SFC working group
Internet-Draft
Intended status: Informational
Expires: October 7, 2016

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

April 5, 2016

SFC Header Mapping for Legacy SF
draft-song-sfc-legacy-sf-mapping-07

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 having support for the SFC header, or even being aware of it. This document provides a mechanism between an SFC proxy and an SFC-unaware service function (herein termed "legacy SF"), to identify the SFC header associated with a packet that is returned from a legacy SF, without an SFC header being explicitly carried in the wired protocol between SFC proxy and legacy SF.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 7, 2016.

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, and they neither have to be aware of the SFC header, nor interpret it. It is a straightforward function for an SFC proxy to remove an SFC header to send a packet to a legacy SF, but it is not obvious what SFC header should be added to packets arriving at the SFC proxy from the legacy SF.

This document provides a mechanism between an SFC proxy and a legacy SF, to identify the SFC header associated with a packet that is returned from a legacy SF, without anything in the SFC header being explicitly carried in the wired protocol between SFC proxy and legacy SF. The motivation for supporting legacy SF is that existing service functions don't need to be upgraded to support SFC, removing one barrier to wide adoption of SFC.

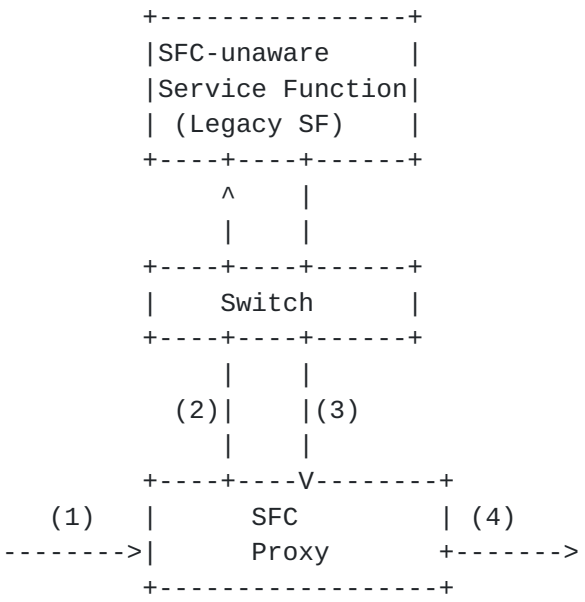


Figure 1: Procedure of a packet processed by a legacy SF

The legacy service function (i.e., "SFC-unaware Service Function" in the Figure 1) only handles packets without SFC header, because it does not understand the SFC header. Note that different classes of legacy SF may have varying 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.).

This document focuses heavily on legacy SFs that are transparent at layer 2. In particular, we assume the following about layer-2-transparent legacy SFs:

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 following 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, and removes its SFC header, which may optionally contain metadata, and store the SFC header locally, and then sends the de-encapsulated packet to the SF. After the SF processes the packet, the packet will be sent back 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.

The key problem contemplated in this document is: what layer 2 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? Below, 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.

When metadata is sent without any associated payload (congruent metadata) and the associated service function is a legacy one, then SFF MUST relay the metadata to the next hop SFF, without sending the metadata to SFC proxy. For some types of metadata, the metadata should be saved in case it needs to be added to packets injected by the legacy SF.

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.

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 original service packet header (Layer 2, layer 3, and layer 4) sent to it, but it may drop packets.

Non-transparent SF: A service function that changes some part of the original service packet header sent to it.

Original Service Packet: The payload in an SFC encapsulation packet or a packet constructed based on the original payload.

SFC Proxy: A network function that operates as an SF node within the SFC architecture while delegating application functions to one or more attached Legacy SFs by acting as an adapter or bridge between the SFC protocol and SF wire protocols understood by the legacy SF.

3. Mechanisms

The mechanisms used in this document require that each forwarding entity and its connected service functions in the same layer 2 network. The following are considerations mainly for transparent SFs. If the original payload packet is a layer 2 packet, and the mapping method used is layer 2 MAC address, then the assumption is that the SF does not need to look into the layer 2 header. If it does, other mechanisms should be used.

3.1. For Transparent Service Functions

If the service function is transparent to packet headers, the following methods can be used for SFC header mapping.

3.1.1. Layer 2 MAC Address

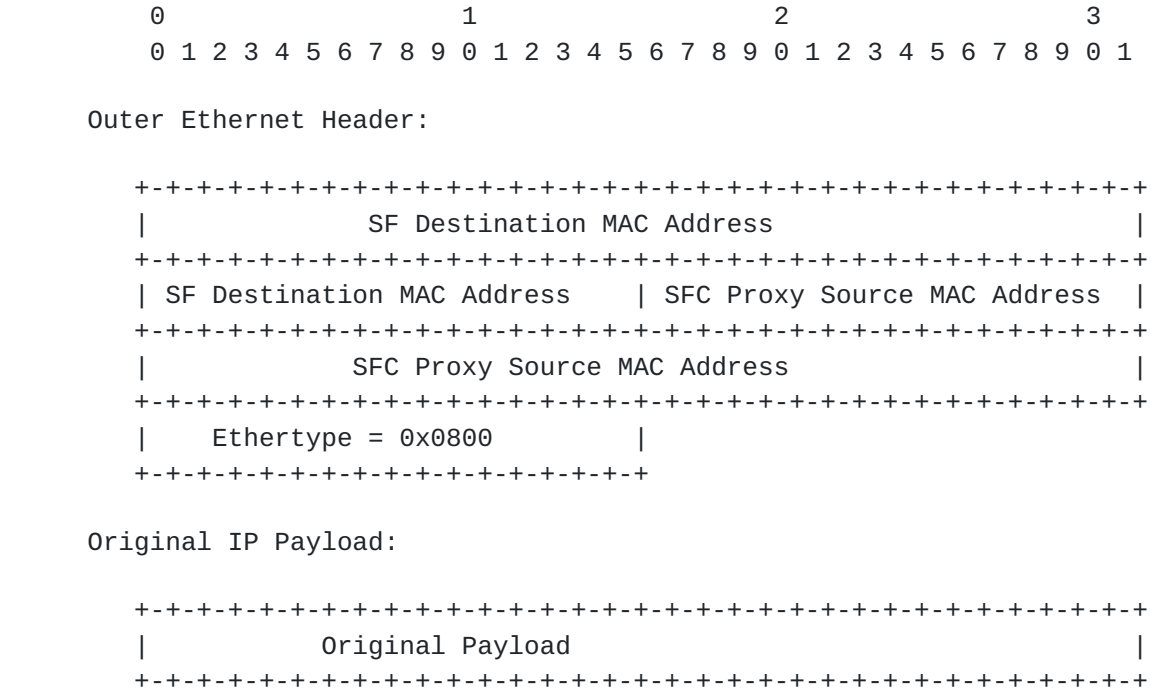
The layer 2 MAC address is used to associate a SFC header between SFC proxy and SF; i.e., each SFC header will be assigned a source MAC address on the SFC proxy. If SFC header can be changed per packet,

then SFC proxy assigns a new source MAC address for each packet it received, otherwise, it assigns a new MAC address for each unique SFC header that must be applied to returning packets. (It is not necessary to have a unique MAC address for each flow received.)

When SFC proxy received 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.

Open issue: usually the MAC address table size in a switch is no more than 16K. When there is a requirement that per packet metadata needs to be restored to each packet after the packet returns from the SF instance, it may require more MAC addresses than the MAC table size in the switch. This may overflow the MAC table, thus the packet cannot route back to the SFC proxy correctly.

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. A variation that is symmetric assigns a unique source/destination pair for each unique SFC header.



3.1.2. VLAN

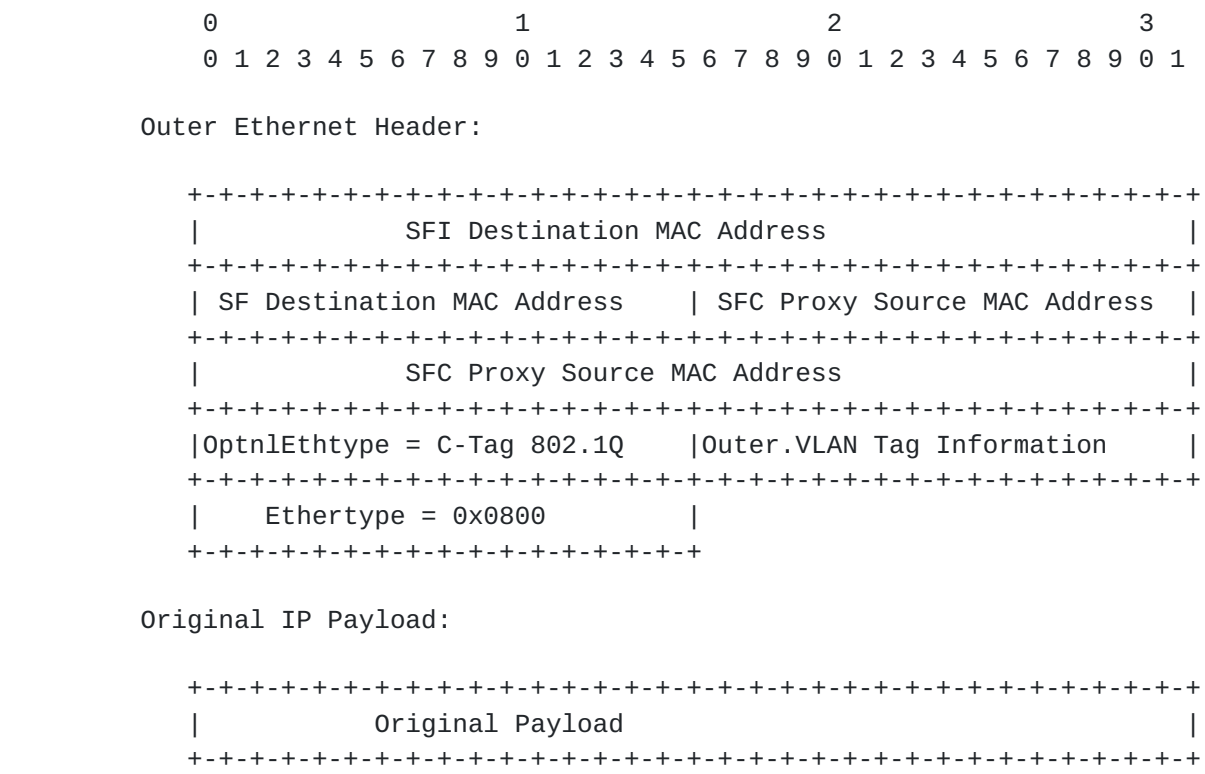
If the network between the SFC proxy and SF is a layer 2 network, and in the case that an SF needs to look into the MAC address of the packet, then VLAN can be used for the mapping between them. The SFC proxy removes the SFC header and sends the packet to the SF, with encapsulating a certain VLAN ID. It is a new encapsulation, supposing that the legacy App can be configured to accept VLAN-tagged packets and to send them back on the same VLAN. It is assumed that the receiving service function host/VM can support multiple VLANs. The SFC proxy locally maintains the mapping between VLAN ID/direction and the SFC header.

When it gets 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.

If metadata is not used, the number of VLAN tags required is exactly the number of SFC paths that pass through the SF, and it can be known at configuration time how many are required.

[I-D.dolson-sfc-vlan] describes an approach for service function chaining by using the input interface and VLAN number to select the next output interface and new VLAN number. SF devices that work with the dolson-sfc-vlan scheme will work with the VLAN scheme described here.

One open issue with VLAN tag is that if the use case requires per packet metadata, then the address space of VLAN digits cannot be enough.

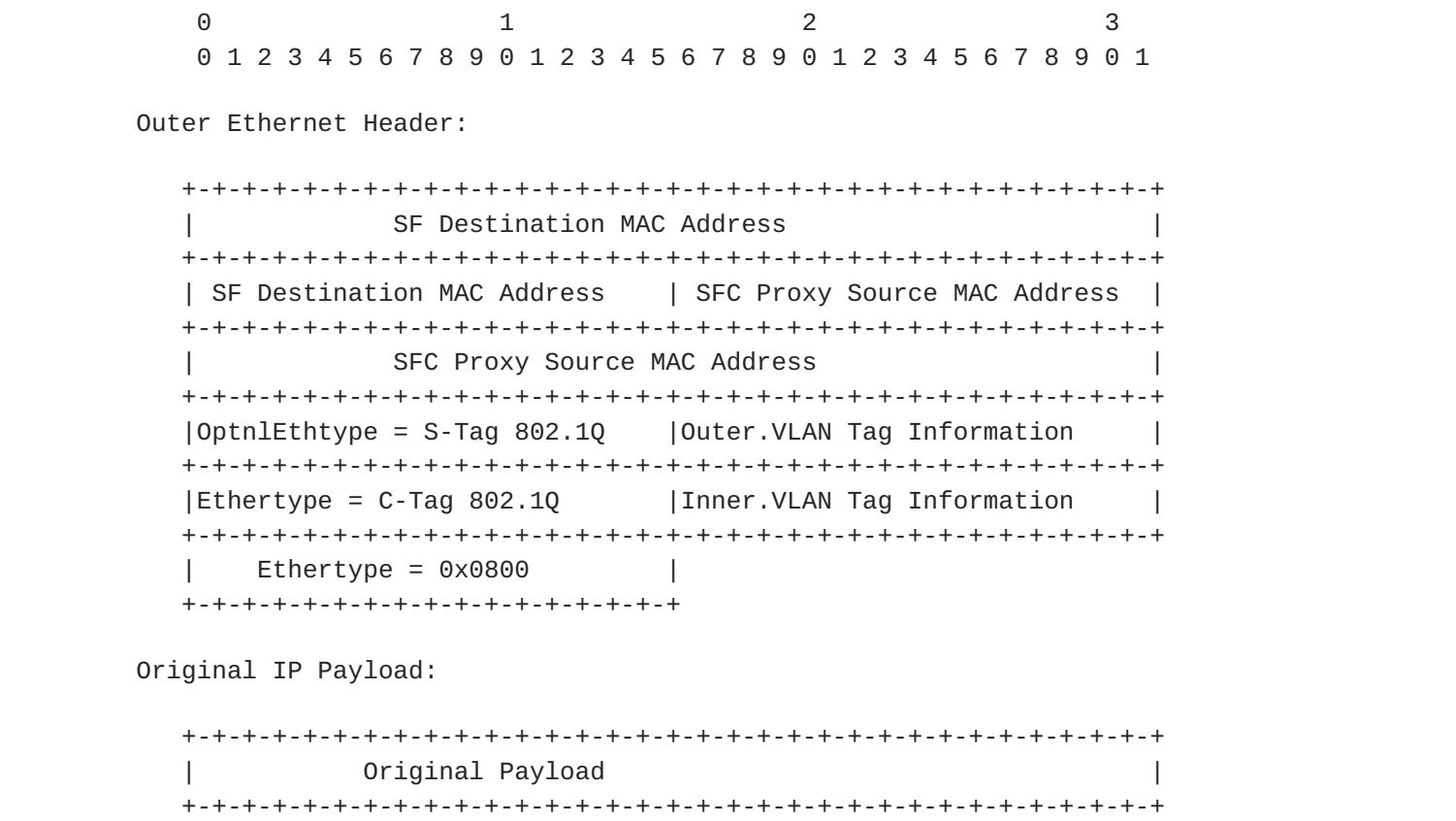


3.1.3. QinQ

If the network between the SFC proxy and SF is already a VLAN network, and the SF needs to look into the MAC address, then QinQ is used for the communication between SFC proxy and SF. The SFC proxy removes the SFC header and sends the original traffic to legacy SF with a certain outer VLAN ID. It locally maintains the mapping between outer VLAN ID and the SFC header.

If the network between SFC proxy and SF is not a VLAN network, then QinQ can be used for either per flow mapping or per packet mapping, using two layer VLAN fields. Because of the increase in address space, QinQ can be used in two-layer VLAN: outer VLAN-id per flow, and inner VLAN-id per packet. If the network between SFC proxy and SF is a VLAN network, then QinQ can only be used for per flow mapping, using one VLAN field.

It is assumed that the receiving service function host/VM can support multiple service VLAN IDs with multiple inner VLAN IDs.

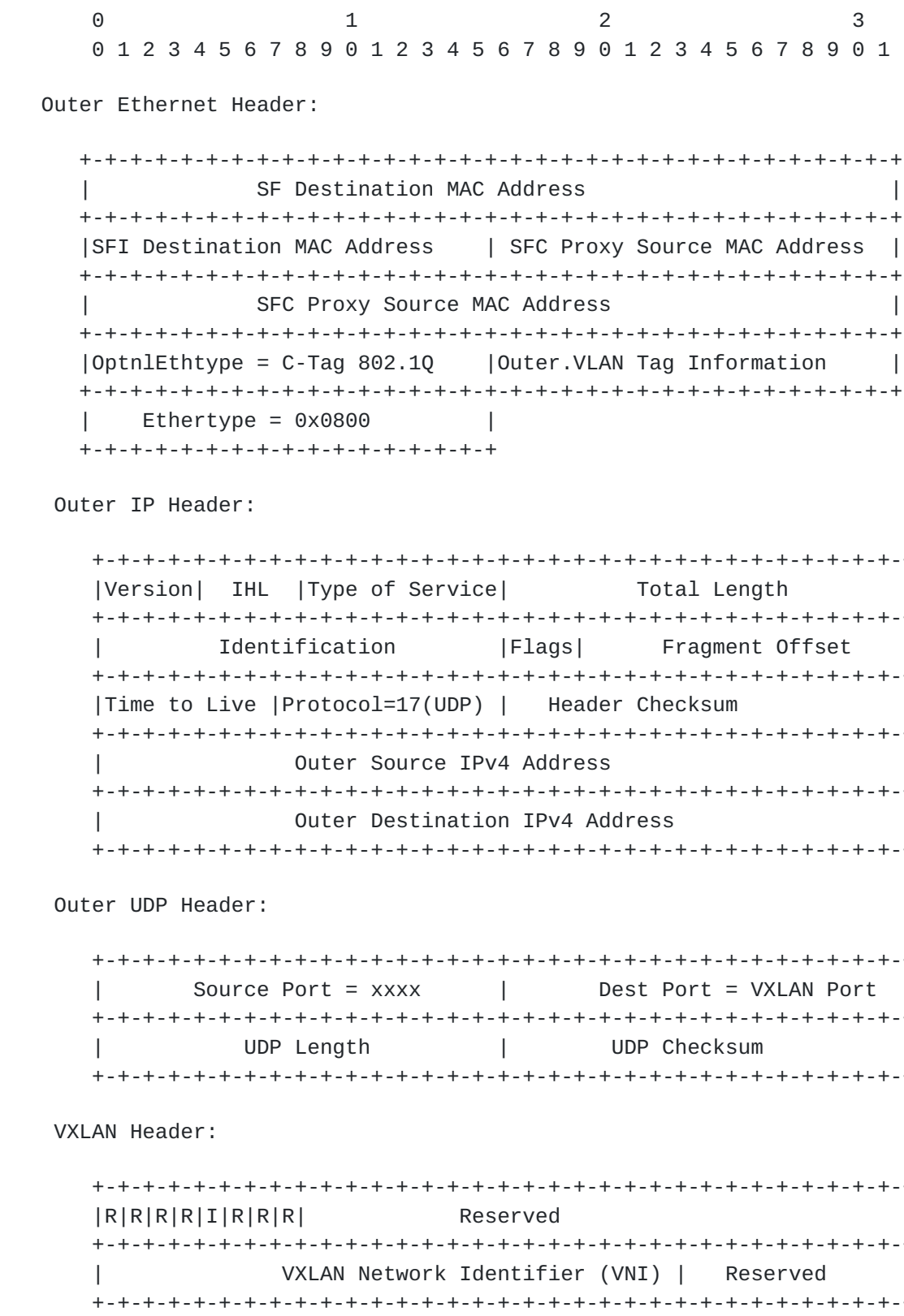


3.1.4. VXLAN

If the SFC proxy and SF are already deployed in a QinQ network, then VXLAN [RFC7348] can be used for the mapping, i.e. VNI can be used for the mapping between them. This tunneling technology is only used when the original packet type is at layer 2 and the SF has to look into the layer 2 MAC header.

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 VLANs is larger.


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
Outer Ethernet Header:
SF Destination MAC Address
SFI Destination MAC Address
SFC Proxy Source MAC Address
SFC Proxy Source MAC Address
OptnlEthtype = C-Tag 802.1Q
Outer.VLAN Tag Information
Ethertype = 0x0800
Outer IP Header:
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol=17(UDP)
Header Checksum
Outer Source IPv4 Address
Outer Destination IPv4 Address
Outer UDP Header:
Source Port = xxxx
Dest Port = VXLAN Port
UDP Length
UDP Checksum
VXLAN Header:
R R R R I R R R
Reserved
VXLAN Network Identifier (VNI)
Reserved

3.1.5. 5-tuple

The 5-tuple of an SFC packet can be used as a key to associate an SFC header in the SFC proxy 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 organizations, as such unicity 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.

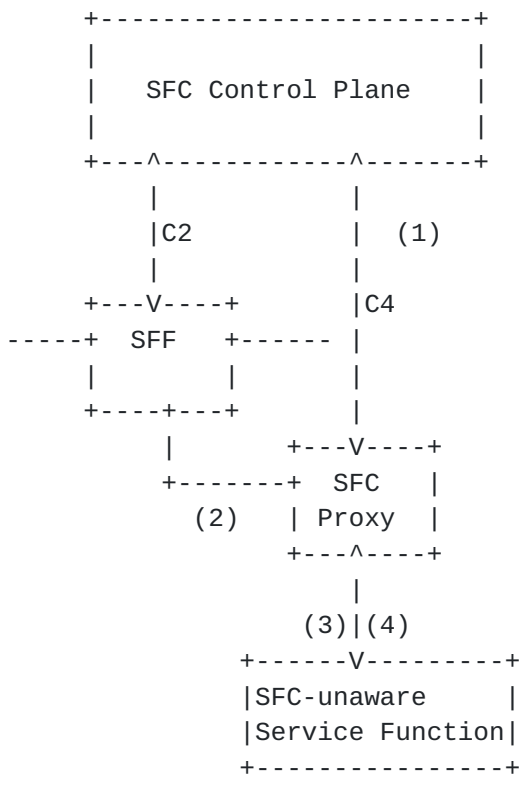
Another similar use case could be that a client and a server use http 80 port for transporting different types of data, and if each type has its specific SFC header with metadata, then 5-tuple does not work either.

This method cannot support per-packet metadata.

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



4. Operation Considerations

The following table shows all the methods and the conditions to use.

Table 1: Operation Consideration

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

4.1. Metadata Consideration

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 look into the layer 2 header, it may cause security threats. It also provides diagrams of the main entities that the information model is comprised of.

6. Acknowledgement

The authors would like to thank Ron Parker and Joel Halpern for their 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.dolson-sfc-vlan]
Dolson, D., "VLAN Service Function Chaining", draft-dolson-sfc-vlan-00 (work in progress), February 2014.

[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-03 (work in progress), January 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