SFC working group
Internet-Draft
Intended status: Informational
Expires: April 2, 2015

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

September 29, 2014

SFC Header Mapping for Legacy SF
draft-song-sfc-legacy-sf-mapping-03

Abstract

A Service Function Chain (SFC) 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 legacy service function can participate in the service function chain, but they are not aware of the SFC header, nor interpret it. This document provides a mechanism between an SFC proxy and an SFC-unaware Service Function (i.e. legacy SF), to identify the SFC header associated with a packet that is returned from a legacy SF, without SFC header being explicitly carried in the wired protocol between SFC Proxy and legacy SF.

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

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 2, 2015.

Table of Contents

1. Introduction

A Service Function Chain (SFC) [I-D.ietf-sfc-architecture] 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 are kept as legacy, and they do not have to be aware of the SFC header, nor interpret it. 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 an SFC proxy and a legacy SF.

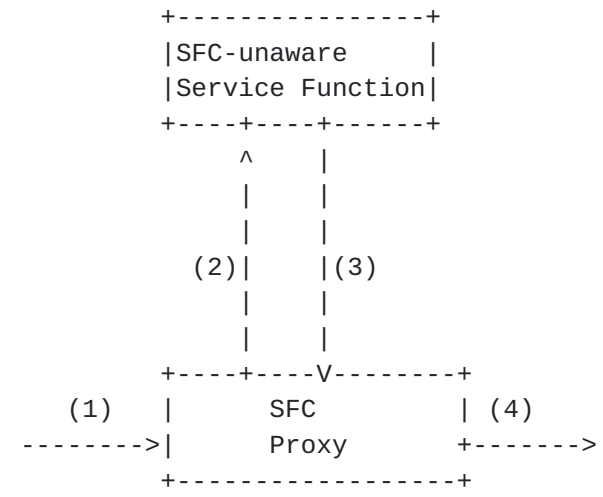


Figure 1: Procedure of a packet processed by a legacy SF

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. One advantage is that the existing service functions don't need to be upgraded to support SFC. Otherwise it may be a hindrance for the widely adoption of SFC.

Assuming that for some legacy SFs, the packet header is transparent to them, i.e., this kind of SFs will not modify the layer 2 or layer 3 packet headers. If the payload in the SFC encapsulation is layer 3 traffic, it will be kept as it is, and a new layer 2 header will be added before sending to the SF. However if the payload in the SFC encapsulation is layer 2 traffic, the SFC proxy may modify the original source MAC address and use the new source MAC address for mapping to the stored SFC header. This will not impact the SF processing. The SF will send the traffic back after processing. For the current stage, we leave the legacy SFs which modify the original packet headers as an open issue for further study.

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 original packet to the SF. After SF processing the packet, the traffic will be sent back to the SFC proxy. The SFC proxy retrieves the pre- stored SFC header accordingly, and encapsulates the packet with the SFC header, and then sends the packet to next-hop service function. The key problem here is how to map the packet to its original SFC header.

If the SFC header is not changed per flow at a certain point, e.g., a specific SFC proxy (i.e. each flow has a specific SFC header in a SFC proxy, but in another SFC proxy, the SFC header is different), then the SFC proxy needs to find the original SFC header per flow. If the SFC header is changed per packet for a specific flow at a certain point, then the SFC proxy needs to find the original SFC header per packet. The second case may happen if different packets in a flow carry different metadata (e.g. the metadata can be injected to the packet by a DPI appliance). It's also the reason why five-tuple cannot be used for the mapping to retrieve the original SFC header.

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.

[Open Issue: Should the authors of 'SFC header' consider the issue of having a flag on metadata specifying if it is per flow, per packet? Only the sender of the metadata knows.]

An expiration time can be used for each mapping entry in the SFC proxy. If the SFC header in that entry has not been retrieved after the expiration time, the entry will be deleted from the entry table.

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 a SFC encapsulation packet or a packet constructed based on the original payload.

3. Mechanisms

The mechanisms used in this document require that each forwarding entity and its connected service functions in a 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

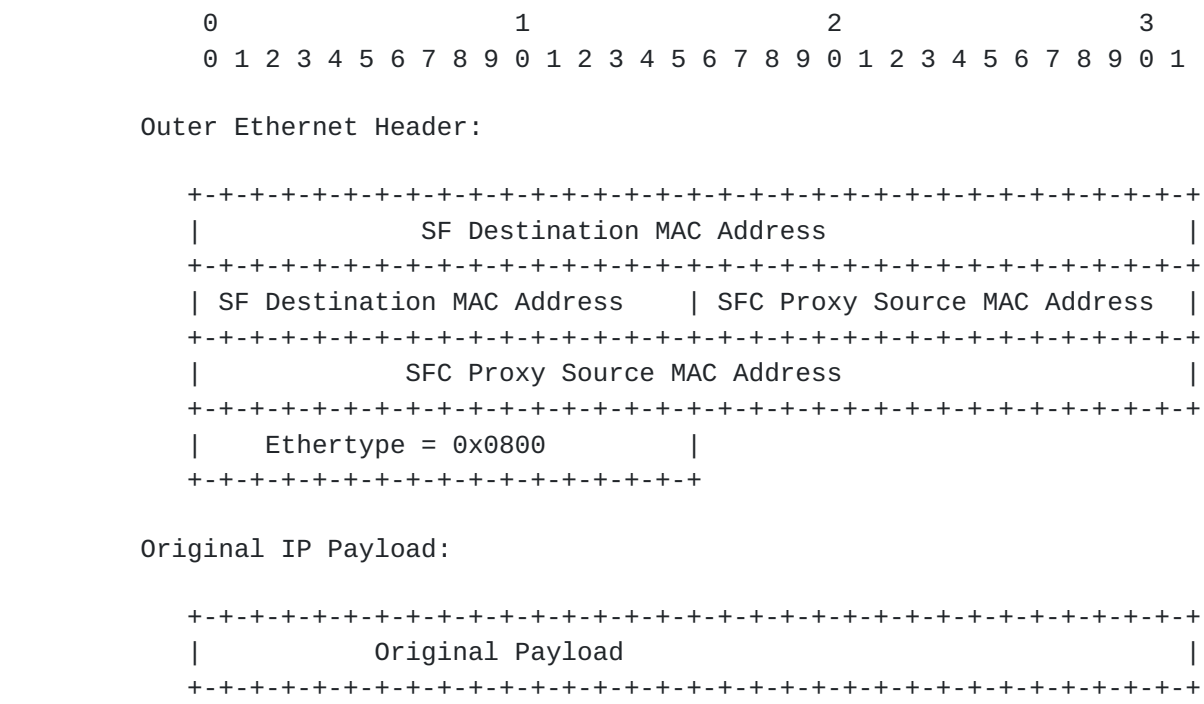
If the service function is transparent to packet headers, the following methods can be used for SFC header mapping.

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 flow it received.

When SFC proxy received the returned packet from the SF, it retrieves the packet's original SFC header by using the 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.



3.1.2. VLAN

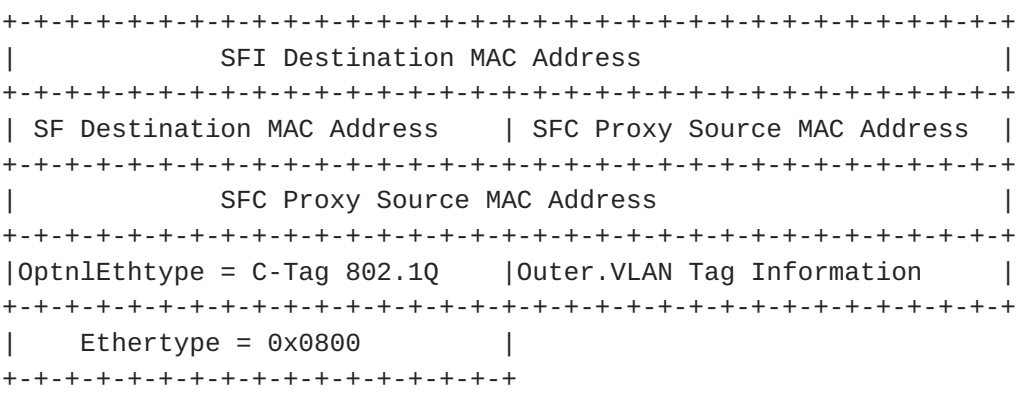
If the network between the SFC proxy and SF is a layer 2 network, and in the case that an SF needs to look into the MAC address of the packet, then VLAN can be used for the mapping between them. The SFC proxy removes the SFC header and sends the packet to the SF, with encapsulating a certain VLAN ID. It is a new encapsulation, this supposes that the legacy App can be configured to accept encapsulated packets and to send them back on the same VLAN. It is assumed that the receiving service function host/VM can support multiple VLANs. It locally maintains the mapping between VLAN ID and the SFC header. When it gets the returned packet from the SF, it removes the VLAN part from the packet and retrieves the corresponding SFC header according to the VLAN ID, and then encapsulates SFC header into that packet before sending to the next service function.

The VLAN ID may be used for mapping per flow, i.e. each flow will be assigned a new VLAN ID. If SFC header could be changed per packet, the length of VLAN ID is not enough for mapping.

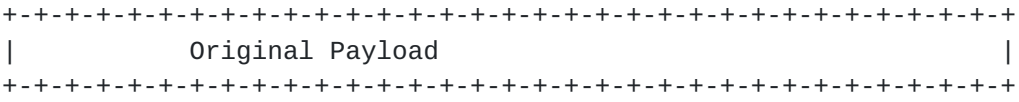
[Open issue: [I-D.dolson-sfc-vlan] describes an approach for service function chaining by using the input interface and VLAN number to select the next output interface and new VLAN number. However the mechanism discussed in this section is not necessary to use a pair of VLAN IDs to identify the uplink and downlink streams respectively. Only in the condition that the path IDs for the symmetric flows are same, then it is necessary to assign different VLAN IDs for the

uplink and downlink streams respectively in order to associate the VLAN ID with a specific service chain header.]

Outer Ethernet Header:



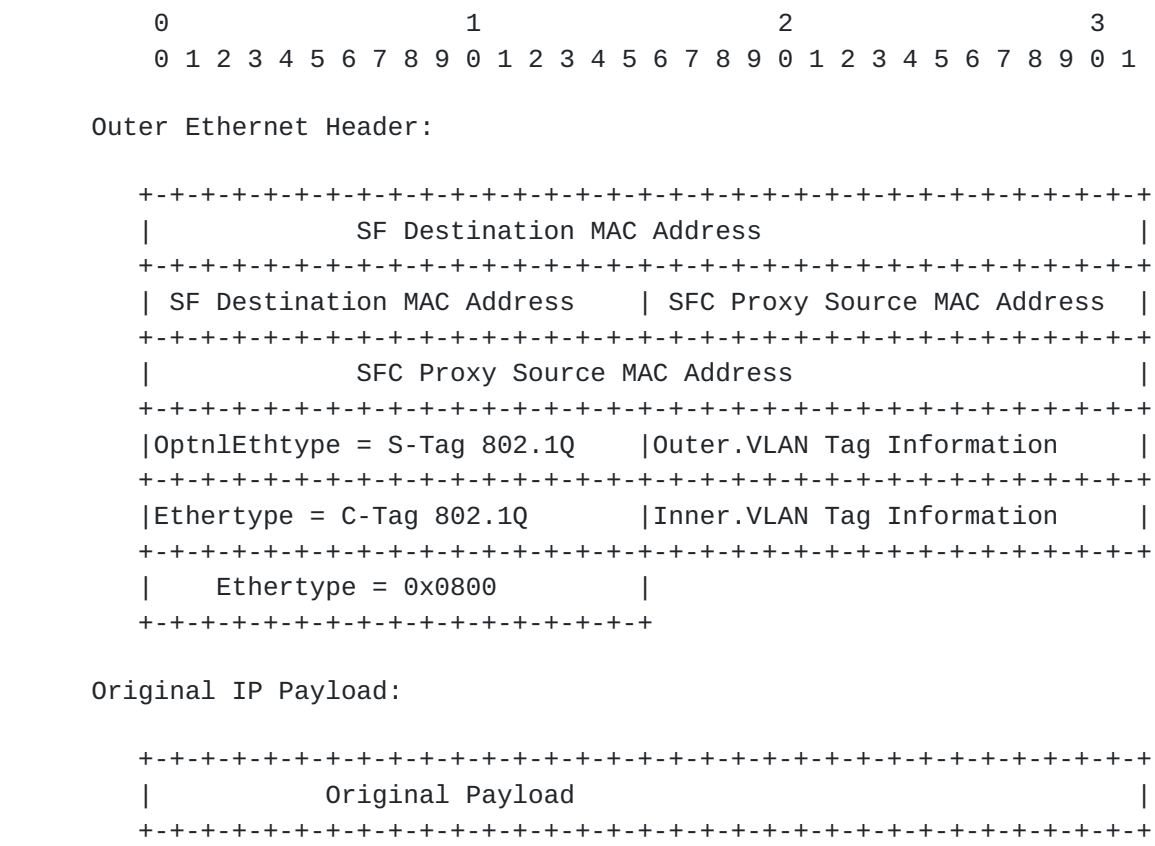
Original IP Payload:



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 remove the SFC header and send the original traffic to SF with a certain outer VLAN ID. It locally maintains the mapping between outer VLAN ID and the SFC header.

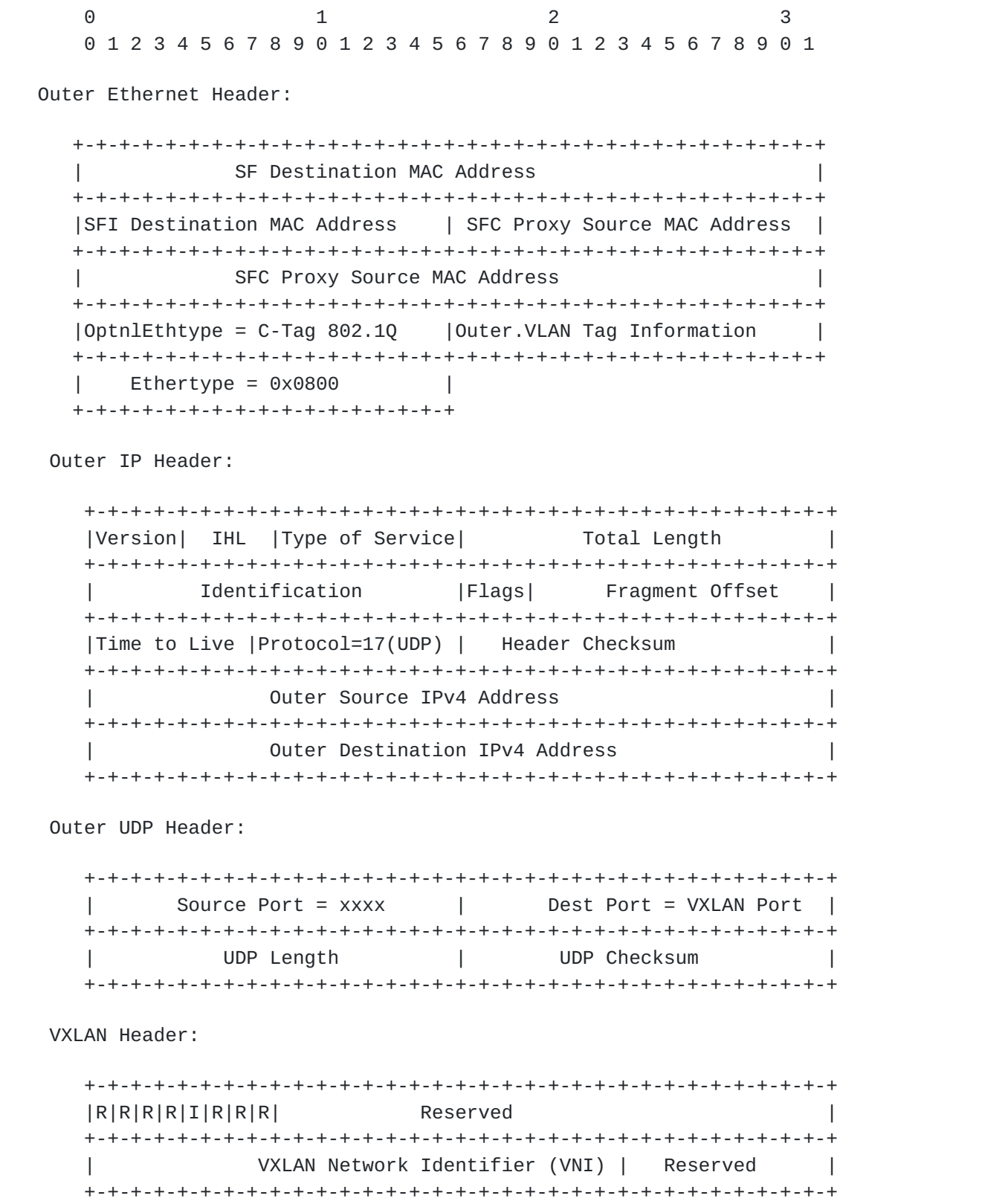
If the network between SFC proxy and SF is not a VLAN network, then QinQ can be used for either per flow mapping or per packet mapping, using two layer VLAN fields. Because of the increase in address space, QinQ can be used in two-layer VLAN: outer VLAN-id per flow, and inner VLAN-id per packet. If the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



3.1.4. VXLAN

If the SFC proxy and SF are already deployed in a QinQ network, then VXLAN [I-D.mahalingam-dutt-dcops-vxlan] 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.


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

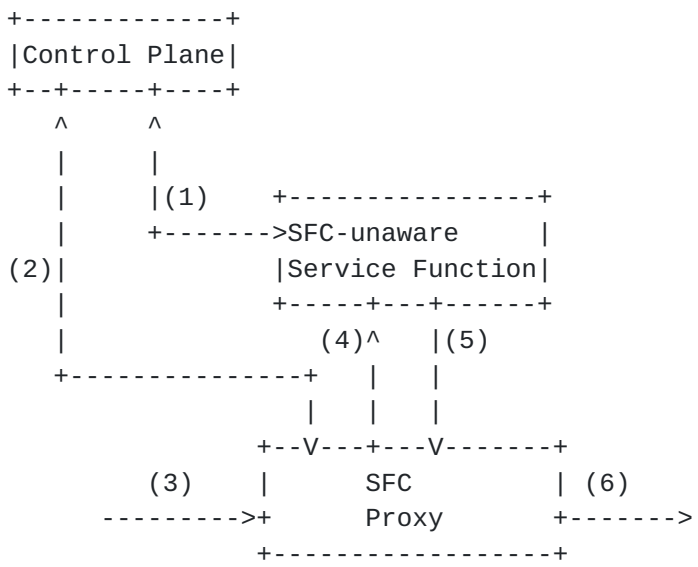
3.1.5. 5-tuple

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.

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 use case below is just one 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 (step 1), and then the control plane can notify the SFC proxy the mapping information (step 2). 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 5), the SFC proxy retrieves the SFC header from the mapping table by using the processed header as a key.



4. Operation Consideration

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 path ID	L2 header won't be modified by the SF.
	VLAN	(VLAN ID, SFC header) e.g. assign a VLAN ID per path ID	L2 header won't be modified by the SF.
	QinQ	(Outer VLAN ID, SFC header) e.g. assign an outer VLAN ID per path ID	The SF is required to support QinQ. L2 header won't be modified by the SF.
	VXLAN	(VNI, SFC header) e.g. assign a VNI per path ID	The SF is required to support VXLAN.
	5-tuple	(5-tuple, SFC header) The SFC proxy maintains the mapping table for 5-tuple and the SFC header.	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.

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 for his comments.

7. References

7.1. Normative References

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

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-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-02 (work in progress), September 2014.

[I-D.mahalingam-dutt-dcops-vxlan]
Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", draft-mahalingam-dutt-dcops-vxlan-09 (work in progress), April 2014.

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