-threats-13 (work in progress), August 2015.

[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-01 (work in progress), July 2015.

[I-D.rodrigueznatal-lisp-multi-tuple-eid] Rodriguez-Natal, A., Cabellos-Aparicio, A., Barkai, S., Ermagan, V., Maino, F., Lewis, D., and D. Farinacci, ""LISP support for Multi-Tuple EIDs" draft-rodrigueznatal-lisp-multi-tuple-eids-00 (work in progress)", June 2015.

Authors' Addresses

Albert Cabellos
UPC-BarcelonaTech
c/ Jordi Girona 1-3
Barcelona, Catalonia 08034
Spain

Email: acabello@ac.upc.edu

Sharon Barkai
Hewlett Packard Enterprise
3000 Hanover Street
Palo Alto, CA
USA

Email: sharon.barkai@hpe.com

Barak Perlman
Hewlett Packard Enterprise
3000 Hanover Street
Palo Alto, CA
USA

Email: barak.perlman@hpe.com

Vina Ermagan
Cisco Systems Inc
170 W Tasman Drive
San Jose, CA 95134
USA

Email: vermagan@cisco.com

Fabio Maino
Cisco Systems Inc
170 W Tasman Drive
San Jose, CA 95134
USA

Email: fmaino@cisco.com

Alberto Rodriguez-Natal
UPC-BarcelonaTech
c/ Jordi Girona 1-3
Barcelona, Catalonia 08034
Spain

Email: arnatal@ac.upc.edu

Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: December 29, 2016

D. Dolson
Sandvine
S. Homma
NTT
D. Lopez
Telefonica I+D
M. Boucadair
Orange
D. Liu
Alibaba Group
T. Ao
ZTE Corporation
V. Vu
SSU
June 27, 2016

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

Abstract

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

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

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

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

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

outside of the scope of the document to consider domains operated by different organizations.

2. Hierarchical Service Function Chaining (hSFC)

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

2.1. Top Level

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

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

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

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

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

At this network-wide level, the number of SFPs required is a linear function of the number of ways in which a packet is required to traverse different sub-domains and egress the network. Note that the various paths which may be taken within a sub-domain are not represented by distinct network-wide SFPs; specific policies at the ingress nodes of each sub-domain bind flows to sub-domain paths.

Packets are classified at the edge of the network to select the paths by which sub-domains are to be traversed. At the ingress of each sub-domain, paths are reclassified to select the paths by which SFs in the sub-domain are to be traversed. At the egress of each sub-domain, packets are returned to the top-level paths. Contrast this with an approach requiring the top-level classifier to select paths to specify all of the SFs in each sub-domain.

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

```
                  +------------+
                  |Sub-domain#1|
                  |  in DC1    |
                  +----+-------+
                       |
                .---- SFF1 ------.   +--+
        +--+   /     /  |         \--|CF|
   --->|CF|--/---->'   |          \ +--+
        +--+ /  SC#1    |           \
             |          |            |
             |          V    .------>|--->
             |         /    /        |
             \         |   /        /
        +--+  \        |  /        /  +--+
        |CF|---\       | /        /---|CF|
        +--+    '---- SFF2 ------'    +--+
                       |
                  +----+-------+
                  |Sub-domain#2|
                  |   in DC2   |
                  +------------+
```

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

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

2.2. Lower Levels

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

Unlike the top level, data packets entering the sub-domain are already SFC-encapsulated. Figure 2 shows a sub-domain interfaced with a higher-level domain by means of an Internal Boundary Node

(IBN). It is the purpose of the IBN to apply classification rules and direct the packets to the selected local SFPs terminating at an egress IBN. The egress IBN finally restores packets to the original SFC shim and hands them off to SFFs.

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

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

```
  +----+    +-----+  +----------------------+   +-----+
  |    |    | SFF |  |   IBN 1  (in DC 1)   |   | SFF |
  |    |SC#1|     |  |  +----------------+  |   |     |
->|    |===============>|      SFF       |===============>
  |    |    +-----+  |  +----------------+  |   +-----+
  | CF |             |   |              ^   |
  |    |             |   v              |   |
  |    |             |+--------------------+|   Top domain
  |    |             ||CF, fwd/rev mapping ||
  |    |    * * * * *||  and "glue"        || * * * * *
  |    |    *        |+--------------------+|         *
  +----+    *        | | |              | | |    Sub  *
            *        +-o-o--------------o-o-+   domain*
            *     SC#2 | |SC#1          ^ ^       #1  *
            *    +-----+ |              | |           *
            *    |       V              | |           *
            *    |     +---+  +------+  | |           *
            *    |     |SFF|->|SF#1.1|--+ |           *
            *    |     +---+  +------+    |           *
            *    V                        |           *
            *  +---+  +------+  +---+  +------+       *
            *  |SFF|->|SF#2.1|->|SFF|->|SF#2.2|       *
            *  +---+  +------+  +---+  +------+       *
            * * * * * * * * * * * * * * * * * * * * * *
```

*** Sub-domain boundary; === top-level chain; --- low-level chain.

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

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

3. Internal Boundary Node (IBN)

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

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

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

3.1. IBN Path Configuration

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

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

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

1. Saving SPI and SI in transport-layer flow state,

2. Pushing SPI and SI into metadata,

3. Using unique lower-level paths per upper-level path coordinates,

4. Nesting NSH headers, encapsulating the higher-level NSH headers within the lower-level NSH headers,

5. Saving upper-level by a flow ID and placing an hSFC flow ID into metadata,

3.1.1. Flow-Stateful IBN

An IBN can be flow-aware, returning packets to the correct higher-level SFP on the basis of the transport-layer coordinates (typically, a 5-tuple) of packets exiting the lower-level SFPs.

When packets are received by the IBN on a higher-level path, the encapsulated packets are parsed for IP and transport-layer (TCP, UDP, etc.) coordinates. State is created, indexed by these coordinates ({source-IP, destination-IP, source-port, destination-port and transport protocol} typically). The state contains at least critical fields of the encapsulating SFC header (or perhaps the entire header).

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

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

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

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

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

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

If an SFC domain handles multiple network regions (e.g., multiple private networks), the coordinates may be augmented with additional parameters, perhaps using some metadata to identify the network region.

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

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

3.1.2. Encoding Upper-Level Paths in Metadata

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

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

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

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

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

3.1.3. Using Unique Paths per Upper-Level Path

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

Whenever the upper-level domain provisions a path via the lower-level domain, the lower-level domain controller must provision corresponding paths to traverse the lower-level domain.

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

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

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

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

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

Figure 3: Encapsulation of NSH within NSH

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

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

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

However, the down-side is it does require SFs in the lower-level domain to be able to parse multiple layers of NSH. If the SF injects

packets, it must also be able to deal with adding appropriate multiple layers of headers to injected packets.

3.1.5. Stateful / Metadata Hybrid

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

hSFC Flow ID	SPI	SI	Context1	Context2	Context3	Context4
1	45	254	100	2112	12345	7

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

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

```
 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 Identifer               | Service Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      hSFC Flow ID                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Mandatory Context Header                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Mandatory Context Header                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Mandatory Context Header                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Advantages of this approach include:

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

- o Does not require all domains to have the same metadata scheme,
- o Can be used to restore any upper-domain information, not just service path,
- o The lower domain only requires a single item of metadata regardless of the number of items of metadata used in the upper domain. (For MD-Type 1, this leaves 3 slots for use in the lower domain.)
- o No special functionality is required of the SF, other than the usual ability to preserve metadata and to apply metadata to injected packets.

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

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

3.2. Gluing Levels Together

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

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

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

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

3.3. Decrementing Service Index

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

A good strategy seems to be to do this when the packet is first received by the IBN, before applying any of the strategies of Section 3.1, immediately prior to classification.

4. Sub-domain Classifier

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

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

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

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

5. Control Plane Elements

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

- o Each control-plane instance manages a single level of hierarchy of a single domain.

- o Each control-plane is agnostic about other levels of hierarchy. This aspect allows humans to reason about the system within a single domain and allows control-plane algorithms to use only domain-local inputs. Top-level control does not need visibility to sub-domain policies, nor does sub-domain control need visibility to higher-level policies.

- o Sub-domain control-planes are agnostic about control-planes of other sub-domains. This allows both humans and machines to manipulate sub-domain policy without considering policies of other domains.

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

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

6. Extension for Adopting to NSH-Unaware Service Functions

The hierarchical approach can be used for dividing networks into NSH-aware and NSH-unaware domains by converting NSH encapsulation to other forwarding techniques (e.g., 5-tuple-based routing with OpenFlow), as shown in Figure 5.

```
                  * * * * * * * * * * * * * * * * * *
                 *   NSH-aware domain                 *
                 *       +-------+       +-------+    *
                 *       | SF#1  |       | SF#5  |    *
                 *       +-o---o-+       +-o---o-+    *
                 *         ^   |           ^   |      *
                 *       +-|---|-+       +-|---|-+    *
                 *       | |SFF| |       | |SFF| |    *
                 *       +-|---|-+       +-|---|-+    *
                 *         .   |           |   .      *
                 * +--+   /    |           |    \     *
                -->|CF|--'     |           |     '------->
                 * +--+        v           |          *
                 *         +---o-----------o---+      *
                  .*.*.*.*.|  / |   IBN   | \  |*.*.*.
                 .         +-o--o---------o--o-+      .
                 .           |  |         ^  ^        .
                 .           |  +-+     +-+  |        .
                 .       +---+    v     |    +---+    .
                 .       |      +-o-----o-+      |    .
                 .       |      |  SF#2   |      |    .
                 .       |      +---------+      |    .
                 .       +--+                 +--+    .
                 .          |   +---------+   |       .
                 .          v   |         v   |       .
                 .        +-o---o-+     +-o---o-+     .
                 .        | SF#3  |     | SF#4  |     .
                 .        +-------+     +-------+     .
                 .   NSH-unaware domain               .
                  . . . . . . . . . . . . . . . . . .
```

SF#1 and SF#5 are NSH-aware and SF#2, SF#3 and SF#4 are NSH-unaware. In the NSH-unaware domain, packets are conveyed in a format supported by SFs which are deployed there.

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

6.1. Purpose

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

- o In a period of transition from legacy SFs to NSH-aware SFs and

- o Supporting multi-tenancy.

6.2. Requirements for IBN

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

- o Saving SPI and SI in transport-layer flow state (refer to Section 3.1.1) and

- o Using unique lower-level paths per upper-level NSH coordinates (refer to Section 3.1.3).

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

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

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

7. Acknowledgements

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

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

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

Ron Parker

Christian Jacquenet

Jie Cao

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

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

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

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

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

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

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

10. References

10.1. Normative References

[I-D.ietf-sfc-control-plane]
Boucadair, M., "Service Function Chaining (SFC) Control Plane Components & Requirements", draft-ietf-sfc-control-plane-06 (work in progress), May 2016.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-05 (work in progress), May 2016.

[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>.

10.2. Informative References

[I-D.ao-sfc-for-dc-interconnect]
Ao, T. and W. Bo, "Hierarchical SFC for DC Interconnection", draft-ao-sfc-for-dc-interconnect-01 (work in progress), October 2015.

[I-D.homma-sfc-forwarding-methods-analysis]
Homma, S., Naito, K., Lopez, D., Stiemerling, M., Dolson, D., Gorbunov, A., Leymann, N., Bottorff, P., and d. don.fedyk@hpe.com, "Analysis on Forwarding Methods for Service Chaining", draft-homma-sfc-forwarding-methods-analysis-05 (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-02 (work in progress), January 2015.

Appendix A. Examples of Hierarchical Service Function Chaining

The advantage of hierarchical service function chaining compared with normal or flat service function chaining is that it can reduce the management complexity significantly. This section discusses examples that show those advantages.

A.1. Reducing the Number of Service Function Paths

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

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

In this example there are five security functions deployed in the Security Domain. The Security Domain operator wants to enforce the five different security policies, and the Optimization Domain operator wants to apply different optimizations (either cache or video optimization) to each of these two types of traffic. If we use

flat SFC (normal branching), 10 SFPs are needed in each domain. In contrast, if we use hierarchical SFC, only 5 SFPs in Security Domain and 2 SFPs in Optimization Domain will be required, as shown in Figure 7.

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

```
             . . . . . . . . . . . .   . . . . . . . . . . . . . .
             . Security Domain     .   .  Optimization Domain  .
             .                     .   .                       .
             .    +-1---[     ]------------------>[Cache  ]------->
             .    |     [ WAF ]    .   .                       .
             .    +-2-->[     ]------------------>[Video Opt.]---->
             .    |                .   .                       .
             .    +-3---[Anti ]------------------>[Cache  ]------->
             .    |     [Virus]    .   .                       .
             .    +-4-->[     ]------------------>[Video Opt.]---->
             .    |                .   .                       .
             .    +-5-->[     ]------------------>[Cache  ]------->
[DPI]--->[CF]---|     [ IPS ]    .   .                       .
             .    +-6-->[     ]------------------>[Video Opt.]---->
             .    |                .   .                       .
             .    +-7-->[     ]------------------>[Cache  ]------->
             .    |     [ IDS ]    .   .                       .
             .    +-8-->[     ]------------------>[Video Opt.]---->
             .    |                .   .                       .
             .    +-9-->[Traffic]---------------->[Cache  ]------->
             .    |     [Monitor]  .   .                       .
             .    +-10->[       ]---------------->[Video Opt.]---->
             . . . . . . . . . . . .   . . . . . . . . . . . . . .
```

The classifier must select paths that determine the combination of Security and Optimization concerns. 1:WAF+Cache, 2:WAF+VideoOpt, 3:AntiVirus+Cache, 4:AntiVirus+VideoOpt, 5: IPS+Cache, 6:IPS+VideoOpt, 7:IDS+Cache, 8:IDS+VideoOpt, 9:TrafficMonitor+Cache, 10:TrafficMonitor+VideoOpt

Figure 6: Flat SFC (normal branching)

```
        . . . . . . . . . . . . . . .    . . . . . . . . . . . . . . . .
        .     Security Domain       .    .   Optimization Domain     .
        .                           .    .                           .
   [CF]---->[  [CF]    IBN      ]---------->[  [CF]   IBN         ]---->
        .    |                  ^   .    .  |                    ^  .
        .    +----->[ WAF ]-----+   .    .  +-->[ Cache ]---------+  .
        .    |                  |   .    .  |                     |  .
        .    +-->[Anti-Virus]---+   .    .  +-->[Video Opt]-------+  .
        .    |                  |   .    .                           .
        .    +----->[ IPS ]-----+   .    . . . . . . . . . . . . . . .
        .    |                  |   .
        .    +----->[ IDS ]-----+   .
        .    |                  |   .
        .    +-->[ Traffic ]----+   .
        .        [ Monitor ]        .
        . . . . . . . . . . . . . .
```

Figure 7: Simplified path management with Hierarchical SFC

A.2. Managing a Distributed Data-Center Network

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

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

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

- o Some service functions are being trialed, introduced or otherwise handle a relatively small amount of traffic. It may be cheaper to manage these service functions in a single central data center and steer packets to the central data center than to manage these service functions in all data centers.

```
                   +-----------+
                   |Central DC |
                   +-----------+
                      ^  ^   ^
                      |  |   |
                  .---|--|---|----.
                /   /   |   |      \
               /   /    |    \      \
     +-----+   /   /     |     \      \    +-----+
     |Local|  |   /      |      \     |    |Local|
     |DC#1 |--|--.       |       .----|----|DC#3 |
     +-----+  |          |            |    +-----+
               \         |            /
                \        |           /
                 \       |          /
                  '----------------'
                         |
                      +-----+
                      |Local|
                      |DC#2 |
                      +-----+
```

Figure 8: Simplify inter-DC SFC management

For large data center operators, one local DC may have tens of thousands of servers and hundred of thousands of virtual machines. SFC can be used to manage user traffic. For example, SFC can be used to classify user traffic based on service type, DDoS state etc.

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

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

Authors' Addresses

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

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

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

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

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

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

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Dapeng Liu
Alibaba Group
Beijing 100022
China

Email: max.ldp@alibaba-inc.com

Ting Ao
ZTE Corporation
No.889,Bibo Rd.,Zhangjiang Hi-tech Park
Shanghai 201203
China

Phone: +86-21-688976442
Email: ao.ting@zte.com.cn

Vu Anh Vu
Soongsil University
369 Sangdo-ro
Seoul, Dongjak-gu 06978
Korea

Email: vuva@dcn.ssu.ac.kr

Service Function Chaining
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2017

S. Kumar, Ed.
L. Kreeger, Ed.
Cisco Systems, Inc.
S. Majee
F5 Networks
W. Haeffner
Vodafone
R. Manur
Broadcom
D. Melman
Marvell

February 22, 2017

UDP Transport for Network Service Header
draft-kumar-sfc-nsh-udp-transport-03

Abstract

This draft describes the transport encapsulation to carry Network Service Header (NSH) over UDP transport protocol. This enables applications and services using NSH to communicate over a simple layer-3 network without topological constraints. It brings down the barrier to deploy NSH by not requiring additional overhead as is typical of overlay encapsulation mechanisms designed on top of UDP.

As a first benefit, this method eases the deployment of Service Function Chaining (SFC) by allowing SFC components to utilize the basic UDP/IP stack available in virtually all network elements and end systems to setup the virtual network and realize Service Function Chains (SFCs).

Copyright Notice

Table of Contents

1. Introduction

NSH is an encapsulation designed to carry SFC specific information and metadata. It is very flexible in providing fixed and variable length encapsulation options while allowing for a high degree of

extensibility. NSH in addition allows for carrying a variety of packets as payload, there by acting as a shim header between the inner payload and the outer transport.

NSH focuses on the application aspect of the encapsulation while leaving the transport mechanisms out of scope. This design choice is meant to allow NSH to be carried on any transport as required by the application and the use cases.

The transport independence aspect of NSH makes it necessary for existing transport protocols or new ones to carry NSH encapsulated packet as a payload. Given that IP networks are ubiquitous with virtually every device, element, node connected to the IP network possessing the ability to support UDP datagram transport over IP layer, it is one of the most basic of the transports to carry NSH.

UDP as a transport provides many benefits which has made it the de-facto choice for creating virtual networks such as VxLAN [RFC7348]. By nature it is a datagram service and trades reliability for simplicity and reduced overhead. It allows for sufficient entropy, for the network to exploit, in load balancing packets across paths in the network. Likewise, end hosts exploit it to distribute packets between the NICs and processor cores, for optimum performance. To this end, network elements and end hosts, both hardware and software, implement specific mechanisms to optimize UDP packet processing.

UDP datagram service and efficient implementations of it in existing networks is thus a forgone conclusion. These benefits among others, coupled with extensibility aspect of NSH - to implement security, header verification, etc., makes UDP a very simple, widely available and foundational choice for transporting NSH encapsulated packets.

This draft describes the considerations for the creation of on-demand point-to-point lightweight UDP tunnels to transport NSH encapsulated packets, hereafter abbreviated as NSH-OVER-UDP.

1.1. Applicability and Operating Environments

NSH is an encapsulation carrying control information and metadata in addition to the packet or frame for service delivery. NSH base header contains the next protocol field to specify the payload type encapsulated. Supported payload types include IPv4, IPv6 and Ethernet, not including the experimental types. UDP as a transport for NSH hence is tunneling IPv4, IPv6 and Ethernet packets or frames encapsulated in NSH.

This draft follows the usage guidelines outlined in [I-D.ietf-tsvwg-rfc5405bis] in specifying the usage of UDP as a

transport for NSH. [I-D.ietf-tsvwg-rfc5405bis] offers guidelines for application and protocol designers to consider and adopt when using UDP as a transport. It primarily focusses on congestion control to prevent congestion collapse and to provide fairness for all users of capacity along the path while also providing other recommendations. In particular, it identifies two types of applicability for the specification of applications, such as this draft: 1) General Internet and 2) Controlled Environment. Former is broadly the specification targeting the use of UDP for applications over the Internet, which seems to have become inevitable for successful applications even when those applications start out in limited networks. The latter on the other hand is the specification targeting the use of UDP for applications in a controlled environment. Controlled environments are assumed to be well coordinated and well managed. Further, such environments have additional means, in the form carrier grade or other tools and hardware to manage congestion rather than rely on application built-in mechanisms.

NSH and more broadly SFC, in its initial specification, targets only a single administrative domain and falls into the applicability type 2: controlled environment. It is assumed that SFC is deployed over a single or even multiple connected IP networks that are all under the same administrative domain or cooperating domains. It is thus assumed that these controlled networks are traffic engineered and manage congestion through external means. Deploying SFC over the Internet and hence the use of UDP to carry NSH over the Internet is out of scope for this draft.

1.2. 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 some terms defined in SFC architecture [I-D.ietf-sfc-architecture] and NSH [I-D.ietf-sfc-nsh] drafts as mere examples for ease of understanding.

3. NSH UDP Overlay Transport Encapsulation

3.1. Stacking And Layering

A NSH encapsulated packet when carried over an UDP transport looks as depicted in Figure 1.

The original payload, L2 frame, L3 packet, NSH OAM message, etc., is first encapsulated in NSH shim header. The NSH encapsulated packet then becomes the payload for the UDP packet carried over an IPv4 or IPv6 network. The UDP header serves as the L4 transport for NSH and its payload.

Although depicted as a layer3 IP over an L2 network, nothing is assumed about how the L3 network is designed and deployed. It is entirely possible for IPinIP or MPLS or other underpinnings.

```
+-------------------------------------------------------------+
|  NSH Payload                                                |
|  (Original L2/L3 frame/packet or other as signaled by NSH)  |
+-------------------------------------------------------------+
|                                                             |
|  Network Service Header (NSH)                               |
+-------------------------------------------------------------+
|                                                             |
|  L4 UDP Header                                              |
+-------------------------------------------------------------+
|                                                             |
|  L3 (IPv4|IPv6) Header                                      |
+-------------------------------------------------------------+
|                                                             |
|  L2 (Ethernet) Header                                       |
+-------------------------------------------------------------+
```

Figure 1: NSH UDP Stack

3.2. NSH UDP Overlay Packet Format

Figure 2 shows the format of the NSH encapsulation transported over UDP.

Rest of the document assumes UDP destination port to be set to NSH-UDP-PORT unless stated otherwise explicitly, when carrying a NSH encapsulated payload packet in UDP transport.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
|  Source Port = XXXXX          |  Dest Port = NSH-UDP-PORT     | UDP
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Length                       |  Checksum                     | Hdr
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
|                                                               | UDP
~  Network Service Header (NSH)                                 ~
|                                                               |  P
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  a
|                                                               |  y
|                                                               |  l
~  NSH Payload                                                  ~  o
|  (Original L2/L3 frame/packet or other as signaled by NSH)    |  a
|                                                               |  d
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
```

Figure 2: NSH UDP Overlay Encapsulation Format

Source Port :
The UDP port number computed to provide entropy. See Section 4.2 for details.

Dest Port :
UDP port number assigned to NSH: NSH-UDP-PORT.

Length :
Length of the UDP payload. This includes both the UDP header and payload as specified in [RFC0768].

Checksum :
UDP checksum computed over the pseudo header, NSH and NSH payload or zero. See Section 4.3

NSH :
The NSH encapsulation header.

NSH Payload :
The original frame or packet being carried or OAM message, etc.

4. UDP Encapsulation Considerations

4.1. UDP Transport End-points

The UDP transport extends between the two end-points involved in carrying the NSH traffic. The control plane provisioning the NSH encapsulation MUST specify the location of the transport destination when using UDP as the transport, such as the IPv4 or IPv6 address of the end-point.

In the case of SFC, this UDP transport extends between two SFC components: Classifier and SFF or any two SFFs or SFF and SF or SFF and SFC-proxy. The destination of the UDP transport is thus the IP address used by these components to receive the NSH encapsulated traffic. When UDP transport is required to carry NSH encapsulated traffic, SFC control plane MUST provision the UDP transport destination and the use of UDP as transport.

4.2. UDP Source Port Considerations

The source port used in the UDP transport SHOULD be computed to provide entropy for load balancing along the transmission path, including network elements such as routers and switches as well as end points such as servers. This behavior may in turn be controlled by local-policy at the encapsulating entity.

The source port number SHOULD stay constant and not change for the flow represented within the NSH payload. This is typically done by computing the source UDP port number as a hash over the invariant part of the NSH payload. This could be IP and UDP or IP and TCP part of the NSH payload when the next-protocol field in NSH base header is set to IPv4, for instance. This avoids inducing packet reordering due to the use of NSH UDP transport.

The recommended selection of source port as per [RFC6335], is the dynamic range: 49152-65535. A number in this range SHOULD be selected to reflect the flow contained in NSH payload.

The source port number SHOULT NOT be set to zero by the UDP transport encapsulating entity to mitigate data injection attacks by off-path devices.

In case of carrying UDP over IPv6, network flow label [RFC6437] SHOULD be used with the same value as set in the UDP source port. This allows devices processing NSH encapsulated UDP packets to ECMP [RFC6438] load balance and route at the IP level as opposed to looking inside the UDP header.

The end point receiving and processing the UDP transported NSH packets SHOULD perform checks to filter packets with invalid source port numbers.

4.3. Checksum Considerations

UDP header checksum is essential to ensure UDP payload is not corrupted. In case of IPv4, IP header checksum enables detection of delivering to the wrong destination. UDP checksum over IPv6 on the other hand enables the detection of UDP payload corruption in addition to the detection of wrong destination. UDP checksum MUST be handled as per [RFC0768] [RFC2460] by both the decapsulator and the encapsulator.

NSH is capable of carrying both IP and non-IP packets. In case of IP packets, NSH payload may already have checksum protection. In such cases the vulnerable portion of the UDP transport carrying NSH is just the NSH header part of the UDP payload. However, given the controlled network environment, this may be low risk and hence checksum protection is optional as per discussion below.

4.3.1. IPv4 Checksum Processing

IPv4 allows for zero checksum and hence the decapsulator MUST accept UDP datagrams received with zero checksum. Checksum in the UDP header MAY be set to zero for performance or other implementation specific reasons by the encapsulator of NSH packet (classifier, SFF, SF-proxy or SF). When a non-zero checksum is set by the encapsulator, it MUST be computed over the IP, UDP headers and the data as defined in the UDP specification [RFC0768]. The receiving entity thus MUST accept a UDP encapsulated NSH packet with non-zero UDP checksum. Receiving entities, of UDP encapsulated NSH packets with non-zero checksum, MUST verify the checksum before accepting the packet.

4.3.2. IPv6 Checksum Processing

IPv6 header does not itself have a checksum but relies on the upper layers such as UDP. UDP over IPv6 hence protects both the source and destination addresses in addition to the payload.

[RFC2460] does not allow the use of zero checksum with UDP. [RFC6935] and [RFC6936] define the guidelines and requirements to be met for using zero checksum with IPv6. Since SFC and NSH are constrained within a single administrative domain, zero checksum method MUST be configurable to override the default option.

The following are the requirements and recommendations for using NSH-OVER-UDP, over an IPv6 transport not performing UDP checksum to verify the integrity of the transport end points.

1. By default, NSH encapsulator node SHOULD use non zero UDP checksum for NSH-OVER-UDP packets.

2. By default, NSH encapsulator node SHOULD NOT send packets with zero checksum for NSH-OVER-UDP packets.

3. By default NSH decapsulator node SHOULD discard NSH-OVER-UDP packets received with zero checksum.

4. NSH encapsulator and decapsulator nodes MUST be configurable to use the zero checksum method for NSH-OVER-UDP packets.

5. When zero checksum method is enabled for use with NSH-OVER-UDP packets, it is RECOMMENDED that it be done for specific IP addresses. The decapsulator MUST check for the validity of the source and destination IP addresses by comparing them against the set of IP address enabled for zero checksum method.

6. NSH encapsulator MAY send both zero and non-zero checksum NSH-OVER-UDP packets to the same destination and the decapsulator nodes MUST receive both zero and non-zero checksum NSH-OVER-UDP packets from the same source

7. A middlebox, if present in the path of NSH-OVER-UDP packets, MUST allow forwarding of NSH-OVER-UDP packets with both zero and non-zero checksum.

8. While using zero checksum, the network administrator MUST ensure that the corruption of packets in the environment is low through means outside the scope of this draft, such as monitoring and analysis of network traffic.

9. Encapsulator and decapsulator components MUST verify the non-zero checksum when in use and provide an integrity mechanism to isolate the cause of corruption when the corruption rate increases.

The above requirements do not change the requirements specified in [RFC2460], further updated in [RFC6935] or, the requirements in [RFC6936] but are adopted for transporting NSH over UDP.

Since NSH is specified for controlled environments, which provides visibility and control over packet corruption in the environment, the

network operator is expected to keep the packet corruption to an acceptably low level. The above requirements further contribute to reducing the corruption rates and hence they are not expected to increase in any way as compared to using UDP win non NSH-OVER-UDP packets in the same environment. Requirements 2, 3 and 5 in section 5 of [RFC6936] are hence satisfied.

Since NSH is an encapsulation header with no requirement on sate maintenance at either the encapsulator or the decapsulator and NSH-OVER-UDP adds no additional state requirements, requirement 4 in section 5 of [RFC6936] is not applicable.

NSH-OVER-UDP has no control feedback mechanism as it does not specify a protocol or additional semantics for its use, requirements 6 and 7 in section 5 of [RFC6936] do not apply.

Since NSH-OVER-UDP is unidirectional, requirement 10 in section 5 of [RFC6936] does not apply.

In summary, NSH-OVER-UDP may use zero checksum method while carried over IPv6 in accordance with the drafts sighted in the above discussion.

4.3.3. UDP-Lite Considerations

NSH when transported over UDP with zero checksum method, either in IPv4 or IPv6 packets, looses the integrity verification provided by the checksum. However, as discussed in Section 4.3.2, deployment in a controlled environment is expected to minimize packet corruption.

NSH payload may consist of packets that may or may not have their own integrity verification mechanisms as in IPv4 or TCP or UDP packets inside NSH. In light of this, if implementations require integrity verification of the payload but want to avoid the redundant integrity checks or, require integrity checks only for the NSH header, should seriously consider UDP-Lite [RFC3828]. UDP-lite shares the UDP name space but uses IP protocol identifier to distinguish itself from UDP.

4.4. Congestion Considerations

Congestion control with a connection less protocol like UDP is a very important consideration to prevent congestion collapse as discussed in depth in [RFC5405] updated by [I-D.ietf-tsvwg-rfc5405bis]. In particular, senders of UDP traffic are expected to control the rate at which packets are sent to a destination in order to minimize the congestions effects along the path to the destination which is shared with other flows between different source and destination tuples.

NSH-OVER-UDP is expected to transport both IP and non IP traffic although former is expected to be the dominant use case. Where IP traffic is carried, it is assumed to be congestion controlled. In such scenarios, additional congestion control in NSH-OVER-UDP is assumed to be unnecessary. However, when non-IP traffic is carried, such as link layer traffic, the rate at which packets are sent to the destination by an NSH-OVER-UDP encapsulator may not be controlled and hence may run into congestion issues in the network impacting throughput of not only other senders and receivers but the throughput of this sender as well. The network operator(s) can minimize or avoid these situations by carful planning to control the rate of transmission of such packets through means outside the scope of NSH-OVER-UDP.

Since NSH-OVER-UDP is expected to be deployed in controlled environments, it is suitable for deployment in such network environments. It MUST NOT be deployed over general Internet unless explicit guarantees are in place to control the sender of such packets to prevent congestion.

The operator of the networks where NSH-OVER-UDP is deployed is expected to impose checks at the egress points of those networks to ensure any traffic that is not congestion controlled does not escape to the Internet.

4.5. MTU and Fragmentation Considerations

Fragmentation severely impacts the performance and efficiency of the elements that process the packet fragments, which includes the routers and middleboxes among others, and the network in general. Fragmentation creates more packets in the network, requires resources in the network elements to buffer and reassemble, which only gets worse if the fragments are re-ordered (for instance they take different paths in the network), adds processing overhead, to name a few disadvantages. Further, it leads to loss of an entire packet and even flow, if a single fragment is lost. A firewall enforcing policies on the packet content requires entire packet to be reassembled and a loss of a fragment results in dropping the all fragments and blocking the corresponding flow.

An application, as a result, SHOULD NOT send packets that exceed the MTU along the path of the packet. This requires Operators of networks deploying NSH-OVER-UDP are RECOMMENDED to configure the MTU of the network to accommodate NSH and UDP transport encapsulation overhead. This is only feasible when the networks are under single administrative domain or co-operating administrative domains or managed networks. Where it is not possible to set the network-wide MTU to accommodate NSH-OVER-UDP packet overhead, NSH-OVER-UDP

encapsulators SHOULD use path MTU (PMTU) discovery to determine the MTU along the path. Ideally, such PMTU discovery SHOULD be performed by the end application and lower the packet MTU. Such a method fails when the packet traverses multiple administrative domains or the Internet as ICMP messages required for successful operation of PMTU are increasingly being filtered.

Within the same administrative domain, PMTU discovery SHOULD be used by the NSH-OVER-UDP encapsulators to determine the MTU along the path. The determined PMTU MUST over-ride the configured MTU. NSH-OVER-UDP encapsulators MUST fragment the packet being encapsulated prior to encapsulating in UDP. This may result in NSH itself spread across multiple fragments in extreme cases and hence reassembly becomes a requirement to process NSH.

When fragmentation is indeed performed by the NSH-OVER-UDP encapsulators, same source port number MUST be used on all the fragments of the same packet.

4.6. Middlebox Considerations

Middle boxes typically build state and have a notion of flow. Policies are often applied by the operator of such networks and middleboxes to the flow in one direction while expecting the same policy to be applied automatically in the opposite direction of the flow by the middlebox. State-full behavior of the middleboxes enables such functionality.

NSH-OVER-UDP creates a point-to-point unidirectional tunnel. NSH-UDP-PORT is the destination port while a random port is chosen as the source port as explained in Section 4.2. Since NSH is deployed in a controlled environment, the operator MUST update the middleboxes to allow packets destined to NSH-UDP-PORT. It may further be constrained to specific source and destination IP addresses.

In case of SFC, NSH is used to steer traffic to middleboxes and the middleboxes are expected to parse NSH-OVER-UDP packets to service the NSH payload packets and hence presence of middle boxes are expected.

4.7. Differentiated Services and ECN Considerations

IP Packets carried as payload of NSH inside NSH-OVER-UDP may have differentiated services (DS) [RFC2475] or ECN [RFC3168] or both markings on them. It becomes important to determine the markings for the encapsulating IP packet such as NSH-OVER-UDP when carrying such a marked packet as payload. [RFC2983] and [RFC6040] discuss DS and ECN topics in depth, respectively, as it applies to tunnels.

5. Acknowledgements

The authors would like to thank David Black, Alia Atlas and others for their feedback, review comments and guidance.

6. IANA Considerations

IANA is requested to assign a well-known UDP port number for the purpose defined in this draft, referred to as NSH-UDP-PORT.

7. Security Considerations

Encapsulating NSH in UDP does not alter the security risk of NSH encapsulation and payload and hence security of the payload is as per [I-D.ietf-sfc-nsh]

NSH-OVER-UDP is expected to predominantly carry IP traffic which is checksumed. In increasing number of cases, as in mobile service provider networks, the traffic is also encrypted. Although it is allowed to use zero UDP checksum with NSH-OVER-UDP, non-zero checksum SHOULD be used to protect against corruption to mitigate privacy concerns.

Use of computed port number for NSH-OVER-UDP source port, as discussed in Section 4.2, provides minimal protection against off-path attacks although it is not a substitute for encryption techniques. However, where source port computation for entropy is disabled a random port SHOULD be selected to mitigate exposure to off-path attacks as described in [RFC6056].

8. References

8.1. Normative References

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-11 (work in progress), February 2017.

[I-D.ietf-tsvwg-rfc5405bis]
Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", draft-ietf-tsvwg-rfc5405bis-19 (work in progress), October 2016.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[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>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <http://www.rfc-editor.org/info/rfc2475>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <http://www.rfc-editor.org/info/rfc2983>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., Ed., and G. Fairhurst, Ed., "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, DOI 10.17487/RFC3828, July 2004, <http://www.rfc-editor.org/info/rfc3828>.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <http://www.rfc-editor.org/info/rfc6040>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <http://www.rfc-editor.org/info/rfc6056>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <http://www.rfc-editor.org/info/rfc6437>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <http://www.rfc-editor.org/info/rfc6438>.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <http://www.rfc-editor.org/info/rfc6935>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <http://www.rfc-editor.org/info/rfc6936>.

8.2. Informative 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.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <http://www.rfc-editor.org/info/rfc6335>.

[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>.

Authors' Addresses

Surendra Kumar (editor)
Cisco Systems, Inc.
170 W. Tasman Dr.
San Jose, CA 95134
US

Email: smkumar@cisco.com

Larry Kreeger (editor)
Cisco Systems, Inc.
170 W. Tasman Dr.
San Jose, CA 95134
US

Email: kreeger@cisco.com

Sumandra Majee
F5 Networks
90 Rio Robles
San Jose, CA 95134
US

Email: S.Majee@F5.com

Walter Haeffner
Vodafone
Ferdinand-Braun-Platz 1
Duesseldorf 40549
DE

Email: walter.haeffner@vodafone.com

Rajeev Manur
Broadcom

Email: rmanur@broadcom.com

David Melman
Marvell

Email: davidme@marvell.com

service function chain
Internet-Draft
Intended status: Standards Track
Expires: April 9, 2016

C. Xie
China Telecom
W. Meng
C. Wang
ZTE Corporation
B. Khasnabish
ZTE TX, Inc.
October 7, 2015

service function chain Use Cases in Broadband
draft-meng-sfc-broadband-usecases-04

Abstract

This document discusses about service function chain use cases in different scenarios of broadband network. The document provides analysis of different solutions and also describes the suitable scenarios that each solution may be deployed in.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 9, 2016.

Table of Contents

1. Introduction

The object of SFC is trying to unload services from legacy devices in traditional network and deal with such services through corresponding service functions which are topologically independent from physical devices.

As increasingly large number of customers, the possibility of deployment SFC in broadband network seems emergency. And this document aims to illustrate the possibly typical and unified service function chains in Broadband Networks and analyze the possible deployments of diverse service function chains in broadband network.

In figure 1, here outlines the possible SFC deployment architecture in Broadband Networks. This architecture tries to simplify and unify the services in CPEs and unloads the services from CPEs to the SFCs in Access Networks to achieve virtual CPE functions. And as well, extracts the services in BNASs and offloads the services from BNASs to the SFCs in Barrier Networks to accomplish virtual BNAS functions. As a result of that, the Internet Service Provider can manage and maintain the whole Broadband Networks more flexibly.

```
                 +-----------+       +-----------+
                 |    Host1  |  ...  |    HostN  |
                 +-----+-----+       +-----+-----+
                       |                   |
                 +-----+-----+       +-----+-----+
                 | classifer |       | classifer |
                 +-----+-----+       +-----+-----+
                       |                   |
                       |---------|---------|
                                 |
                         +-------+---------+
                         |    SFCs/vCPEs   |
          .              +-------+---------+          .
          .                        |                   .
   +-------|---------+   +-------|---------+  +-------|---------+
   |switch/classifier|   |switch/classifier|  |switch/classifier|  ...
   +-------|---------+   +-------|---------+  +-------|---------+
           |                     |                    |
           +---------------------|--------------------+
                                 |
                          +------+------+
                          | SFCs/vBNASs |
                          +------+------+
                                 |
                         --------|--------
                       /         |         \
                      |       Internet      |
                       \         |         /
                         --------|--------
```

Figure 1: SFC Architecture of Broadband Network

2. Convention and Terminology

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

The terms about SFC are defined in [I-D.ietf-sfc-problem-statement].

The terms about CGN/DS-Lite/Lightweight 4o6/MAP/NAT64 are defined in [RFC6888]/[RFC6333]/ [I-D.ietf-softwire-lw4over6]/ [I-D.ietf-softwire-map]/ [RFC6146].

3. Use cases

The following sections highlight some of the most common broadband network use case scenarios and are in no way exhaustive.

3.1. Internet Access from Homes

Figure 2 illustrates an abstract architecture of broadband network, including CPE sitting in home access network, BNAS as broadband access network gateway,CR located in bone network and the Internet.

```
+---+      +------+        +-----+      +---------+
|CPE|------| BNAS |--------| CR  |------| Internet|
+---+      +------+        +-----+      +---------+
```

Figure 2: Architecture of Broadband Network

Also, the Broadband Forum(BBF) is developing a study document(SD-326), which aims to study market requirements and usecases for Flexible Service Chaining. Except that, this document tries to develop more typical usecases in Broadband Networks.

3.1.1. Native IPv4 Network or Native IPv6 Network

```
     ---------------------------------------------------------------------------
->
                   +--- +   +-------+   +-----+   +----------+
                   |    |   |DPI/DFI|   | LB/ |   |URL Filter|---|
                |--| UM |---|  /Qos |---| FRR |---|  /FW/PC  |   |
                |  +----+ | +-------+ | +-----+ | +----------+   |
    +---+  +------+       |           |         |             +-----+     +-----
----+
    |CPE|--| BNAS-|-------|-----------|---------|-------------| CR  |-----| Inte
rnet|
    +---+  +------+                                           +-----+     +-----
----+
```

Figure 3: Native IPv4 Network or Native IPv6 Network

Figure 3 shows possible deployment of SFC in native IPv4 network or native IPv6 network. As UM(users management) service, which is the main service of BNAS device in traditional network, consumes large memory and resources, it seems reasonable to decouple UM service from legacy BNAS device and treat it as a service node , which may include DHCP,AAA functions and some other functions related to users management. And what!_s more, only users subscription messages (protocol messages) go through UM service. Once a user is approved by UM service, the following data flow of this user go through the other service function chain which is assigned to this data flow.

’BNAS-’ means some functions have been unburdened from traditional BNAS,such as user management,Qos,Load Balance and so forth.

Given that SFC is applied in Broadband network, the main SFs may cover: User Management, DPI, DFI, Qos, Load Balance, Fast Reroute, URL Filter,Firewall,Parental Control and so forth. And the possible order is not as strict as above. The upstream/downstream traffic may go through different permutations and combination of these SNs. For example:

SFC1: UM

This SFC stands for the process of subsribers!_ log-in and log-out. All the broadband subscribers!_ log-in messages and log-out messages need go through this SFC. After approved by this SFC, then the users flow can access the Internet or other services.

SFC2: Qos

This SFC shows some bandwidth restrictions or several priority-based schedules are applied to this approved subscriber. Almost each home subscriber has a corresponding subscribed bandwidth, different services from a home have distinctive priority as well. As a result, this is a basic SFC used in internet access from homes.

SFC3: Qos--LB

This SFC extends SFC2, which utilizes load balance to offload approved subscribers!_ flow from an overload path. This is also a typical scenario in broadband network, especially in metropolitan area network.

SFC4: Qos--LB--URL Filter

Based on SFC3, this SFC gives extra restrictions to the content that the approved subscriber wants to access.

SFC5: Qos--Parental Control

This is similar to SFC4,except there is no Load Balance. Another difference is that SFC5 offers some restrictions to downstream traffic in terms of content. SFC5 allows some legal or appropriate contents to flow to subscribers, while some illegal or inappropriate contents are blocking.

3.1.2. IPv4/IPv6 Coexist Network

As showed below in figure 4, the main difference between IPv4/IPv6 native network and IPv4/IPv6 coexist network is whether there exists a NAT funtion. Although in IPv4 native network, there maybe exist NAT44 function as a result of limited IPv4 address, we try to put

this scenario together with other IPv6 transition scenarioes in this section and discuss them in detail.

```
     ---------------------------------------------------------------------------
->
                 +--- +   +-------+   +-----+   +----------+
                 |    |   |DPI/DFI|   | LB/ |   |URL Filter|---|
           |-----|NAT |---|  /Qos |---| FRR |---|  /FW/PC  |   |
           |     +----+ | +-------+ | +-----+ | +----------+   |
+---+  +------+         |           |         |             +-----+   +---------
+
|CPE|--| BNAS-|---------|-----------|---------|-------------| CR  |---| Internet
|
+---+  +------+                                             +-----+   +---------
+
```

Figure 4: IPv4/IPv6 Coexist Network

Whether NAT stands for NAT44 or NAT64 or NAT46 depends on the the Internet Server Provider. It may be NAT44, which reflects the communication between IPv4 private customer and IPv4 public server. Or it may be NAT64, which means the communication between IPv6 customer and IPv4 Server. And where NAT is deployed is the preferance of the Internet Server Provider as well. It may be besides BNAS,which stands for distributed deployment, or besides CR,which represents central deployment.

Above figure 4 just gives a simple example of a possible deployment position in distributed deployment scenario. Actually, there are some other complicated IPv6 transition scenarioes . And this section tries to give some typical examples in IPv4/IPv6 coexist network, and conclude a feasible SFC architecture in IPv4/IPv6 coexist network. Also, in the following sections, the other SFs emphasized in section 3.1.1 are not highlighted, just try to keep the diagram simple and suitable for the draft’s specification.

3.1.2.1. NAT44

Figure 5 illustrates a simple NAT44 scenario how SF-NAT is deployed and how SF-NAT may work.

```
                                  .
                                  :
                                  |
                     ............ |
                  External realm  |
                  ISP network-->  |-------------+
                                  |        ++---|----++
                                  |        | SF-NAT44 |
                                  |        ++---|----++
                                  |-------------+
                              ++--|----+
                     ........... |  BNAS-  |
                 Internal realm  ++------++
                                   |    |
                  ISP network-->   |    |
                                   |    |
                           ++------++  ++------++
                           |  CPE1  |  |  CPE2  |  etc.
                           ++------++  ++------++
```

Figure 5: NAT44

In distributed broadband networks, SNs may be deployed beside BNAS. These SNs may contain or logically connect to SF-NAT and other service functions such as UM,QOS,Load Balance,etc.

Here gives an example of possible SFC in IPv4/IPv6 coexist network, which combines NAT function with the service functions in native IPv4/IPv6 network.

SFC6: Qos--NAT--LB--URL Filter

SFC6 combines NAT function with SFC4, and represents the classical scenario in IPv4/IPv6 coexist network. After customers have subscribed, apply subscriber-based Qos policy, then transform IPv4/ IPv6 address into IPv4 address, and do five-tuple load balance for the outbound traffic.

At last, monitor the outbound traffic and decide whether to permit them to the internet or block them.

After the first packet of an outbound flow has been processed by this SFC, this SFC can do SFP optimization to bypass NAT service function to improve the experience of this subscriber. Then, for the following packets of this outbound flow, the SFF connects to NAT service function can forward them according to the forwarding table which is derived from the NAT service function.

As for the inbound flow of this subscriber, there exists an open issue: how the inbound flow is steered to the same NAT service function or the same SFF which connects to the same NAT service function.

3.1.2.2. DS-Lite

Figure 6 describes a scenario of DS-lite, which completes IPv4 communication between IPv4 private customer and IPv4 server across IPv6 network through tunnels. And the main principle of DS-Lite is to encapsulate IPv4 packets in IPv6 Header and forward this IPv4-in-IPv6 packets to CGN device and enforce NAT function in CGN device. Generally, CGN device resides in BNAS device.

```
                    +-----------+
                    | IPv4 Host |
                    +-----+-----+
                          |
                +---------|---------+
                |       CPE-        |
                +---------|---------+
                          +-----------------+
                          |           +-----|--------+
                          |           | SF-Dslite-Enc|
                          |           +-----|--------+
                          +-----------------+
                         |||
                         |||<-IPv4-in-IPv6 softwire
                         |||
                +--------|||--------+
                |        BNAS-       |
                +--------|||--------+
                          +----------------------+
                          |           +----------|----------+
                          |           |     SF-Dslite-Dec   |
                          |           |        SF-NAT       |
                          |           +----------|----------+
                          +-----------------+
                          |
                          |
                  --------|--------
                /         |         \
               |  IPv4 Internet     |
                \         |         /
                  --------|--------
```

Figure 6: DS-Lite

SFC7: Dslite-Enc---Dslite-Dec---NAT---LB---URL Filter

When the outbound flow are received by the CPE, the CPE sends them to a specific classifier which determines the flow should be forwarded directly or dealed with DS-Lite process. if the flow should be dealed with DS-Lite process, then the classifier sends the datagram within service header encapsulation to Softwire-SN which contains SF-Dslite-Encapsulation instance. In this instance, it fulfils DS-Lite encapsulate and then encapsulates overlay header and forwards this flow to nexthop in the traditional network.

Next, the BNAS- receives the processed flow, the BNAS- sends them to a classifier and finds they are legal flow and need to be dealed with DS-Lite process. then, this flow are forwarded to SF-Dslite-Decapsulation to decapsulate DS-Lite encapsulation. And as well, forwarded to SF-NAT to create and maintain the NAT mapping table for DS-Lite subscriber. SF-Dslite-decapsulation and SF-NAT can reside in one service function or two different service functions. After that, completes the subsequent SFs.

In other words, BNAS-,itself, would decouple DS-lite-related functions to specific service function(s). What!_s more, if SFP optimization function is enabled, BNAS- acts as SFF which connectes to SF-NAT, and derives the NAT/forwarding table from SF-NAT and bypasses SF-NAT to improve the experience of this subscriber.

If deploy SFC7 in this scenario, there also exists a consideration: how to address the relationship between the access side SFC domain and the network side SFC domain. If they are deployed in two different SFC domain, how to cooperate between the SF-Dslite-Encapsulation service function and SF-Dslite-Decapsulation service function. On the other hand, if they are deployed in one big SFC domain, it seems more feasible to carry out this SFC7.

3.2. Internet Access from Enterprises

```
 ------------------------------------------------------------------------------
----->
                                   +-------+   +-----+   +----------+
                                   |DPI/DFI|   | LB/ |   |URL Filter|---|
 +-----+                      |----|  /Qos |---| FRR |---|  /FW/PC  |   |
 |host1|--|  +----------+     |    +-------+ | +-----+ | +----------+   |
 +-----+  |  |URL Filter|  +-----+           |         |             +-----+   +
---------+
          ---|  /FW/PC  |--| SR- |-----------|---------|-------------| CR  |---|
 Internet|
 +-----+  |  +----------+  +-----+                                   +-----+   +
---------+
 |host2|--|
 +-----+
```

Figure 7: Internet access from enterprises

Internet access from enterprises is another network service. They lease some ports or even some devices from Internet Server Providers. In addition to internal s service functions which are situated in the internal enterprise network, there maybe deploy many external ISP!_s service functions which are sitted on the way to the internet. -.And what!_s more, there maybe deploy IPsec along with VPN users for the sake of the security of enterprise network.

Internal service functions may include: Firewall, NAT function, etc.

As for external service functions deployed by ISP, typical service functions are VPN, like L2VPN,L3VPN,IPsec,IPsec VPN etc. Conventionally, there is a NAT function residing on SR, converting VPN traffic to public traffic to access the internet.

In some cases, service providers need to assign differentiated services to VPN users. In other words, different VPN users may go through differentiated SFC. But, VPN traffic are all encapsulated in outer MPLS header or some other transport headers, how the public network elements classify them to different SFCs? At this time, there maybe need create a mapping between VPN ID/VPN Name and corresponding SFC on the service provider edge device.

Other external service functions involved in Internet access from enterprise network maybe similar to home network, for example, DPI,DFI,Qos,Load Balance, URL Filter,Firewall,Parental Control and so on.

SFC8: URL Filter--FW---NAT---Qos---Load Balance----FW

Here, you may see two FW functions. One is in the inner of enterprise, which represents the URL constrains from the perspective of enterprise. While the other one is sitted in the ISP network, out of the inner enterprise, and stands for the URL restrictions from the standpoint of ISP.

3.3. Internet Access from Campuses

TBD

3.4. Added-value Service Access

To promote their primary service, ISP try to provide value-added services to add value to the standard service offering. Here maybe focus on some significant value-added services in broadband network such as IPTV,VOIP,etc.

3.4.1. Destination Address Accounting(DAA)

Figure 8 illustrates a possible deployment of DAA funtion in broadband network.

```
 +-----+
 |host1|--|
 +-----+  | +-------+   +----------+      +-----+   +---------+
          --|  CPE  |---| BRAS+DAA |------| CR  |---| Internet|
 +-----+  | +-------+   +----------+      +-----+   +---------+
 |host2|--|
 +-----+
```

Figure 8: DAA Deployment in broadband network

In this diagram, DAA assists BRAS to accomplish finer-granularity outbound filter or/and inbound filter based on destination IP address. But,in central deployment scenario of DS-Lite, there is a IPv4-in-IPv6 tunnel from CPE to CR. As a result of that, BRAS cannot identify the true IPv4 destination address in this IPv4-in-IPv6 packets. And then, BRAS cannot enforce DAA function to manage the subscriber more flexibly.

SFC9: DAA----Dslite-Enc----Dslite-Dec----NAT

```
          +-----------+       +-----------+
          |    Host1  |       |    Host2  |
          +-----+-----+       +-----+-----+
                |                   |
          +-----|-----+       +-----|-----+
          |   CPE2-   |       |   CPE2-   |
          +-----|-----+       +-----|-----+
                |                   |
                |---------|---------|
                          |
                  +-------|---------+
                  |    SF-DAA       |
                  +-------|---------+
                          |
                  +-------|---------+
                  | SF-Dslite-Enc   |
                  +-------|---------+
                          |
                         |||
                         |||<--softwire
                         |||
                +--------|||--------+
                |        CR         |
                +--------|||--------+
                          +----------------------+
                          |           +----------|-----+
                          |           | SF-Dslite-Dec  |
                          |           |   SF-NAT       |
                          |           +----------|-----+
                          +----------------------+
                          |
                          |
                  --------|--------
                /         |         \
               |       Internet      |
                \         |         /
                  --------|--------
```

Figure 9: DAA + Softwire Deployment in broadband network

3.4.2. IPTV

Figure 10 illustrates a possible deployment of IPTV network via SFC.

```
<-----------------------------------------------------
+----+
|STB1|-------|
+----+       |
             |
+----+   +-------+    +------+
|STB2|---| BNAS1-|----| Qos1 |---------|
+----+   +-------+    +------+         |
             |                         |
+----+       |                         |
|STB3|-------|                         |
+----+                                 |
                                   +---------+     +-----+
+----+                             | DPI/DFI |-----| CDN |
|STB4|-------|                     +---------+     +-----+
+----+       |                         |
             |                         |
+----+   +-------+    +------+         |
|STB5|---| BNAS2-|----| Qos2 |---------|
+----+   +-------+    +------+
             |
+----+       |
|STB6|-------|
+----+
```

Figure 10: IPTV network via SFC

IPTV is a IP multicast service, in which multi-subscribers should receive the same traffic from the multicast source like Content Distribution Network. Supposed there are six IPTV subscribers, from STB1 to STB6, they are located in different districts and they all need to receive traffic from Program 1. A possible SFC abstract here is :

```
SFC10: DPI--Qos1

        |---Qos2
```

In SFC10, as for the inbound traffic, there are two different outputs, Qos1 and Qos2. Firstly, traffic from multicast source go through DPI, which used for detecting whether the multicast traffic are legal or unmalicious. After that, legal traffic propagate to different Qos, and next, each goes through different BNAS- to different STB subscirbers separately.

3.4.3. VoIP/MoIP

TBD

4. Considerations

4.1. Service Function Chain Symmetry

A complete end-to-end access in broadband network should consist of a set of service function instances in a specific order. Such as:

4.2. Deploying consideration

4.2.1. Standalone mode

In broadband networks, service function components are hanging next to routers such as CPEs/BNASs/CRs. All traffics would be received and steered by routers. Routers send the traffic to classifier in which traffic that matches classification criteria is forwarded along a given SFP to realize the specifications of an SFC.

```
                        +-----------+
                        |    Host   |
                        +-----+-----+
                              |
                              |
                     +---------|---------+    +-------------------+
                     |                   |    |    -------------  |
                     |        CPE        ----->   |    SFP      | |
                     |                   <-----   --------------  |
                     +---------|---------+    +-------------------+
                               |
                               |
                       --------|-------
                     /         |         \
                    |   ISP core network  |
                     \         |         /
                       ------- | -------
                               |
                               |
                     +---------|---------+    +-------------------+
                     |                   |    |    -----------    |
                     |       BNAS        |---->   |   SFP     |   |
                     |                   <----|    -----------    |
                     +---------|---------+    +-------------------+
                               |
                               |
                       --------|--------
                     /         |         \
                    |       Internet      |
                     \         |         /
                       --------|--------
```

Figure 11: Standalone mode

Take DS-Lite CGN for example.

Outbound traffic:

In the example shown in Figure X, a datagram received by the CPE from the host at address 10.0.0.1, using TCP DST port 10000, will be translated to a datagram with IPv4 SRC address 192.0.2.1 and TCP SRC port 5000 in the Internet.

When the datagram 1 is received by the CPE, the CPE sent it to a specific classifier which determines the datagram should be forwarded directly or dealed with DS-Lite process. Then the classifier sends the datagram within service header encapsulated to the first element

of SFP. SF-SOFTWIRE encapsulates the datagram in another datagram (datagram 2) and forwards it BACK to CPE over the softwire. The datagram 2 would be sent to the Dual-Stack Lite carrier-grade NAT by CPE.

When the BNAS receives datagram 2, the BNAS sends it to a classifier and find it need to be dealed with DS-Lite process. Then the classifier send the datagram within service header encapsulated to the first element of SFP.

SF-SOFTWIRE decapsulates the datagram 2 to datagram 1 and forwards it SF-NAT, which determines from its NAT table that the datagram received on the softwire with TCP SRC port 10000 should be translated to datagram 3 with IPv4 SRC address 192.0.2.1 and TCP SRC port 5000.

The translated datagram would be also sent back to BNAS for next forwarding.

Inbound traffic:

Figure x shows an inbound message received at the classifer. When the BNAS receives datagram 1, the BNAS sends it to a classifier. Then the classifier sends the datagram within service header encapsulated to the first element of SFP. SF- NAT looks up the IP/ TCP DST information in its translation table. In the example in Figure 3, the NAT changes the TCP DST port to 10000, sets the IP DST address to 10.0.0.1, and it will be sent back to BNAS to forwards the datagram to the softwire. The SF-SOFTWIRE of the CPE decapsulates the IPv4 datagram inbound softwire datagram and forwards it to the host.

4.2.2. Directly connecting mode

There is another mode to deploy service function components. In broadband home networks, service function components are directly connected to the network. They are connected straight to a BNAS or Routers.

Under this scenario, it seems like more costly than standalone mode during transition period.

```
                      |  +-----------+
                     out |    Host   |
                      |  +-----+-----+
                      v        |
                     +---------|---------+     +-------------+
                     |                   |-out->classifier A |
                     |                   |     +------|------+
                     |        CPE        |            |
                     |                   |            |
                     |                   |           out
                     +---------/\--------+            |
                               ||                     |
                               +<===== in =====+------v------+
                                               |             |
                                               |     SFP  A  |
                                               |             |
                               +<----- out-----+------/\-----+
                               |                      ||
                     +---------v---------+            ||
                     |                   |            ||
                     |                   |            ||
                     |       BNAS        |            ||
                     |                   |     +------||-----+
                     |                   |==in=>classifier B |
                     +---------|---------+     +-------------+
                       --------|--------   /\
                     /         |         \ ||
                    |    METRO NETWORK   | in
                     \         |         / ||
                      ---------^--------
                               .
                               .
                     +---------+---------+
                     |                   |
                     |                   |    +-------------+
                     |         CR        |    |    SFP N    |
                     |                   |    +-------------+
                     |                   |
                     +-------------------+
```

Figure 12: Directly connecting mode

Take NAT44 for example.

Outbound traffic:

For directly connecting mode, the difference in dealing with traffic

is whether the network steer the traffic loopback. That means service function node could send datagrams directly to the next hop.

For example, when the outbound datagram is received by the BNAS and processed by classifer A and SF-NAT which forward the processed datagram straight next to router.

Inbound traffic:

It is quite similar with the process of dealing with outbound traffic. when the inbound datagram is received by the router and processed by classifer B and SF-NAT which forward the processed datagram straight next to NAT BNAS.

4.3. Pool consideration

In traditional networks, pools are configured in router one by one. Pool configuration means these IP addresses in each pool MUST be advertised for creating forward routing path to ensures that the message is routed to the correct target, especially to inbound traffic. Thus, pool location is a problem we must face to in SFC framework.

In standalone mode shown in figure 6, pool could be configured in the classifier beside gateway and advertised by the gateway itself. The classifier would assign IP addresses to service functions for creating mapping table. Both-bound traffic should be forward to gateway first and then for NAT treatment in relative service function components.

In Directly connecting mode shown in figure 7, pool could be configured in classifier B and advertised by classifier B for creating inbound routing path.

There is a mechanism to manage the address pools centrally. Pools could be assigned to classifiers by management server which is handled by Operators centrally.

4.4. NAT traversal

TBD

4.5. Unify home router

TBD

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

TBD

7. Normative References

[I-D.ietf-sfc-problem-statement]
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-13 (work in progress), February 2015.

[I-D.ietf-softwire-lw4over6]
Cui, Y., Qiong, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-ietf-softwire-lw4over6-13 (work in progress), November 2014.

[I-D.ietf-softwire-map]
Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, "Mapping of Address and Port with Encapsulation (MAP)", draft-ietf-softwire-map-13 (work in progress), March 2015.

[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>.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <http://www.rfc-editor.org/info/rfc2865>.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <http://www.rfc-editor.org/info/rfc6052>.

[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>.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <http://www.rfc-editor.org/info/rfc6333>.

[RFC6519] Maglione, R. and A. Durand, "RADIUS Extensions for Dual-Stack Lite", RFC 6519, DOI 10.17487/RFC6519, February 2012, <http://www.rfc-editor.org/info/rfc6519>.

[RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888, April 2013, <http://www.rfc-editor.org/info/rfc6888>.

Authors' Addresses

Xie Chongfeng
China Telecom
Room 502, No.118, Xizhimennei Street
Beijing
China

Email: xiechf01@gmail.com,xiechf@ctbri.com.cn

Wei Meng
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: meng.wei2@zte.com.cn,vally.meng@gmail.com

Cui Wang
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: wang.cui1@zte.com.cn

Bhumip Khasnabish
ZTE TX, Inc.
55 Madison Avenue, Suite 160
Morristown, New Jersey 07960
USA

Email: bhumip.khasnabish@ztetx.com

SFC Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

D. Migault, Ed.
Ericsson
C. Pignataro
T. Reddy
Cisco
C. Inacio
CERT/SEI/CMU

October 19, 2015

SFC environment Security requirements
draft-mglt-sfc-security-environment-req-00.txt

Abstract

This document provides environment security requirements for the SFC architecture. Environment security requirements are independent of the protocols used for SFC - such as NSH for example. As a result, the requirements provided in this document are intended to provide good security practices so SFC can be securely deployed and operated. These security requirements are designated as environment security requirements as opposed to the protocol security requirements.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 21, 2016.

Table of Contents

1. Requirements notation

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

2. Introduction

This document provides environment security requirements for the SFC architecture [I-D.ietf-sfc-architecture]. Environment security requirements are independent of the protocols used for SFC - such as NSH [I-D.ietf-sfc-nsh]. As a result, the requirements provided in this document are intended to provide good security practice so SFC can be securely deployed and operated. These security requirements are designated as environment security requirements as opposed to the

protocol security requirements. This document is built as follows. Section [SFC Environment Overview] provides an overall description of the SFC environment with the introduction of the different planes (SFC Control Plane, the SFC Management Plane, the Tenant's user Plane and the SFC Data Plane). Section [Threat Analysis] describes potential threats to the SFC architecture. Section [Plane Isolation] provides recommendations to limit the attack surface for outsider's attacks. More specifically, it describes how to contain the SFC Data Plane and control access to the SFC Control(?) Plane outside of the SFC Data Plane. Section [SFC Data Plane Requirements] provides recommendations and requirements on how to limit the attack surface for an attacker inside the SFC Data Plane.

This document assumes the reader is familiar with the SFC architecture defined in [I-D.ietf-sfc-architecture].

3. Terminology and Acronyms

- Tenant: A tenant is one organization that is using SFC. A tenant may use SFC on one's own private infrastructure or on a shared infrastructure.

- Tenant's User Data Plane: The tenant may be using SFC to provide service to its customers or users. The communication of these users is designated as Tenant's user Data Plane and includes all communications involving the tenant's users. As a result, if a user is communicating with a server or a user from another domain, the communication with that tenant's user is part of the Tenant's Users Data Plane.

4. SFC Environment Overview

This section provides an overview of SFC. It is not in the scope to this document to provide an explicit description of SFC. Instead, the reader is expected to read [I-D.ietf-sfc-architecture], [I-D.ietf-sfc-control-plane] and other SFC related documents.

Service Function Chaining (SFC) architecture is defined in [I-D.ietf-sfc-architecture]. This section briefly illustrates the main concepts of the SFC architecture and positions the architecture within an environment.

```
                      SFC Control Plane
                                          ^             ^    ^
   -------------------------|---------------|-------------|----|--------
               |         C1 |            C3 |          C2 |    | C4
               |            |            +-----+          |    |
               |            |            |  SF |          |    |
               |            |            +-----+          |    |
               |            v               |             |    |
               |       +------------+     +-----+      +-----+ |
               |    +->|    SFC     |---->| SFF |----->| SFF |------+
    SFC        |    |  | Classifier |<----|     |<-----|     | |    |
    Management |    |  +------------+     +-----+      +-----+ |    |
    Plane      |    |         ^                           |    |    |
               |    |         |                      +-----------+   |
               |    |         |                      | SFC Proxy |   |
               |    |         |                      +-----------+   |
               |    |         |                            |         |
               |    |         |                         +------+     |
               |    |         v                         |  SF  |     |
               |    |    +---------+                    +------+     |
               |    |    |  SFC 2  | <------------ SFC 1 ------->   |
               |    |    +---------+                                 |
               |    |            SFC Data Plane                      |
   -----------------|-----------------------------------------------|--
       SFC incoming |                                  SFC outgoing |
       Data traffic |                                  Data traffic v
```

SFC Tenant’s Users Data Plane

SFC Environment Overview

SFC defined a Service Function Path (SFP) which is an ordered set of Service Functions (SF) applied to packets. SFP is defined at the SF level. A SF may be performed by different instances of SF located at different positions. As a result, a specific packet may pass through different instances of SFC. The ordered set of SF instances a packet goes through is called the Rendered SF Path (RSFP).

Upon the receipt of an incoming packet from the tenant’s user, the SFC Classifier determines, according to Classifiers, which SFP is associated to that packet. The packet is forwarded from Service Function Forwarders (SFF) to SFF. SFF are then in charge of forwarding the packet to the next SFF or to a SF. Forwarding decisions may be performed using SFP information provided by the SFC Encapsulation. As described in [I-D.ietf-sfc-nsh] the SFC Encapsulation contains SFP information such as the SFP ID and Service Index and eventually (especially for the MD-2 in NSH) some additional metadata. SF may be SFC aware or not. In the case the SFC functions

are not SFC aware, a SFC Proxy performs the SFC Decapsulation (resp. SFC Encapsulation) before forwarding the packet to the SF (resp. after receiving the packet from the SF).

The environment associated to SFC may be separated into three main planes:

- SFC Management Plane and Control Plane are defined in [I-D.ietf-sfc-control-plane].

- SFC Data Plane consists in all SF components as well as the data exchanged between the SF components. Communications between SF components includes the packet themselves, their associated metadata, the routing logic - similar to RIB - or SF logic, i.e. what they retuned values are for example.

- SFC Tenant’s Users Data Plane consists in the traffic data provided by the different users of the tenants. When a user is communicating with a server or another user -eventually from another administrative domain - , the communication belongs to the SFC Tenant’s Users Data Plane whenever packets are provided by the server of by the user.

5. Threat Analysis

This section describes potential threats the SFC Data Plane may be exposed. The list of threats is not expected to be complete.

Attacks may be performed from inside the SFC Data Plane or from outside the SFC Data plane, in which case, the attacker is in at least one of the following planes: SFC Control Plane, SFC Management Plane or SFC Tenants’ Users Plane. Some most sophisticated attacks may involve a coordination of attackers in multiple planes.

5.1. Attacks performed from the SFC Control Plane

Attacks related to the control plane have been detailed in section 5 of [I-D.ietf-sfc-control-plane].

5.2. Attacks performed from the SFC Management Plane

Attacks performed on the SFC Management Plane are similar to those performed from the SFC Control Plane. The main difference is that the SFC Management Plan provides usually a greater control of the SFC component that the SFC Control Plane.

In addition, the actions performed by the SFC Management Plane have fewer restrictions, which means it may be harder to enforce strong control access policies.

5.3. Attacks performed from the Tenant’s Users Plane

The SFC Tenant’s User Plane is not expected to have fine access control policies on the packets sent or received by users. Unless they are filtered, all packets are good candidate to the SFC Classifier. This provides the user some opportunities to test the behavior of the SFC.

In addition, the Tenant’s Users Plane is not controlled by the SFC Tenant, and users may initiate communications where both ends - the client and the server- are under the control of the same user. Such communications may be seen as user controlled communications (UCC).

UCC may enable any user to monitor and measure the health of the SFC. This may be an useful information to infer information on the tenant’s activity or to define when a DoS attack may cause more damage. One way to measure the health or load of the tenant’s SFC is to regularly send a packet and measure the time it takes to be received, in order to estimate the processing time within the SFC.

UCC may enable any user to test the consistency of the SFC. One example of inconsistency could be that SFC decapsulation is not performed - or inconsistently performed - before leaving the SFC, which could leak some metadata with private information. For example, a user may send spoofed packet. Suppose for example, that a request HTTP GET video.example.com/movie is received with some extra header information such as CLIENT_ID: 1234567890, or CLIENT_EMAIL: client@example.foo. If these pieces of information are derived from the source IP address, the attacker may collect them by changing the IP address for example. In this case, the spoofed packets as used to collect private and confidential information of the tenant’s users. Note that such threat is not specific to SFC, and results from the combination of spoofed IP and non-authenticated IP address are used to identify a user. What is specific to SFC is that metadata are likely to carry multiple pieces of information - potentially non-authenticated - associated to the user. In the case above, meta-data is carried over the HTTP header. Inserting the metadata in the HTTP header may be performed by a SF that takes its input from the SFC encapsulation. In addition, SFC encapsulation may also leak this information directly to a malicious node if that node belongs to the SFC plane. In this later case, the user builds on the top of and intrusion to the SFC Data Plane that is detailed later.

In some case, spoofed packet may impersonate other’s tenants. Suppose for example that the same infrastructure is used by multi tenants, and which are identified by the IP address of their users. In this case, spoofing an IP address associated to another tenant may be sufficient to collect the information confidential and private information.

Similarly, UCC may enable any user to infer packet has been dropped or is in a loop. Suppose a user send a spoofed packet and receives no response. The attacker may infer that the packet has been dropped or is in a loop. A loop is expect to load the system and sending a "well known packet" over the UCC and measuring the response time may determine whether the packet has been dropped or is in a loop.

Correlation of time measurement and spoofed packet over a UCC may provide various type information that could be used by an attacker.

- The attacker may correlate spoofed packet and time measurement in order discover the SFC topology or the logic of the SFC Classifier. Typically, it may infer when new SFs are placed in the SFC for example. In addition, as metadata are placed in band, the time response may also provide an indication of the size of the metadata associated to the packet. The combination of these pieces of information may help an attacker to orchestrate a future attack on a specific SF either to maximize the damages or to collect some metadata - like identification credentials.

- The attacker may also define the type of packets that require the SFC the more processing. Additional processing may be due a large set of additional metadata that require fragmentation, some packets that are not treated in a coherent and consistent manner within the SFC. Such information may be used for example to optimize a DoS attack. In addition, it could also be used in order to artificially increase the necessary resource of the Tenant in order to increase the cost of operation for running its service.

Time measurement and spoofed packet in combination with variable query rate over a UCC may provide information on the orchestration of the SFC itself. For example, the user may be able to detect when elasticity mechanisms are triggered.

An attacker may be able to leverage the knowledge that SFC is in use by specific carriers to effect the processing of data using the SFC system as a processor in the attack. This leads to a number of potential weaknesses in the Internet ecosystem.

An attacker may be able to characterize the type of client platforms using a web site by carefully crafting data streams that will be modified by the SFC system versus client systems that would view web data unmodified. For example, leveraging SFC and carefully crafted data, a malicious web site operator may be able to create a particularly formatted common file that when modified by a cellular operator for bandwidth savings creates a file that may crash, (creating a DoS attack) on a select set of clients. Clients not accessing that web site using the same RSFP would not experience any issues. Additionally, external examination of the malicious site would not demonstrate any malicious content, relying on the SF to modify the content.

A well crafted site could potentially leverage the variances of functionality from different RSFPs in order to GEO locate a user. An example would be creating an image file which when recompressed creates image artifacts rendering the image unusable, but allowing the user to respond to such an event, thereby letting the web site operator know the user has potentially moved from a higher to lower bandwidth network location within the area of a specific network operator.

5.4. Attacks performed from the SFC Data Plane

This section considers an attacker has been able to take control of an SFC component. As a result, the attacker may become able to modify the traffic and perform, on-path attacks, it may also be able to generate traffic, or redirect traffic to perform some kind of Man-in-the-middle attacks. This is clearly a fault, and security policies should be set to avoid this situation. This section analyses in case this intrusion occurs, the potential consequences on the SFC.

The traffic within the SFC Data Plane is composed of multiple layers. The traffic is composed of communications between SFC components. The transport between the SFC component is the transport protocol and is not considered in the SFC. It can typically be a L2 transport layer, or an L3 transport layer using various encapsulation techniques (vLAN, VxLAN, GRE, IPsec tunnels for example). The transport layer carries SFC Encapsulated that are composed of an SFC Encapsulation envelope that carries metadata and a SFC payload that is the actual packet exchanged between the two end points.

As a result, attacker may use the traffic to perform attacks at various layers. More specifically, attacks may be performed at the transport layer, the SFC Encapsulation layer or the SFC payload layer.

- Attacks performed at the transport layer may be related to SFC in the sense that illegitimate SFC traffic could be provided to the SF. Typically, a malicious node that is not expected to communicate with that SF may inject packets into the SFC, such malicious node may eventually spoof the IP address of legitimate SF, so the receiving SF may not be able to detect the packet is not legitimate. Threats related to IP spoofing are described in [RFC6959] and may be addresses by authenticated traffic (e.g. using IPsec). Such threats are not related to SFC even though they may impact a given SF.

- the SFC Encapsulation as well as the SFC payload are usually considered as input by a SF. As such they may represent efficient vector of attacks for the SF. Attacks performed through SFC payload are similar as the ones described in the Tenant’s Users Data Plane section. As a result, such attacks are not considered in this section, and this section mostly considers attacks based on the SFC Encapsulation and malicious metadata.

When an attacker is within the SFC Data Plane, it may have a full or partial control of one SF component in which case, the attacker is likely to compromise the associated SFCs. It could for example, modify the expected operation of the SFC. Note that in this case, the SFC may be appropriately provisioned and set, however, the SFC does not operate as expected this may only be detected by monitoring and auditing the SFC Data Plane.

Although traffic authentication may be performed at various layers L2 L3 or at the SFC Encapsulation layer, this section considers the SFC traffic. As a result, the SFC traffic is authenticated if the SF is able to authenticate the incoming SFC packet.

When SFC traffic is not authenticated, an attacker may inject spoofed packet in any SFC component. The attacker may use spoofed packet to discover the logic of the SFC. On the other hand, the attacker may also inject packet in order to perform DoS attack via reflection. In fact, some SF may carry large metadata, which may provide a vector for amplification within the SFC Data Plane and thus either load the network or the next SF. Note that amplification may be generated by metadata, the SFC payload, and the attacker may replay packets or completely craft new packets. In addition, the attacker may choose a spoofed packet to increase the CPU load on the SFC components. For example, it could insert additional metadata to generate fragmentation. Similarly, it may also insert unnecessary metadata that may need to be decapsulated and analyzed even though they may not be considered for further actions. Spoofed packet may not only be generated to attack the SFC component at the SFC layer. In fact

spoofed packet may also target applications of the SF. For example an attacker may also forge packet for HTTP based application - like a L7 firewall - in order to perform a slowloris [SLOWLORIS] like attack. Note that in this case, such attacks are addressed in the Tenant's Users Data Plane section. The specificity here is that the attacker has a more advanced understanding of the processing of the SFC, and can thus be more efficient.

When SFC traffic is not authenticated, an attacker may also modify on-path the packet. By changing some metadata contained in the SFC Encapsulation, the attacker may test and discover the logic of the SFF. Similarly, when the attacker is aware of the logic of a SFC component, the attacker may modify some metadata in order to modify the expected operation of the SFC. Such example includes for example redirection to a SF which could result in overloading the SF and overall affect the complete SFC. Similarly, the attacker may also create loops within the SFC. Note that redirection may not occur only in a given SFC. In fact, the attacker may use SFC branching to affect other SFC. Another example would also include a redirection to a node owned by the attacker and which is completely outside the SFC. Motivation for such redirection would be that the attacker has full administrator privileges on that node, whereas it only has limited capabilities on the corrupted node. Such attack is a man-in-the-middle attack. The important thing to note is that in this case the traffic is brought outside the legitimate SFC domain. In fact, performing a man-in-the-middle attack as described above means that the SFC domain has been extended. This can be easily performed in case all node of the data center or the tenant's virtual network is likely to host a SFC component. A similar scenario may also consider that the traffic could be redirected outside the data center or the tenant's virtual network if the routing of firewall rule enables such policies.

A direct consequence is that a corrupted SFC component may affect the whole SFC. This also means that the trust of a given SFC decreases with the number of SF involved as each SF presents a surface of attack.

An attacker may also perform passive attacks by listening to traffic exchanged throughout the SFC Data Plane. Such attacks are described in [RFC7258]. Metadata are associated to each packet. These metadata are additional pieces of information not carried in the packet and necessary for each SF to operate. As a result, metadata may contain private information such as identifiers or credentials. In addition, observing the traffic may provide information on the tenant's activity. Note that encryption only may not prevent such attacks, as activity may be inferred by the traffic load.

6. Plane Isolation Requirements

Plane Isolation consists in limiting the surface of attack of the SFC Data Plane by controlling the interfaces between the SFC Data Plane and the other planes.

Complete isolation of the planes is not possible, as there are still some communications that must be enabled in order to benefit from the benefits of SFC. As a result, isolation should be understood as enabling communications between planes in a controlled way.

This section lists the recommendations so communication between planes can be controlled. This involves controlling communications between planes as well as controlling communication within a plane.

The requirements listed below applies to all planes, whereas the following subsection are more specific to each plane, providing recommendations on the interface with the SFC Data Plane.

REQ1: In order to increase isolation it is recommended that every plane communicates with another plane using a dedicated interface. In our case, the SFC Management Plane, the SFC Control Plane and the SFC Data Plane SHOULD use dedicated networks and dedicated interfaces. Isolation of inter-plane communication may be enforced using different ways. How isolation is enforced depends on the type of traffic, the network environment for example, and within a given SFC architecture different techniques may be used for the different planes. One way to isolate communications is to use completely different network on dedicated NICS. On the other hand, depending on the required level of isolation, a logical isolation may be performed using different IP addresses or ports with network logically isolated - that is using for example different VxLAN, or GRE tunnels. In this case, isolation relies on the trust associated to the different switches and router. In case of a lake of trust on the on-path elements, authenticated encryption may be used to provide a logical isolation. With authenticated encryption, trust is placed on the end points. Note also that encryption can also be used in combination of other isolation mechanisms in order to increase the level of isolation.

REQ2: Activity on each interface between planes MUST be monitored and regularly audited.

REQ3: Each interface between planes MUST be provided means to filter traffic or rate-limit the traffic. Filtering and rate-

limiting policies may be finer grained and may apply for a subset of traffic.

6.1. SFC Control Plane Isolation

In order to limit the risks of an attack from the SFC Control Plane, effort should be made in order to restrict the capabilities and the information provided by the SFC Data Plane to the SFC Control Plane to the authorized tenants only. In this case the authorized tenants are the users or organizations responsible for the SFC domain.

REQ4: Tenants of the SFC Control Plane SHOULD authenticate in order to prevent tenant’s usurpation or communication hijacking.

REQ5: Communications between SFC Control Plane and the SFC Data Plane MUST be authenticated and encrypted in order to preserve privacy. The purpose of encryption in this case prevents an attacker to be aware of the action performed by the SFC Control Plane. Such information may be used to orchestrate an attack - especially when SFC component report their CPU/ network load.

REQ6: Strong access control policies SHOULD be enforced. Control SHOULD be performed on the engaged resource (e.g. CPU, memory, disk access for example) and SHOULD be associated explicitly to authorized tenants. By default, a tenant SHOULD be denied any access to resource, and access SHOULD be explicit.

When possible, the use of API is recommended in order to limit the scope of possible interactions between the SFC Control Plane and the SFC Data Plane. This is one way to limit the possibilities of the tenants. In addition, each of these actions should be associated an authorized tenant, as well as authorized parameters.

REQ7: Audit SHOULD be performed regularly to check access control policies are still up-to-date and prevent non-authorized users to control the SFC Data Plane.

6.2. SFC Management Plane Isolation

The requirements for the SFC Control Plane and SFC Management Plane are similar. The main difference of the interfaces between the SFC Management Plane and the SFC Control Plane is that it is less likely that APIs could be used to configure the different SFC components. As a result, users of the SFC Management Plane are likely to have a broader and wider control over the SFC component.

REQ8: it is REOMMENDED to enforce stronger authentication mechanisms (for example relying on hardware tokens or keys) and to limit the scope of administrative roles on a per component basis.

REQ9: SFC Control Plane and SFC Management Plane may present some overlap. Each SFC component MUST have clear policies in case these two planes enter in conflict.

6.3. Tenant's Users Data Plane Isolation

The Tenant's Users Data Plane is supposed to have less restricted access control than the other SFC Management Plane and SFC Control Planes. A typical use case could be that each tenant are controlling and managing the SFC in order to provide services to their associated users. The number of users interacting with the SFC Data Plane is expected to be larger than the number of tenants interacting with the SFC Control and SFC Management Planes. In addition, the scope of communications initiated or terminating at the user end points is likely to be unlimited compared to the scope of communications between the tenants and the SFC Control Plane or SFC Management Plane. In such cases, the tenant may be provided two roles. One to grant access to the SFC, and another one to control and manage the SFC. These two roles should be able to interact and communicate.

REQ10: Users SHOULD be authenticated, and only being granted access to the SFC if authorized. Authorization may be provided by the SFC itself or outside the SFC.

REQ11: Filtering policies SHOULD prevent access to a user, or traffic when a malicious behavior is noticed. A malicious activity may be noticed once a given behavioral pattern is detected or when unexpected load is monitored in the SFC Data Plane.

REQ12: Tenant's User Plane SHOULD be monitored, in order to detect malicious behaviors.

REQ13: When SFC is used by multiple tenants, each tenant's traffic SHOULD be isolated based on authenticated information. More specifically, the use of a Classifier that can easily be spoofed like an IP address SHOULD NOT be used.

7. SFC Data Plane Requirements

This section provides requirements and recommendation for the SFC Data Plane.

REQ14: Communications within the SFC Data Plane MUST be authenticated in order to prevent the traffic to be modified by an attacker.

As a result, authentication includes the SFC Encapsulation as well as the SFC payload.

REQ15: Communication MUST NOT reveal privacy sensitive metadata.

REQ16: The metadata provided in the communication MUST be limited in in term of volume as to limit the amplification factor as well as fragmentation.

REQ17: Metadata SHOULD NOT be considered by the SFF for forwarding decision. In fact, the inputs considered for switching the packet to the next SFF or a SF should involve a minimum processing operation to be read. More specifically, these inputs are expected fixed length value fields in the SFC Encapsulation header rather than any TLV format.

REQ18: When multiple tenants share a given infrastructure, the traffic associated to each tenant MUST be authenticated and respective Tenant’s Users Planes MUST remain isolated. More specifically, if for example, a SFC Classifier is shared between multiple tenants. The Classifier used to associate the SFC MUST be authenticated. This is to limit the use of spoofed Classifiers. In any case, the SFC component that receives traffic from multiple tenants is assumed to be trusted.

REQ19: Being a member of a SFC domain SHOULD be explicitly mentioned by the node and means should be provided so the SFC domain the node belongs to may be checked. Such requirement intends to prevent a packet to go outside a SFC domain, for example in the case of a man-in-the-middle attacks, where a redirection occurs outside the SFC domain. It is expected that most deployment will rely on border / port mechanisms that prevent outsider users from injecting packets with spoofed metadata. Although such mechanisms are strongly recommended to deploy, in case of failure, they do not prevent man-in-the-middle attack outside the SFC domain.

In addition, the following operational requirements have been identified:

REQ20: SFC components should be uniquely identified and have their own cryptographic material. In other words the use of a shared secret for all nodes SHOULD NOT be considered as one corrupted node would be able to impersonate any node of the SFC Data Plane. This is especially useful for audit.

REQ21: Activity in the SFC Data Plane MUST be monitored and Audit regularly.

REQ22: Isolate the Plane with border and firewall rules.

8. Additional Requirements

REQ23: SFC Encapsulation SHOULD carry some identification so it can be associated to the appropriated SFP as well as its position within the SFC or SFP. Indicating the SFP ID may be sufficient as long as a SFP can uniquely be associated to a single SFC. Otherwise, the SFC should be also indicated. This is especially useful for audit and to avoid traffic coming from one SFC to mix with another SFC.

REQ24: SFC Encapsulation MUST be integrity protected to prevent attackers from modifying the SFP ID.

9. Security Considerations

10. Privacy Considerations

11. IANA Considerations

12. Acknowledgments

The authors would like to thank Joel Halpern for his valuable comments.

13. References

13.1. Normative References

[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>.

[RFC6959] McPherson, D., Baker, F., and J. Halpern, "Source Address Validation Improvement (SAVI) Threat Scope", RFC 6959, DOI 10.17487/RFC6959, May 2013, <http://www.rfc-editor.org/info/rfc6959>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

13.2. Informative References

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-01 (work in progress), July 2015.

[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-control-plane]
Li, H., Wu, Q., Huang, O., Boucadair, M., Jacquenet, C., Haeffner, W., Lee, S., Parker, R., Dunbar, L., Malis, A., Halpern, J., Reddy, T., and P. Patil, "Service Function Chaining (SFC) Control Plane Components & Requirements", draft-ietf-sfc-control-plane-00 (work in progress), August 2015.

[SLOWLORIS]
Wikipedia, "Slowloris", <https://en.wikipedia.org/wiki/Slowloris_%28software%29>.

Authors' Addresses

Daniel Migault (editor)
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 514-452-2160
Email: daniel.migault@ericsson.com

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

Phone: +1 919-392-7428
Email: cpignata@cisco.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Bangalore, Karnataka 560103
India

Phone: +91 9886
Email: tireddy@cisco.com

Christopher Inacio
CERT, Software Engineering Institute, Carnegie Mellon University
4500 5th Ave
Pittsburgh, PA 15213
USA

Phone: +1 412-268-3098
Email: inacio@cert.org

Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: May 7, 2016

J. Napper
S. Kumar
Cisco Systems, Inc.
P. Muley
W. Hendericks
Alcatel-Lucent

November 4, 2015

NSH Context Header Allocation -- Mobility
draft-napper-sfc-nsh-mobility-allocation-02

Abstract

This document provides a recommended allocation of the mandatory fixed context headers for a Network Service Header (NSH) within the mobility service provider network context. NSH is described in detail in [ietf-sfc-nsh]. This allocation is intended to support uses cases as defined in [ietf-sfc-use-case-mobility].

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 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Service function chaining provides a mechanism for network traffic to be forced through multiple service functions in a sequence. Metadata can be useful to service functions. Network Service Headers (NSH) provides support for carrying shared metadata between service functions (and devices) using 4 fixed-length 32-bit context headers as defined in [ietf-sfc-nsh]. NSH is then encapsulated within an outer header for transport.

This document provides a recommended default allocation scheme for the fixed-length context headers in the context of service chaining within fixed and mobile broadband service provider networks. Supporting use cases describing the need for a metadata header in these contexts are described in [ietf-sfc-use-case-mobility]. This draft does not address control plane mechanisms.

2. Definition Of Terms

This document uses the terms as defined in [RFC7498] and [RFC7665].

3. Network Service Header (NSH) Context Headers

In Service Function Chaining, the Network Service Header is composed of a 4-byte base header (BH1), a 4-byte service path header (SH1) and four mandatory 4-byte context headers (CH1-CH4) as described in [ietf-sfc-nsh].

```
 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 = 0x01| Next Protocol | BH1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Service Path ID                      | Service Index | SH1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Mandatory Context Header 1                     | CH1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Mandatory Context Header 2                     | CH2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Mandatory Context Header 3                     | CH3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Mandatory Context Header 4                     | CH4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Network Service Header - MD Type 0x01

4. Recommended Mobility Context Allocation

The following context header allocation provides information to support service function chaining in a mobile service provider network as described in [ietf-sfc-use-case-mobility].

The set of context headers can be delivered to service functions that can use the metadata within to enforce policy, communicate between service functions, provide subscriber information and other functionality. Several of the context headers are typed allowing for different metadata to be provided to different service functions or even to the same service function but on different packets within a flow. Which metadata are sent to which service functions is decided in the SFC control plane and is thus out of the scope of this document.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| R | Sub | Tag |                Context ID                   | CH1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Sub/Endpoint ID                         ~ CH2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                   Sub/Endpoint ID (cont.)                     | CH3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          ServiceTag                           | CH4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: NSH Mobility Context Allocation

Figure 2 provides a high-level description of the fields in the recommended allocation of the fixed context headers for a mobility context.

5. Broadband Allocation Specifics

The intended use for each of the context header allocations is as follows:

R - Reserved.

Sub - Sub/Endpoint ID type field. These bits determine the type of the 64-bit Sub/Endpoint ID field that spans CH2 and CH3.

Tag - The Tag field indicates the type of the ServiceTag field in CH4.

Context ID - The Context ID field allows the Subscriber/Endpoint ID field to be scoped. For example, the Context ID field could contain the incoming VRF, VxLAN VNID, VLAN, or policy identifier within which the Subscriber/Endpoint ID field is defined.

Sub/App ID - 64-bit length Subscriber/Endpoint identifier (e.g., IMSI, MSISDN, or implementation-specific Endpoint ID) of the corresponding subscriber/machine/application for the flow. This field is typed by the value of the Sub field as follows:

000 - If the Sub field is not set, then the 64-bit Sub/Endpoint ID field is an opaque field that can be used or ignored by service functions as determined by the control plane.

001 - The Sub/Endpoint ID field contains an IMSI [itu-e-164].

010 - The Sub/Endpoint ID field contains an MSISDN (8-15 digit) [itu-e-164].

011 - The Sub/Endpoint ID field contains a 64-bit identifier that can be used to group flows (e.g., in Machine-to-Machine, M2M).

100-111 - Reserved.

ServiceTag - A ServiceTag is a unique identifier that can carry metadata specific to the flow or subscriber identified in the Sub/App ID field. Some types for this field are specified by the Tag field as follows:

000 - If the Tag field is not set, then the ServiceTag field in CH4 is an opaque field that can be used or ignored by service functions as determined by the control plane.

001 - The ServiceTag field in CH4 contains information related to the Radio Access Network (RAN) for the subscriber as follows in Figure 3. Note that these values should correspond to those that can be obtained for the flow from the corresponding 3GPP PCRF (Policy and Charging Rules Function) component using Diameter as described in [TS.29.230].

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| CAN |      QoS      |U| Con |    App Id               | Rsvd  | CH4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Service Tag RAN Allocation

CAN - IP-CAN-Type (Diameter AVP code 1027).

QoS - QoS-Class-Identifier AVP (Diameter AVP code 1028).

U - QoS-Upgrade AVP (Diameter AVP code 1030).

Con - Congestion level.

App Id - Application ID describing the flow type. Allocation of IDs is done in the control plane and is out of the scope of this document.

Rsvd - Reserved.

010-111 - Reserved.

6. Context Allocation and Control Plane Considerations

This document describes an allocation scheme for the mandatory context headers in the context of mobile service providers. This suggested allocation of context headers should be considered as a guideline and may vary depending on the use case. The control plane aspects of specifying and distributing the allocation scheme among different service functions within the Service Function Chaining environment to guarantee consistent semantics for the metadata is beyond the scope of this document.

7. Security Considerations

The context header allocation recommended by this document includes numbers that must be distributed consistently across a Service Function Chaining environment. Protocols for distributing these numbers securely are required in the control plane, but are out of scope of this document.

Furthermore, some of the metadata carried in the context headers require secure methods to prevent spoofing or modification by service function elements that may themselves be exposed to subscriber traffic and thus might be compromised. This document does not address such security concerns.

8. IANA Considerations

This document has no actions for IANA.

9. Acknowledgments

The authors would like to thank Jim Guichard for his assistance structuring the document.

10. References

10.1. Normative References

[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>.

10.2. Informative References

[ietf-sfc-nsh] Quinn, P. and U. Elzur, "Network Service Header", I-D draft-ietf-sfc-nsh-01 (work in progress), July 2015.

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

[itu-e-164]
"The international public telecommunication numbering plan", ITU-T E.164, November 2010.

[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>.

[TS.29.230]
"Diameter applications; 3GPP specific codes and identifiers", 3GPP TS 29.230 13.2.0, September 2015.

Authors' Addresses

Jeffrey Napper
Cisco Systems, Inc.

Email: jenapper@cisco.com

Surendra Kumar
Cisco Systems, Inc.

Email: smkumar@cisco.com

Praveen Muley
Alcatel-Lucent

Email: praveen.muley@alcatel-lucent.com

Wim Hendericks
Alcatel-Lucent

Email: Wim.Henderickx@alcatel-lucent.com

SFC
Internet-Draft
Intended status: Standards Track
Expires: October 31, 2016

R. Penno
C. Pignataro
C. Yen
E. Wang
K. Leung
Cisco Systems
D. Dolson
Sandvine
April 29, 2016

Packet Generation in Service Function Chains
draft-penno-sfc-packet-03

Abstract

Service Functions (e.g., Firewall, NAT, Proxies and Intrusion Prevention Systems) generate packets in the reverse flow direction to the source of the current in-process packet/flow. In this document we discuss and propose how to support this required functionality within the SFC framework.

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

1. Introduction

Service Functions (e.g., Firewall, NAT, Proxies and Intrusion Prevention Systems) generate packets in the reverse flow direction destined to the source of the current in-process packet/flow. In some cases, devices generate packets without any in-process packet. Packet generation is a basic intrinsic functionality and therefore needs to be supported in a service function chaining deployment.

2. Problem Statement

The challenge of this functionality in service chain environments is that generated packets need to traverse in the reverse order the same Service Functions traversed by original packet that triggered the packet generation.

Although this might seem to be a straightforward problem, on further inspection there are a few interesting challenges that need to be solved. First and foremost a few requirements need to be met in order to allow a packet to make its way through back to its source through the service path:

- o A symmetric path ID needs to exist. Symmetric path is discussed in [SymmetricPaths]

- o The SF needs to be able to encapsulate such error or proxy packets in a encapsulation transport such as VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe] + NSH header [I-D.ietf-sfc-nsh]

- o The SF needs to be able to determine, directly or indirectly, the symmetric path ID and associated next service-hop index or, alternatively, indicate reverse path for the service path ID in the original packet

3. Definitions and Acronyms

The reader should be familiar with the terms contained in [I-D.ietf-sfc-nsh] ,[I-D.ietf-sfc-architecture] and [I-D.ietf-nvo3-vxlan-gpe]

4. Assumptions

We make the following assumption throughout this document

1. An SF could be connected to more than one SFF directly. In other words, a SF can be multi-homed and each connection can use different encapsulations.

2. After forwarding a packet to an SF, the SFF always has connectivity to the next hop SFF to complete the path. This means the following Figure 1 scenario is not permitted. (SFF2 cannot complete the forward path which contains SFF3 and potentially SFs connected to SFF3.)

```
            .-.           .-.
           /   \         /   \
          ( SF1 )       ( SF2 )
           \   /         \   / \
            '+'           '+'   \
             |             |     \
             |             |      \
          +--+---+      +--+---+   \+------+
   ...---+ SFF1 +------+ SFF2 |    | SFF3 +---...
          +------+      +--+---+    +------+
                           |
                           |
                           +-----...
```

RSFP Forward -> SFF1 : SF1 : SFF1 : SFF2 : SF2 : SFF3 : ...

Figure 1: Arrangement not supported

3. Forward and reverse paths may be required to utilize different service function forwarders. In the Figure 2 below, if SF2 is directly connected to SFF2A and SFF2B, there could be a case that SFF2A only has the forwarding rules for the forward path, and SFF2B only has the forwarding rules for the reverse path.

```
                .-.             .-.             .-.
               /   \           /   \           /   \
              ( SF1 )         ( SF2 )         ( SF3 )
               \   /\          \   /\          \   /\
                '+'  \          '+'  \          '+'  \
                 |    \          |    \          |    \
                 |     |         |     |         |     |
            +---+---+ |     +-------+ |     +---+---+ |
...---+ SFF1A +-|-----+ SFF2A +-|-----+ SFF3A +-|---...
            +-------+ |     +-------+ |     +-------+ |
                      |               |               |
                  +---+---+       +---+---+       +---+---+
            ...---+ SFF1B +-------+ SFF2B +-------+ SFF3B +-----...
                  +-------+       +-------+       +-------+


Symmetric Paths:

RSFP Forward -> SFF1A : SF1 : SFF1A : SFF2A : SF2 :
                SFF2A : SFF3A : SF3 : SFF3A ...
RSFP Reverse <- SFF1B : SF1 : SFF1B : SFF2B : SF2 :
                SFF2B : SFF3B : SF3 : SFF3B


Asymmetric Paths (skipping SF2 on reverse):

RSFP Forward -> SFF1A : SF1 : SFF1A : SFF2A : SF2 : SFF2A :
                SFF3A : SF3 : SFF3A ...
RSFP Reverse <- SFF1B : SF1 : SFF1B         :       SFF2B :
                SFF3B : SF3 : SFF3B
```

Figure 2: Supported SFF arrangement

Assumption #2 allows an SF to always bounce a packet back to the SFF that originally sent the packet. Due to #3, an SF has to determine which SFF to send the generated packet to. It cannot treat generated packet the same way as forwarded packet, as in #2.

These assumptions make sense for certain implementation. However, some implementations are free of the constraints in #3, which will simplify the SF logic in handling generated traffic.

5. Service Function Behavior

When a Service Function wants to send packets to the reverse direction back to the source it needs to know the symmetric service path ID (if it exists) and associated service index. This information is not available to Service Functions since they do not

need to perform a next-hop service lookup. There are four recommended approaches to solve this problem and we assume different implementations might make different choices.

1. The SF can receive service path forwarding information in the same manner a SFF does.

2. The SF can send the packet in the forward direction but set appropriate bits in the NSH header requesting a SFF to send the packet back to the source

3. The classifier can encode all information the SF needs to send a reverse packet in the metadata header

4. The controller uses a deterministic algorithm when creating the associated symmetric path ID and service index.

We will discuss the ramifications of these approaches in the next sections.

5.1. SF receives Reverse Forwarding Information

This solution is easy to understand but brings a change on how traditionally service functions operate. It requires SFs to receive and process a subset of the information a SFF does. When a SF wants to send a packet to the source, the SF uses information conveyed via the control plane to impose the correct NSH header values.

Advantages:

- o Changes are restricted to SF and controller, no changes to SFF
- o Incremental deployment possible
- o No protocol between SF and SFF, which avoids interoperability issues
- o No performance penalty on SFF due to in or out-of-band protocol

Disadvantages:

- o SFs need to process and understand Rendered Service Path messages from controller

This solution can be characterized by putting the burden on the SF, but that brings the advantage of being self-contained (as well as providing a mechanism for other features). Also, many SFs have

policy or classification function which in fact makes them a classifier and SF combination in practice.

5.2. SF requests SFF cooperation

These solutions can be characterized by distributing the burden between SF and SFF. In this section we discuss two possible in-band solutions: using OAM header and using a reserved bit 'R' in the NSH header.

5.2.1. OAM Header

When the SF needs to send a packet in the reverse direction it will set the OAM bit in the NSH header and use an OAM protocol [I-D.penno-sfc-trace] to request that the SFF impose a new, reverse path NSH header. Post imposition, the SFF forwards the packet correctly.

SF Reverse Packet Request

```
 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|1|C|R|R|R|R|R|R|   Length  |  MD-type=0x1  |  OAM Protocol | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|          Service Path ID                      | Service Index | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                Mandatory Context Header                       | |S
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |F
|                Mandatory Context Header                       | |C
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                Mandatory Context Header                       | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                Mandatory Context Header                       | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ <
|Rev. Pkt Req   |         Original NSH headers (optional)      | |O
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |A
                                                                |M
                                                               /
```

(postamble)

Ver: 1

OAM Bit: 1

Length: 6

MD-Type: 1

Next Protocol: OAM Protocol

Rev. Pkt Req: 1 Reverse packet request

Advantages:

- o SF does not need to process and understand control plane path messages.
- o Clear division of labor between SF and SFF.
- o Extensible
- o Original NSH header could be carried inside OAM protocol which leaves metadata headers available for SF-SFF communication.

Disadvantages:

- o SFFs need to process and understand a new OAM message type
- o Possible interoperability issues between SF-SFF
- o SFF Performance penalty

5.2.2. Service Function Forwarder Behavior

In the case where the SF has all the information to send the packet back to the origin no changes are needed at the SFF. When an SF requests SFF cooperation the SFF MUST be able to process the OAM message used to signal reverse path forwarding.

- o Process/decode OAM message
- o Examine and act on any metadata present in the NSH header
- o Examine its forwarding tables and find the reverse path-id and index of the next service-hop

The reverse path can be found in the Rendered Service Path Yang model [RSPYang] that conveyed to the SFF when a path is constructed.

If a SFF does not understand the OAM message it just forwards the packet based on the original path-id and index. Since it is a special OAM packet, it tells other SFFs and SFs that they should process it differently. For example, a downstream intrusion detection SF might not associate flow state with this packet.

5.2.3. Reserved bit

In this solution the SF sets a reversed bit in the NSH that carries the same semantic as in the OAM solution discussed previously. This solution is simpler from a SF perspective but requires allocating one of the reserved bits. Another issue is that the metadata in the original packet might be overwritten by SFs or SFFs in the path.

When a SFF receives a NSH packet with the reversed bit set, it shall look up a preprogrammed table to map the Service Path ID and Index in the NSH packet into the reverse Service Path ID and Index. The SFF would then use the new reverse ID and Index pair to determine the SF/SFF which is in the reverse direction.

Advantages:

- o No protocol header overhead
- o Limited performance impact on SF

Disadvantages:

- o Use of a reserved bit
- o SFF Performance penalty
- o Not extensible

5.3. Classifier Encodes Information

This solution allows the Service Function to send a reverse packet without interactions with the controller or SFF, therefore it is very attractive. Also, it does not need to have the OAM bit set or use a reserved bit. The penalty is that for a MD Type-1 packet a significant amount of information (48 bits) need to be encoded in the metadata section of the packet and this data cannot be overwritten. Ideally this metadata would need to be added by the classifier.

The Rendered Service Path yang model [RSPYang] already provides all the necessary information that a classifier would need to add to the metadata header. An explanation of this method is better served with an examples.

5.3.1. Symmetric Service Paths

Figure 3 below shows a simple SFC with symmetric service paths comprising three SFs.

```
   ......................SFP2 Forward........................>

     Forward SI  253          252          251

   +---+         .-.          .-.          .-.            +---+
   |   |        /   \        /   \        /   \           |   |
   | A +-------( SF1 )------( SF2 )------( SF3 )----------+ B |
   |   |        \   /        \   /        \   /           |   |
   +---+         '-'          '-'          '-'            +---+

     Reverse SI  253          254          255

    <....................SFP3 (Reverse of SFP2)...................


     SFP2 Forward ->  SF1 : SF2 : SF3
     SFP3 Reverse <-  SF1 : SF2 : SF3

     RSP2 Forward -> SF1 : SF2 : SF3
     RSP3 Reverse <- SF1 : SF2 : SF3
```

Figure 3: SFC example with symmetric path

Below we see the JSON objects of the two symmetric paths depicted above.

```
RENDERED_SERVICE_PATH_RESP_JSON = """
{
  "rendered-service-paths": {
    "rendered-service-path": [
      {
        "name": "SFC1-SFP1-Path-2-Reverse",
        "transport-type": "service-locator:vxlan-gpe",
        "parent-service-function-path": "SFC1-SFP1",
        "path-id": 3,
        "service-chain-name": "SFC1",
        "starting-index": 255,
        "rendered-service-path-hop": [
          {
            "hop-number": 0,
            "service-index": 255,
            "service-function-forwarder-locator": "eth0",
            "service-function-name": "SF3",
            "service-function-forwarder": "SFF3"
          },
          {
            "hop-number": 1,
            "service-index": 254,
```

```
              "service-function-forwarder-locator": "eth0",
              "service-function-name": "SF2",
              "service-function-forwarder": "SFF2"
            },
            {
              "hop-number": 2,
              "service-index": 253,
              "service-function-forwarder-locator": "eth0",
              "service-function-name": "SF1",
              "service-function-forwarder": "SFF1"
            }
          ],
          "symmetric-path-id": 2
        },
        {
          "name": "SFC1-SFP1-Path-2",
          "transport-type": "service-locator:vxlan-gpe",
          "parent-service-function-path": "SFC1-SFP1",
          "path-id": 2,
          "service-chain-name": "SFC1",
          "starting-index": 253,
          "rendered-service-path-hop": [
            {
              "hop-number": 0,
              "service-index": 253,
              "service-function-forwarder-locator": "eth0",
              "service-function-name": "SF1",
              "service-function-forwarder": "SFF1"
            },
            {
              "hop-number": 1,
              "service-index": 252,
              "service-function-forwarder-locator": "eth0",
              "service-function-name": "SF2",
              "service-function-forwarder": "SFF2"
            },
            {
              "hop-number": 2,
              "service-index": 251,
              "service-function-forwarder-locator": "eth0",
              "service-function-name": "SF3",
              "service-function-forwarder": "SFF3"
            }
          ],
          "symmetric-path-id": 3
        }
      ]
    }
```

```
}"""
```

We will assume the classifier will encode the following information in the metadata:

- o symmetric path-id = 2 (24 bits)
- o symmetric starting index = 253 (8 bits)
- o symmetric number of hops = 3 (8 bits)
- o starting index = 255 (8 bits)

In the method below we will assume SF will generate a reverse packet after decrementing the index of the current packet. We will call that current index.

If SF1 wants to generate a reverse packet it can find the appropriate index by applying the following algorithm:

```
current_index = 252

remaining_hops = symmetric_number_hops - (starting_index - current_index)
remaining_hops = 3 - (255 - 252) = 0
reverse_service_index = symmetric_starting_index - remaining_hops - 1
reverse_service_index = next_service_hop_index = 253 - 0 - 1 = 252
The "-1"  is necessary for the service index to point to the next service_hop.
```

If SF2 wants to send reverse packet:

```
current index = 253

remaining_hops = 3 - (255 - 253) = 1
reverse_service_index = next_service_hop_index = 253 - 1 - 1 = 251
```

If SF3 wants to send reverse packet:

```
current index = 254

remaining_hops = 3 - (255 - 254) = 2
reverse_service_index = next_service_hop_index = 253 - 2 - 1 = 250
```

The following tables in Figure 4 summarize the service indexes as calculated by each SF in the forward and reverse paths respectively.

```
Fwd SI = forward Service Index
Cur SI = Current Service Index
Gen SI = Service Index for Generated packets


RSFP1 Forward -
  Number of Hops: 3
  Forward Starting Index: 253
  Reverse Starting Index: 255

+-------+--------+--------+--------+
|  SF   |  SF1   |  SF2   |  SF3   |
+-------+--------+--------+--------+
|Fwd SI |  253   |  252   |  251   |
+-------+--------+--------+--------+
|Cur SI |  252   |  251   |  250   |
+-------+--------+--------+--------+
|Gen SI |  252   |  253   |  254   |
+-------+--------+--------+--------+

 RSFP1 Reverse -
  Number of Hops: 3
  Reverse Starting Index: 255
  Forward Starting Index: 253

+-------+--------+--------+--------+
|  SF   |  SF1   |  SF2   |  SF3   |
+-------+--------+--------+--------+
|Rev SI |  253   |  254   |  255   |
+-------+--------+--------+--------+
|Cur SI |  252   |  253   |  254   |
+-------+--------+--------+--------+
|Gen SI |  252   |  251   |  250   |
+-------+--------+--------+--------+
```

Figure 4: Service indexes generated by each SF in the symmetric forward and reverse paths

5.3.2. Symmetric Service Paths, Optimized

This approach is effectively the same as Section 5.3.1, but with redundant information removed such that the reverse-path information can be packed into 32 bits. This approach is obtained by observing that the same arithmetic is always done on the same constants of starting_index, symmetric_starting_index and symmetric_number_hops.

As before, we require symmetric paths, meaning there are two paths that are exactly the reverse of each other. We assume that the classifier at each end has available the following information:

- o symmetric path-id (24 bits)
- o starting index (8 bits)
- o symmetric starting index (8 bits)
- o symmetric number of hops, which is the same in both directions (8 bits)

The classifier computes, for each path, a "reverse service offset":

```
# Compute using 8-bit, two’s-complement arithmetic:
# (Overflow or underflow are okay)
reverse_service_offset = symmetric_starting_index
                         + starting_index
                         - symmetric_number_of_hops
```

This reverse_service_offset is an 8-bit value that is encoded in metadata along with the 24 bits of reverse_path_id.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                               |    Reverse    |
|          Reverse Path ID                      |    Service    |
|                                               |    Offset     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Metadata format of reverse_info_metadata (32 bits)

We’ll refer to the 32-bit value as reverse_info_metadata. Any Service Function may compute the NSH fields of a reverse packet as follows from the NSH fields of a forward packet.

```
reverse.NSH.Service_Path_ID =
    forward.NSH.reverse_info_metadata.Reverse_Path_ID
# Compute using 8-bit two’s-complement arithmetic:
# (Overflow or underflow are okay)
reverse.NSH.Service_Index :=
    forward.NSH.reverse_info_metadata.Reverse_Service_Offset
    - forward.NSH.Service_Index - 1
reverse.NSH.reverse_info_metadata.Reverse_Service_Offset =
    forward.NSH.reverse_info_metadata.Reverse_Service_Offset
reverse.NSH.reverse_info_metadata.Reverse_Path_ID =
    forward.NSH.Service_Path_ID
```

As you can see, this approach has the convenient property that the reverse_info_metadata can be determined by a Service Function while being agnostic about both forward and reverse paths.

Using the example of Section 5.3.1, these values are used for the SFP2 Forward path:

- o starting_index=253
- o symmetric_starting_index=255
- o symmetric_number_of_hops=3
- o reverse_service_offset=(253+255-3)=249 in 8-bit two’s complement arithmetic

At SF2 on the SFP2 Forward path, where the service index is 251 after decrementing the index, the reverse service index is calculated as:

- o reverse_service_index = 249-251-1 = 253 using 8-bit two’s complement arithmetic

This is the correct index to forward to SF1 on SFP3.

5.3.3. Analysis

Advantages of encoding information in the NSH frame:

- o SF does not need to request SFF cooperation or contact controller
- o No SFF performance impact

Disadvantages:

- o Metadata overhead in case MD-Type 2 is used or use of a metadata slot in case MD-Type 1 is used.
- o Relies on classifier to encode metadata information
- o Requires perfectly symmetrical paths. E.g., one direction cannot have more SFs than the other direction.
- o If classifier will encode information it needs to receive and process rendered service path information

5.4. Algorithmic Reversed Path ID Generation

In these proposals no extra storage is required from the NSH and SFF does not need to know how to handle the reversed packet nor does it know about it. Reverse Path is programmed by Orchestrator and used by SF having the need to send upstream traffic.

5.4.1. Same Path-ID and Disjoint Index Spaces

Instead of defining a new Service Path ID, the same Service Path ID is used. The Orchestrator must define the reverse chain of service using a different range of Service Path Index. It is also assumed that the reverse packet must go through the same number of Services as its forward path. It is proposed that Service Path Index (SPI) 1..127 and 255..129 are the exact mirror of each other.

Here is an example: SF1, SF2, and SF3 are identified using Service Path Index (SPI) 8, 7 and 6 respectively.

Path 100 Index 8 - SF1

Path 100 Index 7 - SF2

Path 100 Index 6 - SF3

Path 100 Index 5 - Terminate

At the same time, Orchestrator programs SPI 248, 249 and 250 as SF1, SF2 and SF3. Orchestrator also programs SPI 247 as "terminate". Reverse-SPI = 256 - SPI.

Path 100 Index 247 - Terminate

Path 100 Index 248 (256 - 8) - SF1

Path 100 Index 249 (256 - 7) - SF2

Path 100 Index 250 (256 - 6) - SF3

If SF3 needs to send the packet in reverse direction, it calculates the new SPI as 256 - 6 (6 is the SPI of the packet) and obtained 250. It then subtract the SPI by 1 and send the packet back to SFF

Subsequently, SFF received the packet and sees the SPI 249. It then diverts the packet to SF2, etc. Eventually, the packet SPI will drop to 247 and the SFF will strip off the NSH and deliver the packet.

The same mechanism works even if SF1 later decided to send back another upstream packet. The packet can ping-pong between SF1 and SF3 using existing mechanism.

Note that this mechanism is a special case of Section 5.3.2 wherein Reverse_Path_ID is the forward path ID and Reverse_Service_Offset=255.

Advantages:

- o No precious NSH area is consumed
- o SF self-contained solution
- o No SFF performance impact and no cooperation needed
- o No Special Classification required

Disadvantages:

- o SPI range is reduced and may become incompatible with existing topology
- o Assumption that the reverse path Service Functions are the same as forward path, only in reverse
- o Reverse paths need to use Service Index = 128 for loop detection instead of SI = 0.

In either case, the SF must have the knowledge through Orchestrator that the reverse path has been programmed and the method (SPI only or SPI + SPID bit) to use.

The symmetrization mechanism keep reverse path symmetric as described in section 6 can be applied in this method as well.

5.4.2. Flip Path-Id and Index High Order bits

An alternative to reducing Service Path Index range is to make use of a different Service Path ID, e.g. the most significant bit. The bit can be flipped when the SF needs to send packet in reverse. However, the negation of the SPI is still required, e.g. SPI 6 becomes SPI 134

This approach is fully compatible with the current NSH protocol standard and provides a fully deterministic way of determining reverse paths. It is the recommended approach.

Advantages:

- o No precious NSH area is consumed
- o SF self-contained solution
- o No SFF performance impact and no cooperation needed
- o No Special Classification required

Disadvantages:

- o Assumption that the reverse path Service Functions are the same as forward path, only in reverse
- o Forward and Reverse Path IDs are algorithmically linked and can not be chosen arbitrarily.

6. Asymmetric Service Paths

In real world the forward and reverse paths can be asymmetric, comprising different set of SFs or SFs in different orders. The following Figure 5 illustrates an example. The forward path is composed of SF1, SF2, SF4 and SF5, while the reverse path skips SF5 and has SF3 in place of SF2.

```
                    ..........        .........
                   .         .      .         .
                  .     249    .    .    246    .
                 .               .  .             .
                .       .-.      ..      .-.      .
  ..............       /   \            /   \      ....SFP1 Forward....>
                      ( SF2 )   247    ( SF5 )
  Forward SI   250   / \   / \        / \   /\
                    /   '-'   \      /   '-'  \
                   /           \    /          \
 +---+         .-./             '-./            \               +---+
 |   |        /   \            /   \             \              |   |
 | A +-------( SF1 )----------( SF4 )-------------+-------------+ B |
 |   |        \   /            \   /                            |   |
 +---+         '-'\             ,-'                             +---+
                   \           /
                    \   .-.   /
   Reverse SI  251   \ /   \ /  254
  <...........        ( SF3 )         .................SFP2 Reverse.....
              .        \   /         .
               .        '-'         .
                .                  .
                 .                .
                  .     253      .
                   ..............
```

```
SFP1 Forward ->  SF1 : SF2 : SF4 : SF5
SFP2 Reverse <-  SF1 : SF3 : SF4
```

Figure 5: SFC example with asymmetric paths

An asymmetric SFC can have completely independent forward and reverse paths. An SF's location in the forward path can be different from that in the reverse path. An SF may appear only in the forward path but not reverse (and vice-versa). In order to use the same algorithm to calculate the service index generated by an SF, one design option is to insert special NOP SFs in the rendered service paths so that each SF is positioned symmetrically in the forward and reverse rendered paths. The SFP corresponding to the example above is:

SFP1 Forward -> SF1 : SF2 : NOP : SF4 : SF5

SFP2 Reverse <- SF1 : NOP : SF3 : SF4 : NOP

The NOP SF is assigned with a sequential service index the same way as a regular SF. The SFF receiving a packet with the service path ID

and service index corresponding to a NOP SF should advance the service index till the service index points to a regular SF. Implementation can use a loopback interface or other methods on the SFF to skip the NOP SFs.

Once the NOP SF is inserted in the rendered service paths, the forward and reverse paths become symmetric. The same algorithm can be applied by the SFs to generate service indexes in the opposite directional path. The following tables list the service indexes corresponding to the example above.

Fwd SI = forward Service Index
Cur SI = Current Service Index
Gen SI = Service Index for Generated packets

RSP1 Forward -
Number of hops: 5
Forward Starting Index: 250
Reverse Starting Index: 255

SF	SF1	SF2	NOP	SF4	SF5
Fwd SI	250	249	248	247	246
Cur SI	249	248	247	246	245
Gen SI	250	251	N/A	253	254

RSP1 Reverse -
Number of hops: 5
Reverse Starting Index: 255
Forward Starting Index: 250

SF	SF1	NOP	SF3	SF4	NOP
Rev SI	251	252	253	254	255
Cur SI	250	251	252	253	254
Gen SI	249	N/A	247	246	N/A

This symmetrization of asymmetric paths could be performed by a controller during path creation.

7. Metadata

A crucial consideration when generating a packet is which metadata should be included in the context headers. In some scenarios if the metadata is not present the packet will not reach its intended destination. Although one could think of many different ways to convey this information, we believe the solution should be simple and require little or no new Service Function functionality.

We assume that a Service Function normally needs to know the semantics of the context headers in order to perform its functions. But clearly knowing the semantics of the metadata is not enough. The issue is that although the SF knows the semantics of the metadata when it receives a packet, it might not be able to generate or retrieve the correct metadata values to insert in the context headers when generating a packet. It is usually the classifier that inserts the metadata in the context headers.

7.1. Service-Path-Invariant Metadata

In order to solve this problem we propose the notion of service-path-invariant metadata. This is metadata that is the same for all packets traversing a certain path. For example, if all packets exiting a service-path need to be routed to a certain VPN, the VPN id would be a path-invariant metadata.

To implement this, the controller needs to configure appropriate fixed values of the metadata present in the context headers for each path identifier in each Service Function that needs to inject packets. The Service Function must store this information so that when the Service Function generates a packet it can insert the minimum required metadata for a packet to reach its destination.

A disadvantage to path-invariant metadata is that it is a type of metadata that adds no information beyond the information available in the path identifier itself. The corollary is that if different metadata is required, a different service paths must be created.

7.2. Service-Path-Default Metadata

We also propose the notion of service-path-default metadata. This is metadata that could vary for different packets on a path but has a default value acceptable for any packet injected onto a certain path. For example, metadata might indicate a quality-of-service (QoS) treatment, and an operator considers it acceptable for injected

packets to have a default QoS treatment. It might also be considered acceptable to not send a particular type of metadata.

To implement this, the controller configures appropriate default metadata values for each path identifier in Service Functions that need to inject packets. The controller may also indicate a particular type may be omitted. The Service Function must store this information so that it can insert the minimum required metadata for a packet to reach its destination.

The disadvantage of this approach is that it relies on the assumption that there is a meaningful default metadata value, which may not exist.

7.3. Bidirectional Clonable Metadata

Some types of metadata may use values applicable to both directions of traffic. An example is routing domain, for which an identifier indicates a private network such that the value is the same for both directions of traffic and may be copied from one packet to another.

To implement this, the controller must indicate to each Service Function that a particular metadata type is bidirectional-clonable. The Service Function can therefore clone the metadata value from one packet to a new packet that it creates, even in the reverse direction. For this type, it is also considered safe to save a copy of metadata for the transport flow. (E.g., to retransmit a TCP packet using metadata cloned from another TCP packet of the same connection.)

Note that the Service Function need not know the meaning of the metadata; it just needs to know it is safe to clone in this manner.

7.4. Unidirectional Clonable Metadata

Some types of metadata may use values applicable to only one direction of traffic, but a value may be cloned from one packet to another in the same direction. An example is a destination identifier, in which meatadata indicates a network egress point. Another example is metadata indicating a property of either the source or destination end-point of the packet.

To implement this, the controller must indicate to each Service Function that a particular metadata type is unidirectional-clonable. A transport-layer-stateful Service Function can therefore save away metadata values that it has witnessed. An injected packet can therefore be assigned a clone of metadata taken from an earlier packet going in the same direction. For example, a Service Function

can send a TCP packet using metadata cloned from another TCP packet of the same connection and direction.

Note that the Service Function need not know the meaning of the metadata; it just needs to know it is safe to clone in this manner.

A disadvantage of unidirectional clonable metadata is that a device cannot respond to a packet unless it has previously witnessed a packet for the same connection in the opposite direction. For example, a firewall cannot respond to the first packet of a connection (since both directions have not been witnessed). However, having seen a full hand-shake, a cache or optimizing proxy can inject or retransmit packets.

7.5. Service-Function-Mastered Metadata

The easiest case to reason about is a type of metadata for which the Service Function can provide the appropriate values: specifically the metadata that it would be responsible for inserting for all packets as part of packet processing. We can assume this is configured by Service-Function-Specific methods.

7.6. Metadata from Reclassification

Finally if the packet needs crucial metadata values that cannot be supplied by the methods above then a reclassification is needed. This reclassification would need to be done by the classifier that would normally process packets in the reverse path or a SFF that had the same rules and capabilities. Ideally the first SFF that processes the generated packet.

If a packet needs to be sent to classifier then it should be carried inside a NSH OAM packet that in turn is tunneled with a protocol such as VXLAN-GPE with the classifier as its tunnel endpoint.

8. Other solutions

We explored other solution that we deemed too complex or that would bring a severe performance penalty:

- An out-of-band request-response protocol between SF-SFF. Given that some service functions need to be able to generate packets quite often this will would create a considerable performance penalty. Specially given the fact that path-ids (and their symmetric counterpart) might change and SF would not be notified, therefore caching benefits will be limited.

- o An out-of-band request-response protocol between SF-Controller. Given that admin or network conditions can trigger service path creation, update or deletions a SF would not be aware of new path attributes. The controller should be able to push new information as it becomes available to the interested parties.

- o SF (or SFF) punts the packet back to the controller. This solution obviously has severe scaling limitations.

9. Implementation

The solutions "Flip Path-Id and Index High Order bits" and "SF receives Reverse Forwarding Information" were implemented in Opendaylight.

10. IANA Considerations

TBD

11. Security Considerations

Service Functions must be trusted entities, being permitted to rewrite service path headers.

12. Acknowledgements

Paul Quinn, Jim Guichard

13. Changes

14. References

14.1. Normative References

[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>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <http://www.rfc-editor.org/info/rfc2616>.

14.2. Informative References

[I-D.ietf-nvo3-vxlan-gpe]
Kreeger, L. and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-02 (work in progress), April 2016.

[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-04 (work in progress), March 2016.

[I-D.penno-sfc-trace]
Penno, R., Quinn, P., Pignataro, C., and D. Zhou, "Services Function Chaining Traceroute", draft-penno-sfc-trace-03 (work in progress), September 2015.

[I-D.penno-sfc-yang]
Penno, R., Quinn, P., Zhou, D., and J. Li, "Yang Data Model for Service Function Chaining", draft-penno-sfc-yang-14 (work in progress), January 2016.

[RSPYang] Opendaylight, , "Rendered Service Path Yang Model", February 2011, <https://github.com/opendaylight/sfc/blob/master/sfc-model/src/main/yang/rendered-service-path.yang>.

[SymmetricPaths]
IETF, , "Symmetric Paths", February 2011, <https://tools.ietf.org/html/draft-ietf-sfc-architecture-11#section-2.2>.

Authors' Addresses

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

Email: repenno@cisco.com

Carlos Pignataro
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: cpignata@cisco.com

Chui-Tin Yen
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: tin@cisco.com

Eric Wang
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: ejwang@cisco.com

Kent Leung
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: kleung@cisco.com

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

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

SFC
Internet-Draft
Intended status: Standards Track
Expires: December 23, 2018

B. Sarikaya
Denpel Informatique
M. Boucadair
Orange
D. von Hugo
Deutsche Telekom
June 21, 2018

Service Function Chaining: Subscriber and Service Identification Use Cases and Variable-Length NSH Context Headers
draft-sarikaya-sfc-hostid-serviceheader-07

Abstract

This document discusses how to inform Service Functions about service- and subscriber-related information for the sake of policy enforcement and appropriate SFC-inferred forwarding. Once the information is consumed by SFC-aware elements of an SFC-enabled domain, it is stripped from packets when they leave the SFC-enabled domain. Thus privacy-sensitive information is not leaked outside the domain.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 23, 2018.

Table of Contents

1. Introduction

This document discusses how to inform Service Functions about service- and subscriber-related information when required for the sake of policy enforcement. Indeed, subscriber-related information may be required to enforce subscriber-specific, SFC-based traffic forwarding policies, since the information carried in packets may not be sufficient.

The enforcement of SFC-based differentiated traffic forwarding policies may also be inferred by QoS considerations. QoS information may serve as an input to classification of SFP for path computation and establishment.

The dynamic structuring of service function chains and their subsequent enforcement may be conditioned by QoS requirements that will affect SF instance identification, location and sequencing.

We refer here to the definition of a logical network slice as a sub-network being isolated from other sub-networks using the same physical infrastructure. Each of these slices are constructed to provide service specific QoS requirements (such as low latency, high availability, or high reliability) efficiently.

SFs and SF Forwarders (SFFs) involved in an SFC have to contribute to the respective QoS requirements characterized by low transmission delay between each other, by exposing a high availability of resources to process function tasks, or by redundancy provided by stand-by machines for seamless execution continuation in case of failures. These requirements may be satisfied by means of control protocols, but in some contexts, carrying QoS-related information in packets may improve the overall SFC operation instead of relying upon the potential complexity of SFC control plane features.

This document adheres to the architecture defined in [RFC7665]. This document assumes the reader is familiar with [RFC7665] and [I-D.ietf-sfc-hierarchical].

Subscriber-related information may be required to implement services such as, but not limited to, traffic policy control, parental control, traffic offload. Such features are often provided by operators as part of their service portfolio.

Another example is the applicability of service chaining in the context of mobile networks (typically, in the 3GPP defined (S)Gi Interface) [I-D.ietf-sfc-use-case-mobility]. Because of the widespread use of private addressing in those networks, if advanced SFs to be invoked are located after a NAT device (that can reside in the Packet Data Network (PDN) Gateway (PGW) or in a distinct operator-specific node), the identification based on the internal IP address is not anymore possible once the NAT has been crossed. As such, means to allow passing the internal information may optimise packet traversal within an SFC-enabled mobile network domain. Furthermore, some SFs that are not enabled on the PGW may require a subscriber identifier e.g., International Mobile Subscriber Identity (IMSI), to properly operate. Other use cases that suffer from identification problems further are discussed in [RFC7620].

Subscriber-specific information can be useful for optimized SFC design and SF placement/invocation, let alone slice/VPN design and operation.

To ensure a service specific quality and performance per use case logically separated network slices will be deployed. Each one is flagged by a corresponding service-related information in terms of a service identifier. Examples are ’tactile internet’, ’eHealth’ or

'industry control' requiring, e.g. granted low latency or extreme high reliability.

This document does not make any assumption about the structure of service identifiers; each such service-related information is treated as an opaque value by the SFC operations and protocols. The semantics and validation of these identifiers are up to the control plane used for SFC. Expectations to SFC control plane protocols are laid down in [I-D.ietf-sfc-hierarchical].

Subscriber- and service-related information is stripped from packets exiting an SFC-enabled domain for the sake of privacy protection in particular. See Section 8 for more discussion on privacy.

The use cases discussed in this document assume the NSH is used exclusively within a single administrative domain.

2. Conventions and Terminology

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].

The reader should be familiar with the terms defined in [RFC7665].

3. Problem Space and Sample Use Cases

Enforcing Policies based on an internal IP address:

> Because of the address sharing, implicit CPE/UE identification that relies on the source IP address cannot be implemented within the administrative domain because the same global IPv4 address is shared by various connected devices (CPE for the fixed case or UE for the mobile case). In the meantime, policies are something provisioned based on the internal IP address assigned to those devices. Means to pass the internal IP address beyond an address sharing device for the sake of per-subscriber policy enforcement is needed in some SFC deployments.
>
> Also, identifiers like a MAC address, or an IMSI may be required to optimize the corresponding SFC operation.

Enforcing Policies based on a subscriber identifier:

> In case some deployments may require per-subscriber policies, carrying subscriber ID information may be required for the sake of proper SFC operation..

Enforcing Policies based on a service specific identifier:

SFCs can be structured according to QoS/QoE requirements that may be shared by different services. In that case, service identification information is required to be accessible across the SFC for the sake of proper SFC operation.

Below we present some use cases where problems related to enforcing policies based on subscriber identifiers and those based on service and/or slice identifiers cannot be addressed by service function chaining. It is important to note that subscriber identification issues raised by address sharing environments are not specific to service function chaining.

3.1. Parental Control Use Case

Parental control service function searches each packet for certain content. Parental control function should have permanent access to corresponding specific information (URL and source IP address), e.g. in a cache, so that all packets of the corresponding flow(s) can be filtered [WT317].

Parental control function receives next packet from the recorded URL. Enforcing the parental control policies may depend on the internal IP address, i.e., the address of the subscriber’s host that is being subject to the parental control. Parental control function must be able to identify incoming traffic to be filtered, e.g., specific URL information. All other traffic is not subject to parental control filtering. Parental control function filters all traffic coming from the indicated URL only for the specific subscriber’s hosts identified by the service logic.

For the virtual CPE case, the access node will receive privately-addressed packets. Because private IPv4 addresses are likely to overlap between several subscribers, the internal private IPv4 address will need to be copied into a dedicated header in the NSH packet so that SFs responsible for parental control can process the packets appropriately. Furthermore, the subscriber identifier may also be required for authorization purposes.

3.2. Traffic Offload Use Case

A traffic offload service function is invoked for each flow/service originated from a mobile terminal and this SF decides whether traffic should be offloaded to the broadband network or sent back to the mobile network. In this use case, policy enforcement is based on the subscriber identifier. The broadband network must obtain the subscription profile from the mobile network and decide if the

traffic coming from this subscriber needs to be offloaded or not. If offloading is needed, this usually means that the subscriber identifier needs to be known by SFFs.

3.3. Mobile Network Use Cases

Many SFs can be executed in different combinations in a mobile network [I-D.ietf-sfc-use-case-mobility]. In particular, placement of NAT function (if used) plays an important role.

If a NAT function is collocated with P-GW as in [TR23.975] or right after the P-GW (i.e. between P-GW and (S)Gi-LAN) then all service functions located upstream can only see the translated IPv4 address as the source address from all User Equipments (UEs). Internal IPv4 address-related part of their policy set won’t be able to execute their service logic. As a consequence, means to inform the various SFs of a given chain about the IPv4 address assigned to the UE and which will be translated into a global IPv4 address may be needed.

Note that the same problem occurs in case IPv6 is being used by UEs, whenever such UEs communicate with an IPv4-only web site. In that case, a NAT64 function is deployed at the P-GW. So in the case of chaining NAT64 SF needs to be invoked as part of a given chain, the IPv6 address used by the UE may be required for the service function chain to work properly.

[I-D.ietf-sfc-use-case-mobility] identifies the following information:

- o Charging ID
- o Subscriber ID
- o GGSN or PGW IP address
- o Serving Gateway Support Node (SGSN) or SGW IP address
- o International Mobile Equipment Identity (IMEI)
- o International Mobile Subscriber Identity (IMSI)
- o Mobile Subscriber ISDN Number (MSISDN)
- o UE IP address

Several other use cases where support of traffic classification with respect to service chain selection to achieve efficient and flexible

mobile service steering are described in [TR22.808]. A set of potential solutions are proposed and discussed in [TR23.718].

3.4. Extreme Low Latency Service Use Cases

Extreme or ultra-low latency requirements may be addressed by specific architectural and protocol characteristics to allow for rapid execution and low transmission delay of packets. Candidate services for such requirements include e-health or vehicular applications. This can be granted by forwarding all packets via the shortest paths only and/or via the service function instances with the lowest processing delay, possibly as a function of the location of the user.

The corresponding service function chain should be configured based on the service demanding for the performance, but policies are also tightly related to the subscriber, i.e. whether being entitled to request the specific service.

3.5. High Reliability Applications Use Cases

Another set of use cases that require very (or ultra-) high reliability of services assume committed QoS parameter values and its possibility to act upon an expected change of the network fulfillment of the QoS targets [TR22.862]. That means: the QoS fulfillment is controlled such that in case of expected or predicted deviation a countermeasure by the network is invoked, e.g. either resources for that session are increased or a backup path is assigned in case no improvement is possible at least the application is informed on the current performance to react upon. This can be granted by forwarding all packets via the most reliable and secure paths only.

4. Subscriber Identification NSH Variable-Length Context Header

Subscriber Identifier is defined as an optional variable-length NSH context header. Its structure is shown in Figure 1.

The subscriber identifier is used to convey an identifier already assigned by the service provider to uniquely identify a subscriber or an information that is required to enforce per-subscriber policies, the structure of the identifier being deployment-specific. Typically, this header may convey the IMSI, an opaque subscriber Identifier, an IP address, etc.

The classifier and SFC-aware Service Functions MAY be instructed via a control interface to inject or strip a subscriber identifier context header. Also, the data to be injected in such header SHOULD be configured to nodes authorized to inject such headers. Failures

to inject such headers SHOULD be logged locally while a notification alarm MAY be sent to a Control Element. The details of sending notification alarms (i.e. the parameters affecting the transmission of the notification alarms depend on the information in the context header such as frequency, thresholds, and content in the alarm: full header, header ID, timestamp etc.) SHOULD be configurable by the control plane.

This document adheres to the recommendations in [RFC8300] for handling the context headers at both ingress and egress SFC boundary nodes. That is, to strip such context headers.

SFC-aware SFs and proxies MAY be instructed to strip a subscriber identifier from the packet or to pass the data to the next SF in the chain after processing the content of the headers. If no instruction is provided, the default behavior is to maintain such context headers so that the information can be passed to next SFC-aware hops.

SFC-aware functions MAY be instructed via the control plane about the validation checks to run on the content of these context headers (e.g., accept only some lengths, accept some subtypes) and the behavior to adopt. For example, SFC-aware nodes may be instructed to ignore the context header, to remove the context header from the packet, etc. Nevertheless, this specification does not require nor preclude such additional validation checks. These validation checks are deployment-specific. If validation checks fail on a context header, an SFC-aware node ignores that context header. The event SHOULD be logged locally while a notification alarm MAY be sent to a control element if the SFC-aware node is instructed to do so.

Multiple subscriber Identifier context TLVs MAY be present in the NSH each carrying a distinct sub-type.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Metadata Class       |      Type     |U|    Length   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SubT        |                                               |
+-+-+-+-+-+-+-+-+
~                       Subscriber Identifier                   ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Subscriber Identifier Variable-Length Context Header

The description of the fields is as follows:

o Metadata Class: MUST be set to 0x0 [RFC8300].

- o Type: TBD1 (See Section 6)

- o SubT field of 8 bits indicates the sub-type of the information conveyed in the "Subscriber Identifier" field. The following values are defined:

  - * 0x00: Opaque value

  - * 0x01: Charging ID. The structure of this ID is deployment-specific.

  - * 0x02: Subscriber ID. The structure of this ID is deployment-specific.

  - * 0x03: GGSN or PGW IP address/prefix

  - * 0x04: Serving Gateway Support Node (SGSN) or SGW IP address/prefix

  - * 0x05: International Mobile Equipment Identity (IMEI)

  - * 0x06: International Mobile Subscriber Identity (IMSI)

  - * 0x07: Mobile Subscriber ISDN Number (MSISDN)

  - * 0x08: UE IP address

- o Subscriber Identifier: Carries an opaque subscriber identifier or an identifier that corresponds to the sub-type.

5. Slice and Service Identification NSH Variable-Length Context Headers

Dedicated service- and slice-specific performance identifiers are defined to differentiate between services requiring specific treatment to exhibit a performance characterized by, e.g., ultra-low latency (ULL) or ultra-high reliability (UHR). These parameters are related to slice and service identifiers, among others. They are contained in the service Identifier. The service Identifier thus allows for the enforcement of a per service policy such as a service classification function to only consider specific Service Function instances during service function path establishment. Details of this process are implementation- specific. For illustration purposes, the classifier may retrieve the details of usable service functions based upon the corresponding service or slice ID. Typical criteria for instantiating specific service functions include location, performance or proximity considerations. For UHR services, the stand-by operation of back-up capacity or the deployment of multiple service function instances may be requested.

In other words, the classifier uses this kind of information to decide about the set of SFFs to invoke to honor the latency or reliability requirement (e.g., compute an Rendered Service Path, RSP, or insert a pointer to be shared with involved SFFs).

Slice and Service Identifiers are defined as optional variable length context headers. Their structure is shown in Figure 2 and Figure 3, respectively.

Service/Slice Identifier context header MAY convey a user or service provider defined unique identity which can be described by an opaque value.

The service requirements in terms of, e.g., maximum latency or minimum outage probability are specified by service providers and are out of the scope of this document.

Only one Slice Identifier context header (as described in Section 1) MUST be present in the NSH.

Multiple Service Identifier context headers MAY be present in the NSH; each carrying a distinct sub-type.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Metadata Class       |      Type     |U|    Length   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                       Slice Identifier                        ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Slice Identifier Variable-Length Context Header

The description of the fields is as follows:

- o Metadata Class: MUST be set to 0x0 [RFC8300].
- o Type: TBD2 (See Section 6)
- o Slice Identifier: The structure of the identifier is deployment-specific. This field carries an identifier that uniquely identifies a slice within a network, e.g. it could be an opaque value with an arbitrary number of characters.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Metadata Class       |      Type     |U|    Length   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SubT        |                                               |
+-+-+-+-+-+-+-+-+
~                       Service Identifier                      ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Service Identifier Variable-Length Context Header

The description of the fields is as follows:

- o Metadata Class: MUST be set to 0x0 [RFC8300].
- o Type: TBD3 (See Section 6)
- o SubT: 8-bit field that carries the sub-type of the information conveyed in the "Service Identifier" field. The following values are defined:
  - * 0x00: Opaque value
  - * 0x01: Ultra-low latency ID. The structure of this ID is service deployment-specific.
  - * 0x02: Ultra-high reliability ID. The structure of this ID is service deployment-specific.
  - * 0x03: Slice Identifier. The structure of this ID is service deployment-specific.
  - * 0x04 - 0x08: reserved
- o Service Identifier: Represents a specific service performance characteristic reflected in the SubT field, but also denotes a default basic (best effort) service without specifically defined requirements. It MAY also be an opaque value which semantic is defined by the operator.

6. IANA Considerations

This document requests IANA to assign the followiing types from the "NSH IETF- Assigned Optional Variable-Length Metadata Types" (0x0000 IETF Base NSH MD Class) registry available at: https://www.iana.org/assignments/nsh/nsh.xhtml#optional-variable-length-metadata-types.

C1	C2	
TBD1	Subscriber Identifier	[ThisDocument]
TBD2	Slice Identifier	[ThisDocument]
TBD3	Service Identifier	[ThisDocument]

7. Security Considerations

Data plane SFC-related security considerations are discussed in [RFC7665] and [RFC8300].

A misbehaving node can inject subscriber Identifiers to disturb the service offered to some subscribers. Also, a misbehaving node can inject subscriber identifiers as an attempt to be granted access to some services. To prevent such misbehavior, only trusted nodes MUST be able to inject such context headers. Nodes that are involved in a SFC-enabled domain are assumed to be trusted ([RFC8300]). Means to check that only authorized nodes are solicited when a packet is crossing an SFC-enabled domain.

8. Privacy Considerations

The metadata defined in this document for subscriber identifiers may reveal private information about the subscriber. Some privacy-related considerations for Internet Protocols are discussed in [RFC6973] and [RFC6967]. In the light of these privacy considerations, it is important to state that the subscriber metadata MUST NOT be exposed outside the operator’s domain. This requirement is already supported by the NSH [RFC8300]. That is, NSH is stripped systematically at the egress of a service chain.

The information conveyed in subscriber identifiers is already known to an administrative entity managing an SFC-enabled domain. Some of that information is already conveyed in the original packets from a host (e.g., internal IP address) while other information is collected from various sources (e.g., GTP tunnel, line identifier, etc.). Conveying such sensitive information in packets may expose subscribers’ sensitive data to entities that are not allowed to receive such information. Misbehaving SFC egress nodes is a threat that may have negative impacts on privacy (e.g., some operational networks leak the MSISDN outside). Operators MUST ensure their SFC-enabled domain is appropriately conforming to the NSH specification so that any privacy-related information is not exposed outside the SFC-enabled domain.

Some use cases that rely upon the solution defined in this document may disclose some additional privacy-related information (e.g., a host identifier of a terminal within a customer premises for the parental control case). It is assumed that this information is provided upon approval from a subscriber [RFC8165]. For example, a customer may provide the information as part of its service management interface or as part of explicit subscription form.

9. Acknowledgements

Comments from Joel Halpern on a previous version and by Carlos Bernardos are appreciated. Contributions by Christian Jacquenet are thankfully acknowledged.

This work has been partially performed in the framework of the EU-funded H2020-ICT-2014-2 project 5G NORMA. Contributions of the project partners are gratefully acknowledged. The project consortium is not liable for any use that may be made of any of the information contained therein.

10. References

10.1. Normative References

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

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

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

10.2. Informative References

[I-D.ietf-sfc-hierarchical] Dolson, D., Homma, S., Lopez, D., and M. Boucadair, "Hierarchical Service Function Chaining (hSFC)", draft-ietf-sfc-hierarchical-09 (work in progress), June 2018.

[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-08 (work in progress), May 2018.

[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, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6967] Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Potential Solutions for Revealing a Host Identifier (HOST_ID) in Shared Address Deployments", RFC 6967, DOI 10.17487/RFC6967, June 2013, <https://www.rfc-editor.org/info/rfc6967>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC7620] Boucadair, M., Ed., Chatras, B., Reddy, T., Williams, B., and B. Sarikaya, "Scenarios with Host Identification Complications", RFC 7620, DOI 10.17487/RFC7620, August 2015, <https://www.rfc-editor.org/info/rfc7620>.

[RFC8165] Hardie, T., "Design Considerations for Metadata Insertion", RFC 8165, DOI 10.17487/RFC8165, May 2017, <https://www.rfc-editor.org/info/rfc8165>.

[TR22.808]
"3GPP TR22.808, Technical Specification Group Services and System Aspects; Study on flexible mobile service steering", 2015.

[TR22.862]
"3GPP TR22.862, Feasibility Study on New Markets and Technology Enablers - Critical Communications; Stage 1 (Release 14)", 2015.

[TR23.718]
"3GPP TR23.718, Technical Specification Group Services and System Aspects; Architecture enhancement for flexible mobile service steering", 2015.

[TR23.975]
"3GPP TR23.975, IPv6 Migration Guidelines", June 2011.

[TS23.003]
"3GPP TS23.003, Technical Specification Group Core Network and Terminals; Numbering, addressing and identification", 2015.

[TS29.212]
"3GPP TS29.212, Policy and Charging Control (PCC) over Gx/Sd reference point", December 2011.

[WT317] BBF, "Network Enhanced Residential Gateway", August 2015.

Authors' Addresses

Behcet Sarikaya
Denpel Informatique

Email: sarikaya@ieee.org

Mohamed Boucadair
Orange
Rennes 3500, France

Email: mohamed.boucadair@orange.com

Dirk von Hugo
Telekom Innovation Laboratories
Deutsche-Telekom-Allee 7
D-64295 Darmstadt
Germany

Email: Dirk.von-Hugo@telekom.de

Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: March 27, 2016

E. Wang
K. Leung
J. Felix
J. Iyer
Cisco Systems Inc.
September 24, 2015

Service Function Chaining Use Cases for Network Security
draft-wang-sfc-ns-use-cases-00

Abstract

Enterprise networks deploy a variety of security devices to protect the network, hosts and endpoints. Network security devices, both hardware and virtual, operate at all OSI layers with scanning and analysis capabilities for application content. Multiple specific devices are often deployed together for breadth and depth of defense. This document describes use cases of Service Function Chaining (SFC) when deploying network security devices in the manner described above and also puts forth requirements for their effective operation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Network security service nodes participate in Service Function Chaining (SFC) to provide comprehensive solutions for securing campus and data center enterprise networks. Often, network operators deploy various types and instances of security service nodes. These nodes

are complementary to one another for the purpose of coverage, depth of defense, scalability and availability.

In addition to packet forwarding, network security devices can buffer, inject or block certain packets, as well as proxy entire connections. Most of the network security devices maintain state at the connection, session or transaction levels. When used in a SFC environment these security Service Function actions and properties require careful design and extension including the Service Classifier and Service Function itself. This document attempts to describe the detailed use cases that lead to the requirements to support network security functions in 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 terms as defined in RFC 7498 [RFC7498], [I-D.ietf-sfc-architecture] and [I-D.ietf-sfc-nsh].

In addition the following terms are defined.

Security Service Function (Security SF): A Security Service Function is a Service Function that carries out specific security tasks. We limit the scope of security functions to network security in this document (as opposed to functions such as endpoint security). In addition to the general forwarding action, a Security Service Function can buffer, proxy, inject or block certain packets based on its policy. A Security Service Function can maintain state at the connection, session or transaction levels. Sample Security Service Functions are: Firewall, Intrusion Prevention/Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy.

Flow: A flow is a uni-directional traffic stream identified by network layer attributes, specifically IP addresses and TCP/UDP ports for TCP/UDP traffic.

Connection: A connection is a bi-directional traffic stream composed of two flows sharing the same network layer attributes.

3. Characteristics of Security Service Functions

Most Security Service Functions are stateful. They maintain state at the connection, session or transaction levels, depending on the OSI layers that they act on. Many Security Functions require seeing both directions of the client-server traffic in order to maintain state properly. Asymmetric traffic must be normalized before packets reach the Security Functions.

Security Service Functions operate on network layer data with specific behaviors. For example:

1. A Firewall tracks TCP state between the TCP client and server. TCP packets that do not correspond to the Firewall's maintained state are likely to be dropped.

2. A Firewall can modify the L3/L4 headers for NAT translation. The flow attributes in the packet header may be changed after the packet egresses the Firewall.

3. A Firewall can proxy a TCP connection by sending a TCP ACK on behalf of the endpoint. From the SFC perspective, this results in Service Function generated packets being injected into the service path in the reverse direction.

4. A Firewall or DDoS mitigator can inject TCP layer challenges to the originating client before the intended server receives a packet from the client.

Security Functions also handle packets and examine data at higher OSI layers. For example:

1. A Firewall can inspect the HTTP header and body data. Based on the inspection results, the firewall can decide to drop the packet and/or block the connection completely.

2. A Web proxy can inject an HTTP challenge page into an HTTP transaction for the purposes of authentication and identity collection.

3. At the enterprise edge, a TLS proxy, when authorized, operates as a trusted Man-in-the-Middle to proxy the TLS handshake and decrypt the packet data. The TCP payload may be completely different between ingress and egress of TLS Proxy.

4. A stream scanning service examines a certain set of application data. File scanning engines examine file streams of specific types.

4. Use Cases

4.1. Service Classification Use Cases

4.1.1. Service classification for bi-directional traffic

Many Security Service Functions require receiving bi-directional traffic of a connection. For example, a DDoS mitigator requires to see the return traffic to maintain proper state.

Return traffic (i.e. server to client response) should be classified based on the forward traffic (i.e. the client to server request). This allows server’s return traffic to be associated with the clients forward traffic. The forward and return traffic forms a single bi-directional connection and shares Service Function Paths with similar set of Service Functions.

In the figure below, the Service Classifier handling traffic from Host B must be able to identify return traffic (flow 2) and select the Service Function Path with "DDoS". Flow 1 and 2 form a connection and traverse DDoS in both directions.

```
                     ,--------------.
                    /                \
                   /      .--.        \
                  /      /    \        \
                 /      ( DDoS )        \
    ------------'      / \    / \        '--(1) Forward Flow (A=>B)-->
                      /   '--'   \
                     /            \
   +---+        .--./              '-------.                      +---+
   |   |       /    \             /         \                     |   |
   | A +------(  SC  )-----------(  Firewall )--------------------+ B |
   |   |       \    /             \         /                     |   |
   +---+        '--'               '-------'                      +---+
```

(a) Flows from Host A

```
                     ,------------.
                    /              \
                   /     .--.       \
                  /     /    \       \
                 /     ( DDoS )       \
    <------------'     / \    / \       '---(2) Return Flow (A<=B)----
                      /   '--'   \
                     /            \
   +---+            /              '-------.         .--.        +---+
   |   |           /              /         \       /    \       |   |
   | A +----------+--------------(  Firewall )------(  SC  )------+ B |
   |   |                          \         /       \    /       |   |
   +---+                           '-------'         '--'        +---+

    <-------------(3) Forward Flow (A<=B)---------------------------
```

(b) Flows from Host B

Figure 1: Forward and return flows between two hosts

4.1.2. Service Classifier to distinguish initiator and responder

Even if a Security Service Function requires receiving bi-directional traffic of a connection, it should not necessarily receive traffic initiated from all network segments for performance, availability, and scalability reasons. For instance, a DDoS mitigator is configured to receive bi-directional traffic initiated from the Internet, but skip traffic initiated from the internal network.

Traffic initiated from a network segment should be classified independently. In Figure 1(b), the Service Classifier for Host B must identify traffic initiated by Host B (flow 3) and classify it

independently. Such traffic bypasses the DDoS Service Function in this example.

The Service Classifier must distinguish between flow 2 and flow 3, both of which are from Host B to Host A. In other words, it must be able to identify the initiator and responder of a connection.

A Service Classifier that keeps certain state would be able to handle the above requirements with ease. The state should be accessible by each Service Classifier if there are multiple instances handling traffic sources from various network segments.

4.1.3. Service Classification based on network and application criteria

The Service Classifier evaluates SFC Policies (i.e. Service Policies) in order to determine the traffic and associated Service Function Paths. In the case of Security Service Functions, the Service Policies can contain match criteria derived from all OSI layers of the packet.

SFC classification is often based on network data, including but not limited to: Network interface port, VLAN, source and destination IP addresses, source and destination TCP and UDP ports, IP protocol, etc. These properties can be derived from the packet headers and are consistent across every packet of a flow.

There are match criteria that are desired by Security Service Functions that are either not present in the first packet, or are not present in every packet.

Those criteria may comprise "application data" from above the network layer, referred to as "application criteria". For example, a policy rule may state:

> for all TLS traffic, run the traffic through Service Function "TLS Proxy"

Another example of an application layer policy rule is:

> for all HTTP traffic with content containing file types of interest, run the traffic through Service Function "File Stream Scanner"

The Service Classifier for Security Service Functions needs to handle complex Service Policy. In some cases, this can be achieved by embedding the Service Classifier function into a Security Service Function, such that it can evaluate the application data as it becomes available.

4.1.4. Switching Service Function Paths based on inspection and scanning results

Network data is likely to be available on the first packet of the flow. When only network data is used as Service Policy match criteria, a stateful Service Classifier will be able to determine the forward and reverse Service Function Paths from the first packet (initial classification). The forward and reverse Service Function Paths remain unchanged for the entire life of the flow for these types of policies.

When the Service Policy contains application criteria, the policy rule may not be fully evaluated until several packets have passed through the chain. For example, TLS traffic can be identified only after the TLS Client Hello handshake message is observed.

Multiple classifiers may be required to provide sufficient classification granularity and complete a full evaluation of the Service Policy. In many cases, classification will be co-located with a Security Service Function that has the ability to inspect and scan the application data.

A new Service Function Path may be selected by a non-initial classification, different from the one determined by the initial classification.

The selection of a new Service Function Path can be reflected in the NSH Service Path Header as a new Service Path ID for the Service Function Forwarder to direct the packet accordingly.

The decision of a new Service Function Path often needs to be stored in Service Function and/or Service Classifier to ensure that subsequent packets of the flow follow the new path. This is because the data that triggers a new Service Function Path may be available from one particular packet only. For example, the packet with the TLS Client Hello message is used to identify a TLS session. Subsequent packets may not contain information for identifying the TLS sessions. All subsequent packets, without being classified again, must travel through the path with the "TLS Proxy" Service Function.

```
                      ,-----.
                     /       \
                    /  .---.  \
                   /  /     \  \
                  /  (  TLS  )  \
      -----------'   ( Proxy )   '-SFP-2. AVC:TLS Proxy:Firewall->
                     /\     / \
                    /  '---'   \                 .--.
    +---+      .-./            .'------.        (    )-.      +---+
    |   |     /   \          /         \     .'        )     |   |
    | A +----( AVC )--------(  Firewall )---(  Internet )----+ B |
    |   |     \   /          \         /     (        -'     |   |
    +---+      '-'            '-------'       '-(     )      +---+
                                                  '---'

      --------------SFP-1. AVC:Firewall ------------------------->
```

Figure 2: Mid-stream service function path update

Figure 2 illustrates a simple set of Security Functions deployed at the Internet edge. The default Service Function Path is SFP-1, with Service Functions "AVC" and "Firewall". When a TLS session is detected (e.g. by detecting the TLS Client Hello in the AVC Service Function), packets of the flow from that point on are switched to SFP-2, which contains "TLS Proxy" between "AVC" and "Firewall" to decrypt the TLS traffic for inspection.

Packets	Service Function Path
TCP Handshake	SFP-1. AVC:Firewall
TLS Client Hello	SFP-1; Switched to SFP-2 after AVC
Rest of TLS HS	SFP-2. AVC:TLS Proxy:Firewall
HTTPS Data	SFP-2. AVC:TLS Proxy:Firewall

Table 1: SFP taken by each packet in an HTTPS connection

Table 1 lists the Service Function Path for each packet in an HTTPS connection, from the TCP 3-way handshake to the HTTPS data packets. A new Service Function Path is selected in the middle of the connection after the TLS Client Hello is observed.

4.2. Service Function Use Cases

4.2.1. Service Classifier-capable Service Function

Service Functions that are capable of selecting a new Service Function Path must have the Service Classifier function integrated. Such Service Functions are often responsible for classification using their inspection and scanning results and updating Service Function Paths based on the Service Policy.

4.2.2. Service Functions operating on L5 or L7 data

Certain Security Service Functions operate on L5 to L7 data. For example, a "TLS Proxy" consumes a TCP stream without retransmitted or overlapping TCP segments. A "Web Proxy" operates on TCP stream of HTTP traffic. The data consumed by such Service Functions may not be in the original packet frame format, and the data may not contain the original L2-L4 header information. Such Service Functions can obtain the session or flow information from the SFC metadata carried in NSH.

4.2.3. Service Function mid-stream pick-up

When a new Service Function Path is selected as a result of Service Policy re-evaluation with application layer policy metadata, a new Service Function may need to start handling packet frames in the middle of a flow. This is referred to as "mid-stream pick-up". Although this is mid-stream from a flow perspective, it is still a complete data stream from the Service Function perspective (e.g., although "TLS Proxy" Service Function may not see the prior TCP handshake packets, it still sees the entire TLS stream). Similarly, transaction based Service Functions only handle packets belonging to a particular transaction. Such Service Function may use the flow ID metadata carried in NSH to link the session back to the flow.

Packet	AVC	TLS Proxy	Firewall
TCP SYN	X		X
TCP SYN/ACK	X		X
TCP ACK	X		X
TLS Client Hello	X	X	X
Rest of TLS HS	X	X	X
HTTPS Data	X	X	X

Table 2: Service Functions visited by each packet in an HTTPS connection

Table 2 lists the Service Functions visited by each packet from an HTTPS connection. The first packet that the Service Function "TLS Proxy" receives is the TLS Client Hello, as opposed to the TCP handshake packets prior to it.

4.2.4. Bypassing for a particular Service Function

Certain Security Service Functions can be compute-intensive while only serving a particular task. It may be required to bypass such a Service Function in the middle of a flow. For example:

- "Firewall" may request offloading of certain flows to fast forwarding engine with minimal inspection

- "HTTP Inspector" may decide to not inspect video streams from a site with a high reputation

- "TLS Proxy" may have to avoid decryption of banking traffic for compliance reasons

The decision to bypass a Service Function is made by the Service Function with its static policy, the inspection results and/or mid-stream evaluation of Service Policy.

Even if a flow is offloaded or bypassed, the Security Service Function may want to continue receiving critical packets for state tracking purposes. For example, "Firewall" may want to receive TCP control packets, and "HTTP Inspector" may want to track each transaction in the same flow.

```
                        .-------.
                       /         \
                      (   HTTP    )
                      ( Inspector )
                   ,---------------------------.
                  /      '---+---'              \
                 /           |                   \
                |            |                    |
                |       .---+---.                 |
                |      /         \                |
                |     (  Firewall )               |
                |   ,---------------.             |
                |  /    '---+---'    \            |
                | /         |         \           |
                | |         |          |          |
                | |     .---+---.     |TCP       |HTTP
                | |    /         \    |control   |Headers
                | |   ( Offloading)   |packets   |
                | |   (    Node   )   |          |
                | |  ,-------------.  |          |
                | | /   '-------'   \ |          |
                | | |               | |          |
                | | |Offloaded      | |          |
                | | |packets        | |          |
                | | |               | |          |
                v v v               v v          v
```

Figure 3: Service function bypass examples

The offloading node can be either the Service Function Forwarder or a capable Service Function with a built-in stateful offloading path (Figure 6). The offloading path tracks the flow state and identifies critical packets to be sent to the bypassed Service Function.

```
                                        Offloading Node
                                       .--------------.
                                      /                \
                                     /        SFF       \
                                    (                    )
                                    (   +++++++++++++   )
                                    (   + Offloading +   )
                                     \.'+    Path    +  /
                  +++++++++++++      .'\ +++++++++++++ /
                  + /-------\  +    .'   '--------------'
                  + | Flow  |  +  .'
                  + | State |  +.'
                  + \-------/  +
                  +            +.
                  + Offloading + '.       Offloading Node
                  +    Path    +   '.    .--------------.
                  +++++++++++++     '. /                \
                                     /.       SF         \
                                    (  '.                )
                                    (   +++++++++++++   )
                                    (   + Offloading +   )
                                     \  +    Path    +  /
                                      \ +++++++++++++ /
                                       '--------------'
```

Figure 4: Service function offloading node

To steer traffic to the path that avoids the bypassed Service Function, a Service Function may update the SFC metadata in the packet if the Service Function has knowledge of the relevant Service Function Paths. Alternatively, a Service Function may signal the Service Classifier to update the Service Function Path to exclude the Service Function. Service Function Path updates may be accomplished by selecting a new path (i.e. a new Service Path ID) with the Service Function excluded.

Service Function bypass may also follow the procedure described in "Service Function Simple Offloads" [I-D.kumar-sfc-offloads], where the Service Function signals the Service Function Forwarder to offload a flow. The Service Function Forwarder caches the offload request and bypasses the Service Function in the service path for the remainder of the flow.

4.2.5. Tap mode Service Functions

Certain Service Functions such as an IDS may operate in "tap" mode, i.e. they consume a packet instead of passing the packet through.

The Service Function Forwarder should send copies of packets to tap mode Service Functions.

```
                              .---.
                             /     \
                            ( [IDS] )
                             \     /
                              '-+-'
                         .---.  /|\      .-------.
                        /     \  |      /         \
                       (  DDoS ) |     (  Firewall )
                        \     /  |      \         /
                         '-+-'   |copy   '---+---'
                          /|\    |          /|\
                           |     |           |
                +---+      \|/    |          \|/        +---+
                |   |     ,--+-----+-----------+----.    |   |
                | C +----(            SFF           )----+ S |
                |   |     '-------------------------'    |   |
                +---+                                    +---+

                [ ] denotes a packet sink
```

Figure 5: Tap mode service functions in SFC

Figure 3 illustrates an example of tap mode Service Function and their insertion into a Service Function Chain. The IDS Service Function receives copies of packets from the Service Function Forwarder.

4.3. Service Data Handling Use Cases

4.3.1. Dropping packets and closing flows

A Security Service Function may decide to drop the current packet or close a particular flow based on its inspection and scanning results, and the associated security policy.

A Service Function may drop packets without forwarding them out, or it may forward and mark such packets to be dropped by the Service Function Forwarder, referencing the flow by its flow ID in the SFC metadata.

A flow-close action usually needs to be taken by multiple stateful Service Functions, as well as the Service Function Forwarder and the Service Classifier, in order to clear their state for such a flow. Any subsequent packets of the closed flow are denied.

```
                          .-----.       .-----.       .-----.
                         /       \     /       \     /       \
                        (  SF-1  )   (  SF-2  )   (  SF-3  )
                         \       /     \       /     \       /
                          '+--+'       '+--+'       '----'
                          /|\ |        /|\ |
                           |  |         |  |Block
                           | (P)        |  |Flow(P)
                           |  |         |  |
                           |  |         |  |
                          (P) |        (P) |
                           |  |         |  |
                           | \|/        | \|/
                  .--.    ,--+--+---------+- : ------------.    .--.
                 /    \  /                                  \  /    \
      --(P)--->(      )(               State Update         )(      )
               (  SC  )(                                    )(  SC  )
      --(Q)-X->(      )(     SFF                            )(      )
                 \    /  \                                  /  \    /
                  '--'    '--------------------------------'    '--'
```

Figure 6: Flow close action example

Figure 4 shows an example of closing a flow after SF-2 processes packet P. The flow close indication can be included in the packet or message returned from SF-2 to the Service Function Forwarder. The flow state update may be distributed to the Service Function Forwarder, Service Classifier and other Service Functions. The distribution mechanism is outside the scope of this document.

4.3.2. Service Function injected new packet

Security Service Functions may inject new packets into an existing flow in either direction. For example,

- "Web Proxy" inserts an HTTP page challenging the client to login, in order to obtain the client's identity. This is in response to a packet (likely HTTP Request) but in the opposite direction of the flow.

- "Firewall" checks an idle TCP connection by sending TCP keepalives to the client and/or server (known as "TCP dead connection detection"). This is on existing flows but not responding to a prior packet.

- "Firewall" sends ICMP error message after dropping a packet. This is in response to the prior packet but on a new flow.

The Service Function or Service Classifier needs to conduct a lookup of the reverse Service Function Path and populate the NSH Service Path Header. The approaches described in [I-D.penno-sfc-packet] may be adopted to support this use case.

4.3.3. Service Function initiated connections

A Service Function may need to create its own connections that are not associated with any client connection. Use cases include probing of servers behind a web proxy. In such cases, there will be no existing metadata for the Service Function to use to establish this connection. Such connections should be classified just like any other connections traversing the Service Function Path, as there may be Service Functions that are required to perform operations such an NAT on such connections in order for it to reach its destination.

```
                <------ SFC-1. Firewall:LB:IPS ------>

           +---+     .-------.       .--.       .-.     +---+
           |   |    /         \     /    \     /   \    |   |
           | C +---(  Firewall )---(  LB  )---( IPS )---+ S |
           |   |    \         /     \    /     \   /    |   |
           +---+     '-------'       '--'       '-'     +---+

                                <-- SFC-2. LB:IPS -->
```

Figure 7: SFC for service function initiated connection

A Service Classifier-capable Service Function may conduct service classification to determine the Service Function Path for the Service Function initiated connection. It can add an NSH with the proper Service Path Headers to the packets, and the Service Function would be the first SF on the chain. Response traffic follows a reverse Service Function Path and terminates at the Service Function. The number of Service Path Identifiers increases with more Service Functions bearing such capability.

A Service Function may send native packets without NSH when it is not capable of service classification. Such traffic is handled by the Service Classifier, which will populate the traffic with the appropriate NSH.

4.3.4. Security classification results

Security Service Functions may generate security classification results (e.g. policy actions and inspection results) while processing the packet data. Certain actions such as packet drop and flow closure can be taken immediately.

However, Service Functions can choose not to take any action immediately. Instead, it may pass the classification results to the subsequent Service Functions or to a control point.

Security classification results may be carried in NSH metadata as a score value. The score can be relayed and refined by other Security Service Functions along the path. Figure 8 below depicts an example of accumulating the client’s score based on the Service Function’s classfication result. The client’s reputation score is 6 as reported by the Service Function "Reputation", and the score is then passed to the next Service Function "Web Proxy" as the initial score for the connection. "Web Proxy" reduces the score to 3 after detecting access to a low reputation website. The Service Function "File Scanner" is involved due to the low score so far. After the "File Scanner" conducts scanning on the downloaded file and identifies it to be a malware, it updates the score to be -5 which is below the threshold for the connection to be blocked.

```
                                                  .------.
       +---+     .--------.      .--------.     /        \    +---+
       |   |    /          \    /          \   (   File   )   |   |
       | C +---( Reputation )--(  Web Proxy )--(  Scanner )---+ S |
       |   |    \          /    \          /    \        /    |   |
       +---+     '--------'      '--------'      '------'     +---+
                      |              |              |
                      V              V              V
      +----------------------+--------------+-------------+
      |        | Client      | Accessing    | Malware     |
      | Reason | Reputation  | website rep  | found in    |
      |        | score 6     | score 2      | download    |
      +--------+------+------+-------+------+-------+-----+
                      |              |              |
                  +---+---+          |              |
                  |   6   |          V              |
                  +---+---+          |              |
                      '------>-------+              V
                                     |              |
                                 +---+---+          |
                                 |   3   |          |
                                 +---+---+          |
                                     '------>-------+
                                                    |
                                                +---+---+
                                                |  -5   |
                                                +---+---+
                                                    |       /-------\
                                                    |       |       |
                                                    '------>| Block |
                                                            |       |
                                                            \-------/
```

Figure 8: Security classification result with accumulated client score

Alternatively, each participating Service Function may send its own classification result to a central Service Function or control point for aggregation. Actions are then taken by a specific Service Function or control point based on the accumulated results. Figure 9 illustrates this option.

```
                                                  .------.
    +---+     .--------.      .--------.     /        \    +---+
    |   |    /          \    /          \   (   File   )   |   |
    | C +---( Reputation )--(  Web Proxy )--(  Scanner )---+ S |
    |   |    \          /    \          /    \        /    |   |
    +---+     '--------'      '--------'      '------'     +---+
                  |                |              |
                  V                V              V
    +--------+-------------+--------------+------------+
    |   SF   | Client      | Accessing    | Malware    |
    |Classify| reputation  | website rep  | found in   |
    | Results| score 6     | score 2      | download   |
    +--------+------+------+-------+------+-----+------+
                    |              |            |
                    |              |            |
                    |              |            |
                    |              |            |  +------+  /-------\
                    |              |            '->|      |  |       |
                    |              '-------------->| Eval +->+ Block |
                    '------------------------------>|      |  |       |
                                                    +------+  \-------/
```

Figure 9: Aggregation of security classification results

5. General Requirements

The above use cases lead to the following requirements for applying SFC to security traffic.

1. SFC MUST support the use of stateful Service Classifiers and Service Functions if present.

2. Service Classifiers MUST have the ability to classify forward and the corresponding reverse Service Function Paths.

3. SFC MUST support the use of Service Policies with network and application layer match criteria if supported by Service Classifier.

4. SFC MUST support Service Function Path update or selection of a new path by a Service Classifier in the middle of a flow.

5. SFC SHOULD allow packet frames carrying only L5 and upper layer traffic data without L2-L4 headers.

6. SFC MUST allow tap mode Service Functions.

7. SFC policies MUST support tap mode Service Functions.

8. SFC MUST support packet injection to the opposite direction of a Service Function Path.

9. SFC SHOULD support bypass of a Service Function in the middle of a connection while allowing necessary control packets to reach the Service Function.

6. Security Considerations

This document describes use cases for Security Service Functions to participate in SFC. There are cases such as picking up traffic from the middle of a packet stream or handling packets without L2-L4 headers. Security Service Functions must process those types of traffic properly and associate them with the appropriate internal state.

While each Security Service Function applies its own implementation to secure the internal data, communications between Service Functions need to be secured as well. Measures must be taken to ensure metadata such as security classifications carried in NSH is not tampered.

7. Acknowledgments

The authors would like to thank Paul Quinn, Reinaldo Penno and Jim Guichard for their detailed review, comments and contributions.

8. IANA Considerations

This document includes no request to IANA.

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-08 (work in progress), May 2015.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-00 (work in progress), March 2015.

[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>.

[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>.

9.2. Informative References

[I-D.kumar-sfc-offloads]
Surendra, S., Guichard, J., Quinn, P., and J. Halpern, "Service Function Simple Offloads", draft-kumar-sfc-offloads-01 (work in progress), September 2015.

[I-D.penno-sfc-packet]
Penno, R., Pignataro, C., Yen, C., Wang, E., and K. Leung, "Packet Generation in Service Function Chains", draft-penno-sfc-packet-00 (work in progress), September 2015.

Authors' Addresses

Eric Wang
Cisco Systems Inc.
170 W Tasman Dr
San Jose, CA 95134
U.S.A.

Email: ejwang@cisco.com

Kent Leung
Cisco Systems Inc.
170 W Tasman Dr
San Jose, CA 95134
U.S.A.

Email: kleung@cisco.com

Jeremy Felix
Cisco Systems Inc.
170 W Tasman Dr
San Jose, CA 95134
U.S.A.

Email: jefelix@cisco.com

Jay Iyer
Cisco Systems Inc.
170 W Tasman Dr
San Jose, CA 95134
U.S.A.

Email: jiyer@cisco.com

Network Working Group
Internet Draft
Intended status: Standards Track
Expires: August 5, 2016

X. Yang
L. Zhu
G. Karagiannis
Huawei Technologies

February 5, 2016

SFC Trace Issue Analysis and Solutions
draft-yang-sfc-trace-issue-analysis-01.txt

Abstract

This document analyzes and provides solutions for some unaddressed SFC Traceroute issues.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-sfc-oam-framework] provides a reference framework for SFC OAM and lists several OAM functions that help to monitor the SFC components. [I-D.penno-sfc-trace] describes a solution of SFC traceroute based on NSH header, but only a subset of the requirements provided in [I-D.ietf-sfc-oam-framework] are addressed. The goal of this draft is to provide solutions for the rest of the requirements and as well as analyze other potential issues.

2. Terminology

The reader should be familiar with the terms contained in [I-D.ietf-sfc-architecture], [I-D.ietf-sfc-oam-framework], [I-D.ietf-sfc-nsh] and [I-D.penno-sfc-trace].

3. SFC Trace

In [I-D.ietf-sfc-oam-framework], four requirements on the SFC trace function are provided:

o) Ability to trigger action from every transit device on the tested layer towards an SF or through an SFC, using TTL (Time To Live) or other means.

o) Ability to trigger every transit device to generate response with OAM code(s) on the tested layer towards an SF or through an SFC, using TTL or other means.

o) Ability to discover and traverse ECMP paths within an SFC.

o) Ability to skip un-supported SF’s while tracing SF’s in an SFC.

The first two requirements are met and solved in [I-D.penno-sfc-trace], but the third and fourth requirements are not yet addressed.

Besides these two requirements, there are several issues that need to be analyzed, such as reporting SFF information, TTL-agnostic solution, etc. These issues are further described in the following sub-sections of this document.

3.1. Skip Unsupported SFs

As stated above, the SFC trace function is preferred to skip un-supported SF while tracing. The current solution depends on the SF to provide this information. This means that if the SF will not support the SFC trace function, then no information will be reported back. The result is similar to an error situation, and may disrupt the optimal control plane operation.

One possible solution is to move all trace related functionalities to the SFF, without making any assumptions on the SF for supporting the trace functionality. If the SF does not support the trace function, then the SFF can provide additional information, such as the IP address of the SF instead.

3.2. ECMP Support

When ECMP is deployed, there can be multiple rendered service paths corresponding to one service path. One trace packet can only traverse one of the rendered service path and trigger reports along that path. Furthermore, trace packets sent at different time may follow different rendered service path, which makes it harder to monitor the overall situation of the service path.
To fulfill the need of "discover and traverse all ECMP paths ", one possible solution for the SFF is to broadcast the trace packet to all possible next hops. To identify the exact rendered service path that the packet traversed, information needs to be recorded in the trace packet. The most straightforward way is to add each SF/SFF ’s information, e.g., name, to the packet. However, uncontrolled broadcasting can generate a significant amount of traffic on the data plane, which may impact the normal forwarding of the service traffic. Using TTL-agnostic solutions can help to reduce the number of broadcasting packets. More study is needed on this topic, but it is considered to be out of the scope of this document.

3.3. Reporting SFF Information

Providing information of SFFs can help identifying errors on the service path in situations like locating the place where forwarding errors occurred, detecting loops, etc.

3.4. TTL-agnostic Solution

Because NSH is not containing a TTL field, the SFC trace function does not necessarily need to follow a traditional TTL based trace solution. In other words, the trace can be done by sending one trace packet to trigger every traversed SFF to send reports of SFs and/or SFFs along the traversed path.

3.5. Sending Report Message to OAM Controller

The SFC OAM control plane can be centralized or distributed. In the centralized case, the trace report packet can be forwarded to the control plane directly. In the distributed case, however, the OAM control entity may not be directly connected with the SFF, so a dedicated control path or a reverse path is needed to forward the report packet.

3.6. More Command Parameters

Information like service path ID, starting service index, and report address are needed to perform a trace. As described in the above sub-sections, there are many aspects impacting the behavior of a particular trace process. They all can be captured as trace command parameters. The following list gives several command parameters that are worth to be taken into consideration:

- o) service path identification
- o) starting service index
- o) Service Index Limit (SIL, described in Section 3.7)
- o) report destination IP address and port
- o) report object: sending report of SF, SFF or both
- o) ECMP support
- o) number of queries to send per hop
- o) time to wait for a response/report
- o) number of queries that can be sent out simultaneously
- o) time interval between sending queries

3.7. Basic SFC Trace Header

The trace headers shown in Figure 1 (Trace Request Header) and in Figure 2 (Trace Report Header) are used as a basis for the SF trace operation described in the following sections.

```
 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|1|C|R|R|R|R|R|R|   Length  |  MD-type=0x2  |  OAM Protocol | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|          Service Path ID                      | Service Index | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                Mandatory Context Header                       | |S
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |F
|                Mandatory Context Header                       | |C
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                Mandatory Context Header                       | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                Mandatory Context Header                       | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ <
|Trace Msg Type |     SIL       |     LSI       | Number Index  | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|          Dest Port            |          Reserved Flags       | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |T
|                       Dest IP Address                         | |R
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |A
|                       Dest IP Address                         | |C
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |E
|                       Dest IP Address                         | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                       Dest IP Address                         | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+/
|  Next Hop Len |        Next Hop Info ...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Trace Request Header

Trace Msg Type: 1 for Trace Request and 2 for Trace Report
SIL: Service Index Limit: At least one less than the Starting Index
LSI: Last Service Index, record the service index of the last service function which processed the packet, default valve is the starting SI
Number Index (NI): number of hops the packet has traversed, default value is 0
Reserved Flags: can be used to indicate the function blocks that need to send reports, whether uses ECMP, etc.
Dest Port: The trace report must be sent to this destination Port

Dest IP: the trace report must be sent to this destination IP address, IPv6 format.

Next Hop Len: The length of Next Hop Info in 4-byte words. The field only exists when needed.

Next Hop Info: A string that records the identification of the next hop, e.g., name, IP address, etc. The field only exists when needed.

```
 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|1|C|R|R|R|R|R|R|   Length  |  MD-type=0x2  |  OAM Protocol | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|          Service Path ID                      | Service Index | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                Mandatory Context Header                       | |S
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |F
|                Mandatory Context Header                       | |C
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                Mandatory Context Header                       | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                Mandatory Context Header                       | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ <
|Trace Msg Type |     SIL       |     LSI       | Number Index  | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|          Dest Port            |          Reserved Flags       | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                       Dest IP Address                         | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |T
|                       Dest IP Address                         | |R
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |A
|                       Dest IP Address                         | |C
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |E
|                       Dest IP Address                         | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|   SF Info Len |        SF Info ...                            | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|  SFF Info Len |       SFF Info ...                            | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+/
```

Figure 2: Trace Report Header

SF Info Len: The SF Info length in 4-byte words. This field is omitted when reporting an SFF.

SF Info: A string that represents the identification of an SF. This field is omitted when reporting an SFF.

SFF Info Len: The SFF Info length in 4-byte words. This field is omitted when reporting an SF.

SFF Info: A string that represents the identification of an SFF. This field is omitted when reporting an SFF.

4. Service Function Behavior

As stated in 3.1, in order to skip unsupported SFs the trace functionalities is moved to the SFFs. In this situation, the SF only needs to have the ability to process the NSH and no assumption is made on whether the SF is supporting the trace function.

When an SF receives a trace packet, it performs the following actions:

1. Decrement Service Index in NSH

2. (Only conducted when trace function is supported) If Service Index is equal to the Services Index Limit, replace the Next Hop Info field with its identification information

3. Send packet back to SFF

5. Service Function Forwarder Behavior

The trace functionality is mainly implemented in the SFF. Section 5.1 describes the basic behavior of the SFF. Sections 5.2 and 5.3 describe the changes to the SFF default behavior, assuming either that the SFF information reporting is enabled or by adopting the TTL-agnostic solution.

5.1. Skip Unsupported SFs

When an SFF receives a trace request packet, it performs the following actions:

1. Checking if the trace packet should be dropped

2. If SI is 1 greater than SIL, and if LSI is greater than SI, the SFF will add the Next Hop Info field to the trace header with its next hop information. If SI is 1 greater than SIL, and if LSI is equal to SI, the SFF will overwrite the Next Hop Info field in the header.

   NOTE: This assumes that the SFF cannot identify whether the next hop is an SF or an SFF. If the SFF can identify the type of the next hop, it can then add the Next Hop Info field to the trace header until finding the SI is 1 greater than SIL and the next hop is an SF.

3. If LSI is greater than SI, change the LSI to be equal to SI.

4. Forward the trace packet to the next hop

If at least one of the following conditions is met, the trace packet will be dropped and a trace report packet is generated:

o) the SI is equal or less than SIL

o) the SFF cannot find the next hop to forward the packet

o) the SI is equal to zero

The following steps are applied to generate a trace report packet:

i. Fill in the NSH header with proper values

ii. Copy the information from the trace request header to the trace report header. (The Next Hop Info field's information will be copied to SF Info field.)

iii. Change the Trace Msg Type field to 2 (Trace Report).

5.2. Reporting SFF Information

As described in Section 5.1, a trace request packet will only trigger one report packet which contains the information of the last hop SF. To completely monitor a service path, several trace request packets are needed. When reporting SFF information, similar behavior is needed to avoid redundant reports of SFFs, i.e., a trace request packet will only trigger report packets generated on SFFs between the last hop SF and the second last hop SF.

Compared to the default behavior described in Section 5.1, only the step 2 is changed when an SFF receives a trace request packet:

o) If SI is 1 greater than SIL, and if LSI is greater than SI, the SFF will add information of the next hop to the trace header. If SI is 1 greater than SIL, and if LSI is equal to SI, the SFF will overwrite next hop information in the header, increase NI and trigger an SFF report.

The NI field is used to record the order of the report packets, which helps to sequence the reports in the control plane.

The following steps are taken when an SFF report is triggered:

1. Fill in the NSH header with proper values

2. Copy the information from the trace request header to the trace report header except for the Next Hop Info field (if it exists).

3. Add the SFF Info Len and SFF Info fields to the report header with the SFF's identification information

4. Change the Trace Msg Type field to 2 (Trace Report).

5.3. TTL-agnostic Solution

As described in Section 3.4, when using TTL-agnostic solution, only one trace request packet is needed to conduct a complete trace process.

Compared to the default behavior described in Section 5.1, only the step 2 is changed when an SFF receives a trace request packet:

o) If LSI is greater than SI, the SFF will add information of the next hop to the trace header, increase NI and trigger an SF report

o) If LSI is equal to SI, the SFF will overwrite next hop information in the header, increase NI and trigger an SFF report

In this scenario the Next Hop Len and Next Hop Info fields are always needed in the trace header, except in the situation that the SFF can identify the type of the next hop. In that situation, the two fields are only needed when the next hop is an SF and when its ID needs to be added to the trace header by the SFF.

The report packet generation process is similar to the ones described in Section 5.2 and 5.3.

6. NSH-unware SF

As stated in section 5, the trace functionality described in this draft is mainly implemented by the SFF instead of SF. As a result, the functionality can also be used in the case where the SFs do not support NSH.

In such a case (or more broadly, in a case where the SFs do not support SFC encapsulation), a proxy is needed between the SFF and SF to process the NSH header. The proxy will remove the NSH header before forwarding the packet to the SF and apply the encapsulation to the packet when it’s returned back. It is expected that proxy can conserve the trace request header when removing the NSH header and restore it when applying the encapsulation. Like a SF supporting the trace functionality (as stated in section 3.1), if the proxy supports the trace functionality, it can provides additional information of the SF by modifying the Next Hop Info field. It can apply the modification either when receiving the packet or when applying the encapsulation.

7. IANA Considerations

IANA considerations are needed for the registration of (1) OAM Protocol Type and (2) OAM protocol Message type.

8. Security Considerations

As stated in section 3.5, if the SFC OAM control plane is centralized, the trace report packets can be forwarded to the control plane directly. In this case, sending one trace request packet can cause one or more report packets (based on whether reporting SFF information and/or adopting TTL-agnostic solution) sent to the controller. This may bring potential security issue, like DDoS attack. One possible way to solve this problem is to authenticate the trace request packet. Further study is needed for this aspect.

9. Acknowledgements

To be done.

10. References

10.1. Normative References

10.2. Informative 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-oam-framework] Alfrin, S., Krishnan, R., Akiya, N., Pignataro, C., and A. Ghanwani, "Service Function Chaining Operation, Administration and Maintenance Framework", draft-ietf-sfc-oam-framework-00 (work in progress), August 2015.

[I-D.penno-sfc-trace] Penno, R., Quinn, P., Pignataro, C., and D. Zhoud, "Service Function Chaining Traceroute", draft-penno-sfc-trace-03 (work in progress), September 2015.

[I-D.ietf-sfc-nsh] Quinn, P., and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-01 (work in progress), July 2015.

Authors' Addresses

Xu Yang
Huawei Technologies
Huawei Building,No. 3, Xinxi Road, Haidian District, Beijing
China
Email: yangxu5@huawei.com

Lei Zhu
Huawei Technologies
Huawei Building,No. 3, Xinxi Road, Haidian District, Beijing
China
Email: Lei.zhu@huawei.com

Georgios Karagiannis
Huawei Technologies
Hansaallee 205,
40549 Dusseldorf,
Germany
Email: Georgios.Karagiannis@huawei.com