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

Abstract

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

Status of This Memo

Table of Contents

1. Introduction

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

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

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

```
                +----------------+
                |SFC-unaware     |
                |Service Function|
                | (Legacy SF)    |
                +----+----+------+
                     ^    |
                     |    |
                +----+----+------+
                |Switch(optional)|
                +----+----+------+
                     |    |
                  (2)|    |(3)
                     |    |
                +----+----V--------+
          (1)   |      SFC         | (4)
        -------->|     Proxy       +------->
                +------------------+
```
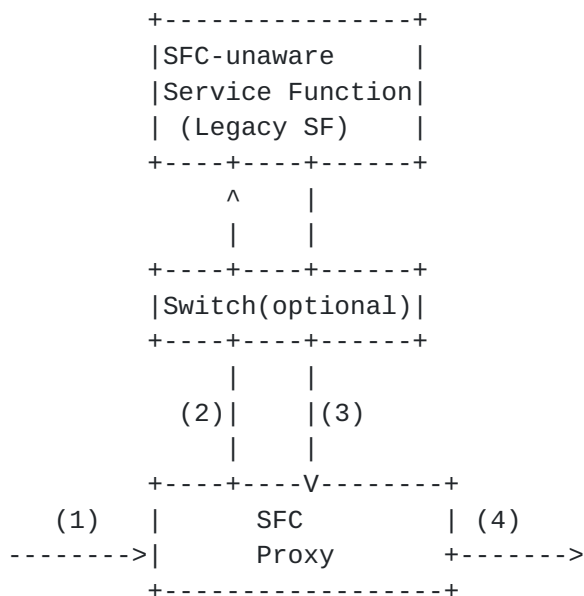
        Figure 1: Procedure of a packet processed by a legacy SF

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

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

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

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

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

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

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

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

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

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

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

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

2. Terminology

The terminology used in this document is defined below:

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

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

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

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

3. Mechanisms

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

3.1. For Transparent Service Functions

3.1.1. VLAN

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

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

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

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

3.1.2. VXLAN

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

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

3.1.3. Ethernet MAC Address

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

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

3.1.4. 5-tuple

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

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

3.2. For Non-transparent Service Functions

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

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

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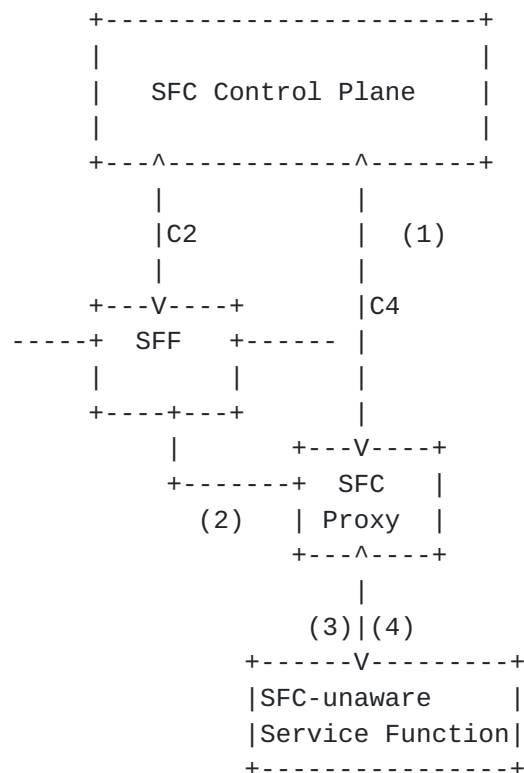
4. Operation Considerations

4.1. Examplar Mechanisms

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

Table 1: Mapping Examples

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

4.2. Challenges to Support Legacy SF

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

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

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

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

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

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

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

4.3. Metadata

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

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

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

5. Security Considerations

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

6. Acknowledgement

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

7. References

7.1. Normative References

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

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

7.2. Informative References

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

Authors' Addresses

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

   Email: haibin.song@huawei.com

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

   Email: youjianjie@huawei.com

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

   Email: lucy.yong@huawei.com

   Yuanlong Jiang
   Huawei
   Bantian, Longgang district
   Shenzhen 518129
   China

   Email: jiangyuanlong@huawei.com

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

Email: ldunbar@huawei.com

Nicolas Bouthors
Qosmos

Email: nicolas.bouthors@qosmos.com

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

Email: ddolson@sandvine.com