

INTERNET-DRAFT  
Intended status: Proposed Standard

Donald Eastlake  
Huawei  
Bob Briscoe  
Independent  
Andrew Malis  
Huawei  
July 2, 2018

Expires: January 1, 2019

Explicit Congestion Notification (ECN) and Congestion Feedback  
Using the Network Service Header (NSH)  
<draft-eastlake-sfc-nsh-ecn-support-01.txt>

Abstract

Explicit congestion notification (ECN) allows a forwarding element to notify downstream devices of the onset of congestion without having to drop packets. Coupled with a means to expose congestion by feeding back information about it to upstream nodes, this can improve network efficiency through better congestion control, frequently without packet drops. This document specifies ECN and congestion feedback support through use of the Network Service Header (NSH, RFC 8300) and IP Flow Information Export (IPFIX, draft-ietf-tsvwg-tunnel-congestion-feedback).

Status of This Memo

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

Distribution of this document is unlimited. Comments should be sent to the SFC Working Group mailing list <sfc@ietf.org> or to the authors.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>. The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

## Table of Contents

|                                                |    |
|------------------------------------------------|----|
| 1. Introduction.....                           | 3  |
| 1.1 NSH Background.....                        | 3  |
| 1.2 ECN Background.....                        | 5  |
| 1.3 Tunnel Congestion Feedback Background..... | 5  |
| 1.4 Conventions Used in This Document.....     | 6  |
| 2. The NSH ECN Field.....                      | 8  |
| 3. ECN Support in the NSH.....                 | 10 |
| 3.1 At The Ingress.....                        | 11 |
| 3.2 At Transit Nodes.....                      | 11 |
| 3.2.1 At NSH Transit Nodes.....                | 12 |
| 3.2.2 At an SF/Proxy.....                      | 12 |
| 3.2.3 At Other Forwarding Nodes.....           | 13 |
| 3.3 At Exit/Egress.....                        | 13 |
| 3.4 Conservation of Packets.....               | 14 |
| 4. Tunnel Congestion Feedback Support.....     | 15 |
| 5. IANA Considerations.....                    | 16 |
| 6. Security Considerations.....                | 17 |
| 7. Acknowledgements.....                       | 17 |
| Normative References.....                      | 18 |
| Informative References.....                    | 19 |
| Authors' Addresses.....                        | 20 |

## 1. Introduction

Explicit congestion notification (ECN [RFC3168]) allows a forwarding element to notify downstream devices of the onset of congestion without having to drop packets. Coupled with a means to expose congestion by feeding back information about it to upstream nodes, this can improve network efficiency through better congestion control, frequently without packet drops. This document specifies ECN and congestion feedback support through use of the Network Service Header (NSH [RFC8300]) and IP Flow Information Export (IPFIX [TunnelCongFeedback]).

This section provides background information on NSH, ECN, congestion feedback, and terminology used in this document.

### 1.1 NSH Background

The Service Function Chaining (SFC [RFC7665]) architecture calls for the encapsulation of traffic within a service function chaining domain with a Network Service Header (NSH [RFC8300]) added by the "Classifier" (ingress node) on entry to the domain and the NSH being removed on exit from the domain at the egress node. The NSH is used to control the path of a packet in an SFC domain. The NSH is a natural way, in a domain where traffic is NSH encapsulated, to both

- (1) note congestion, avoiding possible confusion due, for example, to changes in the outer transport header in different parts of the domain, and,
- (2) direct congestion information feedback to the domain ingress so that it can take action when appropriate to alleviate congestion.

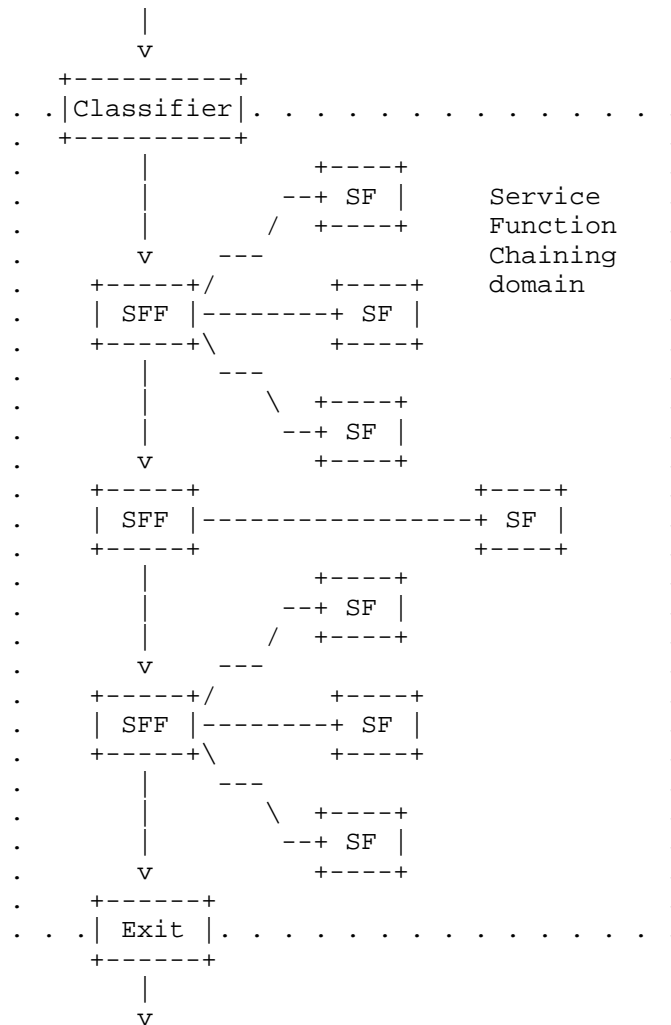


Figure 1. Example SFC Path Forwarding Nodes

Figure 1 shows an SFC domain for the purpose of illustrating the use of NSH. Traffic passes through a sequence of Service Function Forwarders (SFFs) each of which sends the traffic to one or more Service Functions (SFs). Each SF performs some operation on the traffic, for example firewall or Network Address Translation (NAT), and then returns it to the SFF from which it was received.

Logically, during the transit of each SFF, the outer transport header that got the packet to the SFF is stripped, the SFF decides on the next forwarding step, either adding a transport header or, if the SFF is the exit/egress, removing the NSH header. The transport headers

added may be different in different regions of the SFC domain. For example, IP could be used for some SFF-to-SFF communication and MPLS used for other such communication.

## 1.2 ECN Background

Explicit congestion notification (ECN [RFC3168]) allows a forwarding element (such as a router or an Service Function Forwarder (SFF) or Service Function (SF)) to notify downstream devices of the onset of congestion without having to drop packets. This can be used as an element in active queue management (AQM) [RFC7567] to improve network efficiency through better traffic control without packet drops. The forwarding element can explicitly mark some packets in an ECN field instead of dropping the packet. For example, a two-bit field is available for ECN marking in IP headers [RFC3168].

## 1.3 Tunnel Congestion Feedback Background

Tunnel Congestion Feedback [TunnelCongFeedback] is a building block for various congestion mitigation methods that supports feedback of congestion information from an egress node to an ingress node. Examples of actions that can be taken by an ingress node when it has knowledge of downstream congestion include those listed below. Details of implementing these traffic control methods, beyond those given here, are outside the scope of this document.

- (1) Traffic throttling (policing), where the downstream traffic flowing out of the ingress node is limited to reduce or eliminate congestion.
- (2) Upstream congestion feedback, where the ingress node sends messages upstream to or towards the ultimate traffic source, a function that can throttle traffic generation/transmission.
- (3) Traffic re-direction, where the ingress node configures the NSH so that some future traffic avoids congested paths.

NOTE: With this method 3 great care must be taken to avoid (a) significant re-ordering of traffic in flows that it is desirable to keep in order and (b) oscillation/instability in traffic paths due to alternate congestion of previously idle paths and the idling of previously congested paths. For example, it is preferable to classify traffic into flows or a sufficiently coarse granularity that the flows are long lived and use a stable path per flow sending only newly appearing flows on apparently uncongested paths.

Figure 2 shows an example path from an origin sender to a final receiver passing through an example chain of service functions between the ingress and egress of an SFC domain. The path is also likely to pass through other network nodes outside the SFC domain (not shown). The figure shows typical congestion feedback that would be expected from the final receiver to the origin sender, which controls the load the origin sender applies to all elements on the path. The figure also shows the congestion feedback from the egress to the ingress of the SFC domain that is described in this document, to control or balance load within the SFC domain.

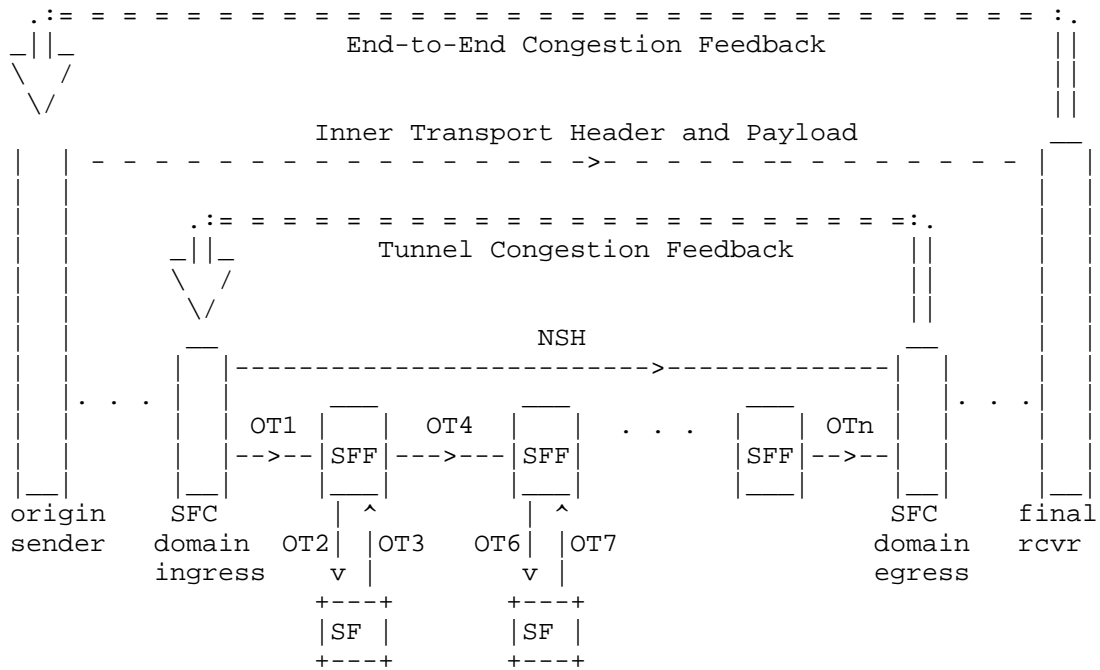


Figure 2: Congestion Feedback across an SFC Domain

SFC Domain congestion feedback in Figure 2 is shown within the context of an end-to-end congestion feedback loop. Also shown is the encapsulated layering of NSH headers within a series of outer transport headers (OT1, OT2, ... OTn).

#### 1.4 Conventions Used in This Document

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Acronyms:

AQM - Active Queue Management [RFC7567]

CE - Congestion Experienced [RFC3168]

downstream - The direction from ingress to egress

ECN - Explicit Congestion Notification [RFC3168]

ECT - ECN Capable Transport [RFC3168]

IPFIX - IP Flow Information Export [RFC7011]

Not-ECT - Not ECN-Capable Transport [RFC3168]

NSH - Network Service Header [RFC8300]

SF - Service Function [RFC7665]

SFC - Service Function Chaining [RFC7665]

SFF - Service Function Forwarder [RFC7665] - A type of node that forwards based on the NSH.

TLV - Type Length Value

upstream - The direction from egress to ingress

## 2. The NSH ECN Field

The NSH header is used to encapsulate and control the subsequent path of traffic (see Section 2 of [RFC8300]). The NSH also provides for metadata inclusion, as shown in Figure 3.

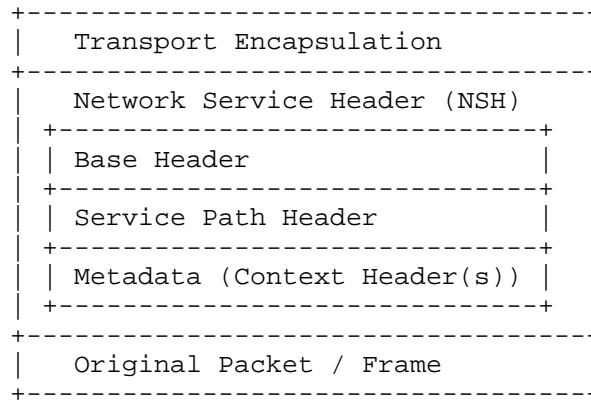


Figure 3. Data Encapsulation with the NSH

Two currently unused bits (indicated by "U") in the NSH Base Header (Section 2.2 of [RFC8300]) are allocated for ECN as shown in Figure 4.

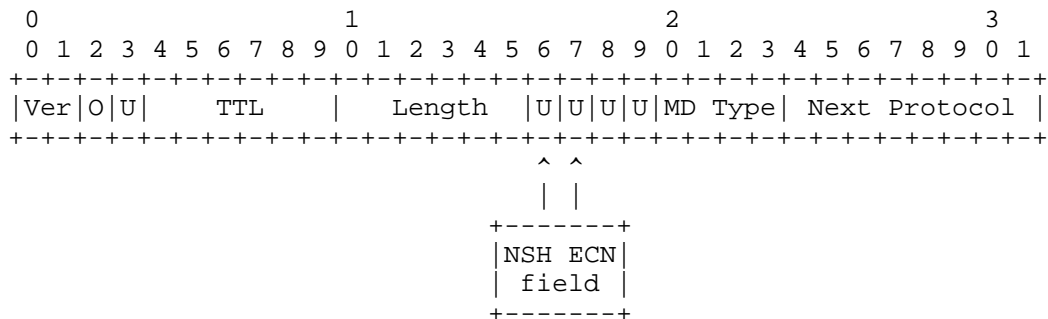


Figure 4: NSH Base Header

Note to RFC Editor: The above figure should be adjusted based on the bits assigned by IANA (see Section 5) and this note deleted.

Table 1 shows the meaning of the code points in the NSH ECN field. These have the same meaning as the ECN field code points in the IPv4 or IPv6 header as defined in [RFC3168].



| Binary | Name    | Meaning                   |
|--------|---------|---------------------------|
| -----  | -----   | -----                     |
| 00     | Not-ECT | Not ECN-Capable Transport |
| 01     | ECT(1)  | ECN-Capable Transport     |
| 10     | ECT(0)  | ECN-Capable Transport     |
| 11     | CE      | Congestion Experienced    |

Table 1. ECN Field Code Points

### 3. ECN Support in the NSH

This section describes the required behavior to support ECN using the NSH. There are two aspects to ECN support:

1. ECN propagation during encapsulation or decapsulation
2. ECN marking during congestion at bottlenecks.

While this section covers all combinations of ECN-aware and not ECN-aware, it is expected that in most cases the NSH domain will be uniform so that, if this document is applicable, all SFFs will support ECN; however, some legacy SFs might not support ECN.

#### ECN Propagation:

The specification of ECN tunneling [RFC6040] explains that an ingress must not propagate ECN support into an encapsulating header unless the egress supports correct onward propagation of the ECN field during decapsulation. We define Compliant ECN Decapsulation here as decapsulation compliant with either [RFC6040] or an earlier compatible equivalent ([RFC4301], or full functionality mode of [RFC3168]).

The procedures in Section 3.2.1 ensure that each ingress of the large number of possible transport links within the SFC domain does not propagate ECN support into the encapsulating outer transport header unless the corresponding egress of that link supports Compliant ECN Decapsulation.

Section 3.3 requires that all the egress nodes of the SFC domain support Compliant ECN Decapsulation in conjunction with tunnel congestion feedback, otherwise the scheme in this document will not work.

#### ECN Marking:

At transit nodes the marking behavior specified in 3.2.1 is recommended and if not implemented at such transit nodes, there may be unmanaged congestion.

Detection of congestion will be most effective if ECN marking is supported by all potential bottlenecks inside the domain in which NSH is being used to route traffic as well as at the ingress and egress. Nodes that do not support ECN marking, or that support AQM but not ECN, will naturally use drop to relieve congestion. The gap in the end-to-end packet sequence will be detected as congestion by the final receiving endpoint, but not by the NSH egress (see Figure 2).

### 3.1 At The Ingress

When the ingress/Classifier encapsulates an incoming IP packet with an NSH, it MUST set the NSH ECN field using the "Normal mode" specified in [RFC6040] (i.e., copied from the incoming IP header).

Then, if the resulting NSH ECN field is Not-ECT, the ingress SHOULD set it to ECT(0), in order to indicate that the NSH encapsulation is an ECN-Capable Transport. It MAY instead be set to ECT(1) if the NSH domain supports the experimental L4S capability [RFC8311], [ecnL4S].

Packets arriving at the ingress might not use IP. If the protocol of arriving packets supports an ECN field similar to IP, the procedures for IP packets can be used. If arriving packets do not support an ECN field similar to IP, they MUST be treated as if they are Not-ECT IP packets.

Then, as the NSH encapsulated packet is further encapsulated with a transport header, if ECN marking is available for that transport (as it is for IP [RFC3168] and MPLS [RFC5129]), the ECN field of the transport header MUST be set using the "Normal mode" specified in [RFC6040] (i.e., copied from the NSH ECN field).

A summary of these normative steps is given in Table 2.

| Incoming Header<br>(also equal to<br>departing Inner<br>Header) | Departing NSH and Outer Headers |                 |
|-----------------------------------------------------------------|---------------------------------|-----------------|
|                                                                 | Classic ECN<br>Mode             | L4S ECN<br>Mode |
| Not-ECT                                                         | ECT(0)                          | ECT(1)          |
| ECT(0)                                                          | ECT(0)                          | ECT(0)          |
| ECT(1)                                                          | ECT(1)                          | ECT(1)          |
| CE                                                              | CE                              | CE              |

Table 2. Setting of ECN fields by an ingress/Classifier

The requirements in this section apply to all ingress nodes for the domain in which NSH is being used to route traffic.

### 3.2 At Transit Nodes

This section described behavior at nodes that forward based on the NSH such as SFF and other forwarding nodes such as IP routers. Figure 5 shows a packet on the wire between forwarding nodes.

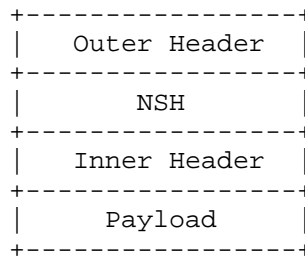


Figure 5. Packet in Transit

### 3.2.1 At NSH Transit Nodes

When a packet is received at an NSH based forwarding node N1, such as an SFF, the outer transport encapsulation is removed and its ECN marking **SHOULD** be combined into the NSH ECN marking as specified in [RFC6040]. If this is not done, any congestion encountered at non-NSH transit nodes between N1 and the next upstream NSH based forwarding node will be lost and not transmitted downstream.

The NSH forwarding node **SHOULD** use a recognized AQM algorithm [RFC7567] to detect congestion. If the NSH ECN field indicates ECT, it will probabilistically set the NSH ECN field to the Congestion Experienced (CE) value or, in cases of extreme congestion, drop the packet.

When the NSH encapsulated packet is further encapsulated for transmission to the next SFF or SF, ECN marking behavior depends on whether or not the node that will decapsulate the outer header supports Compliant ECN Decapsulation (see Section 3). If it does, then the ingress node propagates the NSH ECN field to this outer encapsulation using the "Normal Mode" of ECN encapsulation [RFC6040] (it copies the ECN field). If it does not, then the ingress **MUST** clear ECN in the outer encapsulation to non-ECT (the "Compatibility Mode" of [RFC6040]).

### 3.2.2 At an SF/Proxy

If the SF is NSH and ECN-aware, the processing is essentially the same at the SF as at an SFF as discussed in Section 3.2.1.

If the SF is NSH-aware but not ECN-aware, then the SFF transmitting the packet to the SF will use Compatibility Mode. Congestion encountered in the SFF to SF and SF to SFF paths will be unmanaged.

If the SF is not NSH-aware, then an NSH proxy will be between the SFF and the SF to avoid exposure of the NSH at the SF that does not understand NSHs. This is described in Section 4.6 of [RFC7665]. The SF and proxy together look to the SFF like an NSH-aware SF. The behavior at the proxy and SF in this case is as below:

If such a proxy is not ECN-aware then congestion in the entire path from SFF to proxy to SF back to proxy to SFF will be unmanaged.

If the proxy is ECN-aware the proxy uses an AQM to indicate congestion in the proxy itself in the NSH that it returns to the SFF. The outer header used for the proxy to SF path uses Normal Mode. The outer head used for the proxy return to SFF path uses Normal Mode based copying the NSH ECN field to the outer header. Thus congestion in the proxy will be managed. Congestion in the SF will be managed only if the SF is ECN-aware implementing an AQM.

### 3.2.3 At Other Forwarding Nodes

Other forwarding nodes, that is non-NSH forwarding nodes between NSH forwarding nodes, such as IP routers, might also be potential bottlenecks. If so, they SHOULD implement an AQM algorithm to update the ECN marking in the outer transport header as specified in [RFC3168].

### 3.3 At Exit/Egress

First, any actions are taken based on Congestion Experienced such as forwarding statistics back to the ingress (see Section 4). If the packet being carried inside the NSH is IP, when the NSH is removed the NSH ECN field MUST be combined with IP ECN field as specified in Table 3 that was extracted from [RFC6040]. This requirement applies to all egress nodes for the domain in which NSH is being used to route traffic.

| Arriving<br>Inner<br>Header | Arriving Outer Header |         |         |        |
|-----------------------------|-----------------------|---------|---------|--------|
|                             | Not-ECT               | ECT(0)  | ECT(1)  | CE     |
| Not-ECT                     | Not-ECT               | Not-ECT | Not-ECT | <drop> |
| ECT(0)                      | ECT(0)                | ECT(0)  | ECT(0)  | CE     |
| ECT(1)                      | ECT(1)                | ECT(1)  | ECT(1)  | CE     |
| CE                          | CE                    | CE      | CE      | CE     |

Table 3. Exit ECN Fields Merger

All the egress nodes of the SFC domain MUST support Compliant ECN Decapsulation as specified in this section. If this is not the case, the scheme described in this document will not work, and cannot be used.

### 3.4 Conservation of Packets

The SFC specification permits an SF to absorb packets and to generate new packets as well as to process and forward the packets it receives. Such actions might appear to be packet loss due to congestion or might mask the loss of packets by generating additional packets.

The tunnel congestion feedback approach [TunnelCongFeedback] detects loss by counting payload bytes in at the ingress and counting them out at the egress. This does not work unless nodes conserve the amount of payload bytes. Therefore, it will not be possible to detect loss using this technique if they are not conserved.

Nonetheless, if a bottleneck supports ECN marking, it will be possible to detect the very high level of CE markings that are associated with congestion that is so excessive that it leads to loss. However, it will not be possible for the tunnel congestion feedback approach to detect any congestion, whether slight or severe, if it occurs at a bottleneck that does not support ECN marking.

#### 4. Tunnel Congestion Feedback Support

The collection and storage of congestion information may be useful for later analysis but, unless it can be fed back to a point which can take action to reduce congestion, it will not be useful in real time. Such congestion feedback to the ingress enables it to take actions such as those listed in Section 1.3.

IP Flow Information Export (IPFIX [RFC7011]) provides a standard for communicating traffic flow statistics. As extended by [TunnelCongFeedback], IPFIX can be used to determine the extent of congestion between an ingress and egress.

IPFIX recommends use of SCTP [RFC4960] in partial reliability mode. This mode allows loss of some packets, which is tolerable because IPFIX communicates cumulative statistics. IPFIX over SCTP SHOULD be used directly where there is IP connectivity between the ingress and egress; however, there might be different transport protocols or address spaces used in different regions of an SFC domain that make such direct IP connectivity problematic. The NSH provides the general method of routing of traffic within such domain so the IPFIX over SCTP over IP traffic should be encapsulated in NSH when necessary.

## 5. IANA Considerations

IANA is requested to assign two contiguous bits in the NSH Base Header Bits registry for ECN (bits 16 and 17 suggested) and note this assignment as follows:

| Bit        | Description | Reference       |
|------------|-------------|-----------------|
| -----      | -----       | -----           |
| tbd(16-17) | NSH ECN     | [this document] |



## 6. Security Considerations

For general NSH security considerations, see [RFC8300].

For security considerations concerning tampering with ECN signaling, see [RFC3168]. For security considerations concerning ECN encapsulation, see [RFC6040].

For general IPFIX security considerations, see [RFC7011]. If deployed in an untrusted environment, the signaling traffic between ingress and egress can be protected utilizing the security mechanisms provided by IPFIX (see section 11 in RFC7011).

The solution does not introduce any greater potential to invade privacy than would have been possible without the solution.

## 7. Acknowledgements

The authors wish to thank the following for their comments and suggestion:

Joel Halpern, Xinpeng Wei

## 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>>.
- [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>>.
- [RFC5129] - Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <<https://www.rfc-editor.org/info/rfc5129>>.
- [RFC6040] - Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<http://www.rfc-editor.org/info/rfc6040>>.
- [RFC7011] - Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [RFC7567] - Baker, F., Ed., and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <<http://www.rfc-editor.org/info/rfc7567>>.
- [RFC8174] - Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<http://www.rfc-editor.org/info/rfc8174>>.
- [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>>.
- [TunnelCongFeedback] - Wei, X., Zhu, L., and L. Deng, "Tunnel Congestion Feedback", draft-ietf-tsvwg-tunnel-congestion-feedback, work in progress.

## Informative References

- [RFC4301] - Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4960] - Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<https://www.rfc-editor.org/info/rfc4960>>.
- [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>>.
- [RFC8311] - Black, D., "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation", RFC 8311, DOI 10.17487/RFC8311, January 2018, <<https://www.rfc-editor.org/info/rfc8311>>.
- [ecnL4S] - De Schepper, K., and B. Briscoe, "Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay (L4S)", draft-ietf-tsvwg-ecn-l4s-id, work in progress.

Authors' Addresses

Donald E. Eastlake, 3rd  
Huawei Technologies  
1424 Pro Shop Court  
Davenport, FL 33896 USA

Tel: +1-508-333-2270  
Email: d3e3e3@gmail.com

Bob Briscoe  
Independent  
UK

Email: ietf@bobbriscoe.net  
URI: <http://bobbriscoe.net/>

Andrew G. Malis  
Huawei Technologies

Email: agmalis@gmail.com

## Copyright and IPR Provisions

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License. The definitive version of an IETF Document is that published by, or under the auspices of, the IETF. Versions of IETF Documents that are published by third parties, including those that are translated into other languages, should not be considered to be definitive versions of IETF Documents. The definitive version of these Legal Provisions is that published by, or under the auspices of, the IETF. Versions of these Legal Provisions that are published by third parties, including those that are translated into other languages, should not be considered to be definitive versions of these Legal Provisions. For the avoidance of doubt, each Contributor to the IETF Standards Process licenses each Contribution that he or she makes as part of the IETF Standards Process to the IETF Trust pursuant to the provisions of RFC 5378. No language to the contrary, or terms, conditions or rights that differ from or are inconsistent with the rights and licenses granted under RFC 5378, shall have any effect and shall be null and void, whether published or posted by such Contributor, or included with or in such Contribution.



INTERNET-DRAFT  
Intended status: Proposed Standard

Donald Eastlake  
Huawei  
Bob Briscoe  
Independent  
Andrew Malis  
Huawei  
February 5, 2019

Expires: August 4, 2019

Explicit Congestion Notification (ECN) and Congestion Feedback  
Using the Network Service Header (NSH)  
<draft-eastlake-sfc-nsh-ecn-support-03.txt>

Abstract

Explicit congestion notification (ECN) allows a forwarding element to notify downstream devices of the onset of congestion without having to drop packets. Coupled with a means to feed back information about congestion to upstream nodes, this can improve network efficiency through better congestion control, frequently without packet drops. This document specifies ECN and congestion feedback support within a Service Function Chaining (SFC) domain through use of the Network Service Header (NSH, RFC 8300) and IP Flow Information Export (IPFIX, draft-ietf-tsvwg-tunnel-congestion-feedback).

Status of This Memo

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

Distribution of this document is unlimited. Comments should be sent to the SFC Working Group mailing list <sfc@ietf.org> or to the authors.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>. The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

## Table of Contents

|                                                |    |
|------------------------------------------------|----|
| 1. Introduction.....                           | 3  |
| 1.1 NSH Background.....                        | 3  |
| 1.2 ECN Background.....                        | 5  |
| 1.3 Tunnel Congestion Feedback Background..... | 5  |
| 1.4 Conventions Used in This Document.....     | 7  |
| 2. The NSH ECN Field.....                      | 8  |
| 3. ECN Support in the NSH.....                 | 10 |
| 3.1 At The Ingress.....                        | 11 |
| 3.2 At Transit Nodes.....                      | 12 |
| 3.2.1 At NSH Transit Nodes.....                | 12 |
| 3.2.2 At an SF/Proxy.....                      | 13 |
| 3.2.3 At Other Forwarding Nodes.....           | 13 |
| 3.3 At Exit/Egress.....                        | 13 |
| 3.4 Conservation of Packets.....               | 14 |
| 4. Tunnel Congestion Feedback Support.....     | 15 |
| 5. IANA Considerations.....                    | 16 |
| 6. Security Considerations.....                | 17 |
| 7. Acknowledgements.....                       | 17 |
| Normative References.....                      | 18 |
| Informative References.....                    | 19 |
| Authors' Addresses.....                        | 20 |



## 1. Introduction

Explicit congestion notification (ECN [RFC3168]) allows a forwarding element to notify downstream devices of the onset of congestion without having to drop packets. Coupled with a means to feed back information about congestion to upstream nodes, this can improve network efficiency through better congestion control, frequently without packet drops. This document specifies ECN and congestion feedback support within a Service Function Chaining (SFC [RFC7665]) domain through use of the Network Service Header (NSH [RFC8300]) and IP Flow Information Export (IPFIX [TunnelCongFeedback]).

It requires that all ingress and egress nodes of the SFC domain implement ECN. While congestion management will be the most effective if all interior nodes of the SFC domain implement ECN, some benefit is obtained even if some interior nodes do not implement ECN. In particular, congestion at any bottleneck where ECN marking is not implemented will be unmanaged.

The subsections below in this section provide background information on NSH, ECN, congestion feedback, and terminology used in this document.

### 1.1 NSH Background

The Service Function Chaining (SFC [RFC7665]) architecture calls for the encapsulation of traffic within a service function chaining domain with a Network Service Header (NSH [RFC8300]) added by the "Classifier" (ingress node) on entry to the domain and the NSH being removed on exit from the domain at the egress node. The NSH is used to control the path of a packet in an SFC domain. The NSH is a natural place, in a domain where traffic is NSH encapsulated, to note congestion, avoiding possible confusion due, for example, to changes in the outer transport header in different parts of the domain.

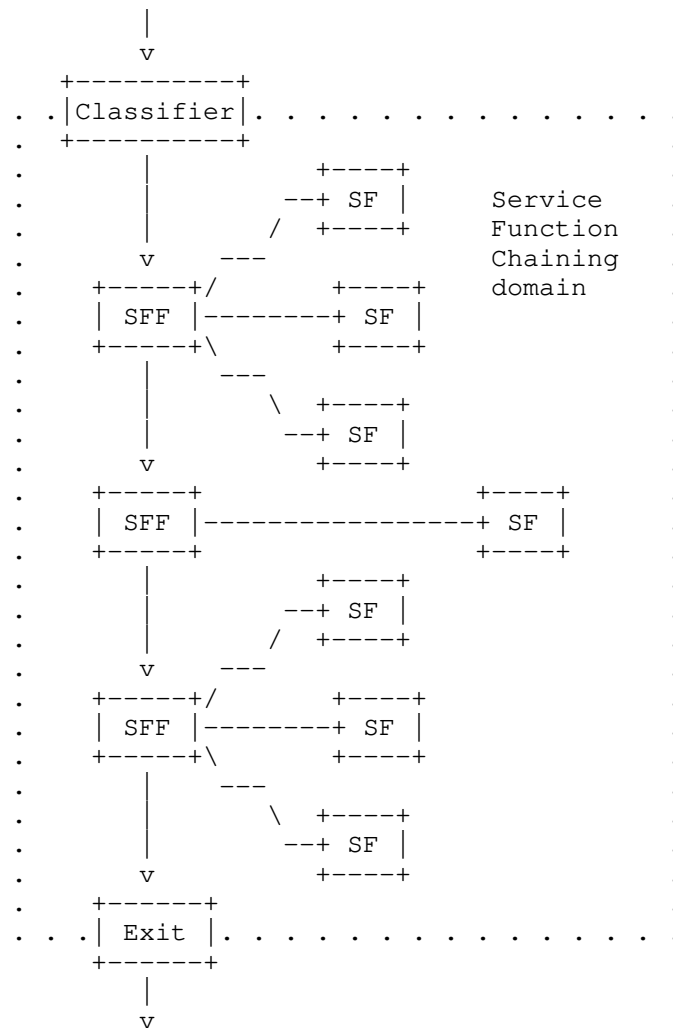


Figure 1. Example SFC Path Forwarding Nodes

Figure 1 shows an SFC domain for the purpose of illustrating the use of NSH. Traffic passes through a sequence of Service Function Forwarders (SFFs) each of which sends the traffic to one or more Service Functions (SFs). Each SF performs some operation on the traffic, for example firewall or Network Address Translation (NAT), and then returns it to the SFF from which it was received.

Logically, during the transit of each SFF, the outer transport header that got the packet to the SFF is stripped, the SFF decides on the next forwarding step, either adding a transport header or, if the SFF is the exit/egress, removing the NSH header. The transport headers

added may be different in different regions of the SFC domain. For example, IP could be used for some SFF-to-SFF communication and MPLS used for other such communication.

## 1.2 ECN Background

Explicit congestion notification (ECN [RFC3168]) allows a forwarding element (such as a router or an Service Function Forwarder (SFF) or Service Function (SF)) to notify downstream devices of the onset of congestion without having to drop packets. This can be used as an element in active queue management (AQM) [RFC7567] to improve network efficiency through better traffic control without packet drops. The forwarding element can explicitly mark some packets in an ECN field instead of dropping the packet. For example, a two-bit field is available for ECN marking in IP headers [RFC3168].

## 1.3 Tunnel Congestion Feedback Background

Tunnel Congestion Feedback [TunnelCongFeedback] is a building block for various congestion mitigation methods. It supports feedback of congestion information from an egress node to an ingress node. Examples of actions that can be taken by an ingress node when it has knowledge of downstream congestion include those listed below. Details of implementing these traffic control methods, beyond those given here, are outside the scope of this document.

Any action by the ingress to reduce congestion needs to allow sufficient time for the end-to-end congestion control loop to respond first, for instance by the ingress taking a smoothed average of the level of congestion signalled by feedback from the tunnel egress.

- (1) Traffic throttling (policing), where the downstream traffic flowing out of the ingress node is limited to reduce or eliminate congestion.
- (2) Upstream congestion feedback, where the ingress node sends messages upstream to or towards the ultimate traffic source, a function that can throttle traffic generation/transmission.
- (3) Traffic re-direction, where the ingress node configures the NSH of some future traffic so that it avoids congested paths. Great care must be taken to avoid (a) significant re-ordering of traffic in flows that it is desirable to keep in order and (b) oscillation/instability in traffic paths due to alternate congestion of previously idle paths and the idling of previously congested paths. For example, it is preferable to classify

traffic into flows of a sufficiently coarse granularity that the flows are long lived and use a stable path per flow sending only newly appearing flows on apparently uncongested paths.

Figure 2 shows an example path from an origin sender to a final receiver passing through an example chain of service functions between the ingress and egress of an SFC domain. The path is also likely to pass through other network nodes outside the SFC domain (not shown). The figure shows typical congestion feedback that would be expected from the final receiver to the origin sender, which controls the load the origin sender applies to all elements on the path. The figure also shows the congestion feedback from the egress to the ingress of the SFC domain that is described in this document, to control or balance load within the SFC domain.

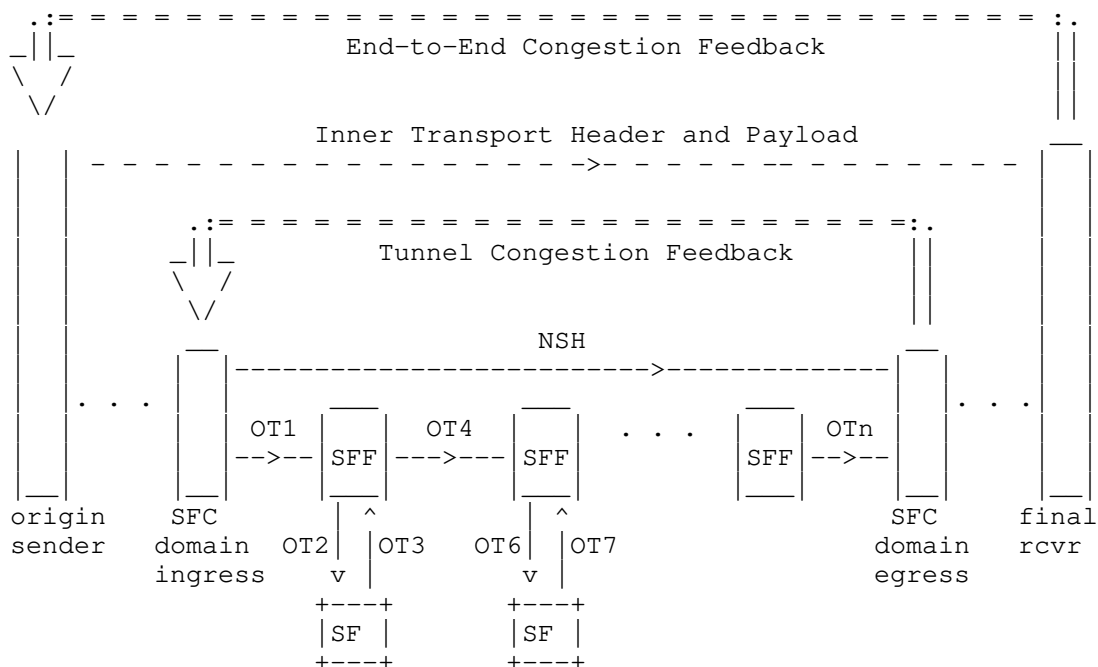


Figure 2: Congestion Feedback across an SFC Domain

SFC Domain congestion feedback in Figure 2 is shown within the context of an end-to-end congestion feedback loop. Also shown is the encapsulated layering of NSH headers within a series of outer transport headers (OT1, OT2, ... OTn).

#### 1.4 Conventions Used in This Document

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

##### Acronyms:

AQM - Active Queue Management [RFC7567]

CE - Congestion Experienced [RFC3168]

downstream - The direction from ingress to egress

ECN - Explicit Congestion Notification [RFC3168]

ECT - ECN Capable Transport [RFC3168]

IPFIX - IP Flow Information Export [RFC7011]

Not-ECT - Not ECN-Capable Transport [RFC3168]

NSH - Network Service Header [RFC8300]

SF - Service Function [RFC7665]

SFC - Service Function Chaining [RFC7665]

SFF - Service Function Forwarder [RFC7665] - A type of node that forwards based on the NSH.

TLV - Type Length Value

upstream - The direction from egress to ingress

## 2. The NSH ECN Field

The NSH header is used to encapsulate and control the subsequent path of traffic (see Section 2 of [RFC8300]). The NSH also provides for metadata inclusion, as shown in Figure 3.

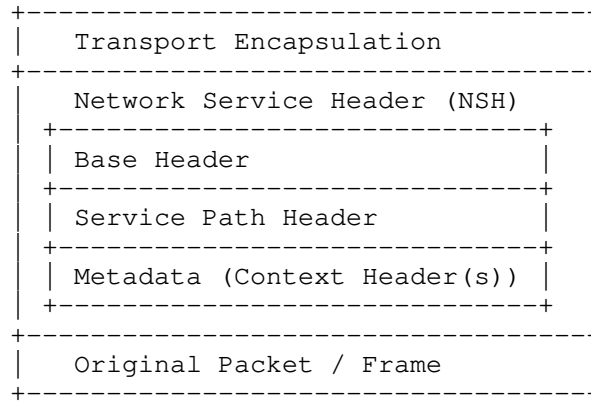


Figure 3. Data Encapsulation with the NSH

Two currently unused bits (indicated by "U") in the NSH Base Header (Section 2.2 of [RFC8300]) are allocated for ECN as shown in Figure 4.

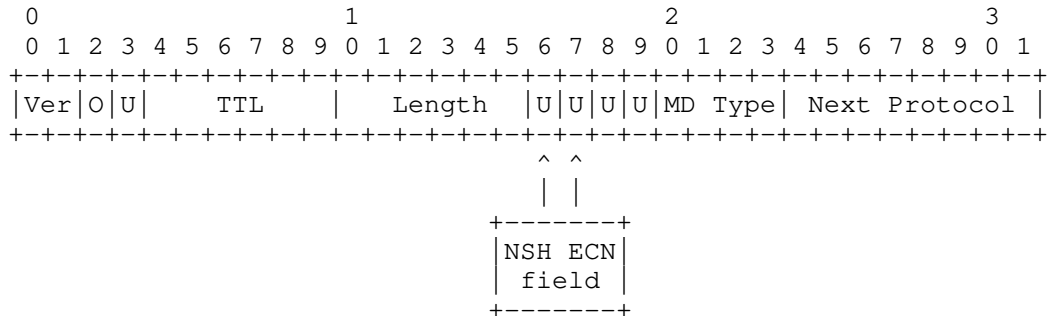


Figure 4: NSH Base Header

Note to RFC Editor: The above figure should be adjusted based on the bits assigned by IANA (see Section 5) and this note deleted.

Table 1 shows the meaning of the code points in the NSH ECN field. These have the same meaning as the ECN field code points in the IPv4 or IPv6 header as defined in [RFC3168].

| Binary | Name    | Meaning                   |
|--------|---------|---------------------------|
| 00     | Not-ECT | Not ECN-Capable Transport |
| 01     | ECT(1)  | ECN-Capable Transport     |
| 10     | ECT(0)  | ECN-Capable Transport     |
| 11     | CE      | Congestion Experienced    |

Table 1. ECN Field Code Points

### 3. ECN Support in the NSH

This section describes the required behavior to support ECN using the NSH. There are two aspects to ECN support:

1. ECN propagation during encapsulation or decapsulation
2. ECN marking during congestion at bottlenecks.

While this section covers all combinations of ECN-aware and not ECN-aware, it is expected that in most cases the NSH domain will be uniform so that, if this document is applicable, all SFFs will support ECN; however, some legacy SFs might not support ECN.

#### ECN Propagation:

The specification of ECN tunneling [RFC6040] explains that an ingress must not propagate ECN support into an encapsulating header unless the egress supports correct onward propagation of the ECN field during decapsulation. We define Compliant ECN Decapsulation here as decapsulation compliant with either [RFC6040] or an earlier compatible equivalent ([RFC4301], or full functionality mode of [RFC3168]).

The procedures in Section 3.2.1 ensure that each ingress of the large number of possible transport links within the SFC domain does not propagate ECN support into the encapsulating outer transport header unless the corresponding egress of that link supports Compliant ECN Decapsulation.

Section 3.3 requires that all the egress nodes of the SFC domain support Compliant ECN Decapsulation in conjunction with tunnel congestion feedback, otherwise the scheme in this document will not work.

#### ECN Marking:

At transit nodes the marking behavior specified in 3.2.1 is recommended and if not implemented at such transit nodes, there may be unmanaged congestion.

Detection of congestion will be most effective if ECN marking is supported by all potential bottlenecks inside the domain in which NSH is being used to route traffic as well as at the ingress and egress. Nodes that do not support ECN marking, or that support AQM but not ECN, will naturally use drop to relieve congestion. The gap in the end-to-end packet sequence will be detected as congestion by the final receiving endpoint, but not by the NSH egress (see Figure 2).



### 3.1 At The Ingress

When the ingress/Classifier encapsulates an incoming IP packet with an NSH, it MUST set the NSH ECN field using the "Normal mode" specified in [RFC6040] (i.e., copied from the incoming IP header).

Then, if the resulting NSH ECN field is Not-ECT, the ingress SHOULD set it to ECT(0). This indicates that, even though the end-to-end transport is not ECN-capable, the egress and ingress of the SFC domain are acting as an ECN-capable transport. This approach will inherently support all known variants of ECN, including the experimental L4S capability [RFC8311], [ecnL4S].

Packets arriving at the ingress might not use IP. If the protocol of arriving packets supports an ECN field similar to IP, the procedures for IP packets can be used. If arriving packets do not support an ECN field similar to IP, they MUST be treated as if they are Not-ECT IP packets.

Then, as the NSH encapsulated packet is further encapsulated with a transport header, if ECN marking is available for that transport (as it is for IP [RFC3168] and MPLS [RFC5129]), the ECN field of the transport header MUST be set using the "Normal mode" specified in [RFC6040] (i.e., copied from the NSH ECN field).

A summary of these normative steps is given in Table 2.

| Incoming Header<br>(also equal to<br>departing Inner<br>Header) | Departing NSH<br>and Outer<br>Headers |
|-----------------------------------------------------------------|---------------------------------------|
| Not-ECT                                                         | ECT(0)                                |
| ECT(0)                                                          | ECT(0)                                |
| ECT(1)                                                          | ECT(1)                                |
| CE                                                              | CE                                    |

Table 2. Setting of ECN fields by an ingress/Classifier

The requirements in this section apply to all ingress nodes for the domain in which NSH is being used to route traffic.

### 3.2 At Transit Nodes

This section described behavior at nodes that forward based on the NSH such as SFF and other forwarding nodes such as IP routers. Figure 5 shows a packet on the wire between forwarding nodes.

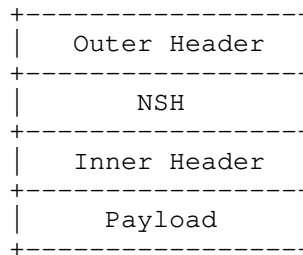


Figure 5. Packet in Transit

#### 3.2.1 At NSH Transit Nodes

When a packet is received at an NSH based forwarding node N1, such as an SFF, the outer transport encapsulation is removed and its ECN marking SHOULD be combined into the NSH ECN marking as specified in [RFC6040]. If this is not done, any congestion encountered at non-NSH transit nodes between N1 and the next upstream NSH based forwarding node will be lost and not transmitted downstream.

The NSH forwarding node SHOULD use a recognized AQM algorithm [RFC7567] to detect congestion. If the NSH ECN field indicates ECT, it will probabilistically set the NSH ECN field to the Congestion Experienced (CE) value or, in cases of extreme congestion, drop the packet.

When the NSH encapsulated packet is further encapsulated for transmission to the next SFF or SF, ECN marking behavior depends on whether or not the node that will decapsulate the outer header supports Compliant ECN Decapsulation (see Section 3). If it does, then the ingress node propagates the NSH ECN field to this outer encapsulation using the "Normal Mode" of ECN encapsulation [RFC6040] (it copies the ECN field). If it does not, then the ingress MUST clear ECN in the outer encapsulation to non-ECT (the "Compatibility Mode" of [RFC6040]).

### 3.2.2 At an SF/Proxy

If the SF is NSH and ECN-aware, the processing is essentially the same at the SF as at an SFF as discussed in Section 3.2.1.

If the SF is NSH-aware but not ECN-aware, then the SFF transmitting the packet to the SF will use Compatibility Mode. Congestion encountered in the SFF to SF and SF to SFF paths will be unmanaged.

If the SF is not NSH-aware, then an NSH proxy will be between the SFF and the SF to avoid exposure of the NSH at the SF that does not understand NSHs. This is described in Section 4.6 of [RFC7665]. The SF and proxy together look to the SFF like an NSH-aware SF. The behavior at the proxy and SF in this case is as below:

If such a proxy is not ECN-aware then congestion in the entire path from SFF to proxy to SF back to proxy to SFF will be unmanaged.

If the proxy is ECN-aware the proxy uses an AQM to indicate congestion in the proxy itself in the NSH that it returns to the SFF. The outer header used for the proxy to SF path uses Normal Mode. The outer head used for the proxy return to SFF path uses Normal Mode based copying the NSH ECN field to the outer header. Thus congestion in the proxy will be managed. Congestion in the SF will be managed only if the SF is ECN-aware implementing an AQM.

### 3.2.3 At Other Forwarding Nodes

Other forwarding nodes, that is non-NSH forwarding nodes between NSH forwarding nodes, such as IP routers, might also be potential bottlenecks. If so, they SHOULD implement an AQM algorithm to update the ECN marking in the outer transport header as specified in [RFC3168].

### 3.3 At Exit/Egress

First, any actions are taken based on Congestion Experienced such as forwarding statistics back to the ingress (see Section 4). If the packet being carried inside the NSH is IP, when the NSH is removed the NSH ECN field MUST be combined with IP ECN field as specified in Table 3 that was extracted from [RFC6040]. This requirement applies to all egress nodes for the domain in which NSH is being used to route traffic.

| Arriving<br>Inner<br>Header | Arriving Outer Header |         |         |        |
|-----------------------------|-----------------------|---------|---------|--------|
|                             | Not-ECT               | ECT (0) | ECT (1) | CE     |
| Not-ECT                     | Not-ECT               | Not-ECT | Not-ECT | <drop> |
| ECT (0)                     | ECT (0)               | ECT (0) | ECT (0) | CE     |
| ECT (1)                     | ECT (1)               | ECT (1) | ECT (1) | CE     |
| CE                          | CE                    | CE      | CE      | CE     |

Table 3. Exit ECN Fields Merger

All the egress nodes of the SFC domain MUST support Compliant ECN Decapsulation as specified in this section. If this is not the case, the scheme described in this document will not work, and cannot be used.

### 3.4 Conservation of Packets

The SFC specification permits an SF to absorb packets and to generate new packets as well as to process and forward the packets it receives. Such actions might appear to be packet loss due to congestion or might mask the loss of packets by generating additional packets.

The tunnel congestion feedback approach [TunnelCongFeedback] detects loss by counting payload bytes in at the ingress and counting them out at the egress. This does not work unless nodes conserve the amount of payload bytes. Therefore, it will not be possible to detect loss using this technique if they are not conserved.

Nonetheless, if a bottleneck supports ECN marking, it will be possible to detect the very high level of CE markings that are associated with congestion that is so excessive that it leads to loss. However, it will not be possible for the tunnel congestion feedback approach to detect any congestion, whether slight or severe, if it occurs at a bottleneck that does not support ECN marking.

#### 4. Tunnel Congestion Feedback Support

The collection and storage of congestion information may be useful for later analysis but, unless it can be fed back to a point which can take action to reduce congestion, it will not be useful in real time. Such congestion feedback to the ingress enables it to take actions such as those listed in Section 1.3.

IP Flow Information Export (IPFIX [RFC7011]) provides a standard for communicating traffic flow statistics. As extended by [TunnelCongFeedback], IPFIX can be used to determine the extent of congestion between an ingress and egress.

IPFIX recommends use of SCTP [RFC4960] in partial reliability mode. This mode allows loss of some packets, which is tolerable because IPFIX communicates cumulative statistics. IPFIX over SCTP SHOULD be used directly where there is IP connectivity between the ingress and egress; however, there might be different transport protocols or address spaces used in different regions of an SFC domain that make such direct IP connectivity problematic. The NSH provides the general method of routing of traffic within such domain so the IPFIX over SCTP over IP traffic should be encapsulated in NSH when necessary.

## 5. IANA Considerations

IANA is requested to assign two contiguous bits in the NSH Base Header Bits registry for ECN (bits 16 and 17 suggested) and note this assignment as follows:

| Bit        | Description | Reference       |
|------------|-------------|-----------------|
| -----      | -----       | -----           |
| tbd(16-17) | NSH ECN     | [this document] |

## 6. Security Considerations

For general NSH security considerations, see [RFC8300].

For security considerations concerning tampering with ECN signaling, see [RFC3168]. For security considerations concerning ECN encapsulation, see [RFC6040].

For general IPFIX security considerations, see [RFC7011]. If deployed in an untrusted environment, the signaling traffic between ingress and egress can be protected utilizing the security mechanisms provided by IPFIX (see section 11 in RFC7011).

The solution in this document does not introduce any greater potential to invade privacy than would have been possible without the solution.

## 7. Acknowledgements

The authors wish to thank the following for their comments and suggestion:

Joel Halpern, Tal Mizrahi, Xinpeng Wei

## 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>>.
- [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>>.
- [RFC5129] - Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <<https://www.rfc-editor.org/info/rfc5129>>.
- [RFC6040] - Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<http://www.rfc-editor.org/info/rfc6040>>.
- [RFC7011] - Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [RFC7567] - Baker, F., Ed., and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <<http://www.rfc-editor.org/info/rfc7567>>.
- [RFC8174] - Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<http://www.rfc-editor.org/info/rfc8174>>.
- [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>>.
- [TunnelCongFeedback] - Wei, X., Zhu, L., and L. Deng, "Tunnel Congestion Feedback", draft-ietf-tsvwg-tunnel-congestion-feedback, work in progress.



## Informative References

- [RFC4301] - Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4960] - Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<https://www.rfc-editor.org/info/rfc4960>>.
- [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>>.
- [RFC8311] - Black, D., "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation", RFC 8311, DOI 10.17487/RFC8311, January 2018, <<https://www.rfc-editor.org/info/rfc8311>>.
- [ecnL4S] - De Schepper, K., and B. Briscoe, "Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay (L4S)", draft-ietf-tsvwg-ecn-l4s-id, work in progress.

Authors' Addresses

Donald E. Eastlake, 3rd  
Huawei Technologies  
1424 Pro Shop Court  
Davenport, FL 33896 USA

Tel: +1-508-333-2270  
Email: d3e3e3@gmail.com

Bob Briscoe  
Independent  
UK

Email: ietf@bobbriscoe.net  
URI: <http://bobbriscoe.net/>

Andrew G. Malis  
Huawei Technologies

Email: agmalis@gmail.com

## Copyright and IPR Provisions

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License. The definitive version of an IETF Document is that published by, or under the auspices of, the IETF. Versions of IETF Documents that are published by third parties, including those that are translated into other languages, should not be considered to be definitive versions of IETF Documents. The definitive version of these Legal Provisions is that published by, or under the auspices of, the IETF. Versions of these Legal Provisions that are published by third parties, including those that are translated into other languages, should not be considered to be definitive versions of these Legal Provisions. For the avoidance of doubt, each Contributor to the IETF Standards Process licenses each Contribution that he or she makes as part of the IETF Standards Process to the IETF Trust pursuant to the provisions of RFC 5378. No language to the contrary, or terms, conditions or rights that differ from or are inconsistent with the rights and licenses granted under RFC 5378, shall have any effect and shall be null and void, whether published or posted by such Contributor, or included with or in such Contribution.



SFC  
Internet-Draft  
Intended status: Informational  
Expires: December 20, 2018

J. Guichard, Ed.  
H. Song  
Huawei  
J. Tantsura  
Nuage Networks  
J. Halpern  
Ericsson  
W. Henderickx  
Nokia  
M. Boucadair  
Orange  
June 18, 2018

NSH and Segment Routing Integration for Service Function Chaining (SFC)  
draft-guichard-sfc-nsh-sr-02

## Abstract

This document describes two application scenarios where Network Service Header (NSH) and Segment Routing (SR) techniques can be deployed together to support Service Function Chaining (SFC) in an efficient manner while maintaining separation of the service and transport planes as originally intended by the SFC architecture.

In the first scenario, an NSH-based SFC is created using SR as the transport between SFFs. SR in this case is just one of many encapsulations that could be used to maintain the transport-independent nature of NSH-based service chains.

In the second scenario, SR is used to represent each service hop of the NSH-based SFC as a segment within the segment-list. SR and NSH in this case are integrated.

In both scenarios SR is responsible for steering packets between SFFs along a given SFP while NSH is responsible for maintaining the integrity of the service plane, the SFC instance context, and any associated metadata.

These application scenarios demonstrate that NSH and SR can work jointly and complement each other leaving the network operator with the flexibility to use whichever transport technology makes sense in specific areas of their network infrastructure, and still maintain an end-to-end service plane using NSH.

## 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 20, 2018.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                             |    |
|-------------------------------------------------------------|----|
| 1. Introduction . . . . .                                   | 3  |
| 1.1. SFC Overview and Rationale . . . . .                   | 3  |
| 1.2. SFC within SR Networks . . . . .                       | 4  |
| 2. NSH-based SFC with SR-based transport tunnel . . . . .   | 5  |
| 3. SR-based SFC with Integrated NSH Service Plane . . . . . | 9  |
| 4. Encapsulation Details . . . . .                          | 11 |
| 4.1. NSH using MPLS-SR Transport . . . . .                  | 11 |
| 4.2. NSH using SRv6 Transport . . . . .                     | 12 |
| 5. Security Considerations . . . . .                        | 13 |
| 6. IANA Considerations . . . . .                            | 13 |
| 7. Acknowledgments . . . . .                                | 13 |
| 8. References . . . . .                                     | 13 |
| 8.1. Normative References . . . . .                         | 13 |
| 8.2. Informative References . . . . .                       | 13 |

|                              |    |
|------------------------------|----|
| Authors' Addresses . . . . . | 14 |
|------------------------------|----|

## 1. Introduction

### 1.1. SFC Overview and Rationale

The dynamic enforcement of a service-derived, adequate forwarding policy for packets entering a network that supports advanced Service Functions (SFs) has become a key challenge for operators and service providers. Particularly, cascading SFs, for example at the Gi interface in the context of mobile network infrastructure, have shown their limits, such as the same redundant classification features must be supported by many SFs in order to execute their function, some SFs are receiving traffic that they are not supposed to process (e.g., TCP proxies receiving UDP traffic), which inevitably affects their dimensioning and performance, an increased design complexity related to the properly ordered invocation of several SFs, etc.

In order to solve those problems and to avoid the adherence with the underlying physical network topology while allowing for simplified service delivery, Service Function Chaining (SFC) techniques have been introduced.

SFC techniques are meant to rationalize the service delivery logic and master the companion complexity while optimizing service activation time cycles for operators that need more agile service delivery procedures to better accommodate ever-demanding customer requirements. Indeed, SFC allows to dynamically create service planes that can be used by specific traffic flows. Each service plane is realized by invoking and chaining the relevant service functions in the right sequence. [RFC7498] provides an overview of the SFC problem space and [RFC7665] specifies an SFC architecture. The SFC architecture has the merit to not make assumptions on how advanced features (e.g., load-balancing, loose or strict service paths) have to be enabled with a domain. Various deployment options are made available to operators with the SFC architecture and this approach is fundamental to accommodate various and heterogeneous deployment contexts.

Many approaches can be considered for encoding the information required for SFC purposes (e.g., communicate a service chain pointer, encode a list of loose/explicit paths, disseminate a service chain identifier together with a set of context information, etc.). Likewise, many approaches can also be considered for the channel to be used to carry SFC-specific information (e.g., define a new header, re-use existing fields, define an IPv6 extension header, etc.). Among all these approaches, the IETF endorsed a transport-independent SFC encapsulation scheme: NSH [RFC8300]; which is the most mature SFC

encapsulation solution. This design is pragmatic as it does not require replicating the same specification effort as a function of underlying transport encapsulation. Moreover, this design approach encourages consistent SFC-based service delivery in networks enabling distinct transport protocols in various segments of the network or even between SFFs vs SF-SFF hops.

## 1.2. SFC within SR Networks

As described in [I-D.ietf-spring-segment-routing], Segment Routing (SR) leverages the source routing technique. Concretely, a node steers a packet through an SR policy instantiated as an ordered list of instructions called segments. While initially designed for policy-based source routing, SR also finds its application in supporting SFC [I-D.xu-clad-spring-sr-service-chaining]. The two SR flavors, namely MPLS-SR [I-D.ietf-spring-segment-routing-mpls] and SRv6 [I-D.ietf-6man-segment-routing-header], can both encode a Service Function (SF) as a segment so that an SFC can be specified as a segment list. Nevertheless, and as discussed in [RFC7498], traffic steering is only a subset of the issues that motivated the design of the SFC architecture. Further considerations such as simplifying classification at intermediate SFs and allowing for coordinated behaviors among SFs by means of supplying context information should be taken into account when designing an SFC data plane solution.

While each scheme (i.e., NSH-based SFC and SR-based SFC) can work independently, this document describes how the two can be used together in concert and complement each other through two representative application scenarios. Both application scenarios may be supported using either MPLS-SR or SRv6:

- o NSH-based SFC with SR-based transport: in this scenario segment routing provides the transport encapsulation between SFFs while NSH is used to convey and trigger SFC policies.
- o SR-based SFC with integrated NSH service plane: in this scenario each service hop of the SFC is represented as a segment of the SR segment-list. SR is responsible for steering traffic through the necessary SFFs as part of the segment routing path and NSH is responsible for maintaining the service plane, and holding the SFC instance context and associated metadata.

It is of course possible to combine both of these two scenarios so as to support specific deployment requirements and use cases.



## 2. NSH-based SFC with SR-based transport tunnel

Because of the transport-independent nature of NSH-based service chains, it is expected that the NSH has broad applicability across different domains of a network. By way of illustration the various SFs involved in a service chain are available in a single data center, or spread throughout multiple locations (e.g., data centers, different POPs), depending upon the operator preference and/or availability of service resources. Regardless of where the service resources are deployed it is necessary to provide traffic steering through a set of SFFs and NSH-based service chains provide the flexibility for the network operator to choose which particular transport encapsulation to use between SFFs, which may be different depending upon which area of the network the SFFs/SFs are currently deployed. Therefore from an SFC architecture perspective, segment routing is simply one of multiple available transport encapsulations that can be used for traffic steering between SFFs. Concretely, NSH does not require to use a unique transport encapsulation when traversing a service chain. NSH-based service forwarding relies upon underlying service node capabilities.

The following three figures provide an example of an SFC established for flow F that has SF instances located in different data centers, DC1 and DC2. For the purpose of illustration, let the SFC's Service Path Identifier (SPI) be 100 and the initial Service Index (SI) be 255.

Referring to Figure 1, packets of flow F in DC1 are classified into an NSH-based SFC and encapsulated after classification as <Inner Pkt><NSH: SPI 100, SI 255><Outer-transport> and forwarded to SFF1 (which is the first SFF hop for this service chain).

After removing the outer transport encapsulation, that may or may not be MPLS-SR or SRv6, SFF1 uses the SPI and SI carried within the NSH encapsulation to determine that it should forward the packet to SF1. SF1 applies its service, decrements the SI by 1, and returns the packet to SFF1. SFF1 therefore has <SPI 100, SI 254> when the packet comes back from SF1. SFF1 does a lookup on <SPI 100, SI 254> which results in <next-hop: DC1-GW1> and forwards the packet to DC1-GW1.

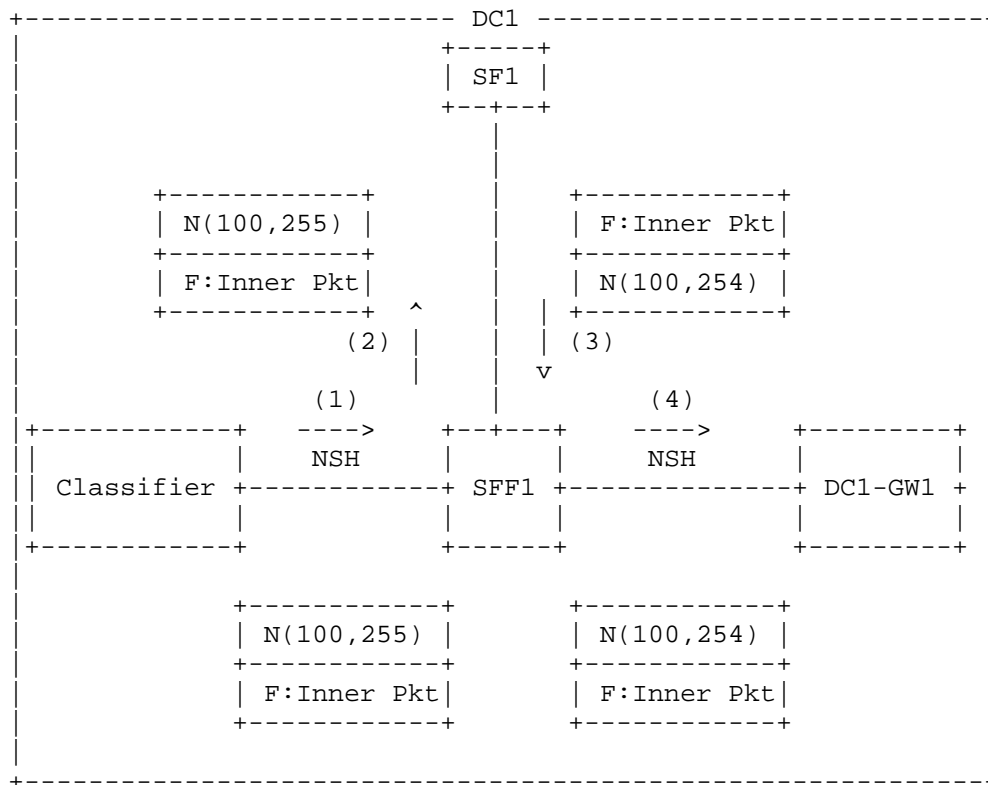


Figure 1: SR for inter-DC SFC - Part 1

Referring now to Figure 2, DC1-GW1 performs a lookup on the information conveyed in the NSH which results in <next-hop: DC2-GW1, encapsulation: SR>. The SR encapsulation has the SR segment-list to forward the packet across the inter-DC network to DC2.

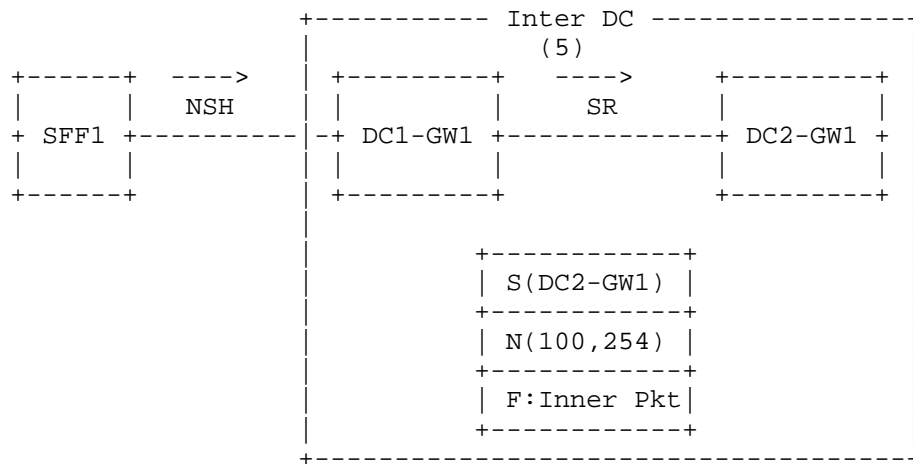


Figure 2: SR for inter-DC SFC - Part 2

When the packet arrives at DC2, as shown in Figure 3, the SR encapsulation is removed and DC2-GW1 performs a lookup on the NSH which results in next-hop: SFF2. The outer transport encapsulation may be any transport that is able to identify NSH as the next protocol.

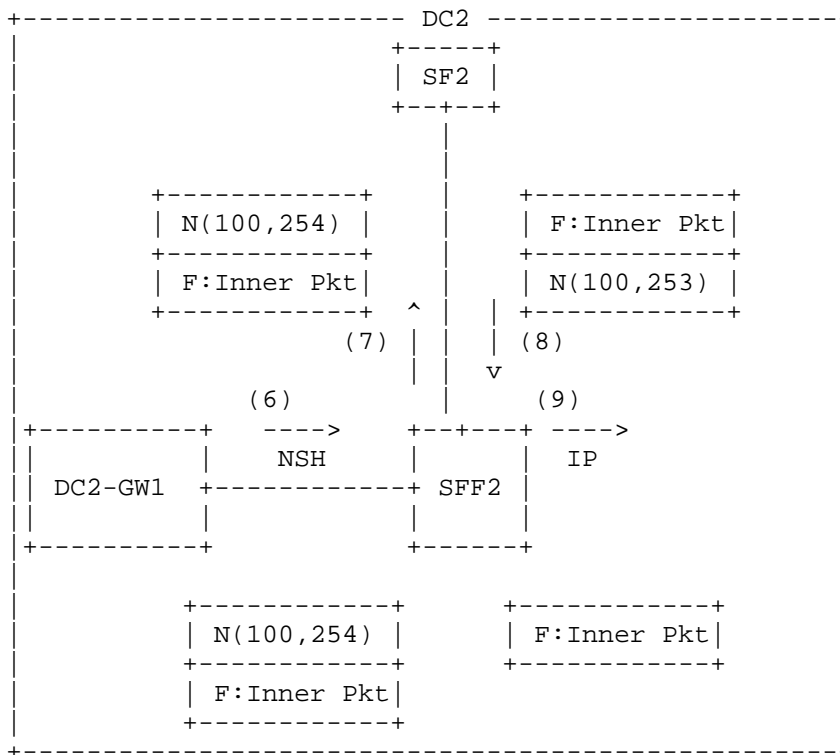


Figure 3: SR for inter-DC SFC - Part 3

The benefits of this scheme are listed hereafter:

- o The network operator is able to take advantage of the transport-independent nature of the NSH encapsulation.
- o The network operator is able to take advantage of the traffic steering capability of SR where appropriate.
- o Light-weight NSH is used in the data center for SFC and avoids more complex hierarchical SFC schemes between data centers.
- o Clear responsibility division and scope between NSH and SR.

Note that this scenario is applicable to any case where multiple segments of a service chain are distributed into multiple domains or where traffic-engineered paths are necessary between SFFs (strict forwarding paths for example).

### 3. SR-based SFC with Integrated NSH Service Plane

In this scenario we assume that the SFs are NSH-aware and therefore it should not be necessary to implement an SFC proxy to achieve Service Function Chaining. The operation relies upon SR to perform SFF-SFF transport and NSH to provide the service plane between SFs thereby maintaining SFC context and metadata.

When a service chain is established, a packet associated with that chain will first encapsulate an NSH that will be used to maintain the end-to-end service plane through use of the SFC context. The SFC context (e.g., the service plane path referenced by the SPI) is used by an SFF to determine the SR segment list for forwarding the packet to the next-hop SFFs. The packet is then encapsulated using the (transport-specific) SR header and forwarded in the SR domain following normal SR operation.

When a packet has to be forwarded to an SF attached to an SFF, the SFF strips the SR information of the packet, updates the SR information, and saves it to a cache indexed by the NSH SPI. This saved SR information is used to encapsulate and forward the packet(s) coming back from the SF.

When the SF receives the packet, it processes it as usual and sends it back to the SFF. Once the SFF receives this packet, it extracts the SR information using the NSH SPI as the index into the cache. The SFF then pushes the SR header on top of the NSH header, and forwards the packet to the next segment in the segment list.

Figure 4 illustrates an example of this scenario.

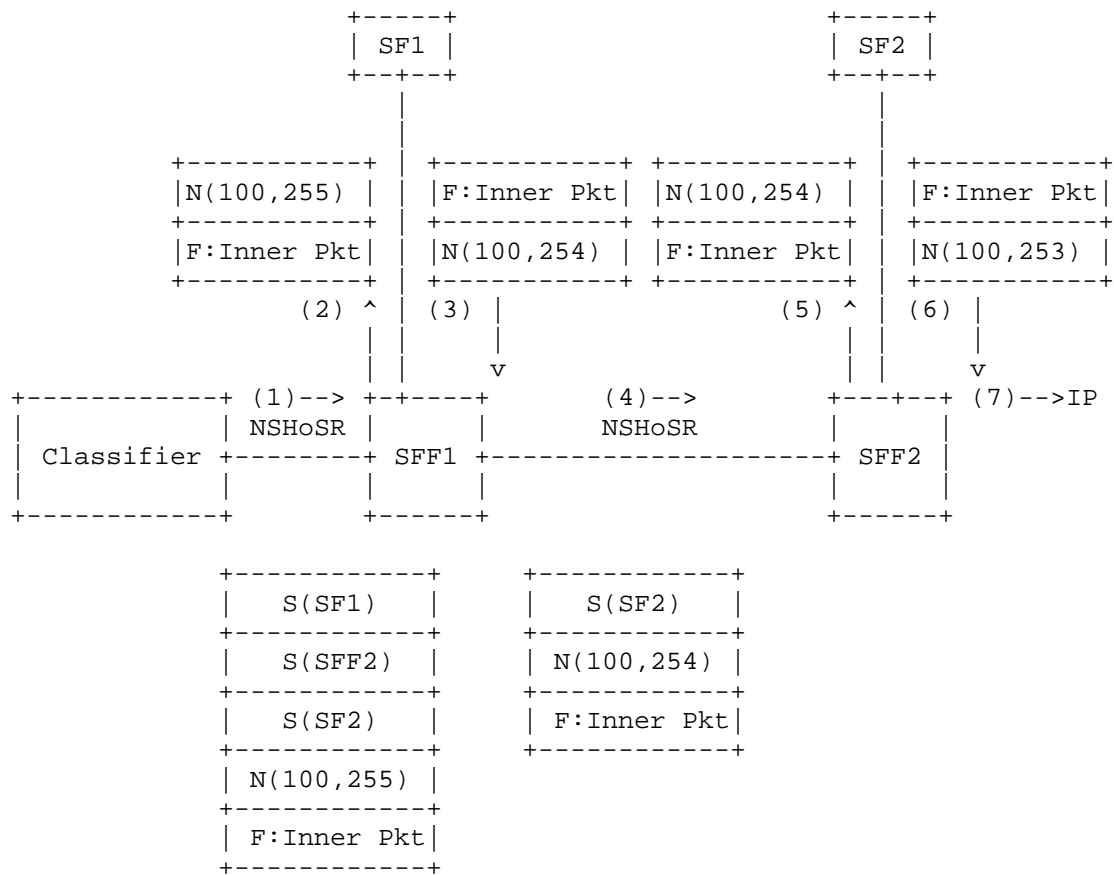


Figure 4: NSH over SR for SFC

The benefits of this scheme include:

- o It is economically sound for SF vendors to only support one unified SFC solution. The SF is unaware of the SR.
- o It simplifies the SFF (i.e., the SR router) by nullifying the needs for re-classification and SR proxy.
- o It provides a unique and standard way to pass metadata to SFs. Note that currently there is no solution for MPLS-SR to carry metadata and there is no solution to pass metadata to SR-unaware SFs.
- o SR is also used for forwarding purposes including between SFFs.

- o It takes advantage of SR to eliminate the NSH forwarding state in SFFs. This applies each time strict or loose SFFs are in use.
- o It requires no interworking as would be the case if MPLS-SR based SFC and NSH-based SFC were deployed as independent mechanisms in different parts of the network.

#### 4. Encapsulation Details

##### 4.1. NSH using MPLS-SR Transport

MPLS-SR instantiates Segment IDs (SIDs) as MPLS labels and therefore the segment routing header is a stack of MPLS labels.

When carrying NSH within an MPLS-SR transport, the full encapsulation headers are as illustrated in Figure 5.

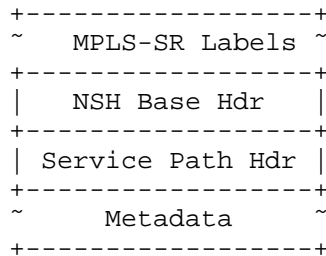


Figure 5: NSH using MPLS-SR Transport

As described in [I-D.ietf-spring-segment-routing] the IGP signaling extension for IGP-Prefix segment includes a flag to indicate whether directly connected neighbors of the node on which the prefix is attached should perform the NEXT operation or the CONTINUE operation when processing the SID. When NSH is carried beneath MPLS-SR it is necessary to terminate the NSH-based SFC at the tail-end node of the MPLS-SR label stack. This is the equivalent of MPLS Ultimate Hop Popping (UHP) and therefore the prefix-SID associated with the tail-end of the SFC MUST be advertised with the CONTINUE operation so that the penultimate hop node does not pop the top label of the MPLS-SR label stack and thereby expose NSH to the wrong SFF. It is RECOMMENDED that a specific prefix-SID be allocated at each node for use by the SFC application for this purpose.

At the end of the MPLS-SR path it is necessary to provide an indication to the tail-end that NSH follows the MPLS-SR label stack.

There are several ways to achieve this but its specification is outside the scope of this document.

#### 4.2. NSH using SRv6 Transport

When carrying NSH within an SRv6 transport the full encapsulation is as illustrated in Figure 6.

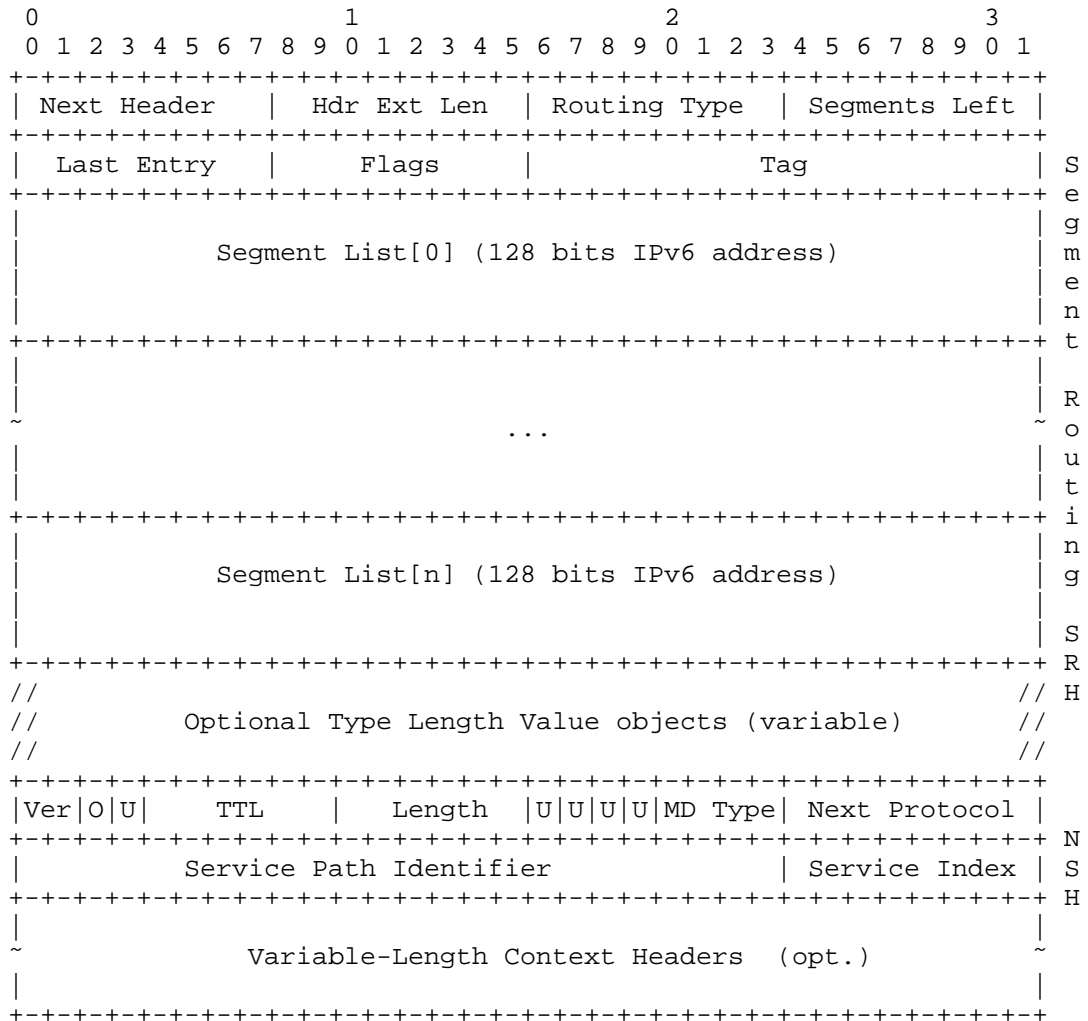


Figure 6: NSH using SRv6 Transport



## 5. Security Considerations

Generic SFC-related security considerations are discussed in [RFC7665]. NSH-specific security considerations are discussed in [RFC8300].

## 6. IANA Considerations

This memo includes no request to IANA.

## 7. Acknowledgments

TBD.

## 8. References

### 8.1. Normative References

- [I-D.ietf-spring-segment-routing]  
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.
- [I-D.ietf-spring-segment-routing-mpls]  
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-12 (work in progress), February 2018.
- [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>>.

### 8.2. Informative References

- [I-D.ietf-6man-segment-routing-header]  
Previdi, S., Filsfils, C., Raza, K., Dukes, D., Leddy, J.,  
Field, B., daniel.voyer@bell.ca, d.,  
daniel.bernier@bell.ca, d., Matsushima, S., Leung, I.,  
Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun,  
D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing  
Header (SRH)", draft-ietf-6man-segment-routing-header-09  
(work in progress), March 2018.
- [I-D.xu-clad-spring-sr-service-chaining]  
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca,  
d., Decraene, B., Yadlapalli, C., Henderickx, W., Salsano,  
S., and S. Ma, "Segment Routing for Service Chaining",  
draft-xu-clad-spring-sr-service-chaining-00 (work in  
progress), December 2017.
- [RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for  
Service Function Chaining", RFC 7498,  
DOI 10.17487/RFC7498, April 2015,  
<<https://www.rfc-editor.org/info/rfc7498>>.

#### Authors' Addresses

James N Guichard (editor)  
Huawei  
2330 Central Express Way  
Santa Clara  
USA

Email: [james.n.guichard@huawei.com](mailto:james.n.guichard@huawei.com)

Haoyu Song  
Huawei  
2330 Central Express Way  
Santa Clara  
USA

Email: [haoyu.song@huawei.com](mailto:haoyu.song@huawei.com)

Jeff Tantsura  
Nuage Networks  
USA

Email: [jefftant.ietf@gmail.com](mailto:jefftant.ietf@gmail.com)

Joel Halpern  
Ericsson  
USA

Email: joel.halpern@ericsson.com

Wim Henderickx  
Nokia  
USA

Email: wim.henderickx@nokia.com

Mohamed Boucadair  
Orange  
USA

Email: mohamed.boucadair@orange.com

sfc  
Internet-Draft  
Intended status: Standards Track  
Expires: December 2, 2018

F. Brockners  
S. Bhandari  
V. Govindan  
C. Pignataro  
Cisco  
H. Gredler  
RtBrick Inc.  
J. Leddy  
Comcast  
S. Youell  
JMPC  
T. Mizrahi  
Marvell  
D. Mozes  
  
P. Lapukhov  
Facebook  
R. Chang  
Barefoot Networks  
May 31, 2018

NSH Encapsulation for In-situ OAM Data  
draft-ietf-sfc-ioam-nsh-00

Abstract

In-situ Operations, Administration, and Maintenance (OAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document outlines how IOAM data fields are encapsulated in the Network Service Header (NSH).

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 December 2, 2018.

#### Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

|                                                         |   |
|---------------------------------------------------------|---|
| 1. Introduction . . . . .                               | 2 |
| 2. Conventions . . . . .                                | 3 |
| 3. IOAM data fields encapsulation in NSH . . . . .      | 3 |
| 4. Considerations . . . . .                             | 5 |
| 4.1. Discussion of the encapsulation approach . . . . . | 5 |
| 4.2. IOAM and the use of the NSH O-bit . . . . .        | 6 |
| 5. IANA Considerations . . . . .                        | 6 |
| 6. Security Considerations . . . . .                    | 7 |
| 7. Acknowledgements . . . . .                           | 7 |
| 8. References . . . . .                                 | 7 |
| 8.1. Normative References . . . . .                     | 7 |
| 8.2. Informative References . . . . .                   | 8 |
| Authors' Addresses . . . . .                            | 8 |

#### 1. Introduction

In-situ OAM (IOAM) records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM data is added to the data packets rather than is being sent within packets specifically dedicated to OAM. This document defines how IOAM data fields are transported as part of the Network Service Header (NSH) [RFC8300] encapsulation. The IOAM data fields are defined in [I-D.ietf-ippm-ioam-data]. An implementation of IOAM which leverages NSH to carry the IOAM data is available from the FD.io open source software project [FD.io].

## 2. Conventions

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].

Abbreviations used in this document:

IOAM: In-situ Operations, Administration, and Maintenance

NSH: Network Service Header

OAM: Operations, Administration, and Maintenance

TLV: Type, Length, Value

## 3. IOAM data fields encapsulation in NSH

NSH is defined in [RFC8300]. IOAM data fields are carried in NSH using a next protocol header which follows the NSH MDx metadata TLVs. An IOAM header is added containing the different IOAM data fields defined in [I-D.ietf-ippm-ioam-data]. In an administrative domain where IOAM is used, insertion of the IOAM header in NSH is enabled at the NSH tunnel endpoints, which also serve as IOAM encapsulating/decapsulating nodes by means of configuration.



Multiple IOAM options MAY be included within the NSH encapsulation. For example, if a NSH encapsulation contains two IOAM options before a data payload, the Next Protocol field of the first IOAM option will contain the value of TBD\_IOAM, while the Next Protocol field of the second IOAM option will contain the "NSH Next Protocol" number indicating the type of the data payload.

#### 4. Considerations

This section summarizes a set of considerations on the overall approach taken for IOAM data encapsulation in NSH, as well as deployment considerations.

##### 4.1. Discussion of the encapsulation approach

This section is to support the working group discussion in selecting the most appropriate approach for encapsulating IOAM data fields in NSH.

An encapsulation of IOAM data fields in NSH should be friendly to an implementation in both hardware as well as software forwarders and support a wide range of deployment cases, including large networks that desire to leverage multiple IOAM data fields at the same time.

Hardware and software friendly implementation: Hardware forwarders benefit from an encapsulation that minimizes iterative look-ups of fields within the packet: Any operation which looks up the value of a field within the packet, based on which another lookup is performed, consumes additional gates and time in an implementation - both of which are desired to be kept to a minimum. This means that flat TLV structures are to be preferred over nested TLV structures. IOAM data fields are grouped into three option categories: Trace, proof-of-transit, and edge-to-edge. Each of these three options defines a TLV structure. A hardware-friendly encapsulation approach avoids grouping these three option categories into yet another TLV structure, but would rather carry the options as a serial sequence.

Total length of the IOAM data fields: The total length of IOAM data can grow quite large in case multiple different IOAM data fields are used and large path-lengths need to be considered. If for example an operator would consider using the IOAM trace option and capture node-id, app\_data, egress/ingress interface-id, timestamp seconds, timestamps nanoseconds at every hop, then a total of 20 octets would be added to the packet at every hop. In case this particular deployment would have a maximum path length of 15 hops in the IOAM domain, then a maximum of 300 octets of IOAM data were to be encapsulated in the packet.



Different approaches for encapsulating IOAM data fields in NSH could be considered:

1. Encapsulation of IOAM data fields as "NSH MD Type 2" (see [RFC8300], section 2.5). Each IOAM data field option (trace, proof-of-transit, and edge-to-edge) would be specified by a type, with the different IOAM data fields being TLVs within this the particular option type. NSH MD Type 2 offers support for variable length meta-data. The length field is 6-bits, resulting in a maximum of 256 ( $2^6 \times 4$ ) octets.
2. Encapsulation of IOAM data fields using the "Next Protocol" field. Each IOAM data field option (trace, proof-of-transit, and edge-to-edge) would be specified by its own "next protocol".
3. Encapsulation of IOAM data fields using the "Next Protocol" field. A single NSH protocol type code point would be allocated for IOAM. A "sub-type" field would then specify what IOAM options type (trace, proof-of-transit, edge-to-edge) is carried.

The third option has been chosen here. This option avoids the additional layer of TLV nesting that the use of NSH MD Type 2 would result in. In addition, this option does not constrain IOAM data to a maximum of 256 octets, thus allowing support for very large deployments.

#### 4.2. IOAM and the use of the NSH O-bit

[RFC8300] defines an "O bit" for OAM packets. Per [RFC8300] the O bit must be set for OAM packets and must not be set for non-OAM packets. Packets with IOAM data included MUST follow this definition, i.e. the O bit MUST NOT be set for regular customer traffic which also carries IOAM data and the O bit MUST be set for OAM packets which carry only IOAM data without any regular data payload.

#### 5. IANA Considerations

IANA is requested to allocate protocol numbers for the following "NSH Next Protocol" related to IOAM:

| Next Protocol | Description | Reference     |
|---------------|-------------|---------------|
| x             | TBD_IOAM    | This document |

## 6. Security Considerations

IOAM is considered a "per domain" feature, where one or several operators decide on leveraging and configuring IOAM according to their needs. Still, operators need to properly secure the IOAM domain to avoid malicious configuration and use, which could include injecting malicious IOAM packets into a domain.

## 7. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Stefano Previdi, Hemant Singh, Erik Nordmark, LJ Wobker, and Andrew Yourtchenko for the comments and advice.

## 8. References

### 8.1. Normative References

- [I-D.ietf-ippm-ioam-data] Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., daniel.bernier@bell.ca, d., and J. Lemon, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-02 (work in progress), March 2018.
- [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>>.
- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <<https://www.rfc-editor.org/info/rfc2784>>.
- [RFC3232] Reynolds, J., Ed., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, DOI 10.17487/RFC3232, January 2002, <<https://www.rfc-editor.org/info/rfc3232>>.
- [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>>.

## 8.2. Informative References

- [FD.io] "Fast Data Project: FD.io", <<https://fd.io/>>.
- [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>>.

## Authors' Addresses

Frank Brockners  
Cisco Systems, Inc.  
Hansaallee 249, 3rd Floor  
DUESSELDORF, NORDRHEIN-WESTFALEN 40549  
Germany

Email: [fbrockne@cisco.com](mailto:fbrockne@cisco.com)

Shwetha Bhandari  
Cisco Systems, Inc.  
Cessna Business Park, Sarjapura Marathalli Outer Ring Road  
Bangalore, KARNATAKA 560 087  
India

Email: [shwethab@cisco.com](mailto:shwethab@cisco.com)

Vengada Prasad Govindan  
Cisco Systems, Inc.

Email: [venggovi@cisco.com](mailto:venggovi@cisco.com)

Carlos Pignataro  
Cisco Systems, Inc.  
7200-11 Kit Creek Road  
Research Triangle Park, NC 27709  
United States

Email: [cpignata@cisco.com](mailto:cpignata@cisco.com)

Hannes Gredler  
RtBrick Inc.

Email: [hannes@rtbrick.com](mailto:hannes@rtbrick.com)

John Leddy  
Comcast

Email: John\_Leddy@cable.comcast.com

Stephen Youell  
JP Morgan Chase  
25 Bank Street  
London E14 5JP  
United Kingdom

Email: stephen.youell@jpmorgan.com

Tal Mizrahi  
Marvell  
6 Hamada St.  
Yokneam 20692  
Israel

Email: talmi@marvell.com

David Mozes

Email: mosesster@gmail.com

Petr Lapukhov  
Facebook  
1 Hacker Way  
Menlo Park, CA 94025  
US

Email: petr@fb.com

Remy Chang  
Barefoot Networks  
2185 Park Boulevard  
Palo Alto, CA 94306  
US

SFC  
Internet-Draft  
Intended status: Standards Track  
Expires: 29 October 2022

F. Brockners, Ed.  
Cisco  
S. Bhandari, Ed.  
Thoughtspot  
27 April 2022

Network Service Header (NSH) Encapsulation for In-situ OAM (IOAM) Data  
draft-ietf-sfc-ioam-nsh-09

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) is used for recording and collecting operational and telemetry information while the packet traverses a path between two points in the network. This document outlines how IOAM data fields are encapsulated with the Network Service Header (NSH).

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 29 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

|                                                                 |   |
|-----------------------------------------------------------------|---|
| 1. Introduction . . . . .                                       | 2 |
| 2. Conventions . . . . .                                        | 2 |
| 3. IOAM encapsulation with NSH . . . . .                        | 3 |
| 4. IANA Considerations . . . . .                                | 4 |
| 5. Security Considerations . . . . .                            | 5 |
| 6. Acknowledgements . . . . .                                   | 5 |
| 7. Contributors . . . . .                                       | 5 |
| 8. References . . . . .                                         | 6 |
| 8.1. Normative References . . . . .                             | 6 |
| 8.2. Informative References . . . . .                           | 7 |
| Appendix A. Discussion of the IOAM encapsulation approach . . . | 7 |
| Authors' Addresses . . . . .                                    | 9 |

## 1. Introduction

In-situ OAM (IOAM), as defined in [I-D.ietf-ippm-ioam-data], is used to record and collect OAM information while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM data is added to the data packets rather than is being sent within packets specifically dedicated to OAM. This document defines how IOAM data fields are transported as part of the Network Service Header (NSH) [RFC8300] encapsulation for the Service Function Chaining (SFC) [RFC7665]. The IOAM-Data-Fields are defined in [I-D.ietf-ippm-ioam-data]. An implementation of IOAM which leverages NSH to carry the IOAM data is available from the FD.io open source software project [FD.io].

## 2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Abbreviations used in this document:

IOAM: In-situ Operations, Administration, and Maintenance

NSH: Network Service Header

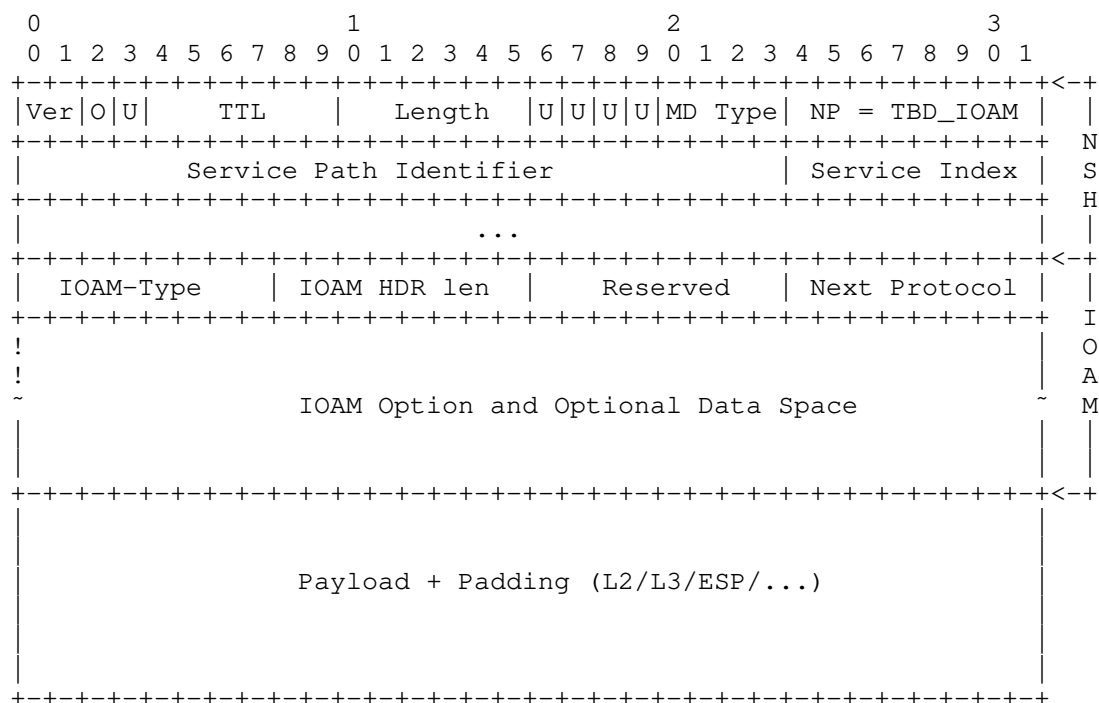
OAM: Operations, Administration, and Maintenance

SFC: Service Function Chaining

TLV: Type, Length, Value

### 3. IOAM encapsulation with NSH

The NSH is defined in [RFC8300]. IOAM-Data-Fields are carried as NSH payload using a next protocol header which follows the NSH headers. An IOAM header is added containing the different IOAM-Data-Fields. The IOAM-Data-Fields MUST follow the definitions corresponding to IOAM-Option-Types (e.g. see Section 5 of [I-D.ietf-ippm-ioam-data] and Section 3.2 of [I-D.ietf-ippm-ioam-direct-export]). In an administrative domain where IOAM is used, insertion of the IOAM header in NSH is enabled at the NSH tunnel endpoints, which also serve as IOAM encapsulating/decapsulating nodes by means of configuration. See [I-D.ietf-ippm-ioam-deployment] for a discussion of deployment related aspects of IOAM-Data-fields.



The NSH header and fields are defined in [RFC8300]. The O-bit MUST be handled following the rules in [I-D.ietf-sfc-oam-packet]. The "NSH Next Protocol" value (referred to as "NP" in the diagram above) is TBD\_IOAM.

The IOAM related fields in NSH are defined as follows:

**IOAM-Type:** 8-bit field defining the IOAM-Option-Type, as defined

in the IOAM Option-Type Registry specified in [I-D.ietf-ippm-ioam-data].

IOAM HDR Len: 8 bit Length field contains the length of the IOAM header in 4-octet units.

Reserved bits: Reserved bits are present for future use. The reserved bits MUST be set to 0x0 upon transmission and ignored upon receipt.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM. The semantics of this field are identical to the Next Protocol field in [RFC8300].

IOAM Option and Data Space: IOAM-Data-Fields as specified by the IOAM-Type field. IOAM-Data-Fields are defined corresponding to the IOAM-Option-Type (e.g. see Section 5 of [I-D.ietf-ippm-ioam-data] and Section 3.2 of [I-D.ietf-ippm-ioam-direct-export]).

Multiple IOAM-Option-Types MAY be included within the NSH encapsulation. For example, if a NSH encapsulation contains two IOAM-Option-Types before a data payload, the Next Protocol field of the first IOAM option will contain the value of TBD\_IOAM, while the Next Protocol field of the second IOAM-Option-Type will contain the "NSH Next Protocol" number indicating the type of the data payload. The applicability of the IOAM Active and Loopback flags [I-D.ietf-ippm-ioam-flags] is outside the scope of this document and may be specified in the future. When a packet with IOAM is received at an NSH based forwarding node such as an Service Function Forwarder (SFF) that does not understand IOAM header, it SHOULD drop the packet. The mechanism to maintain and notify of such events are outside the scope of this document.

#### 4. IANA Considerations

IANA is requested to allocate protocol numbers for the following "NSH Next Protocol" related to IOAM:

| Next Protocol | Description | Reference     |
|---------------|-------------|---------------|
| x             | TBD_IOAM    | This document |



## 5. Security Considerations

IOAM is considered a "per domain" feature, where one or several operators decide on leveraging and configuring IOAM according to their needs. Still, operators need to properly secure the IOAM domain to avoid malicious configuration and use, which could include injecting malicious IOAM packets into a domain. For additional IOAM related security considerations, see Section 10 in [I-D.ietf-ippm-ioam-data]. For additional OAM and NSH related security considerations see Section 5 of [I-D.ietf-sfc-oam-packet].

## 6. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Stefano Previdi, Hemant Singh, Erik Nordmark, LJ Wobker, Andrew Yourtchenko, Greg Mirsky and Mohamed Boucadair for the comments and advice.

## 7. Contributors

In addition to editors listed on the title page, the following people have contributed to this document:

Vengada Prasad Govindan  
Cisco Systems, Inc.  
Email: venggovi@cisco.com

Carlos Pignataro  
Cisco Systems, Inc.  
7200-11 Kit Creek Road  
Research Triangle Park, NC 27709  
United States  
Email: cpignata@cisco.com

Hannes Gredler  
RtBrick Inc.  
Email: hannes@rtbrick.com

John Leddy  
Email: john@leddy.net

Stephen Youell  
JP Morgan Chase  
25 Bank Street  
London E14 5JP  
United Kingdom  
Email: stephen.youell@jpmorgan.com

Tal Mizrahi  
Huawei Network.IO Innovation Lab  
Israel  
Email: tal.mizrahi.phd@gmail.com

David Mozes  
Email: mosesster@gmail.com

Petr Lapukhov  
Facebook  
1 Hacker Way  
Menlo Park, CA 94025  
US  
Email: petr@fb.com

Remy Chang  
Barefoot Networks  
2185 Park Boulevard  
Palo Alto, CA 94306  
US

## 8. References

### 8.1. Normative References

- [I-D.ietf-ippm-ioam-data]  
Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-data-17, 13 December 2021, <<https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-data-17.txt>>.
- [I-D.ietf-sfc-oam-packet]  
Boucadair, M., "OAM Packet and Behavior in the Network Service Header (NSH)", Work in Progress, Internet-Draft, draft-ietf-sfc-oam-packet-01, 25 April 2022, <<https://www.ietf.org/archive/id/draft-ietf-sfc-oam-packet-01.txt>>.
- [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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

[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>>.

## 8.2. Informative References

[FD.io] "Fast Data Project: FD.io", <<https://fd.io/>>.

[I-D.ietf-ippm-ioam-deployment]  
Brockners, F., Bhandari, S., Bernier, D., and T. Mizrahi,  
"In-situ OAM Deployment", Work in Progress, Internet-  
Draft, draft-ietf-ippm-ioam-deployment-01, 11 April 2022,  
<<https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-deployment-01.txt>>.

[I-D.ietf-ippm-ioam-direct-export]  
Song, H., Gafni, B., Zhou, T., Li, Z., Brockners, F.,  
Bhandari, S., Sivakolundu, R., and T. Mizrahi, "In-situ  
OAM Direct Exporting", Work in Progress, Internet-Draft,  
draft-ietf-ippm-ioam-direct-export-07, 13 October 2021,  
<<https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-direct-export-07.txt>>.

[I-D.ietf-ippm-ioam-flags]  
Mizrahi, T., Brockners, F., Bhandari, S., Sivakolundu, R.,  
Pignataro, C., Kfir, A., Gafni, B., Spiegel, M., and J.  
Lemon, "In-situ OAM Loopback and Active Flags", Work in  
Progress, Internet-Draft, draft-ietf-ippm-ioam-flags-07,  
13 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-flags-07.txt>>.

[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>>.

## Appendix A. Discussion of the IOAM encapsulation approach

This section lists several approaches considered for encapsulating IOAM with NSH and presents the rationale for the approach chosen in this document.

An encapsulation of IOAM-Data-Fields in NSH should be friendly to an implementation in both hardware as well as software forwarders and support a wide range of deployment cases, including large networks that desire to leverage multiple IOAM-Data-Fields at the same time.

Hardware and software friendly implementation: Hardware forwarders benefit from an encapsulation that minimizes iterative look-ups of fields within the packet: Any operation which looks up the value of a field within the packet, based on which another lookup is performed, consumes additional gates and time in an implementation - both of which are desired to be kept to a minimum. This means that flat TLV structures are to be preferred over nested TLV structures. IOAM-Data-Fields are grouped into several categories, including trace, proof-of-transit, and edge-to-edge. Each of these options defines a TLV structure. A hardware-friendly encapsulation approach avoids grouping these three option categories into yet another TLV structure, but would rather carry the options as a serial sequence.

Total length of the IOAM-Data-Fields: The total length of IOAM-Data-Fields can grow quite large in case multiple different IOAM-Data-Fields are used and large path-lengths need to be considered. If for example an operator would consider using the IOAM Trace Option-Type and capture node-id, app\_data, egress/ingress interface-id, timestamp seconds, timestamps nanoseconds at every hop, then a total of 20 octets would be added to the packet at every hop. In case this particular deployment would have a maximum path length of 15 hops in the IOAM domain, then a maximum of 300 octets were to be encapsulated in the packet.

Different approaches for encapsulating IOAM-Data-Fields in NSH could be considered:

1. Encapsulation of IOAM-Data-Fields as "NSH MD Type 2" (see [RFC8300], Section 2.5). Each IOAM-Option-Type (e.g. trace, proof-of-transit, and edge-to-edge) would be specified by a type, with the different IOAM-Data-Fields being TLVs within this the particular option type. NSH MD Type 2 offers support for variable length meta-data. The length field is 6-bits, resulting in a maximum of 256 ( $2^6 \times 4$ ) octets.
2. Encapsulation of IOAM-Data-Fields using the "Next Protocol" field. Each IOAM-Option-Type (e.g trace, proof-of-transit, and edge-to-edge) would be specified by its own "next protocol".
3. Encapsulation of IOAM-Data-Fields using the "Next Protocol" field. A single NSH protocol type code point would be allocated for IOAM. A "sub-type" field would then specify what IOAM options type (trace, proof-of-transit, edge-to-edge) is carried.

The third option has been chosen here. This option avoids the additional layer of TLV nesting that the use of NSH MD Type 2 would result in. In addition, this option does not constrain IOAM data to a maximum of 256 octets, thus allowing support for very large deployments.

#### Authors' Addresses

Frank Brockners (editor)  
Cisco Systems, Inc.  
Hansaallee 249, 3rd Floor  
40549 DUESSELDORF  
Germany  
Email: [fbrockne@cisco.com](mailto:fbrockne@cisco.com)

Shwetha Bhandari (editor)  
Thoughtspot  
3rd Floor, Indiqube Orion, 24th Main Rd, Garden Layout, HSR Layout  
Bangalore, KARNATAKA 560 102  
India  
Email: [shwetha.bhandari@thoughtspot.com](mailto:shwetha.bhandari@thoughtspot.com)

Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: December 2, 2018

F. Brockners  
S. Bhandari  
S. Dara  
C. Pignataro  
Cisco  
J. Leddy  
Comcast  
S. Youell  
JPMC  
D. Mozes

T. Mizrahi  
Marvell  
May 31, 2018

Proof of Transit  
draft-ietf-sfc-proof-of-transit-00

Abstract

Several technologies such as Traffic Engineering (TE), Service Function Chaining (SFC), and policy based routing are used to steer traffic through a specific, user-defined path. This document defines mechanisms to securely prove that traffic transited said defined path. These mechanisms allow to securely verify whether, within a given path, all packets traversed all the nodes that they are supposed to visit.

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 December 2, 2018.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                   |    |
|---------------------------------------------------|----|
| 1. Introduction . . . . .                         | 3  |
| 2. Conventions . . . . .                          | 4  |
| 3. Proof of Transit . . . . .                     | 5  |
| 3.1. Basic Idea . . . . .                         | 5  |
| 3.2. Solution Approach . . . . .                  | 6  |
| 3.2.1. Setup . . . . .                            | 7  |
| 3.2.2. In Transit . . . . .                       | 7  |
| 3.2.3. Verification . . . . .                     | 7  |
| 3.3. Illustrative Example . . . . .               | 7  |
| 3.3.1. Basic Version . . . . .                    | 7  |
| 3.3.1.1. Secret Shares . . . . .                  | 8  |
| 3.3.1.2. Lagrange Polynomials . . . . .           | 8  |
| 3.3.1.3. LPC Computation . . . . .                | 8  |
| 3.3.1.4. Reconstruction . . . . .                 | 9  |
| 3.3.1.5. Verification . . . . .                   | 9  |
| 3.3.2. Enhanced Version . . . . .                 | 9  |
| 3.3.2.1. Random Polynomial . . . . .              | 9  |
| 3.3.2.2. Reconstruction . . . . .                 | 10 |
| 3.3.2.3. Verification . . . . .                   | 10 |
| 3.3.3. Final Version . . . . .                    | 11 |
| 3.4. Operational Aspects . . . . .                | 11 |
| 3.5. Alternative Approach . . . . .               | 12 |
| 3.5.1. Basic Idea . . . . .                       | 12 |
| 3.5.2. Pros . . . . .                             | 12 |
| 3.5.3. Cons . . . . .                             | 12 |
| 4. Sizing the Data for Proof of Transit . . . . . | 12 |
| 5. Node Configuration . . . . .                   | 13 |
| 5.1. Procedure . . . . .                          | 14 |
| 5.2. YANG Model . . . . .                         | 14 |
| 6. IANA Considerations . . . . .                  | 17 |
| 7. Manageability Considerations . . . . .         | 17 |

|                                             |    |
|---------------------------------------------|----|
| 8. Security Considerations . . . . .        | 17 |
| 8.1. Proof of Transit . . . . .             | 18 |
| 8.2. Cryptanalysis . . . . .                | 18 |
| 8.3. Anti-Replay . . . . .                  | 19 |
| 8.4. Anti-Preplay . . . . .                 | 19 |
| 8.5. Anti-Tampering . . . . .               | 20 |
| 8.6. Recycling . . . . .                    | 20 |
| 8.7. Redundant Nodes and Failover . . . . . | 20 |
| 8.8. Controller Operation . . . . .         | 20 |
| 8.9. Verification Scope . . . . .           | 21 |
| 8.9.1. Node Ordering . . . . .              | 21 |
| 8.9.2. Stealth Nodes . . . . .              | 21 |
| 9. Acknowledgements . . . . .               | 21 |
| 10. References . . . . .                    | 21 |
| 10.1. Normative References . . . . .        | 21 |
| 10.2. Informative References . . . . .      | 22 |
| Authors' Addresses . . . . .                | 22 |

## 1. Introduction

Several deployments use Traffic Engineering, policy routing, Segment Routing (SR), and Service Function Chaining (SFC) [RFC7665] to steer packets through a specific set of nodes. In certain cases, regulatory obligations or a compliance policy require operators to prove that all packets that are supposed to follow a specific path are indeed being forwarded across an exact set of pre-determined nodes.

If a packet flow is supposed to go through a series of service functions or network nodes, it has to be proven that indeed all packets of the flow followed the path or service chain or collection of nodes specified by the policy. In case some packets of a flow weren't appropriately processed, a verification device should determine the policy violation and take corresponding actions corresponding to the policy (e.g., drop or redirect the packet, send an alert etc.) In today's deployments, the proof that a packet traversed a particular path or service chain is typically delivered in an indirect way: Service appliances and network forwarding are in different trust domains. Physical hand-off-points are defined between these trust domains (i.e. physical interfaces). Or in other terms, in the "network forwarding domain" things are wired up in a way that traffic is delivered to the ingress interface of a service appliance and received back from an egress interface of a service appliance. This "wiring" is verified and then trusted upon. The evolution to Network Function Virtualization (NFV) and modern service chaining concepts (using technologies such as Locator/ID Separation Protocol (LISP), Network Service Header (NSH), Segment Routing (SR), etc.) blurs the line between the different trust domains, because the



hand-off-points are no longer clearly defined physical interfaces, but are virtual interfaces. As a consequence, different trust layers should not to be mixed in the same device. For an NFV scenario a different type of proof is required. Offering a proof that a packet indeed traversed a specific set of service functions or nodes allows operators to evolve from the above described indirect methods of proving that packets visit a predetermined set of nodes.

The solution approach presented in this document is based on a small portion of operational data added to every packet. This "in-situ" operational data is also referred to as "proof of transit data", or POT data. The POT data is updated at every required node and is used to verify whether a packet traversed all required nodes. A particular set of nodes "to be verified" is either described by a set of secret keys, or a set of shares of a single secret. Nodes on the path retrieve their individual keys or shares of a key (using for e.g., Shamir's Secret Sharing scheme) from a central controller. The complete key set is only known to the controller and a verifier node, which is typically the ultimate node on a path that performs verification. Each node in the path uses its secret or share of the secret to update the POT data of the packets as the packets pass through the node. When the verifier receives a packet, it uses its key(s) along with data found in the packet to validate whether the packet traversed the path correctly.

## 2. Conventions

Abbreviations used in this document:

HMAC: Hash based Message Authentication Code. For example, HMAC-SHA256 generates 256 bits of MAC

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

LPC: Lagrange Polynomial Constants

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

NSH: Network Service Header

POT: Proof of Transit

POT-profile: Proof of Transit Profile that has the necessary data for nodes to participate in proof of transit

RND: Random Bits generated per packet. Packet fields that donot change during the traversal are given as input to HMAC-256 algorithm. A minimum of 32 bits (left most) need to be used from the output if RND is used to verify the packet integrity. This is a standard recommendation by NIST.

SEQ\_NO: Sequence number initialized to a predefined constant. This is used in concatenation with RND bits to mitigate different attacks discussed later.

SFC: Service Function Chain

SR: Segment Routing

### 3. Proof of Transit

This section discusses methods and algorithms to provide for a "proof of transit" for packets traversing a specific path. A path which is to be verified consists of a set of nodes. Transit of the data packets through those nodes is to be proven. Besides the nodes, the setup also includes a Controller that creates secrets and secrets shares and configures the nodes for POT operations.

The methods how traffic is identified and associated to a specific path is outside the scope of this document. Identification could be done using a filter (e.g., 5-tuple classifier), or an identifier which is already present in the packet (e.g., path or service identifier, NSH Service Path Identifier (SPI), flow-label, etc.)

The solution approach is detailed in two steps. Initially the concept of the approach is explained. This concept is then further refined to make it operationally feasible.

#### 3.1. Basic Idea

The method relies on adding POT data to all packets that traverse a path. The added POT data allows a verifying node (egress node) to check whether a packet traversed the identified set of nodes on a path correctly or not. Security mechanisms are natively built into the generation of the POT data to protect against misuse (i.e. configuration mistakes, malicious administrators playing tricks with routing, capturing, spoofing and replaying packets). The mechanism for POT leverages "Shamir's Secret Sharing" scheme [SSS].

Shamir's secret sharing base idea: A polynomial (represented by its coefficients) is chosen as a secret by the controller. A polynomial represents a curve. A set of well-defined points on the curve are

needed to construct the polynomial. Each point of the polynomial is called "share" of the secret. A single secret is associated with a particular set of nodes, which typically represent the path, to be verified. Shares of the single secret (i.e., points on the curve) are securely distributed from a Controller to the network nodes. Nodes use their respective share to update a cumulative value in the POT data of each packet. Only a verifying node has access to the complete secret. The verifying node validates the correctness of the received POT data by reconstructing the curve.

The polynomial cannot be constructed if any of the points are missed or tampered. Per Shamir's Secret Sharing Scheme, any lesser points means one or more nodes are missed. Details of the precise configuration needed for achieving security are discussed further below.

While applicable in theory, a vanilla approach based on Shamir's secret sharing could be easily attacked. If the same polynomial is reused for every packet for a path a passive attacker could reuse the value. As a consequence, one could consider creating a different polynomial per packet. Such an approach would be operationally complex. It would be complex to configure and recycle so many curves and their respective points for each node. Rather than using a single polynomial, two polynomials are used for the solution approach: A secret polynomial which is kept constant, and a per-packet polynomial which is public. Operations are performed on the sum of those two polynomials - creating a third polynomial which is secret and per packet.

### 3.2. Solution Approach

Solution approach: The overall algorithm uses two polynomials: POLY-1 and POLY-2. POLY-1 is secret and constant. Each node gets a point on POLY-1 at setup-time and keeps it secret. POLY-2 is public, random and per packet. Each node generates a point on POLY-2 each time a packet crosses it. Each node then calculates (point on POLY-1 + point on POLY-2) to get a (point on POLY-3) and passes it to verifier by adding it to each packet. The verifier constructs POLY-3 from the points given by all the nodes and cross checks whether  $\text{POLY-3} = \text{POLY-1} + \text{POLY-2}$ . Only the verifier knows POLY-1. The solution leverages finite field arithmetic in a field of size "prime number".

Detailed algorithms are discussed next. A simple example is discussed in Section 3.3.

### 3.2.1. Setup

A controller generates a first polynomial (POLY-1) of degree  $k$  and  $k+1$  points on the polynomial. The constant coefficient of POLY-1 is considered the SECRET. The non-constant coefficients are used to generate the Lagrange Polynomial Constants (LPC). Each of the  $k$  nodes (including verifier) are assigned a point on the polynomial i.e., shares of the SECRET. The verifier is configured with the SECRET. The Controller also generates coefficients (except the constant coefficient, called "RND", which is changed on a per packet basis) of a second polynomial POLY-2 of the same degree. Each node is configured with the LPC of POLY-2. Note that POLY-2 is public.

### 3.2.2. In Transit

For each packet, the ingress node generates a random number (RND). It is considered as the constant coefficient for POLY-2. A cumulative value (CML) is initialized to 0. Both RND, CML are carried as within the packet POT data. As the packet visits each node, the RND is retrieved from the packet and the respective share of POLY-2 is calculated. Each node calculates  $(\text{Share}(\text{POLY-1}) + \text{Share}(\text{POLY-2}))$  and CML is updated with this sum. This step is performed by each node until the packet completes the path. The verifier also performs the step with its respective share.

### 3.2.3. Verification

The verifier cross checks whether  $\text{CML} = \text{SECRET} + \text{RND}$ . If this matches then the packet traversed the specified set of nodes in the path. This is due to the additive homomorphic property of Shamir's Secret Sharing scheme.

## 3.3. Illustrative Example

This section shows a simple example to illustrate step by step the approach described above.

### 3.3.1. Basic Version

Assumption: It is to be verified whether packets passed through 3 nodes. A polynomial of degree 2 is chosen for verification.

Choices: Prime = 53.  $\text{POLY-1}(x) = (3x^2 + 3x + 10) \bmod 53$ . The secret to be re-constructed is the constant coefficient of POLY-1, i.e., SECRET=10. It is important to note that all operations are done over a finite field (i.e., modulo prime).

### 3.3.1.1. Secret Shares

The shares of the secret are the points on POLY-1 chosen for the 3 nodes. For example, let  $x_0=2$ ,  $x_1=4$ ,  $x_2=5$ .

$$\text{POLY-1}(2) = 28 \Rightarrow (x_0, y_0) = (2, 28)$$

$$\text{POLY-1}(4) = 17 \Rightarrow (x_1, y_1) = (4, 17)$$

$$\text{POLY-1}(5) = 47 \Rightarrow (x_2, y_2) = (5, 47)$$

The three points above are the points on the curve which are considered the shares of the secret. They are assigned to three nodes respectively and are kept secret.

### 3.3.1.2. Lagrange Polynomials

Lagrange basis polynomials (or Lagrange polynomials) are used for polynomial interpolation. For a given set of points on the curve Lagrange polynomials (as defined below) are used to reconstruct the curve and thus reconstruct the complete secret.

$$\begin{aligned} l_0(x) &= (((x-x_1) / (x_0-x_1)) * ((x-x_2)/(x_0-x_2))) \bmod 53 = \\ &(((x-4) / (2-4)) * ((x-5)/(2-5))) \bmod 53 = \\ &(10/3 - 3x/2 + (1/6)x^2) \bmod 53 \end{aligned}$$

$$\begin{aligned} l_1(x) &= (((x-x_0) / (x_1-x_0)) * ((x-x_2)/(x_1-x_2))) \bmod 53 = \\ &(-5 + 7x/2 - (1/2)x^2) \bmod 53 \end{aligned}$$

$$\begin{aligned} l_2(x) &= (((x-x_0) / (x_2-x_0)) * ((x-x_1)/(x_2-x_1))) \bmod 53 = \\ &(8/3 - 2 + (1/3)x^2) \bmod 53 \end{aligned}$$

### 3.3.1.3. LPC Computation

Since  $x_0=2$ ,  $x_1=4$ ,  $x_2=5$  are chosen points. Given that computations are done over a finite arithmetic field ("modulo a prime number"), the Lagrange basis polynomial constants are computed modulo 53. The Lagrange Polynomial Constant (LPC) would be  $10/3$ ,  $-5$ ,  $8/3$ .

$$\text{LPC}(x_0) = (10/3) \bmod 53 = 21$$

$$\text{LPC}(x_1) = (-5) \bmod 53 = 48$$

$$\text{LPC}(x_2) = (8/3) \bmod 53 = 38$$

For a general way to compute the modular multiplicative inverse, see e.g., the Euclidean algorithm.

#### 3.3.1.4. Reconstruction

Reconstruction of the polynomial is well-defined as

$$\text{POLY1}(x) = l_0(x) * y_0 + l_1(x) * y_1 + l_2(x) * y_2$$

Subsequently, the SECRET, which is the constant coefficient of  $\text{POLY1}(x)$  can be computed as below

$$\text{SECRET} = (y_0 * \text{LPC}(l_0) + y_1 * \text{LPC}(l_1) + y_2 * \text{LPC}(l_2)) \bmod 53$$

The secret can be easily reconstructed using the y-values and the LPC:

$$\begin{aligned} \text{SECRET} &= (y_0 * \text{LPC}(l_0) + y_1 * \text{LPC}(l_1) + y_2 * \text{LPC}(l_2)) \bmod 53 = \bmod (28 * 21 \\ &+ 17 * 48 + 47 * 38) \bmod 53 = 3190 \bmod 53 = 10 \end{aligned}$$

One observes that the secret reconstruction can easily be performed cumulatively hop by hop. CML represents the cumulative value. It is the POT data in the packet that is updated at each hop with the node's respective  $(y_i * \text{LPC}(i))$ , where  $i$  is their respective value.

#### 3.3.1.5. Verification

Upon completion of the path, the resulting CML is retrieved by the verifier from the packet POT data. Recall that verifier is preconfigured with the original SECRET. It is cross checked with the CML by the verifier. Subsequent actions based on the verification failing or succeeding could be taken as per the configured policies.

#### 3.3.2. Enhanced Version

As observed previously, the vanilla algorithm that involves a single secret polynomial is not secure. Therefore, the solution is further enhanced with usage of a random second polynomial chosen per packet.

##### 3.3.2.1. Random Polynomial

Let the second polynomial POLY-2 be  $(\text{RND} + 7x + 10x^2)$ . RND is a random number and is generated for each packet. Note that POLY-2 is public and need not be kept secret. The nodes can be pre-configured with the non-constant coefficients (for example, 7 and 10 in this case could be configured through the Controller on each node). So precisely only RND value changes per packet and is public and the rest of the non-constant coefficients of POLY-2 kept secret.

## 3.3.2.2. Reconstruction

Recall that each node is preconfigured with their respective Share(POLY-1). Each node calculates its respective Share(POLY-2) using the RND value retrieved from the packet. The CML reconstruction is enhanced as below. At every node, CML is updated as

$$\text{CML} = \text{CML} + (((\text{Share}(\text{POLY-1}) + \text{Share}(\text{POLY-2})) * \text{LPC}) \bmod \text{Prime})$$

Let us observe the packet level transformations in detail. For the example packet here, let the value RND be 45. Thus POLY-2 would be  $(45 + 7x + 10x^2)$ .

The shares that could be generated are (2, 46), (4, 21), (5, 12).

At ingress: The fields RND = 45. CML = 0.

At node-1 (x0): Respective share of POLY-2 is generated i.e., (2, 46) because share index of node-1 is 2.

$$\text{CML} = 0 + ((28 + 46) * 21) \bmod 53 = 17$$

At node-2 (x1): Respective share of POLY-2 is generated i.e., (4, 21) because share index of node-2 is 4.

$$\text{CML} = 17 + ((17 + 21) * 48) \bmod 53 = 17 + 22 = 39$$

At node-3 (x2), which is also the verifier: The respective share of POLY-2 is generated i.e., (5, 12) because the share index of the verifier is 12.

$$\text{CML} = 39 + ((47 + 12) * 38) \bmod 53 = 39 + 16 = 55 \bmod 53 = 2$$

The verification using CML is discussed in next section.

## 3.3.2.3. Verification

As shown in the above example, for final verification, the verifier compares:

$$\text{VERIFY} = (\text{SECRET} + \text{RND}) \bmod \text{Prime}, \text{ with Prime} = 53 \text{ here}$$

$$\text{VERIFY} = (\text{RND-1} + \text{RND-2}) \bmod \text{Prime} = (10 + 45) \bmod 53 = 2$$

Since VERIFY = CML the packet is proven to have gone through nodes 1, 2, and 3.

### 3.3.3. Final Version

The enhanced version of the protocol is still prone to replay and preplay attacks. An attacker could reuse the POT metadata for bypassing the verification. So additional measures using packet integrity checks (HMAC) and sequence numbers (SEQ\_NO) are discussed later "Security Considerations" section.

### 3.4. Operational Aspects

To operationalize this scheme, a central controller is used to generate the necessary polynomials, the secret share per node, the prime number, etc. and distributing the data to the nodes participating in proof of transit. The identified node that performs the verification is provided with the verification key. The information provided from the Controller to each of the nodes participating in proof of transit is referred to as a proof of transit profile (POT-profile). Also note that the set of nodes for which the transit has to be proven are typically associated to a different trust domain than the verifier. Note that building the trust relationship between the Controller and the nodes is outside the scope of this document. Techniques such as those described in [I-D.ietf-anima-autonomic-control-plane] might be applied.

To optimize the overall data amount of exchanged and the processing at the nodes the following optimizations are performed:

1. The points (x, y) for each of the nodes on the public and private polynomials are picked such that the x component of the points match. This lends to the LPC values which are used to calculate the cumulative value CML to be constant. Note that the LPC are only depending on the x components. They can be computed at the controller and communicated to the nodes. Otherwise, one would need to distributed the x components to all the nodes.
2. A pre-evaluated portion of the public polynomial for each of the nodes is calculated and added to the POT-profile. Without this all the coefficients of the public polynomial had to be added to the POT profile and each node had to evaluate them. As stated before, the public portion is only the constant coefficient RND value, the pre-evaluated portion for each node should be kept secret as well.
3. To provide flexibility on the size of the cumulative and random numbers carried in the POT data a field to indicate this is shared and interpreted at the nodes.



### 3.5. Alternative Approach

In certain scenarios preserving the order of the nodes traversed by the packet may be needed. An alternative, "nested encryption" based approach is described here for preserving the order

#### 3.5.1. Basic Idea

1. The controller provisions all the nodes with their respective secret keys.
2. The controller provisions the verifier with all the secret keys of the nodes.
3. For each packet, the ingress node generates a random number RND and encrypts it with its secret key to generate CML value
4. Each subsequent node on the path encrypts CML with their respective secret key and passes it along
5. The verifier is also provisioned with the expected sequence of nodes in order to verify the order
6. The verifier receives the CML, RND values, re-encrypts the RND with keys in the same order as expected sequence to verify.

#### 3.5.2. Pros

Nested encryption approach retains the order in which the nodes are traversed.

#### 3.5.3. Cons

1. Standard AES encryption would need 128 bits of RND, CML. This results in a 256 bits of additional overhead is added per packet
2. In hardware platforms that do not support native encryption capabilities like (AES-NI). This approach would have considerable impact on the computational latency

### 4. Sizing the Data for Proof of Transit

Proof of transit requires transport of two data fields in every packet that should be verified:

1. RND: Random number (the constant coefficient of public polynomial)

## 2. CML: Cumulative

The size of the data fields determines how often a new set of polynomials would need to be created. At maximum, the largest RND number that can be represented with a given number of bits determines the number of unique polynomials POLY-2 that can be created. The table below shows the maximum interval for how long a single set of polynomials could last for a variety of bit rates and RND sizes: When choosing 64 bits for RND and CML data fields, the time between a renewal of secrets could be as long as 3,100 years, even when running at 100 Gbps.

| Transfer rate | Secret/RND size | Max # of packets                           | Time RND lasts        |
|---------------|-----------------|--------------------------------------------|-----------------------|
| 1 Gbps        | 64              | $2^{64} = \text{approx. } 2 \cdot 10^{19}$ | approx. 310,000 years |
| 10 Gbps       | 64              | $2^{64} = \text{approx. } 2 \cdot 10^{19}$ | approx. 31,000 years  |
| 100 Gbps      | 64              | $2^{64} = \text{approx. } 2 \cdot 10^{19}$ | approx. 3,100 years   |
| 1 Gbps        | 32              | $2^{32} = \text{approx. } 4 \cdot 10^9$    | 2,200 seconds         |
| 10 Gbps       | 32              | $2^{32} = \text{approx. } 4 \cdot 10^9$    | 220 seconds           |
| 100 Gbps      | 32              | $2^{32} = \text{approx. } 4 \cdot 10^9$    | 22 seconds            |

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

## 5. Node Configuration

A POT system consists of a number of nodes that participate in POT and a Controller, which serves as a control and configuration entity. The Controller is to create the required parameters (polynomials, prime number, etc.) and communicate those to the nodes. The sum of all parameters for a specific node is referred to as "POT-profile". This document does not define a specific protocol to be used between Controller and nodes. It only defines the procedures and the associated YANG data model.

### 5.1. Procedure

The Controller creates new POT-profiles at a constant rate and communicates the POT-profile to the nodes. The controller labels a POT-profile "even" or "odd" and the Controller cycles between "even" and "odd" labeled profiles. The rate at which the POT-profiles are communicated to the nodes is configurable and is more frequent than the speed at which a POT-profile is "used up" (see table above). Once the POT-profile has been successfully communicated to all nodes (e.g., all NETCONF transactions completed, in case NETCONF is used as a protocol), the controller sends an "enable POT-profile" request to the ingress node.

All nodes maintain two POT-profiles (an even and an odd POT-profile): One POT-profile is currently active and in use; one profile is standby and about to get used. A flag in the packet is indicating whether the odd or even POT-profile is to be used by a node. This is to ensure that during profile change the service is not disrupted. If the "odd" profile is active, the Controller can communicate the "even" profile to all nodes. Only if all the nodes have received the POT-profile, the Controller will tell the ingress node to switch to the "even" profile. Given that the indicator travels within the packet, all nodes will switch to the "even" profile. The "even" profile gets active on all nodes and nodes are ready to receive a new "odd" profile.

Unless the ingress node receives a request to switch profiles, it'll continue to use the active profile. If a profile is "used up" the ingress node will recycle the active profile and start over (this could give rise to replay attacks in theory - but with  $2^{32}$  or  $2^{64}$  packets this isn't really likely in reality).

### 5.2. YANG Model

This section defines that YANG data model for the information exchange between the Controller and the nodes.

```
<CODE BEGINS> file "ietf-pot-profile@2016-06-15.yang"
module ietf-pot-profile {

    yang-version 1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-pot-profile";

    prefix ietf-pot-profile;

    organization "IETF xxx Working Group";
```

```
contact "";

description "This module contains a collection of YANG
            definitions for proof of transit configuration
            parameters. The model is meant for proof of
            transit and is targeted for communicating the
            POT-profile between a controller and nodes
            participating in proof of transit.";

revision 2016-06-15 {
  description
    "Initial revision.";
  reference
    "";
}

typedef profile-index-range {
  type int32 {
    range "0 .. 1";
  }
  description
    "Range used for the profile index. Currently restricted to
    0 or 1 to identify the odd or even profiles.";
}

grouping pot-profile {
  description "A grouping for proof of transit profiles.";
  list pot-profile-list {
    key "pot-profile-index";
    ordered-by user;
    description "A set of pot profiles.";

    leaf pot-profile-index {
      type profile-index-range;
      mandatory true;
      description
        "Proof of transit profile index.";
    }

    leaf prime-number {
      type uint64;
      mandatory true;
      description
        "Prime number used for module math computation";
    }

    leaf secret-share {
```

```
        type uint64;
        mandatory true;
        description
            "Share of the secret of polynomial 1 used in computation";
    }

    leaf public-polynomial {
        type uint64;
        mandatory true;
        description
            "Pre evaluated Public polynomial";
    }

    leaf lpc {
        type uint64;
        mandatory true;
        description
            "Lagrange Polynomial Coefficient";
    }

    leaf validator {
        type boolean;
        default "false";
        description
            "True if the node is a verifier node";
    }

    leaf validator-key {
        type uint64;
        description
            "Secret key for validating the path, constant of poly 1";
    }

    leaf bitmask {
        type uint64;
        default 4294967295;
        description
            "Number of bits as mask used in controlling the size of the
            random value generation. 32-bits of mask is default.";
    }
}

container pot-profiles {
    description "A group of proof of transit profiles.";

    list pot-profile-set {
        key "pot-profile-name";
```

```
ordered-by user;
description
  "Set of proof of transit profiles that group parameters
   required to classify and compute proof of transit
   metadata at a node";

leaf pot-profile-name {
  type string;
  mandatory true;
  description
    "Unique identifier for each proof of transit profile";
}

leaf active-profile-index {
  type profile-index-range;
  description
    "Proof of transit profile index that is currently active.
     Will be set in the first hop of the path or chain.
     Other nodes will not use this field.";
}

uses pot-profile;
}
/**** Container: end ****/
}
/**** module: end ****/
}
<CODE ENDS>
```

## 6. IANA Considerations

IANA considerations will be added in a future version of this document.

## 7. Manageability Considerations

Manageability considerations will be addressed in a later version of this document.

## 8. Security Considerations

Different security requirements achieved by the solution approach are discussed here.

### 8.1. Proof of Transit

Proof of correctness and security of the solution approach is per Shamir's Secret Sharing Scheme [SSS]. Cryptographically speaking it achieves information-theoretic security i.e., it cannot be broken by an attacker even with unlimited computing power. As long as the below conditions are met it is impossible for an attacker to bypass one or multiple nodes without getting caught.

- o If there are  $k+1$  nodes in the path, the polynomials (POLY-1, POLY-2) should be of degree  $k$ . Also  $k+1$  points of POLY-1 are chosen and assigned to each node respectively. The verifier can re-construct the  $k$  degree polynomial (POLY-3) only when all the points are correctly retrieved.
- o Precisely three values are kept secret by individual nodes. Share of SECRET (i.e. points on POLY-1), Share of POLY-2, LPC, P. Note that only constant coefficient, RND, of POLY-2 is public.  $x$  values and non-constant coefficient of POLY-2 are secret

An attacker bypassing a few nodes will miss adding a respective point on POLY-1 to corresponding point on POLY-2, thus the verifier cannot construct POLY-3 for cross verification.

Also it is highly recommended that different polynomials should be used as POLY-1 across different paths, traffic profiles or service chains.

### 8.2. Cryptanalysis

A passive attacker could try to harvest the POT data (i.e., CML, RND values) in order to determine the configured secrets. Subsequently two types of differential analysis for guessing the secrets could be done.

- o Inter-Node: A passive attacker observing CML values across nodes (i.e., as the packets entering and leaving), cannot perform differential analysis to construct the points on POLY-1. This is because at each point there are four unknowns (i.e. Share(POLY-1), Share(Poly-2) LPC and prime number P) and three known values (i.e. RND, CML-before, CML-after).
- o Inter-Packets: A passive attacker could observe CML values across packets (i.e., values of PKT-1 and subsequent PKT-2), in order to predict the secrets. Differential analysis across packets could be mitigated using a good PRNG for generating RND. Note that if constant coefficient is a sequence number than CML values become quite predictable and the scheme would be broken.

### 8.3. Anti-Replay

A passive attacker could reuse a set of older RND and the intermediate CML values to bypass certain nodes in later packets. Such attacks could be avoided by carefully choosing POLY-2 as a  $(SEQ\_NO + RND)$ . For example, if 64 bits are being used for POLY-2 then first 16 bits could be a sequence number  $SEQ\_NO$  and next 48 bits could be a random number.

Subsequently, the verifier could use the  $SEQ\_NO$  bits to run classic anti-replay techniques like sliding window used in IPSEC. The verifier could buffer up to  $2^{16}$  packets as a sliding window. Packets arriving with a higher  $SEQ\_NO$  than current buffer could be flagged legitimate. Packets arriving with a lower  $SEQ\_NO$  than current buffer could be flagged as suspicious.

For all practical purposes in the rest of the document RND means  $SEQ\_NO + RND$  to keep it simple.

The solution discussed in this memo does not currently mitigate replay attacks. An anti-replay mechanism may be included in future versions of the solution.

### 8.4. Anti-Preplay

An active attacker could try to perform a man-in-the-middle (MITM) attack by extracting the POT of PKT-1 and using it in PKT-2. Subsequently attacker drops the PKT-1 in order to avoid duplicate POT values reaching the verifier. If the PKT-1 reaches the verifier, then this attack is same as Replay attacks discussed before.

Preplay attacks are possible since the POT metadata is not dependent on the packet fields. Below steps are recommended for remediation:

- o Ingress node and Verifier are configured with common pre shared key
- o Ingress node generates a Message Authentication Code (MAC) from packet fields using standard HMAC algorithm.
- o The left most bits of the output are truncated to desired length to generate RND. It is recommended to use a minimum of 32 bits.
- o The verifier regenerates the HMAC from the packet fields and compares with RND. To ensure the POT data is in fact that of the packet.



If an HMAC is used, an active attacker lacks the knowledge of the pre-shared key, and thus cannot launch preplay attacks.

The solution discussed in this memo does not currently mitigate prereplay attacks. A mitigation mechanism may be included in future versions of the solution.

#### 8.5. Anti-Tampering

An active attacker could not insert any arbitrary value for CML. This would subsequently fail the reconstruction of the POLY-3. Also an attacker could not update the CML with a previously observed value. This could subsequently be detected by using timestamps within the RND value as discussed above.

#### 8.6. Recycling

The solution approach is flexible for recycling long term secrets like POLY-1. All the nodes could be periodically updated with shares of new SECRET as best practice. The table above could be consulted for refresh cycles (see Section 4).

#### 8.7. Redundant Nodes and Failover

A "node" or "service" in terms of POT can be implemented by one or multiple physical entities. In case of multiple physical entities (e.g., for load-balancing, or business continuity situations - consider for example a set of firewalls), all physical entities which are implementing the same POT node are given that same share of the secret. This makes multiple physical entities represent the same POT node from an algorithm perspective.

#### 8.8. Controller Operation

The Controller needs to be secured given that it creates and holds the secrets, as need to be the nodes. The communication between Controller and the nodes also needs to be secured. As secure communication protocol such as for example NETCONF over SSH should be chosen for Controller to node communication.

The Controller only interacts with the nodes during the initial configuration and thereafter at regular intervals at which the operator chooses to switch to a new set of secrets. In case 64 bits are used for the data fields "CML" and "RND" which are carried within the data packet, the regular intervals are expected to be quite long (e.g., at 100 Gbps, a profile would only be used up after 3100 years) - see Section 4 above, thus even a "headless" operation without a Controller can be considered feasible. In such a case, the

Controller would only be used for the initial configuration of the POT-profiles.

#### 8.9. Verification Scope

The POT solution defined in this document verifies that a data-packet traversed or transited a specific set of nodes. From an algorithm perspective, a "node" is an abstract entity. It could be represented by one or multiple physical or virtual network devices, or is could be a component within a networking device or system. The latter would be the case if a forwarding path within a device would need to be securely verified.

##### 8.9.1. Node Ordering

POT using Shamir's secret sharing scheme as discussed in this document provides for a means to verify that a set of nodes has been visited by a data packet. It does not verify the order in which the data packet visited the nodes. In case the order in which a data packet traversed a particular set of nodes needs to be verified as well, alternate schemes that e.g., rely on "nested encryption" could to be considered.

##### 8.9.2. Stealth Nodes

The POT approach discussed in this document is to prove that a data packet traversed a specific set of "nodes". This set could be all nodes within a path, but could also be a subset of nodes in a path. Consequently, the POT approach isn't suited to detect whether "stealth" nodes which do not participate in proof-of-transit have been inserted into a path.

#### 9. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Erik Nordmark, and Andrew Yourtchenko for the comments and advice.

#### 10. References

##### 10.1. Normative References

[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>>.

[SSS] "Shamir's Secret Sharing", <[https://en.wikipedia.org/wiki/Shamir%27s\\_Secret\\_Sharing](https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing)>.

## 10.2. Informative References

[I-D.ietf-anima-autonomic-control-plane]  
Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-03 (work in progress), July 2016.

## Authors' Addresses

Frank Brockners  
Cisco Systems, Inc.  
Hansaallee 249, 3rd Floor  
DUESSELDORF, NORDRHEIN-WESTFALEN 40549  
Germany

Email: [fbrockne@cisco.com](mailto:fbrockne@cisco.com)

Shwetha Bhandari  
Cisco Systems, Inc.  
Cessna Business Park, Sarjapura Marathalli Outer Ring Road  
Bangalore, KARNATAKA 560 087  
India

Email: [shwethab@cisco.com](mailto:shwethab@cisco.com)

Sashank Dara  
Cisco Systems, Inc.  
Cessna Business Park, Sarjapura Marathalli Outer Ring Road  
BANGALORE, Bangalore, KARNATAKA 560 087  
INDIA

Email: [sadara@cisco.com](mailto:sadara@cisco.com)

Carlos Pignataro  
Cisco Systems, Inc.  
7200-11 Kit Creek Road  
Research Triangle Park, NC 27709  
United States

Email: [cpignata@cisco.com](mailto:cpignata@cisco.com)

John Leddy  
Comcast

Email: John\_Leddy@cable.comcast.com

Stephen Youell  
JP Morgan Chase  
25 Bank Street  
London E14 5JP  
United Kingdom

Email: stephen.youell@jpmorgan.com

David Mozes

Email: mosesster@gmail.com

Tal Mizrahi  
Marvell  
6 Hamada St.  
Yokneam 20692  
Israel

Email: talmi@marvell.com

Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: 4 May 2021

F. Brockners, Ed.  
Cisco  
S. Bhandari, Ed.  
Thoughtspot  
T. Mizrahi, Ed.  
Huawei Network.IO Innovation Lab  
S. Dara  
Seconize  
S. Youell  
JPMC  
31 October 2020

Proof of Transit  
draft-ietf-sfc-proof-of-transit-08

Abstract

Several technologies such as Traffic Engineering (TE), Service Function Chaining (SFC), and policy based routing are used to steer traffic through a specific, user-defined path. This document defines mechanisms to securely prove that traffic transited a defined path. These mechanisms allow to securely verify whether, within a given path, all packets traversed all the nodes that they are supposed to visit. This document specifies a data model to enable these mechanisms using YANG.

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 4 May 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                     |    |
|-----------------------------------------------------|----|
| 1. Introduction . . . . .                           | 3  |
| 2. Terminology . . . . .                            | 5  |
| 3. Proof of Transit . . . . .                       | 6  |
| 3.1. Basic Idea . . . . .                           | 6  |
| 3.2. Solution Approach . . . . .                    | 7  |
| 3.2.1. Setup . . . . .                              | 8  |
| 3.2.2. In Transit . . . . .                         | 8  |
| 3.2.3. Verification . . . . .                       | 9  |
| 3.3. Illustrative Example . . . . .                 | 9  |
| 3.3.1. Baseline . . . . .                           | 9  |
| 3.3.1.1. Secret Shares . . . . .                    | 9  |
| 3.3.1.2. Lagrange Polynomials . . . . .             | 10 |
| 3.3.1.3. LPC Computation . . . . .                  | 10 |
| 3.3.1.4. Reconstruction . . . . .                   | 10 |
| 3.3.1.5. Verification . . . . .                     | 11 |
| 3.3.2. Complete Solution . . . . .                  | 11 |
| 3.3.2.1. Random Polynomial . . . . .                | 11 |
| 3.3.2.2. Reconstruction . . . . .                   | 11 |
| 3.3.2.3. Verification . . . . .                     | 12 |
| 3.3.3. Solution Deployment Considerations . . . . . | 12 |
| 3.4. Operational Aspects . . . . .                  | 13 |
| 3.5. Ordered POT (OPOT) . . . . .                   | 13 |
| 4. Sizing the Data for Proof of Transit . . . . .   | 14 |
| 5. Node Configuration . . . . .                     | 16 |
| 5.1. Procedure . . . . .                            | 16 |
| 5.2. YANG Model for POT . . . . .                   | 17 |
| 5.2.1. Main Parameters . . . . .                    | 17 |
| 5.2.2. Tree Diagram . . . . .                       | 18 |
| 5.2.3. YANG Model . . . . .                         | 18 |
| 6. IANA Considerations . . . . .                    | 23 |
| 7. Security Considerations . . . . .                | 23 |
| 7.1. Proof of Transit . . . . .                     | 24 |
| 7.2. Cryptanalysis . . . . .                        | 24 |
| 7.3. Anti-Replay . . . . .                          | 25 |
| 7.4. Anti-Preplay . . . . .                         | 26 |
| 7.5. Tampering . . . . .                            | 26 |
| 7.6. Recycling . . . . .                            | 26 |

|                                             |    |
|---------------------------------------------|----|
| 7.7. Redundant Nodes and Failover . . . . . | 27 |
| 7.8. Controller Operation . . . . .         | 27 |
| 7.9. Verification Scope . . . . .           | 27 |
| 7.9.1. Node Ordering . . . . .              | 28 |
| 7.9.2. Stealth Nodes . . . . .              | 28 |
| 7.10. POT Yang module . . . . .             | 28 |
| 8. Acknowledgements . . . . .               | 29 |
| 9. Contributors . . . . .                   | 29 |
| 10. References . . . . .                    | 29 |
| 10.1. Normative References . . . . .        | 29 |
| 10.2. Informative References . . . . .      | 30 |
| Authors' Addresses . . . . .                | 31 |

## 1. Introduction

Several deployments use Traffic Engineering, policy routing, Segment Routing (SR), and Service Function Chaining (SFC) [RFC7665] to steer packets through a specific set of nodes. In certain cases, compliance policies require operators to prove that all packets that are supposed to follow a specific path are indeed being forwarded across an exact set of pre-determined nodes.

If a packet flow is supposed to go through a series of service functions or network nodes, it has to be proven that indeed all packets of the flow followed the path or service chain or collection of nodes specified by the policy. In case some packets of a flow weren't appropriately processed, a verification device should determine the policy violation and take corresponding actions corresponding to the policy (e.g., drop or redirect the packet, send an alert etc.) In today's deployments, the proof that a packet traversed a particular path or service chain is typically delivered in an indirect way: Service appliances and network forwarding are in different trust domains. Physical hand-off-points are defined between these trust domains (i.e. physical interfaces). Or in other terms, in the "network forwarding domain" things are wired up in a way that traffic is delivered to the ingress interface of a service appliance and received back from an egress interface of a service appliance. This "wiring" is verified and then trusted upon. The evolution to Network Function Virtualization (NFV) and modern service chaining concepts (using technologies such as Locator/ID Separation Protocol (LISP), Network Service Header (NSH), Segment Routing (SR), etc.) blurs the line between the different trust domains, because the hand-off-points are no longer clearly defined physical interfaces, but are virtual interfaces. As a consequence, different trust layers should not to be mixed in the same device. For an NFV scenario a different type of proof is required. Offering a proof that a packet indeed traversed a specific set of service functions or nodes allows operators to evolve from the above described indirect methods of proving that packets visit a predetermined set of nodes.

The solution approach presented in this document is based on a small portion of operational data added to every packet. This "in-situ" operational data is also referred to as "proof of transit data", or POT data. The POT data is updated at every required node and is used to verify whether a packet traversed all required nodes. A particular set of nodes "to be verified" is either described by a set of shares of a single secret. Nodes on the path retrieve their individual shares of the secret using Shamir's Secret Sharing scheme from a central controller. The complete secret set is only known to the controller and a verifier node, which is typically the ultimate node on a path that performs verification. Each node in the path uses its share of the secret to update the POT data of the packets as the packets pass through the node. When the verifier receives a packet, it uses its key along with data found in the packet to validate whether the packet traversed the path correctly. This document defines a data model and specifies it formally using the YANG 1.1 [RFC7950] data modeling language to support enabling the POT solution. The YANG data model defined in this document conforms to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].



Note to RFC Editor: Please replace the date 2020-09-09 in Section 5.2 of the draft with the date of publication of this draft as a RFC. Also, replace reference to RFC XXXX, and draft-ietf-sfc-proof-of-transit with the RFC numbers assigned to the drafts.

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP[RFC2119] [RFC8174]

Abbreviations used in this document:

HMAC: Hashed Message Authentication Code. For example, HMAC-SHA256 generates 256 bits of MAC

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

LPC: Lagrange Polynomial Constants

NFV: Network Function Virtualization

NSH: Network Service Header

POT: Proof of Transit

POT-Profile: Proof of Transit Profile that has the necessary data for nodes to participate in proof of transit

RND: Random Bits generated per packet. Packet fields that do not change during the traversal are given as input to HMAC-256 algorithm. A minimum of 32 bits (left most) need to be used from the output if RND is used to verify the packet integrity. This is a standard recommendation by NIST.

SEQ\_NO: Sequence number initialized to a predefined constant. This is used in concatenation with RND bits to mitigate different attacks discussed later.

SFC: Service Function Chain

SSSS: Shamir's Secret Sharing Scheme

SR: Segment Routing

### 3. Proof of Transit

This section discusses methods and algorithms to provide for a "proof of transit" (POT) for packets traversing a specific path. A path which is to be verified consists of a set of nodes. Transit of the data packets through those nodes is to be proven. Besides the nodes, the setup also includes a Controller that creates secrets and secrets shares and configures the nodes for POT operations.

The methods how traffic is identified and associated to a specific path is outside the scope of this document. Identification could be done using a filter (e.g., 5-tuple classifier), or an identifier which is already present in the packet (e.g., path or service identifier, NSH Service Path Identifier (SPI), flow-label, etc.)

When used in the context of IOAM, the POT information MUST be encapsulated in packets as an IOAM Proof of Transit Option-Type. The details and format of the encapsulation and the IOAM POT Option-Type format are specified in [I-D.ietf-ippm-ioam-data]. When used in conjunction with NSH [RFC8300], the POT Option-Type MUST be carried as specified in [I-D.ietf-sfc-ioam-nsh].

The solution approach is detailed in two steps. Initially the concept of the approach is explained. This concept is then further refined to make it operationally feasible.

#### 3.1. Basic Idea

The method relies on adding POT data to all packets that traverse a path. The added POT data allows a verifying node (egress node) to check whether a packet traversed the identified set of nodes on a path correctly or not. Security mechanisms are natively built into the generation of the POT data to protect against misuse (e.g., configuration mistakes). The mechanism for POT leverages "Shamir's Secret Sharing" scheme [SSS].

Shamir's Secret Sharing base idea: A polynomial (represented by its coefficients) of degree  $k$  is chosen as a secret by the controller. A polynomial represents a curve. A set of  $k+1$  points on the curve define the polynomial and are thus needed to (re-)construct the polynomial. Each of these  $k+1$  points of the polynomial is called a "share" of the secret. A single secret is associated with a particular set of  $k+1$  nodes, which typically represent the path to be verified.  $k+1$  shares of the single secret (i.e.,  $k+1$  points on the curve) are securely distributed from a Controller to the network nodes. Nodes use their respective share to update a cumulative value in the POT data of each packet. Only a verifying node has access to the complete secret. The verifying node validates the correctness of the received POT data by reconstructing the curve.

The polynomial cannot be reconstructed if any of the points are missed or tampered. Per Shamir's Secret Sharing Scheme, any lesser points means one or more nodes are missed. Details of the precise configuration needed for achieving security are discussed further below.

While applicable in theory, a vanilla approach based on Shamir's Secret Sharing Scheme could be easily attacked. If the same polynomial is reused for every packet for a path a passive attacker could reuse the value. As a consequence, one could consider creating a different polynomial per packet. Such an approach would be operationally complex. It would be complex to configure and recycle so many curves and their respective points for each node. Rather than using a single polynomial, two polynomials are used for the solution approach: A secret polynomial as described above which is kept constant, and a per-packet polynomial which is public and generated by the ingress node (the first node along the path). Operations are performed on the sum of those two polynomials - creating a third polynomial which is secret and per packet.

### 3.2. Solution Approach

Solution approach: The overall algorithm uses two polynomials: POLY-1 and POLY-2. POLY-1 is secret and constant. A different POLY-1 is used for each path, and its value is known to the controller and to the verifier of the respective path. Each node gets a point on POLY-1 at setup-time and keeps it secret. POLY-2 is public, random and per packet. Each node generates a point on POLY-2 each time a packet crosses it. Each node then calculates (point on POLY-1 + point on POLY-2) to get a (point on POLY-3) and passes it to verifier by adding it to each packet. The verifier constructs POLY-3 from the points given by all the nodes and cross checks whether  $\text{POLY-3} = \text{POLY-1} + \text{POLY-2}$ . Only the verifier knows POLY-1.

The solution leverages finite field arithmetic in a field of size "prime number", i.e. all operations are performed "modulo prime number".

Detailed algorithms are discussed next. A simple example that describes how the algorithms work is discussed in Section 3.3.

The algorithms themselves do not constrain the ranges of possible values for the different parameters and coefficients used. A deployment of the algorithms will always need to define appropriate ranges. Please refer to the YANG model in Section 5.2 for details on the units and ranges of possible values of the different parameters and coefficients.

### 3.2.1. Setup

A controller generates a first polynomial (POLY-1) of degree  $k$  and  $k+1$  points on the polynomial, corresponding to the  $k+1$  nodes along the path. The constant coefficient of POLY-1 is considered the SECRET, which is per the definition of the Shamir's Secret Sharing algorithm [SSS]. The  $k+1$  points are used to derive the Lagrange Basis Polynomials. The Lagrange Polynomial Constants (LPC) are retrieved from the constant coefficients of the Lagrange Basis Polynomials. Each of the  $k+1$  nodes (including verifier) are assigned a point on the polynomial i.e., shares of the SECRET. The verifier is configured with the SECRET. The Controller also generates coefficients (except the constant coefficient, called "RND", which is changed on a per packet basis) of a second polynomial POLY-2 of the same degree. Each node is configured with the LPC of POLY-2. Note that POLY-2 is public.

### 3.2.2. In Transit

For each packet, the ingress node generates a random number (RND). It is considered as the constant coefficient for POLY-2. A cumulative value (CML) is initialized to 0. Both RND, CML are carried as within the packet POT data. As the packet visits each node, the RND is retrieved from the packet and the respective share of POLY-2 is calculated. Each node calculates  $(\text{Share}(\text{POLY-1}) + \text{Share}(\text{POLY-2}))$  and CML is updated with this sum, specifically each node performs

$$\text{CML} = \text{CML} + ((\text{Share}(\text{POLY-1}) + \text{Share}(\text{POLY-2})) * \text{LPC}) \bmod \text{Prime}$$
, with "LPC" being the Lagrange Polynomial Constant and "Prime" being the prime number which defines the finite field arithmetic that all operations are done over. Please also refer to Section 3.3.2 below for further details how the operations are performed.

This step is performed by each node until the packet completes the path. The verifier also performs the step with its respective share.

### 3.2.3. Verification

The verifier cross checks whether  $CML = SECRET + RND$ . If this matches then the packet traversed the specified set of nodes in the path. This is due to the additive homomorphic property of Shamir's Secret Sharing scheme.

### 3.3. Illustrative Example

This section shows a simple example to illustrate step by step the approach described above. The example assumes a network with 3 nodes. The last node that packets traverse also serves as the verifier. A Controller communicates the required parameters to the individual nodes.

#### 3.3.1. Baseline

Assumption: It is to be verified whether packets passed through the 3 nodes. A polynomial of degree 2 is chosen for verification.

Choices: Prime = 53.  $POLY-1(x) = (3x^2 + 3x + 10) \bmod 53$ . The secret to be re-constructed is the constant coefficient of  $POLY-1$ , i.e., SECRET=10. It is important to note that all operations are done over a finite field (i.e., modulo Prime = 53).

##### 3.3.1.1. Secret Shares

The shares of the secret are the points on  $POLY-1$  chosen for the 3 nodes. For example, let  $x_0=2$ ,  $x_1=4$ ,  $x_2=5$ .

$$POLY-1(2) = 28 \Rightarrow (x_0, y_0) = (2, 28)$$

$$POLY-1(4) = 17 \Rightarrow (x_1, y_1) = (4, 17)$$

$$POLY-1(5) = 47 \Rightarrow (x_2, y_2) = (5, 47)$$

The three points above are the points on the curve which are considered the shares of the secret. They are assigned by the Controller to three nodes respectively and are kept secret.

## 3.3.1.2. Lagrange Polynomials

Lagrange basis polynomials (or Lagrange polynomials) are used for polynomial interpolation. For a given set of points on the curve Lagrange polynomials (as defined below) are used to reconstruct the curve and thus reconstruct the complete secret.

$$\begin{aligned} l_0(x) &= (((x-x_1) / (x_0-x_1)) * ((x-x_2)/(x_0-x_2))) \bmod 53 \\ &= (((x-4) / (2-4)) * ((x-5)/(2-5))) \bmod 53 \\ &= (10/3 - 3x/2 + (1/6)x^2) \bmod 53 \end{aligned}$$

$$\begin{aligned} l_1(x) &= (((x-x_0) / (x_1-x_0)) * ((x-x_2)/(x_1-x_2))) \bmod 53 \\ &= (-5 + 7x/2 - (1/2)x^2) \bmod 53 \end{aligned}$$

$$\begin{aligned} l_2(x) &= (((x-x_0) / (x_2-x_0)) * ((x-x_1)/(x_2-x_1))) \bmod 53 \\ &= (8/3 - 2 + (1/3)x^2) \bmod 53 \end{aligned}$$

## 3.3.1.3. LPC Computation

Since  $x_0=2$ ,  $x_1=4$ ,  $x_2=5$  are chosen points. Given that computations are done over a finite arithmetic field ("modulo a prime number"), the Lagrange basis polynomial constants are computed modulo 53. The Lagrange Polynomial Constants (LPC) would be  $\bmod(10/3, 53)$ ,  $\bmod(-5, 53)$ ,  $\bmod(8/3, 53)$ . LPC are computed by the Controller and communicated to the individual nodes.

$$\text{LPC}(l_0) = (10/3) \bmod 53 = 21$$

$$\text{LPC}(l_1) = (-5) \bmod 53 = 48$$

$$\text{LPC}(l_2) = (8/3) \bmod 53 = 38$$

For a general way to compute the modular multiplicative inverse, see e.g., the Euclidean algorithm.

## 3.3.1.4. Reconstruction

Reconstruction of the polynomial is well-defined as

$$\text{POLY}_1(x) = l_0(x) * y_0 + l_1(x) * y_1 + l_2(x) * y_2$$

Subsequently, the SECRET, which is the constant coefficient of  $\text{POLY}_1(x)$  can be computed as below

$$\text{SECRET} = (y_0 * \text{LPC}(l_0) + y_1 * \text{LPC}(l_1) + y_2 * \text{LPC}(l_2)) \bmod 53$$

The secret can be easily reconstructed using the y-values and the LPC:

```
SECRET = (y0*LPC(l0) + y1*LPC(l1) + y2*LPC(l2)) mod 53
        = (28 * 21 + 17 * 48 + 47 * 38) mod 53
        = 3190 mod 53
        = 10
```

One observes that the secret reconstruction can easily be performed cumulatively hop by hop, i.e. by every node. CML represents the cumulative value. It is the POT data in the packet that is updated at each hop with the node's respective  $(y_i * \text{LPC}(i))$ , where  $i$  is their respective value.

#### 3.3.1.5. Verification

Upon completion of the path, the resulting CML is retrieved by the verifier from the packet POT data. Recall that the verifier is preconfigured with the original SECRET. It is cross checked with the CML by the verifier. Subsequent actions based on the verification failing or succeeding could be taken as per the configured policies.

#### 3.3.2. Complete Solution

As observed previously, the baseline algorithm that involves a single secret polynomial is not secure. The complete solution leverages a random second polynomial, which is chosen per packet.

##### 3.3.2.1. Random Polynomial

Let the second polynomial POLY-2 be  $(\text{RND} + 7x + 10x^2)$ . RND is a random number and is generated for each packet. Note that POLY-2 is public and need not be kept secret. The nodes can be pre-configured with the non-constant coefficients (for example, 7 and 10 in this case could be configured through the Controller on each node). So precisely only the RND value changes per packet and is public and the rest of the non-constant coefficients of POLY-2 is kept secret.

##### 3.3.2.2. Reconstruction

Recall that each node is preconfigured with their respective  $\text{Share}(\text{POLY-1})$ . Each node calculates its respective  $\text{Share}(\text{POLY-2})$  using the RND value retrieved from the packet. The CML reconstruction is enhanced as below. At every node, CML is updated as

$$\text{CML} = \text{CML} + (((\text{Share}(\text{POLY-1}) + \text{Share}(\text{POLY-2})) * \text{LPC}) \bmod \text{Prime})$$

Let us observe the packet level transformations in detail. For the example packet here, let the value RND be 45. Thus POLY-2 would be  $(45 + 7x + 10x^2)$ .

The shares that could be generated are (2, 46), (4, 21), (5, 12).

At ingress: The fields RND = 45. CML = 0.

At node-1 (x0): Respective share of POLY-2 is generated i.e., (2, 46) because share index of node-1 is 2.

$CML = 0 + ((28 + 46) * 21) \bmod 53 = 17$

At node-2 (x1): Respective share of POLY-2 is generated i.e., (4, 21) because share index of node-2 is 4.

$CML = 17 + ((17 + 21) * 48) \bmod 53 = 17 + 22 = 39$

At node-3 (x2), which is also the verifier: The respective share of POLY-2 is generated i.e., (5, 12) because the share index of the verifier is 12.

$CML = 39 + ((47 + 12) * 38) \bmod 53 = 39 + 16 = 55 \bmod 53 = 2$

The verification using CML is discussed in next section.

### 3.3.2.3. Verification

As shown in the above example, for final verification, the verifier compares:

$VERIFY = (SECRET + RND) \bmod Prime$ , with Prime = 53 here

$VERIFY = (RND-1 + RND-2) \bmod Prime = (10 + 45) \bmod 53 = 2$

Since  $VERIFY = CML$  the packet is proven to have gone through nodes 1, 2, and 3.

### 3.3.3. Solution Deployment Considerations

The "complete solution" described above in Section 3.3.2 could still be prone to replay or preplay attacks. An attacker could e.g. reuse the POT metadata for bypassing the verification. These threats can be mitigated by appropriate parameterization of the algorithm. Please refer to Section 7 for details.



### 3.4. Operational Aspects

To operationalize this scheme, a central controller is used to generate the necessary polynomials, the secret share per node, the prime number, etc. and distributing the data to the nodes participating in proof of transit. The identified node that performs the verification is provided with the verification key. The information provided from the Controller to each of the nodes participating in proof of transit is referred to as a proof of transit profile (POT-Profile). Also note that the set of nodes for which the transit has to be proven are typically associated to a different trust domain than the verifier. Note that building the trust relationship between the Controller and the nodes is outside the scope of this document. Techniques such as those described in [I-D.ietf-anima-autonomic-control-plane] might be applied.

To optimize the overall data amount of exchanged and the processing at the nodes the following optimizations are performed:

1. The points (x, y) for each of the nodes on the public and private polynomials are picked such that the x component of the points match. This lends to the LPC values which are used to calculate the cumulative value CML to be constant. Note that the LPC are only depending on the x components. They can be computed at the controller and communicated to the nodes. Otherwise, one would need to distributed the x components to all the nodes.
2. A pre-evaluated portion of the public polynomial for each of the nodes is calculated and added to the POT-Profile. Without this all the coefficients of the public polynomial had to be added to the POT profile and each node had to evaluate them. As stated before, the public portion is only the constant coefficient RND value, the pre-evaluated portion for each node should be kept secret as well.
3. To provide flexibility on the size of the cumulative and random numbers carried in the POT data a field to indicate this is shared and interpreted at the nodes.

### 3.5. Ordered POT (OPOT)

POT as discussed in this document so far only verifies that a defined set of nodes have been traversed by a packet. The order in which nodes where traversed is not verified. "Ordered Proof of Transit (OPOT)" addresses the need of deployments, that require to verify the order in which nodes were traversed. OPOT extends the POT scheme with symmetric masking between the nodes.

1. For each path the controller provisions all the nodes with (or asks them to agree on) two secrets per node, that we will refer to as masks, one for the connection from the upstream node(s), another for the connection to the downstream node(s). For obvious reasons, the ingress and egress (verifier) nodes only receive one, for downstream and upstream, respectively.
2. Any two contiguous nodes in the OPOT stream share the mask for the connection between them, in the shape of symmetric keys. Masks can be refreshed as per-policy, defined at each hop or globally by the controller.
3. Each mask has the same size in bits as the length assigned to CML plus RND, as described in the above sections.
4. Whenever a packet is received at an intermediate node, the CML+RND sequence is deciphered (by XORing, though other ciphering schemas MAY be possible) with the upstream mask before applying the procedures described in Section 3.3.2.
5. Once the new values of CML+RND are produced, they are ciphered (by XORing, though other ciphering schemas MAY be possible) with the downstream mask before transmitting the packet to the next node downstream.
6. The ingress node only applies step 5 above, while the verifier only applies step 4 before running the verification procedure.

The described process allows the verifier to check if the packet has followed the correct order while traversing the path. In particular, the reconstruction process will fail if the order is not respected, as the deciphering process will produce invalid CML and RND values, and the interpolation (secret reconstruction) will finally generate a wrong verification value.

This procedure does not impose a high computational burden, does not require additional packet overhead, can be deployed on chains of any length, does not require any node to be aware of any additional information than the upstream and downstream masks, and can be integrated with the other operational mechanisms applied by the controller to distribute shares and other secret material.

#### 4. Sizing the Data for Proof of Transit

Proof of transit requires transport of two data fields in every packet that should be verified:

1. RND: Random number (the constant coefficient of public polynomial)
2. CML: Cumulative

The size of the data fields determines how often a new set of polynomials would need to be created. At maximum, the largest RND number that can be represented with a given number of bits determines the number of unique polynomials POLY-2 that can be created. The table below shows the maximum interval for how long a single set of polynomials could last for a variety of bit rates and RND sizes: When choosing 64 bits for RND and CML data fields, the time between a renewal of secrets could be as long as 3,100 years, even when running at 100 Gbps.

| Transfer rate | Secret/RND size | Max # of packets                            | Time RND lasts        |
|---------------|-----------------|---------------------------------------------|-----------------------|
| 1 Gbps        | 64              | $2^{64} = \text{approx. } 2 \times 10^{19}$ | approx. 310,000 years |
| 10 Gbps       | 64              | $2^{64} = \text{approx. } 2 \times 10^{19}$ | approx. 31,000 years  |
| 100 Gbps      | 64              | $2^{64} = \text{approx. } 2 \times 10^{19}$ | approx. 3,100 years   |
| 1 Gbps        | 32              | $2^{32} = \text{approx. } 4 \times 10^9$    | 2,200 seconds         |
| 10 Gbps       | 32              | $2^{32} = \text{approx. } 4 \times 10^9$    | 220 seconds           |
| 100 Gbps      | 32              | $2^{32} = \text{approx. } 4 \times 10^9$    | 22 seconds            |

Table 1: Proof of transit data sizing

Table assumes 64 octet packets

If the symmetric masking method for ordered POT is used (Section 3.5), the masks used between nodes adjacent in the path MUST have a length equal to the sum of the ones of RND and CML.

## 5. Node Configuration

A POT system consists of a number of nodes that participate in POT and a Controller, which serves as a control and configuration entity. The Controller is to create the required parameters (polynomials, prime number, etc.) and communicate the associated values (i.e. prime number, secret-share, LPC, etc.) to the nodes. The sum of all parameters for a specific node is referred to as "POT-Profile". For details see the YANG model in Section 5.2. This document defines the procedures and the associated YANG data model.

### 5.1. Procedure

The Controller creates new POT-Profiles at a constant rate and communicates the POT-Profile to the nodes. The controller labels a POT-Profile "even" or "odd" and the Controller cycles between "even" and "odd" labeled profiles. This means that the parameters for the algorithms are continuously refreshed. Please refer to Section 4 for choosing an appropriate refresh rate: The rate at which the POT-Profiles are communicated to the nodes is configurable and MUST be more frequent than the speed at which a POT-Profile is "used up". Once the POT-Profile has been successfully communicated to all nodes (e.g., all NETCONF transactions completed, in case NETCONF is used as a protocol), the controller sends an "enable POT-Profile" request to the ingress node.

All nodes maintain two POT-Profiles (an even and an odd POT-Profile): One POT-Profile is currently active and in use; one profile is standby and about to get used. A flag in the packet is indicating whether the odd or even POT-Profile is to be used by a node. This is to ensure that during profile change the service is not disrupted. If the "odd" profile is active, the Controller can communicate the "even" profile to all nodes. Only if all the nodes have received the POT-Profile, the Controller will tell the ingress node to switch to the "even" profile. Given that the indicator travels within the packet, all nodes will switch to the "even" profile. The "even" profile gets active on all nodes and nodes are ready to receive a new "odd" profile.

Unless the ingress node receives a request to switch profiles, it'll continue to use the active profile. If a profile is "used up" the ingress node will recycle the active profile and start over (this could give rise to replay attacks in theory - but with  $2^{32}$  or  $2^{64}$  packets this isn't really likely in reality).

## 5.2. YANG Model for POT

This section defines that YANG data model for the information exchange between the Controller and the node.

### 5.2.1. Main Parameters

The main parameters for the information exchange between the Controller and the node used in the YANG model are as follows:

- \* `pot-profile-index`: Section 5.1 details that two POT-Profiles are used. Only one of the POT-Profiles is active at a given point in time, allowing the Controller to refresh the non-active one for future use. `pot-profile-index` defines which of the POT-Profiles (the "even" or "odd" POT-Profile) is currently active. `pot-profile-index` will be set in the first hop of the path or chain. Other nodes will not use this field.
- \* `prime-number`: Prime number used for module math computation.
- \* `secret-share`: Share of the secret of polynomial-1 used in computation for the node. If POLY-1 is defined by points  $(x1_i, y1_i)$  with  $i=0, \dots, k$ , then for node  $i$ , the secret-share will be  $y1_i$ .
- \* `public-polynomial`: Public polynomial value for the node.. If POLY-2 is defined by points  $(x2_i, y2_i)$  with  $i=0, \dots, k$ , then for node  $i$ , the secret-share will be  $y2_i$ .
- \* `lpc`: Lagrange Polynomial Coefficient for the node, i.e. for node  $i$ , this would be  $LPC(l_i)$ , with  $l_i$  being the  $i$ -th Lagrange Basis Polynomial.
- \* `validator`: True if the node is a verifier node.
- \* `validator-key`: The validator-key represents the SECRET as described in the sections above. The SECRET is the constant coefficient of  $POLY-1(z)$ . If  $POLY-1(z) = a_0 + a_1*z + a_2*z^2 + \dots + a_k*z^k$ , then the SECRET would be  $a_0$ .
- \* `bitmask`: Number of bits as mask used in controlling the size of the random value generation. 32-bits of mask is default. See Section 4 for details.

### 5.2.2. Tree Diagram

This section shows a simplified graphical representation of the YANG data model for POT. The meaning of the symbols in YANG tree diagrams is defined in [RFC8340].

```

module: ietf-pot-profile
  +--rw pot-profiles
    +--rw pot-profile-set* [pot-profile-name]
      +--rw pot-profile-name      string
      +--rw pot-profile-list* [pot-profile-index]
        +--rw pot-profile-index  profile-index-range
        +--rw status?            boolean
        +--rw prime-number       uint64
        +--rw secret-share       uint64
        +--rw public-polynomial  uint64
        +--rw lpc                uint64
        +--rw validator?         boolean
        +--rw validator-key?     uint64
        +--rw bitmask?          uint64
      +--rw opot-masks
        +--rw downstream-mask*  uint64
        +--rw upstream-mask*    uint64

```

### 5.2.3. YANG Model

```

<CODE BEGINS> file "ietf-pot-profile@2020-09-08.yang"
module ietf-pot-profile {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-pot-profile";

  prefix "pot";

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

  organization "IETF SFC Working Group";

  contact "WG Web:  <https://tools.ietf.org/wg/sfc/>
          WG List:  <mailto:sfc@ietf.org>
          Author   : Frank Brockners <fbrockne@cisco.com>
          Author   : Shwetha Bhandari <shwethab@cisco.com>
          Author   : Tal Mizrahi <tal.mizrahi.phd@gmail.com>";

```

description

"This module contains a collection of YANG definitions for proof of transit configuration parameters. The model is meant for proof of transit and is targeted for communicating the POT-Profile between a controller and nodes participating in proof of transit.

Copyright (c) 2020 IETF Trust and the persons identified as authors of the code. All rights reserved. Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.";

```
revision "2020-09-08" {  
  description  
    "Initial revision.";  
  reference  
    "RFC XXXX: Proof of Transit";  
}
```

```
typedef profile-index-range {  
  type int32 {  
    range "0 .. 1";  
  }  
  description  
    "Range used for the profile index. Currently restricted to  
    0 or 1 to identify the odd or even profiles.";  
}
```

```
grouping pot-profile {
  description "A grouping for proof of transit profiles.";
  list pot-profile-list {
    key "pot-profile-index";
    ordered-by user;
    description "A set of pot profiles.";

    leaf pot-profile-index {
      type profile-index-range;
      mandatory true;
      description
        "Proof of transit profile index.";
    }

    leaf status {
      type boolean;
      default "false";
      description
        "True if this profile is currently active.
        Will be used by the first hop of the path or chain.
        Other nodes will not use this field.";
    }

    leaf prime-number {
      type uint64;
      nacm:default-deny-all;
      mandatory true;
      description
        "Prime number used for module math computation";
    }

    leaf secret-share {
      type uint64;
      nacm:default-deny-all;
      mandatory true;
      description
        "Share of the secret of polynomial-1 used
        in computation for the node. If POLY-1
        is defined by points (x1_i, y1_i) with
        i=0,..k, then for node i, the secret-share
        will be y1_i.";
    }

    leaf public-polynomial {
      type uint64;
      mandatory true;
      description
        "Public polynomial value for the node."
    }
  }
}
```



```
        If POLY-2 is defined by points (x2_i, y2_i)
        with i=0,..k, then for node i,
        the secret-share will be y2_i.";
    }

    leaf lpc {
        type uint64;
        mandatory true;
        description
            "Lagrange Polynomial Coefficient";
    }

    leaf validator {
        type boolean;
        default "false";
        description
            "True if the node is a verifier node";
    }

    leaf validator-key {
        type uint64;
        nacm:default-deny-all;
        description
            "The validator-key represents the secret.
            The secret is the constant coefficient of
            POLY-1(z). If POLY-1(z) =
            a_0 + a_1*z + a_2*z^2+..+a_k*z^k,
            then the SECRET would be a_0.";
    }

    leaf bitmask {
        type uint64;
        default 4294967295;
        description
            "Number of bits as mask used in controlling
            the size of the random value generation.
            32-bits of mask is default.";
    }

    uses opot-profile;
}

grouping opot-profile {
    description "Grouping containing OPoT related data.";
```

```
container opot-masks {
    must "count(downstream-mask) = count(upstream-mask)";
    description "Masking information for OPoT support.";

    leaf-list downstream-mask {
        type uint64;
        max-elements 2;
        description "Secret stream used to demask the PoT metadata.
        The mask is used between nodes adjacent in the path
        and MUST have a length equal to the sum of the ones
        of RND and CML.";
    }

    leaf-list upstream-mask {
        type uint64;
        max-elements 2;
        description "Secret stream used to mask the PoT metadata.
        The mask is used between nodes adjacent in the path
        and MUST have a length equal to the sum of the ones
        of RND and CML.";
    }
}

container pot-profiles {
    description "A group of proof of transit profiles.";

    list pot-profile-set {
        key "pot-profile-name";
        ordered-by user;
        description
            "Set of proof of transit profiles that group parameters
            required to classify and compute proof of transit
            metadata at a node";

        leaf pot-profile-name {
            type string;
            mandatory true;
            description
                "Unique identifier for each proof of transit profile";
        }

        uses pot-profile;
    }
}
/**** Container: end ****/
}
/**** module: end ****/
}
```

<CODE ENDS>

## 6. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in IETF XML Registry [RFC3688], the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-pot-profile

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

IANA is requested to register the following YANG module in the "YANG Module Names" subregistry [RFC7950] within the "YANG Parameters" registry.

Name: ietf-pot-profile

Maintained by IANA: N

Namespace: urn:ietf:params:xml:ns:yang:ietf-pot-profile

Prefix: pot

Reference: RFC XXXX

## 7. Security Considerations

POT is a mechanism that is used for verifying the path through which a packet was forwarded. The security considerations of IOAM in general are discussed in [I-D.ietf-ippm-ioam-data]. Specifically, it is assumed that POT is used in a confined network domain, and therefore the potential threats that POT is intended to mitigate should be viewed accordingly. POT prevents spoofing and tampering; an attacker cannot maliciously create a bogus POT or modify a legitimate one. Furthermore, a legitimate node that takes part in the POT protocol cannot masquerade as another node along the path. These considerations are discussed in detail in the rest of this section.

### 7.1. Proof of Transit

Proof of correctness and security of the solution approach is per Shamir's Secret Sharing Scheme [SSS]. Cryptographically speaking it achieves information-theoretic security i.e., it cannot be broken by an attacker even with unlimited computing power. As long as the below conditions are met it is impossible for an attacker to bypass one or multiple nodes without getting caught.

- \* If there are  $k+1$  nodes in the path, the polynomials (POLY-1, POLY-2) should be of degree  $k$ . Also  $k+1$  points of POLY-1 are chosen and assigned to each node respectively. The verifier can re-construct the  $k$  degree polynomial (POLY-3) only when all the points are correctly retrieved.
- \* Precisely three values are kept secret by individual nodes. Share of SECRET (i.e. points on POLY-1), Share of POLY-2, LPC, P. Note that only constant coefficient, RND, of POLY-2 is public.  $x$  values and non-constant coefficient of POLY-2 are secret

An attacker bypassing a few nodes will miss adding a respective point on POLY-1 to corresponding point on POLY-2, thus the verifier cannot construct POLY-3 for cross verification.

Also it is highly recommended that different polynomials should be used as POLY-1 across different paths, traffic profiles or service chains.

If symmetric masking is used to assure OPOT (Section 3.5), the nodes need to keep two additional secrets: the downstream and upstream masks, that have to be managed under the same conditions as the secrets mentioned above. And it is equally recommended to employ a different set of mask pairs across different paths, traffic profiles or service chains.

### 7.2. Cryptanalysis

A passive attacker could try to harvest the POT data (i.e., CML, RND values) in order to determine the configured secrets. Subsequently two types of differential analysis for guessing the secrets could be done.

- \* Inter-Node: A passive attacker observing CML values across nodes (i.e., as the packets entering and leaving), cannot perform differential analysis to construct the points on POLY-1. This is because at each point there are four unknowns (i.e. Share(POLY-1), Share(Poly-2) LPC and prime number P) and three known values (i.e. RND, CML-before, CML-after). The application of symmetric masking for OPOT makes inter-node analysis less feasible.
- \* Inter-Packets: A passive attacker could observe CML values across packets (i.e., values of PKT-1 and subsequent PKT-2), in order to predict the secrets. Differential analysis across packets could be mitigated using a good PRNG for generating RND. Note that if constant coefficient is a sequence number than CML values become quite predictable and the scheme would be broken. If symmetric masking is used for OPOT, inter-packet analysis could be applied to guess mask values, which requires a proper refresh rate for masks, at least as high as the one used for LPCs.

### 7.3. Anti-Replay

A passive attacker could reuse a set of older RND and the intermediate CML values. Thus, an attacker can attack an old (replayed) RND and CML with a new packet in order to bypass some of the nodes along the path.

Such attacks could be avoided by carefully choosing POLY-2 as a (SEQ\_NO + RND). For example, if 64 bits are being used for POLY-2 then first 16 bits could be a sequence number SEQ\_NO and next 48 bits could be a random number.

Subsequently, the verifier could use the SEQ\_NO bits to run classic anti-replay techniques like sliding window used in IPSEC. The verifier could buffer up to  $2^{16}$  packets as a sliding window. Packets arriving with a higher SEQ\_NO than current buffer could be flagged legitimate. Packets arriving with a lower SEQ\_NO than current buffer could be flagged as suspicious.

For all practical purposes in the rest of the document RND means SEQ\_NO + RND to keep it simple.

The solution discussed in this memo does not currently mitigate replay attacks. An anti-replay mechanism may be included in future versions of the solution.

#### 7.4. Anti-Preplay

An active attacker could try to perform a man-in-the-middle (MITM) attack by extracting the POT of PKT-1 and using it in PKT-2. Subsequently attacker drops the PKT-1 in order to avoid duplicate POT values reaching the verifier. If the PKT-1 reaches the verifier, then this attack is same as Replay attacks discussed before.

Preplay attacks are possible since the POT metadata is not dependent on the packet fields. Below steps are recommended for remediation:

- \* Ingress node and Verifier are configured with common pre shared key
- \* Ingress node generates a Message Authentication Code (MAC) from packet fields using standard HMAC algorithm.
- \* The left most bits of the output are truncated to desired length to generate RND. It is recommended to use a minimum of 32 bits.
- \* The verifier regenerates the HMAC from the packet fields and compares with RND. To ensure the POT data is in fact that of the packet.

If an HMAC is used, an active attacker lacks the knowledge of the pre-shared key, and thus cannot launch preplay attacks.

The solution discussed in this memo does not currently mitigate preplay attacks. A mitigation mechanism may be included in future versions of the solution.

#### 7.5. Tampering

An active attacker could not insert any arbitrary value for CML. This would subsequently fail the reconstruction of the POLY-3. Also an attacker could not update the CML with a previously observed value. This could subsequently be detected by using timestamps within the RND value as discussed above.

#### 7.6. Recycling

The solution approach is flexible for recycling long term secrets like POLY-1. All the nodes could be periodically updated with shares of new SECRET as best practice. The table above could be consulted for refresh cycles (see Section 4).

If symmetric masking is used for OPOT (Section 3.5), mask values must be periodically updated as well, at least as frequently as the other secrets are.

#### 7.7. Redundant Nodes and Failover

A "node" or "service" in terms of POT can be implemented by one or multiple physical entities. In case of multiple physical entities (e.g., for load-balancing, or business continuity situations - consider for example a set of firewalls), all physical entities which are implementing the same POT node are given that same share of the secret. This makes multiple physical entities represent the same POT node from an algorithm perspective.

#### 7.8. Controller Operation

The Controller needs to be secured given that it creates and holds the secrets, as need to be the nodes. The communication between Controller and the nodes also needs to be secured. As secure communication protocol such as for example NETCONF over SSH should be chosen for Controller to node communication.

The Controller only interacts with the nodes during the initial configuration and thereafter at regular intervals at which the operator chooses to switch to a new set of secrets. In case 64 bits are used for the data fields "CML" and "RND" which are carried within the data packet, the regular intervals are expected to be quite long (e.g., at 100 Gbps, a profile would only be used up after 3100 years) - see Section 4 above, thus even a "headless" operation without a Controller can be considered feasible. In such a case, the Controller would only be used for the initial configuration of the POT-Profiles.

If OPOT (Section 3.5) is applied using symmetric masking, the Controller will be required to perform a a periodic refresh of the mask pairs. The use of OPOT SHOULD be configurable as part of the required level of assurance through the Controller management interface.

#### 7.9. Verification Scope

The POT solution defined in this document verifies that a data-packet traversed or transited a specific set of nodes. From an algorithm perspective, a "node" is an abstract entity. It could be represented by one or multiple physical or virtual network devices, or is could be a component within a networking device or system. The latter would be the case if a forwarding path within a device would need to be securely verified.

#### 7.9.1. Node Ordering

POT using Shamir's Secret Sharing scheme as discussed in this document provides for a means to verify that a set of nodes has been visited by a data packet. It does not verify the order in which the data packet visited the nodes.

In case the order in which a data packet traversed a particular set of nodes needs to be verified as well, the alternate schemes related to OPOT (Section 3.5) have to be considered. Since these schemes introduce at least additional control requirements, the selection of order verification SHOULD be configurable the Controller management interface.

#### 7.9.2. Stealth Nodes

The POT approach discussed in this document is to prove that a data packet traversed a specific set of "nodes". This set could be all nodes within a path, but could also be a subset of nodes in a path. Consequently, the POT approach isn't suited to detect whether "stealth" nodes which do not participate in proof-of-transit have been inserted into a path.

#### 7.10. POT Yang module

The YANG module specified in Section 5.2 of this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF [RFC6241] layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF Access Control Module (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content. The nodes defined in this YANG module to specify secret-share, prime-number, and validator-key are read/writeable. These data nodes are considered sensitive and vulnerable to attacks in some network environments. Ability to read from or write into these nodes without proper protection can have a negative effect on the devices that support this feature.



## 8. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Erik Nordmark, Andrew Yourtchenko, Tom Petch, Mohamed Boucadair and Dhruv Dhody for the comments and advice.

## 9. Contributors

In addition to editors and authors listed on the title page, the following people have contributed substantially to this document and should be considered coauthors:

Carlos Pignataro  
Cisco Systems, Inc.  
7200-11 Kit Creek Road  
Research Triangle Park, NC 27709  
United States  
Email: cpignata@cisco.com

John Leddy  
Email: john@leddy.net

David Mozes  
Email: mosesster@gmail.com

Alejandro Aguado  
Universidad Politecnica de Madrid  
Campus Montegancedo, Boadilla del Monte  
Madrid 28660  
Spain  
Phone: +34 910 673 086  
Email: a.aguadom@fi.upm.es

Diego R. Lopez  
Telefonica I+D  
Editor Jose Manuel Lara, 9 (1-B)  
Seville 41013  
Spain  
Phone: +34 913 129 041  
Email: diego.r.lopez@telefonica.com

## 10. References

### 10.1. Normative References

- [I-D.ietf-ippm-ioam-data]  
Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-data-10, 13 July 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-ippm-ioam-data-10.txt>>.
- [I-D.ietf-sfc-ioam-nsh]  
Brockners, F. and S. Bhandari, "Network Service Header (NSH) Encapsulation for In-situ OAM (IOAM) Data", Work in Progress, Internet-Draft, draft-ietf-sfc-ioam-nsh-04, 16 June 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-sfc-ioam-nsh-04.txt>>.
- [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>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [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>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [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>>.
- [SSS] A., S., "How to share a secret", Communications of the ACM (22): 612-613, 1979.

## 10.2. Informative References

- [I-D.ietf-anima-autonomic-control-plane]  
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", Work in Progress, Internet-Draft, draft-ietf-anima-autonomic-control-plane-18, 7 August 2018, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-autonomic-control-plane-18.txt>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.

#### Authors' Addresses

Frank Brockners (editor)  
Cisco Systems, Inc.  
Hansaallee 249, 3rd Floor  
40549 DUESSELDORF  
Germany

Email: [fbrockne@cisco.com](mailto:fbrockne@cisco.com)

Shwetha Bhandari (editor)  
Thoughtspot  
3rd Floor, Indiqube Orion, 24th Main Rd, Garden Layout, HSR Layout  
Bangalore, KARNATAKA 560 102  
India

Email: shwetha.bhandari@thoughtspot.com

Tal Mizrahi (editor)  
Huawei Network.IO Innovation Lab  
Israel

Email: tal.mizrahi.phd@gmail.com

Sashank Dara  
Seconize  
BANGALORE  
Bangalore, KARNATAKA  
India

Email: sashank@seconize.co

Stephen Youell  
JP Morgan Chase  
25 Bank Street  
London  
E14 5JP  
United Kingdom

Email: stephen.youell@jpmorgan.com

SFC WG  
Internet-Draft  
Intended status: Informational  
Expires: January 3, 2019

D. Lachos  
Unicamp  
Q. Xiang  
Tongji/Yale University  
C. Rothenberg  
Unicamp  
Y. Yang  
Tongji/Yale University  
July 2, 2018

Multi-domain Service Function Chaining with ALTO  
draft-lachos-multi-domain-sfc-alto-00

Abstract

Currently, Service Function Chaining (SFC) that span domains with different technology, administration or ownership are being defined by the SFC WG. This document focuses on how the Application Layer Traffic Optimization (ALTO) protocol can be used to advertise and discover abstract topology, resource and service information from different domains, and then compute inter-domain service function paths. Another important concern of this draft is to initiate a discussion (ALTO, SFC as well as other WGs) regarding if, how, and under what conditions ALTO can be useful to improve the multi-domain SFC process.

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

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 January 3, 2019.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                                                        |    |
|----------------------------------------------------------------------------------------|----|
| 1. Introduction . . . . .                                                              | 2  |
| 2. Terminology . . . . .                                                               | 3  |
| 3. Context and Motivation . . . . .                                                    | 3  |
| 3.1. Standardization Activities . . . . .                                              | 4  |
| 3.2. Research projects . . . . .                                                       | 4  |
| 4. ALTO for Multi-domain SFC . . . . .                                                 | 5  |
| 4.1. Advantages of using ALTO . . . . .                                                | 5  |
| 4.1.1. Inter-domain info discovery with ALTO Property Map . . . . .                    | 6  |
| 4.1.2. Inter-domain path computation with ALTO Cost Map . . . . .                      | 6  |
| 5. Motivating Use Cases . . . . .                                                      | 7  |
| 5.1. ALTO as part of the SFC eXchange Platform . . . . .                               | 7  |
| 5.2. Resource Orchestration for Multi-Domain, Geo-Distributed Data Analytics . . . . . | 8  |
| 6. IANA Considerations . . . . .                                                       | 10 |
| 7. Security Considerations . . . . .                                                   | 10 |
| 8. Summary and Outlook . . . . .                                                       | 11 |
| 9. Acknowledgments . . . . .                                                           | 11 |
| 10. References . . . . .                                                               | 12 |
| 10.1. Normative References . . . . .                                                   | 12 |
| 10.2. Informative References . . . . .                                                 | 12 |
| Authors' Addresses . . . . .                                                           | 14 |

## 1. Introduction

The delivery of end-to-end services often requires various Service functions (SFs) or Virtual Network Functions (VNFs). Service Function Chaining (SFC) is constructed as an abstract sequence of SFs or VNFs [RFC7665]. Multi-domain SFC is the ability to deploy SFC across multiple domains with different technology and/or

administration. To do so, an inter-domain communication process between different organizations is necessary to (i) exchange topology, resource and service information, and then (ii) compute inter-domain service function paths.

The ALTO WG has begun started to discuss the uses of ALTO as an information model for representing network resource and services in multi-domain scenarios:

- o [DRAFT-ALTO-BROKER-MDO] proposes an ALTO-based Broker-assisted architecture where a broker plane works as a coordinator between a set of top-level control planes, i.e., Multi-domain Orchestrator (MdOs). The ALTO services (with the proposed extensions) provides abstract maps with a simplified, yet enough information view about MdOs involved in the federation. This information includes the abstract network topology, resource availability (e.g., CPUs, Memory, and Storage) and capabilities (e.g., supported network functions).
- o The document [DRAFT-ALTO-UNICORN] presents Unicorn, a resource orchestration framework for multi-domain, geo-distributed data analytics. This work resorts in ALTO as the information model to support the accurate, yet privacy-preserving resource discovery across different domains. The key information to be provided by the use of ALTO including different types of resources, e.g., the computing, storage, and networking resources.

This draft offers concrete use case examples of how ALTO can be incorporated in the multi-domain SFC architecture. The examples used in this document are based on architectures and assumptions currently being discussed in the SFC WG [DRAFT-HH-MDSFC] and in the ALTO WG [DRAFT-ALTO-BROKER-MDO] [DRAFT-ALTO-UNICORN].

The overall rationale of this document is to begin a discussion between the SFC and the ALTO WG (other WGs are welcome) concerning if, how, and under which conditions ALTO will be helpful in the SFC traversing different administrative domains.

## 2. Terminology

This document makes use of the terminology defined in [DRAFT-HH-MDSFC], [DRAFT-ALTO-UNICORN], and [DRAFT-ALTO-BROKER-MDO].

## 3. Context and Motivation

Nowadays, different standardization activities (e.g., IETF and ETSI) and research projects (e.g., 5GEx [H2020.5GEX], 5G-Transformer

[H2020-5G-TRANSFORMER], T-NOVA [T-NOVA], etc.) have been focused on multi-domain network service chaining.

### 3.1. Standardization Activities

SFC that span domains owned by multiple administrative entities are being discussed in the IETF SFC WG. [DRAFT-HH-MDSFC], for example, describes SFC crossing different domains owned by various organizations (e.g., ISPs) or by a single organization with administration partitions. The proposed architecture uses a SFC eXchange Platform (SXP) to collect and exchange information (topology, service states, policies, etc.) between different organizations and it works both in centralized (Multiple SFC domains connected by a logical SXP) and distributed (SXP server as a broker) environments. Another IETF initiative is the Network Function Virtualization Research Group (NFVRG). The draft "Multi-domain Network Virtualization" [DRAFT-MD-VIRT] envisions a complete end-to-end logical network as stitching services offered by multiple domains from multiple providers. It also points to the need for creating solutions that enable the exchange of relevant information (resources and topologies) across different providers.

The ETSI NFV ISG is paving the way toward viable architectural options supporting the efficient placement of functions in different administrative domains. More specifically, the document [ETSI-NFV-IFA028] reports different NFV MANO architectural approaches with use cases related to network services provided using multiple administrative domains. Besides, it gives a non-exhaustive list of key information to be exchanged between administrative domains (monitoring parameters, topology view, resource capabilities, etc.) and recommendations related to security to permit the correct and proper operation of the final service.

### 3.2. Research projects

Several projects include an architectural model integrating NFV management with SDN control capabilities to address the challenges towards flexible, dynamic, cost-effective, and on-demand service chaining.

[H2020.5GEX] aims to integrate multiple administrations and technologies through the collaboration between operators in the context of emerging 5G networking. [VITAL][T-NOVA] follow a centralized approach where each domain advertises its capabilities to a federation layer which will act as a broker. In order to avoid one network operator per country or regions, [H2020-5G-NORMA] proposes the use of management and control into a single virtual domain. Also, the 5G-Transformer project [H2020-5G-TRANSFORMER] is defining



flexible slicing and federation of transport networking and computing resources across multiple domains.

#### 4. ALTO for Multi-domain SFC

A "dialogue" between potential domains that will provide multi-domain SFC could be beneficial for more efficient use of resources and increasing the SFC performance. However, constrained knowledge of the network services and underlying network topology based only on localized views from the point of view of a single domain limits the potential and scope for multi-domain SFC.

To enable a highly customized multi-domains SFC, [DRAFT-HH-MDSFC] proposes a SFC eXchange Platform to realize inter-domain communication between top-level control planes. The SXP is a logical entity deployed in future Software-defined IXP (as a trusted third-party platform) or built by a single owner between different networks.

On a high level, the scope of the SXP contains two main tasks:

- o Provide end-to-end visibility through the collection of topology information, service states, and policies from different domains.
- o Compute inter-domain service function path to select the service function location from multiple candidate domains.

The ALTO protocol [RFC7285] provides abstract network information in the form of map services that can be consumed by applications in order to become network-aware and to take optimized decisions regarding traffic flows. Recently, ALTO is also being considered in multi-domain orchestration scenarios [DRAFT-ALTO-UNICORN] [DRAFT-ALTO-BROKER-MDO], in which an ALTO server can convey inter-domain network resource and topology information.

In this context, the SXP can take advantage of multi-domain ALTO services to obtain important inter-domain information to "guide" the resource/service provider selection process in that the "best" domain or candidate domains (according to established policies) can be intelligently selected. The following ALTO services can be identified:

##### 4.1. Advantages of using ALTO

ALTO (and customized ALTO extensions) can be used to offer aggregated/abstracted views on various types of information including domain-level topology, storage resources, computation resources, networking resources and PNF/VNF capabilities. This generic

representation contributing to a more simple and scalable solution for resource and service discovery in multi-domain, multi-technology environments.

In case of Multi-domain SFC, the following ALTO services could be identified:

#### 4.1.1. Inter-domain info discovery with ALTO Property Map

Each domain needs a global view of other potential candidate domains to know who can provide part of the NF in the SFC. A brief list of information to be exchanged between different domains includes:

- o Resource capabilities, applicable to both IT (computing and storage) and networking resources participant of the multi-domain SFC, to assist on the decision of SFs placement.
- o Access information (e.g., URL) to the orchestrator entry points and Service Access Points (SAPs) for a corresponding network/domain.

The ALTO Property Map Service [DRAFT-ALTO-PM] can provide a clear global view of the resource information offered by other domains. This information allows discovering which candidate domains may be contacted to deliver the remaining requirements of a requested end-to-end service deployment.

#### 4.1.2. Inter-domain path computation with ALTO Cost Map

Once the candidate domains are discovered, it is necessary to compute inter-domain service function path to select the service function location from those different candidate domains.

The connectivity information among discovered domains can be retrieved by an ALTO Cost Map service, responding, for instance, a path vector with the AS-level topology distance between the source domain and candidate domains. Moreover, path vector constraints (as described in the Multi-Cost Map [RFC8189]) can be applied to filter out the list of unqualified domains.

In case of the Hybrid Hierarchical SFC architecture [DRAFT-HH-MDSFC], the SXP (or the Path Calculation Element in the top-level control plane) could use this information to compute multi-domain service function paths.

## 5. Motivating Use Cases

### 5.1. ALTO as part of the SFC eXchange Platform

As mentioned earlier, the draft [DRAFT-HH-MDSFC] defines a multi-domain SFC architecture that combines control planes to be deployed either into a large domain consisting of smaller sub-domains owned by the same organization or into multiple large domains with different ownership. Figure 1 shows a SXP connecting three different domains (AS1, AS2, AS3). Each domain provides different SFs: AS1 -> SF1; AS2 -> SF2 and SF3; AS3 -> SF3. The SXP includes an ALTO server component to provide abstract topology, resource, and service information for the high-level control plane in each domain.

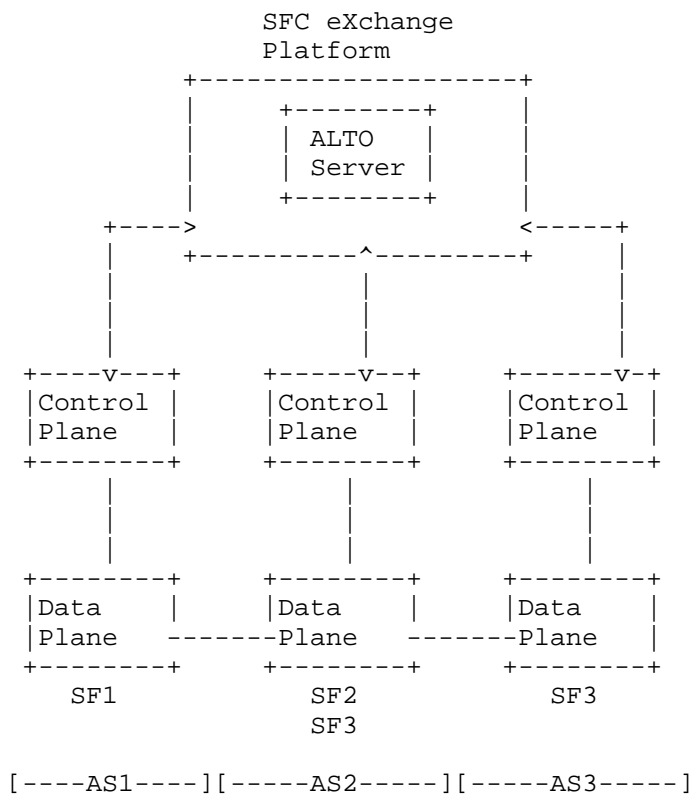


Figure 1: ALTO as part of the SFC eXchange Platform

Every domain has a local Information Base Element; this component can be used by the SXP to create hierarchical databases containing inter-domain resource and topology information. This information source is

used by the ALTO server to create two different ALTO Map Services:  
(i) Property Map and (ii) Cost Map.

The Property Map includes a property value grouped by Autonomous System (AS), this value contains the supported network functions. Additional properties could be considered such as resource availability (e.g., CPUs, Memory, and Storage), orchestrator entry points, etc. An example of the Property Map in our basic scenario is:

|     | Capabilities | Entry Point | CPU | MEM | Storage | ... |
|-----|--------------|-------------|-----|-----|---------|-----|
| AS1 | {SF1}        | http://...  | ... | ... | ...     | ... |
| AS2 | {SF2, SF3}   | http://...  | ... | ... | ...     | ... |
| AS3 | {SF3}        | http://...  | ... | ... | ...     | ... |

Table 1: ALTO Property Map

The Cost Map defines a path vector as an array of ASes, representing the AS-level topological distance for a given SFC request. Table 2 below shows a brief example of a service request and its inter-domain service function path response containing a list of potential domains to be traversed to deliver such service.

| SFC Request   | Multi-domain Service Function Path(s) |
|---------------|---------------------------------------|
| SF1->SF2->SF3 | 1:{AS1:SF1->AS2:SF2->AS2:SF3}         |
|               | 2:{AS1:SF1->AS2:SF2->AS3:SF3}         |

Table 2: ALTO Cost Map

## 5.2. Resource Orchestration for Multi-Domain, Geo-Distributed Data Analytics

In addition to commercial SFC, ALTO is also used as a core information model for collaborative data science networks. The document [DRAFT-ALTO-UNICORN] presents the design of Unicorn, a unified resource orchestration framework for multi-domain, geo-distributed data analytics, currently being developed and deployed in the CMS network, one of the largest scientific experiments in the LHC network.

ALTO is well suited as a fundamental component in Unicorn for providing a generic representation that (1) allows different types of

data analytics jobs to accurately describe their resource requirements and (2) allows member networks to provide accurate information on different types of resources they own and at the same time maintain their privacies.

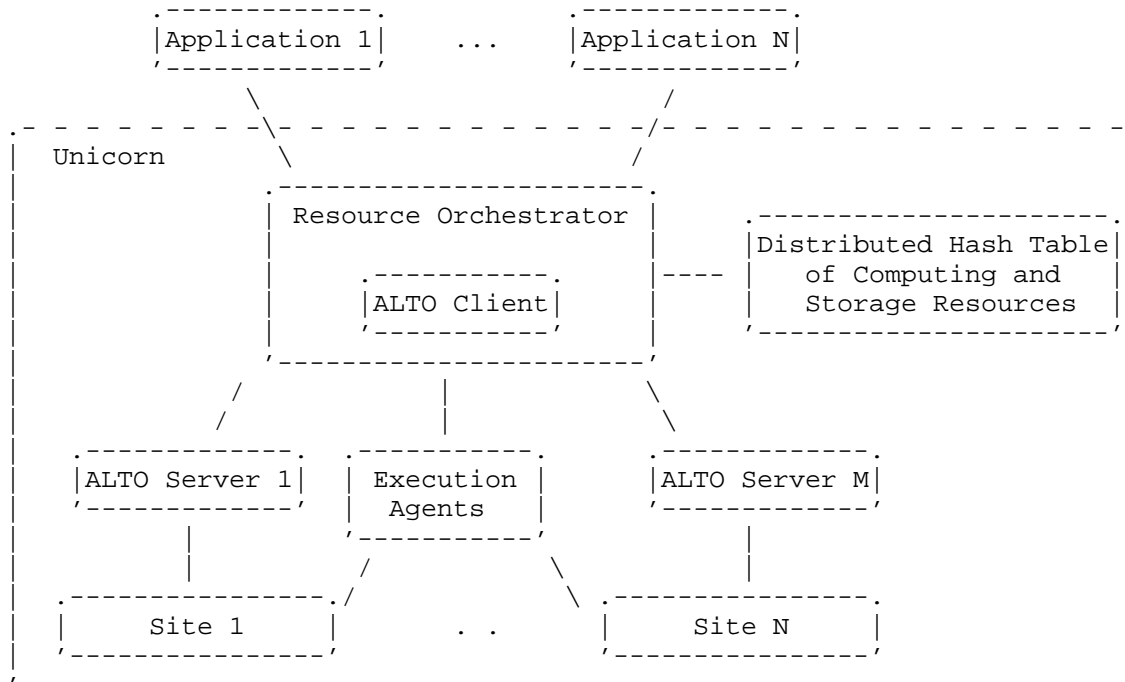


Figure 2: Architecture of Unicorn.

Figure 2 presents the architecture of Unicorn. Specifically, for each member network, one or more ALTO servers are deployed to provide accurate, yet privacy-preserving information of different types of resources owned by the corresponding network. Examples of such information include the link bandwidth between endpoints, the memory I/O bandwidth and the CPU utilization at computing endpoints and the storage space at storage endpoints. In addition to the basic ALTO services defined in [RFC7285], The ALTO servers in Unicorn also provide ALTO extension services such as the ALTO Multi-Cost Service [RFC8189], the ALTO Server-Sent Event Service [DRAFT-ALTO-INCR-UPD] and the ALTO Path Vector Service [DRAFT-ALTO-PV] to provide fine-grained resource information.

Because the ALTO Path Vector service may expose additional private information of each network, Unicorn develops an obfuscating protocol

which ensures that nor the orchestrator or any member networks can associate any path vector information with a corresponding network.

To better address the scalability issue of multi-domain resource discovery, Unicorn also develops a proactive full-mesh discovery mechanism, which precomputes network-level ALTO path vector information and performs projection using such information to compute the fine-grained resource information in response to orchestrator's resource discovery requests.

Details of the obfuscating protocol and the proactive full-mesh discovery mechanism developed in Unicorn can be found in the [DRAFT-ALTO-UNICORN] document.

## 6. IANA Considerations

This document includes no request to IANA.

## 7. Security Considerations

The ALTO base protocol has an extensive discussion on potential security and privacy issues. Using the ALTO base protocol to support multi-domain SFC will not raise new security and privacy issue. However, the information provided by the ALTO base protocol is considered coarse-grained in several recent use cases. As a result, several ALTO extension services have been designed to provide fine-grained network information to the application. Using these ALTO extension services for multi-domain SFC would raise new security and privacy concerns. Next, we list these issues on a per extension basis.

The ALTO unified property extension [DRAFT-ALTO-PM] generalizes the concept of endpoint properties to other entity domains, such as abstract network element. The properties of these entities may contain sensitive service-function-specific information. Exposing such information may discourage networks to provide fine-grained information to support multi-domain SFC.

The ALTO performance cost metrics extension [DRAFT-ALTO-METRICS] proposes a set of ALTO cost metrics derived from traffic engineering tools and protocols. It is stated in this extension that "sharing network TE metric values in numerical mode requires full mutual confidence between the entities managing the ALTO Server and Client." In multi-domain SFC use case, such mutual confidence is needed not only between ALTO server and client, but also among all networks, and third-parties such as broker and a global orchestrator. How to achieve such mutual confidence in multi-domain SFC use case requires further investigation.

The ALTO path vector extension [DRAFT-ALTO-PV] allows ALTO clients to query network information such as capacity region for a given set of flows. Several related studies have shared concerns that this extension may reveal more network internal structures than the more abstract single-node abstraction used in the ALTO base protocol. In multi-domain SFC, this concern will further be amplified as third-party participants may access such information. The recent designed Unicorn system proposes an obfuscating protocol that prevents the receiver of the capacity region information from associating this region to any network. This protocol sheds light for addressing the privacy issue brought by the ALTO path vector extension.

The ALTO cost calendar [DRAFT-ALTO-CALENDAR] and the ALTO incremental update [DRAFT-ALTO-INCR-UPD] extensions allow the ALTO client to get temporal network information. The intention of these extensions is to allow applications to make flexible decisions on when to use network information. However, both extensions expose temporal policy and traffic information of network so that a user may know when the network is most vulnerable for overloading. This issue needs to be carefully addressed in order for both extensions to be used for multi-domain SFC.

## 8. Summary and Outlook

This draft provided arguments why ALTO is a meaningful protocol in the context of SFC traversing different domains, and it presented use case examples about the how ALTO can be used to advertise and discover abstract topology, resource and service information from different domains, and then compute inter-domain service function paths.

The overall objective of this document is to arouse discussions in the SFC WG in order to assess the suitability of the ALTO as a useful protocol for multi-domain SFC scenarios. The result of such discussions will be captured in future versions of this draft.

## 9. Acknowledgments

This work is supported by the Innovation Center of Ericsson S.A., Brazil (grant agreement UNI.64).

Many thanks to Sabine Randriamasy, and Lyle Bertz for their feedback on this draft.

## 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>>.
- [RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <<https://www.rfc-editor.org/info/rfc7285>>.
- [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>>.
- [RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <<https://www.rfc-editor.org/info/rfc8189>>.

### 10.2. Informative References

- [DRAFT-ALTO-BROKER-MDO] Perez, D. and C. Rothenberg, "ALTO-based Broker-assisted Multi-domain Orchestration", draft-lachosrothenberg-alto-brokermdo-00 (work in progress), March 2018.
- [DRAFT-ALTO-CALENDAR] Randriamasy, S., Yang, Y., Wu, Q., Lingli, D., and N. Schwan, "ALTO Cost Calendar", draft-ietf-alto-cost-calendar-05 (work in progress), June 2018.
- [DRAFT-ALTO-INCR-UPD] Roome, W., Yang, Y., and S. Chen, "ALTO Incremental Updates Using Server-Sent Events (SSE)", draft-ietf-alto-incr-update-sse-11 (work in progress), June 2018.
- [DRAFT-ALTO-METRICS] Wu, Q., Yang, Y., Lee, Y., Dhody, D., and S. Randriamasy, "ALTO Performance Cost Metrics", draft-ietf-alto-performance-metrics-04 (work in progress), June 2018.



## [DRAFT-ALTO-PM]

Roome, W., Chen, S., Randriamasy, S., Yang, Y., and J. Zhang, "Unified Properties for the ALTO Protocol", draft-ietf-alto-unified-props-new-03 (work in progress), March 2018.

## [DRAFT-ALTO-PV]

Bernstein, G., Chen, S., Gao, K., Lee, Y., Roome, W., Scharf, M., Yang, Y., and J. Zhang, "ALTO Extension: Path Vector Cost Type", draft-ietf-alto-path-vector-03 (work in progress), March 2018.

## [DRAFT-ALTO-UNICORN]

Xiang, Q., Le, F., Yang, Y., Newman, H., and H. Du, "Unicorn: Resource Orchestration for Multi-Domain, Geo-Distributed Data Analytics", draft-xiang-alto-multidomain-analytics-01 (work in progress), March 2018.

## [DRAFT-HH-MDSFC]

Li, G., Li, G., Li, T., Xu, Q., and H. Zhou, "Hybrid Hierarchical Multi-Domain Service Function chaining", draft-li-sfc-hhsfc-04 (work in progress), April 2018.

## [DRAFT-MD-VIRT]

Bernardos, C., Contreras, L., Vaishnavi, I., Szabo, R., Li, X., Paolucci, F., Sgambelluri, A., Martini, B., Valcarengi, L., Landi, G., Andrushko, D., and A. Mourad, "Multi-domain Network Virtualization", draft-bernardos-nfvrg-multidomain-04 (work in progress), March 2018.

## [ETSI-NFV-IFA028]

ETSI, "Report on architecture options to support multiple administrative domains V3.1.1", Jan 2018, <[http://www.etsi.org/deliver/etsi\\_gr/NFV-IFA/001\\_099/028/03.01.01\\_60/gr\\_NFV-IFA028v030101p.pdf](http://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/028/03.01.01_60/gr_NFV-IFA028v030101p.pdf)>.

## [H2020-5G-NORMA]

H2020, "5G-NORMA -- 5G Novel Radio Multiservice adaptive network Architecture", 2015, <<https://5gnorma.5g-ppp.eu/>>.

## [H2020-5G-TRANSFORMER]

H2020, "5G-Transformer -- 5G Mobile Transport Platform for Vertical", 2017, <<http://5g-transformer.eu/>>.

## [H2020.5GEX]

H2020, "5GEx -- 5G Exchange Project", 2014, <<http://www.5gex.eu/>>.

[T-NOVA] FP7 project T-NOVA, "T-NOVA Project, Network Functions as a Service over Virtualised Infrastructures", 2014, <<http://www.t-nova.eu/>>.

[VITAL] VITAL PROJECT H2020, "VITAL -- Virtualized hybrid satellite-Terrestrial systems for resilient and flexible future networks", 2015, <<http://www.ict-vital.eu/>>.

#### Authors' Addresses

Danny Alex Lachos Perez  
University of Campinas  
Av. Albert Einstein 400  
Campinas, Sao Paulo 13083-970  
Brazil

Email: [dlachosp@dca.fee.unicamp.br](mailto:dlachosp@dca.fee.unicamp.br)  
URI: <https://intrig.dca.fee.unicamp.br/danny-lachos/>

Qiao Xiang  
Tongji/Yale University  
51 Prospect Street  
New Haven, CT  
USA

Email: [qiao.xiang@cs.yale.edu](mailto:qiao.xiang@cs.yale.edu)

Christian Esteve Rothenberg  
University of Campinas  
Av. Albert Einstein 400  
Campinas, Sao Paulo 13083-970  
Brazil

Email: [chesteve@dca.fee.unicamp.br](mailto:chesteve@dca.fee.unicamp.br)  
URI: <https://intrig.dca.fee.unicamp.br/christian/>

Y. Richard Yang  
Tongji/Yale University  
51 Prospect St  
New Haven, CT  
USA

Email: [yang.r.yang@gmail.com](mailto:yang.r.yang@gmail.com)

MPLS Working Group  
Internet-Draft  
Intended status: Informational  
Expires: April 14, 2019

A. Malis  
S. Bryant  
Huawei Technologies  
J. Halpern  
Ericsson  
W. Henderickx  
Nokia  
October 11, 2018

MPLS Encapsulation for SFC NSH  
draft-malis-mpls-sfc-encapsulation-03

Abstract

This document describes how to use a Service Function Forwarder (SFF) Label (similar to a pseudowire label or VPN label) to indicate the presence of a Service Function Chaining (SFC) Network Service Header (NSH) between an MPLS label stack and the packet payload. This allows SFC packets using the NSH to be forwarded between SFFs over an MPLS network, and the selection between multiple SFFs in the destination MPLS node.

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 April 14, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                                               |   |
|-------------------------------------------------------------------------------|---|
| 1. Introduction . . . . .                                                     | 2 |
| 2. MPLS Encapsulation Using an SFF Label . . . . .                            | 3 |
| 2.1. MPLS Label Stack Construction at the Sending Node . . . . .              | 3 |
| 2.2. SFF Label Processing at the Destination Node . . . . .                   | 4 |
| 3. Equal Cost Multipath (ECMP) Considerations . . . . .                       | 4 |
| 4. Operations, Administration, and Maintenance (OAM) Considerations . . . . . | 5 |
| 5. IANA Considerations . . . . .                                              | 5 |
| 6. Security Considerations . . . . .                                          | 5 |
| 7. Acknowledgements . . . . .                                                 | 5 |
| 8. References . . . . .                                                       | 5 |
| 8.1. Normative References . . . . .                                           | 5 |
| 8.2. Informative References . . . . .                                         | 6 |
| Authors' Addresses . . . . .                                                  | 6 |

## 1. Introduction

As discussed in [RFC8300], a number of transport encapsulations for the Service Function Chaining (SFC) Network Service Header (NSH) already exist, such as Ethernet, GRE [RFC2784], and VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe].

This document describes an MPLS transport encapsulation for the NSH, and also describes how to use a Service Function Forwarder (SFF) [RFC7665] Label to indicate the presence of the NSH in the MPLS packet payload. This allows SFC packets using the NSH to be forwarded between SFFs in an MPLS transport network, where MPLS is used to interconnect the network nodes that contain one or more SFFs. The label is also used to select between multiple SFFs in the destination MPLS node.

SFF Labels are similar to other service labels at the bottom of an MPLS label stack that denote the contents of the MPLS payload being other than IP, such as a layer 2 pseudowire, an IP packet that is routed in a VPN context with a private address, or an Ethernet virtual private wire service.

This informational document follows well-established MPLS procedures and does not require any actions by IANA or any new protocol extensions.

## 2. MPLS Encapsulation Using an SFF Label

The encapsulation is a standard MPLS label stack [RFC3032] with an SFF Label at the bottom of the stack, followed by a NSH as defined by [RFC8300] and the NSH payload.

Much like a pseudowire label, an SFF Label is allocated by the downstream receiver of the NSH from its per-platform label space.

If a receiving node supports more than one SFF (i.e., more than one SFC forwarding instance), then the SFF Label can be used to select the proper SFF, by having the receiving node advertise more than one SFF Label to its upstream sending nodes as appropriate.

The method used by the downstream receiving node to advertise SFF Labels to the upstream sending node is out of scope of this document. That said, a number of methods are possible, such as via a protocol exchange, or via a controller that manages both the sender and the receiver using NETCONF/YANG, BGP, PCEP, etc. These are meant as possible examples and not to constrain the future definition of such advertisement methods.

While the SFF label will usually be at the bottom of the label stack, there may be cases where there are additional label stack entries beneath it. For example, when an ACH is carried that applies to the SFF, a GAL [RFC5586] will be in the label stack below the SFF. Similarly, an ELI/EL [RFC6790] may be carried below the SFF in the label stack. This is identical to the situation with VPN labels.

### 2.1. MPLS Label Stack Construction at the Sending Node

When one SFF wishes to send an SFC packet with the NSH to another SFF over an MPLS transport network, a label stack needs to be constructed by the MPLS node that contains the sending SFF in order to transport the packet to the destination MPLS node that contains the receiving SFF. The label can be constructed as follows:

1. Push on zero or more labels that are interpreted by the destination MPLS node, such as the Generic Associated Channel [RFC5586] label (see OAM Considerations below).
2. Push on the SFF Label to identify the desired SFF in the receiving MPLS node.

3. Push on zero or more additional labels such that (a) the resulting label stack will cause the packet to be transported to the destination MPLS node, and (b) when the packet arrives at the destination node, either:
  - \* the SFF Label will be at the top of the label stack, or
  - \* the SFF Label will rise to the top of the label stack before the packet is forwarded to another node and before the packet is dispatched to a higher layer.

## 2.2. SFF Label Processing at the Destination Node

The destination MPLS node performs a lookup on the SFF label to retrieve the next-hop context between the SFF and SF, e.g. to retrieve the destination MAC address in the case where native Ethernet encapsulation is used between SFF and SF. How the next-hop context is populated is out of the scope of this document.

The receiving MPLS node then pops the SFF Label (and any labels beneath it) so that the destination SFF receives the SFC packet with the NSH is at the top of the packet.

## 3. Equal Cost Multipath (ECMP) Considerations

As discussed in [RFC4928] and [RFC7325], there are ECMP considerations for payloads carried by MPLS.

Many existing routers use deep packet inspection to examine the payload of an MPLS packet, and if the first nibble of the payload is equal to 0x4 or 0x6, these routers (sometimes incorrectly, as discussed in [RFC4928]) assume that the payload is IPv4 or IPv6 respectively, and as a result, perform ECMP load balancing based on (presumed) information present in IP/TCP/UDP payload headers or in a combination of MPLS label stack and (presumed) IP/TCP/UDP payload headers in the packet.

For SFC, ECMP may or may not be desirable. To prevent unintended ECMP when it is not desired, the NSH Base Header was carefully constructed so that the NSH could not look like IPv4 or IPv6 based on its first nibble. See Section 2.2 of [RFC8300] for further details.

If ECMP is desired when SFC is used with an MPLS transport network, there are two possible options, Entropy [RFC6790] and Flow-Aware Transport [RFC6391] labels. A recommendation between these options, and their proper placement in the label stack, is for future study.

#### 4. Operations, Administration, and Maintenance (OAM) Considerations

OAM at the SFC Layer is handled by SFC-defined mechanisms [RFC8300]. However, OAM may be required at the MPLS transport layer. If so, then standard MPLS-layer OAM mechanisms such as the Generic Associated Channel [RFC5586] label may be used.

#### 5. IANA Considerations

This document does not request any actions from IANA.

Editorial note to RFC Editor: This section may be removed at your discretion.

#### 6. Security Considerations

This document describes a method for transporting SFC packets using the NSH over an MPLS transport network. It follows well-established MPLS procedures and does not define any new protocol elements or allocate any new code points. It is therefore operationally equivalent to other existing SFC transport encapsulations as defined in [RFC8300]. As such, it should have no effect on SFC security as already discussed in Section 8 of [RFC8300].

#### 7. Acknowledgements

The authors would like to thank Jim Guichard, Eric Rosen, Med Boucadair, Sasha Vainshtein, and Jeff Tantsura for their reviews and comments.

#### 8. References

##### 8.1. Normative References

- [RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <<https://www.rfc-editor.org/info/rfc3032>>.
- [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>>.

## 8.2. Informative References

- [I-D.ietf-nvo3-vxlan-gpe]  
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-06 (work in progress), April 2018.
- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <<https://www.rfc-editor.org/info/rfc2784>>.
- [RFC4928] Swallow, G., Bryant, S., and L. Andersson, "Avoiding Equal Cost Multipath Treatment in MPLS Networks", BCP 128, RFC 4928, DOI 10.17487/RFC4928, June 2007, <<https://www.rfc-editor.org/info/rfc4928>>.
- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <<https://www.rfc-editor.org/info/rfc5586>>.
- [RFC6391] Bryant, S., Ed., Filsfils, C., Drafz, U., Kompella, V., Regan, J., and S. Amante, "Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network", RFC 6391, DOI 10.17487/RFC6391, November 2011, <<https://www.rfc-editor.org/info/rfc6391>>.
- [RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <<https://www.rfc-editor.org/info/rfc6790>>.
- [RFC7325] Villamizar, C., Ed., Kompella, K., Amante, S., Malis, A., and C. Pignataro, "MPLS Forwarding Compliance and Performance Requirements", RFC 7325, DOI 10.17487/RFC7325, August 2014, <<https://www.rfc-editor.org/info/rfc7325>>.
- [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>>.

## Authors' Addresses



Andrew G. Malis  
Huawei Technologies

Email: [agmalis@gmail.com](mailto:agmalis@gmail.com)

Stewart Bryant  
Huawei Technologies

Email: [stewart.bryant@gmail.com](mailto:stewart.bryant@gmail.com)

Joel M. Halpern  
Ericsson

Email: [joel.halpern@ericsson.com](mailto:joel.halpern@ericsson.com)

Wim Henderickx  
Nokia

Email: [wim.henderickx@nokia.com](mailto:wim.henderickx@nokia.com)

RTGWG Working Group  
Internet-Draft  
Intended status: Informational  
Expires: December 30, 2018

G. Mirsky  
ZTE Corp.  
June 28, 2018

Identification of Overlay Operations, Administration, and Maintenance  
(OAM)  
draft-mirsky-rtgwg-oam-identify-00

Abstract

This document analyzes how the presence of Operations, Administration, and Maintenance (OAM) control command and/or special data is identified in some overlay networks, and an impact on the choice of identification may have on OAM functionality.

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 30, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                |   |
|------------------------------------------------|---|
| 1. Introduction . . . . .                      | 2 |
| 2. Conventions used in this document . . . . . | 2 |
| 2.1. Terminology . . . . .                     | 2 |
| 2.2. Keywords . . . . .                        | 3 |
| 3. Overlay Network Encapsulations . . . . .    | 3 |
| 3.1. Encapsulations with Meta-data . . . . .   | 3 |
| 3.2. Fixed-size Encapsulations . . . . .       | 5 |
| 3.3. Source Information Availability . . . . . | 5 |
| 3.4. On-path OAM . . . . .                     | 6 |
| 4. Conclusions . . . . .                       | 6 |
| 5. IANA Considerations . . . . .               | 6 |
| 6. Security Considerations . . . . .           | 7 |
| 7. Acknowledgment . . . . .                    | 7 |
| 8. References . . . . .                        | 7 |
| 8.1. Normative References . . . . .            | 7 |
| 8.2. Informational References . . . . .        | 7 |
| Author's Address . . . . .                     | 9 |

## 1. Introduction

Operations, Administration, and Maintenance (OAM) protocols are used to detect, localize defects in the network, and monitor network performance. Some OAM functions, e.g., failure detection, work in the network proactively, while others, e.g., defect localization, usually performed on-demand. These tasks achieved by a combination of active, passive, and hybrid OAM methods, as defined in [RFC7799].

This document analyzes how the presence of Operations, Administration, and Maintenance (OAM) control command and/or special data, i.e., OAM packet, is identified in some overlay networks, and an impact the choice of identification may have on OAM functionality of active and hybrid OAM methods for the respective overlay network encapsulation.

## 2. Conventions used in this document

## 2.1. Terminology

AMM Alternate Marking method

BIER Bit Indexed Explicit Replication

DetNet Deterministic Networks

GUE Generic UDP Encapsulation

HTS Hybrid Two-step

NSH Network Service Header

NVO3 Network Virtualization Overlays

OAM Operations, Administration and Maintenance

SFC Service Function Chaining

TLV Type-Length-Value

VXLAN-GPE Generic Protocol Extension for VXLAN

Underlay Network or Underlay Layer: The network that provides connectivity between the DetNet nodes. MPLS network providing LSP connectivity between DetNet nodes is an example of underlay layer.

## 2.2. Keywords

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. Overlay Network Encapsulations

New overlay network encapsulations analyzed in two groups:

- o encapsulations that support optional meta-data;
- o fixed-size encapsulations.

### 3.1. Encapsulations with Meta-data

Number of the new encapsulation protocols (e.g., Geneve [I-D.ietf-nvo3-geneve], GUE [I-D.ietf-intarea-gue], and SFC NSH [RFC8300]) support use of Type-Length-Value (TLV) encoding to include optional information into the header. The identification of OAM in these protocols is as the following:

Geneve:

O (1 bit): OAM packet. This packet contains a control message instead of a data payload. Endpoints MUST NOT forward the payload and transit devices MUST NOT attempt to interpret or process it. Since these are infrequent control messages, it is

RECOMMENDED that endpoints direct these packets to a high priority control queue (for example, to direct the packet to a general purpose CPU from a forwarding ASIC or to separate out control traffic on a NIC). Transit devices MUST NOT alter forwarding behavior on the basis of this bit, such as ECMP link selection.

GUE:

Undefined.

SFC NSH:

0 bit: Setting this bit indicates an OAM packet.

Common between Geneve and NSH is the use of the dedicated flag to identify the OAM packet and, at the same time, the presence of the field that identifies the protocol of the payload that immediately follows after the encapsulation header. [RFC8393] points that if the value of that field interpreted as none, i.e., no payload follows the header, then OAM may be included in TLVs, thus creating an active OAM packet. The problem with this mechanism to support active OAM methods may be a limitation of the size of data that can be included in a TLV. For example, the maximum size of data in an NSH Meta-data Type 2, as defined in section 2.5.1 [RFC8300], is 512 octets. The maximum length of data in Geneve Option, per section 3.5 [I-D.ietf-nvo3-geneve], is 128 octets. Thus, using one TLV as active OAM packet, would not allow creating test packets of larger size, which is useful when measuring packet loss and latency with synthetic traffic as part of service activation procedure.

[I-D.ietf-sfc-oam-framework] suggests that the 0 bit used to identify OAM packet and the Next Protocol field identifies the OAM function:

While the presence of OAM marker in the overlay header (e.g., 0 bit in the NSH header) indicates it as OAM packet, it is not sufficient to indicate what OAM function the packet is intended for.

At the same time, some of in-situ OAM proposals, e.g., [I-D.ietf-sfc-ioam-nsh], suggest using TLV to communicate hybrid OAM commands and data. The proposed resolution of using the combination of 0 bit and the Next Protocol field:

... the 0 bit MUST NOT be set for regular customer traffic which also carries IOAM data and the 0 bit MUST be set for OAM packets which carry only IOAM data without any regular data payload.

implies that the O bit only identifies the active OAM packet and not set when hybrid OAM methods used.

### 3.2. Fixed-size Encapsulations

Number of the new encapsulation protocols (e.g., VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe], BIER [RFC8296]) use fixed-size header. The identification of OAM in these protocols is as the following:

#### VXLAN-GPE:

OAM Flag Bit (O bit): The O bit is set to indicate that the packet is an OAM packet.

#### BIER:

OAM packet identified by the value of the Next Protocol field. IANA BIER Next Protocol Identifiers registry includes the identifier for OAM (5).

VXLAN-GPE use of a combination of OAM Flag Bit and the Next Protocol field requires clarification of the header interpretation when the OAM Flag Bit is set and the value of the Next Protocol field is one of defined in section 3.2 of [I-D.ietf-nvo3-vxlan-gpe].

BIER encapsulation, defined in [RFC8296], identifies OAM message immediately following the BIER header by the value of the Next Protocol field.

### 3.3. Source Information Availability

Availability of the packet originator's source information is required for active two-way OAM, e.g., echo request/reply. In cases when the underlay network is IPv4/IPv6 the source information will be provided by the encapsulation of the underlay. But when using MPLS underlay network encapsulation of an active OAM packet have to follow certain rules:

- o if available, use Sender ID in the overlay domain (example BFIR ID in BIER [RFC8296]);
- o use IP/UDP encapsulation of an OAM packet in overlay (similar to Section 4.3 [RFC8029]).

### 3.4. On-path OAM

In addition to active methods, OAM toolset may include methods that don't use specially constructed and injected in the network test packets. [RFC7799] defines OAM methods that are neither entirely active nor passive but are combine both as hybrid methods.

One of the examples of the hybrid OAM method, in-situ OAM, mentioned in Section 3.1. Another example, Alternate Marking method (AMM) [RFC8321], enables on-path OAM functions, e.g., delay and loss measurements, using the data traffic. Because AMM impact on the network can be minimized, measured metrics can be correlated to the network conditions experienced by the specific service. Of all listed in Section 3, BIER allocated the field that may be used for AMM, as discussed in [I-D.ietf-bier-pmmm-oam]. Applicability of AMM to other overlay protocols, i.e. SFC NSH discussed in [I-D.mirsky-sfc-pmamm] and Geneve [I-D.fmm-nvo3-pm-alt-mark], been actively discussed.

Hybrid Two-step (HTS), defined in [I-D.mirsky-ippm-hybrid-two-step], is provides on-path collection and transport of the telemetry information. HTS enables accurate and consistent measurements by separating the measurement action from the transport while ensuring that the follow-up packet that carries the telemetry information does follow the data packet that had triggered the measurement.

## 4. Conclusions

OAM control commands and data may be present as part of the overlay encapsulation header or as a payload that follows the overlay network header. The recommendations:

- o OAM in the overlay header, if supported by the overlay network, identified by the dedicated flag. Use of this method as active OAM is possible but functionality is limited.
- o OAM that follows the overlay header identified as payload type, e.g. by the value of the Next Protocol field.

## 5. IANA Considerations

This document does not propose any IANA consideration. This section may be removed.

## 6. Security Considerations

This document lists the OAM requirements for a DetNet domain and does not raise any security concerns or issues in addition to ones common to networking.

## 7. Acknowledgment

TBD

## 8. References

### 8.1. 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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 8.2. Informational References

- [I-D.fmm-nvo3-pm-alt-mark]  
Fioccola, G., Mirsky, G., and T. Mizrahi, "Performance Measurement (PM) with Alternate Marking in Network Virtualization Overlays (NVO3)", draft-fmm-nvo3-pm-alt-mark-02 (work in progress), June 2018.
- [I-D.ietf-bier-pmmm-oam]  
Mirsky, G., Zheng, L., Chen, M., and G. Fioccola, "Performance Measurement (PM) with Marking Method in Bit Index Explicit Replication (BIER) Layer", draft-ietf-bier-pmmm-oam-04 (work in progress), June 2018.
- [I-D.ietf-intarea-gue]  
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-05 (work in progress), December 2017.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-06 (work in progress), March 2018.



- [I-D.ietf-nvo3-vxlan-gpe]  
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-06 (work in progress), April 2018.
- [I-D.ietf-sfc-ioam-nsh]  
Brockners, F., Bhandari, S., Govindan, V., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., and R. Chang, "NSH Encapsulation for In-situ OAM Data", draft-ietf-sfc-ioam-nsh-00 (work in progress), May 2018.
- [I-D.ietf-sfc-oam-framework]  
Aldrin, S., Pignataro, C., Kumar, N., Akiya, N., Krishnan, R., and A. Ghanwani, "Service Function Chaining (SFC) Operation, Administration and Maintenance (OAM) Framework", draft-ietf-sfc-oam-framework-04 (work in progress), March 2018.
- [I-D.mirsky-ippm-hybrid-two-step]  
Mirsky, G., Lingqiang, W., and G. Zhui, "Hybrid Two-Step Performance Measurement Method", draft-mirsky-ippm-hybrid-two-step-00 (work in progress), February 2018.
- [I-D.mirsky-sfc-pmamm]  
Mirsky, G., Fioccola, G., and T. Mizrahi, "Performance Measurement (PM) with Alternate Marking Method in Service Function Chaining (SFC) Domain", draft-mirsky-sfc-pmamm-03 (work in progress), June 2018.
- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.

- [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>>.
- [RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli,  
L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi,  
"Alternate-Marking Method for Passive and Hybrid  
Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321,  
January 2018, <<https://www.rfc-editor.org/info/rfc8321>>.
- [RFC8393] Farrel, A. and J. Drake, "Operating the Network Service  
Header (NSH) with Next Protocol "None"", RFC 8393,  
DOI 10.17487/RFC8393, May 2018,  
<<https://www.rfc-editor.org/info/rfc8393>>.

## Author's Address

Greg Mirsky  
ZTE Corp.

Email: [gregimirsky@gmail.com](mailto:gregimirsky@gmail.com)

RTGWG Working Group  
Internet-Draft  
Intended status: Informational  
Expires: August 29, 2020

G. Mirsky  
ZTE Corp.  
February 26, 2020

Identification of Overlay Operations, Administration, and Maintenance  
(OAM)  
draft-mirsky-rtgwg-oam-identify-04

Abstract

This document analyzes how the presence of Operations, Administration, and Maintenance (OAM) control command and/or special data is identified in some overlay networks and an impact on the choice of identification may have on OAM functionality.

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 August 29, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                      |    |
|------------------------------------------------------|----|
| 1. Introduction . . . . .                            | 2  |
| 2. Conventions used in this document . . . . .       | 2  |
| 2.1. Terminology . . . . .                           | 2  |
| 2.2. Keywords . . . . .                              | 3  |
| 3. A Control Channel in an Overlay Network . . . . . | 3  |
| 4. Overlay Network Encapsulations . . . . .          | 4  |
| 4.1. Encapsulations with Meta-data . . . . .         | 4  |
| 4.1.1. Available Solutions . . . . .                 | 6  |
| 4.2. Fixed-size Encapsulations . . . . .             | 6  |
| 4.3. Source Information Availability . . . . .       | 7  |
| 4.4. On-path OAM . . . . .                           | 7  |
| 5. Conclusions . . . . .                             | 8  |
| 6. IANA Considerations . . . . .                     | 8  |
| 7. Security Considerations . . . . .                 | 8  |
| 8. Acknowledgment . . . . .                          | 8  |
| 9. References . . . . .                              | 8  |
| 9.1. Normative References . . . . .                  | 9  |
| 9.2. Informational References . . . . .              | 9  |
| Author's Address . . . . .                           | 11 |

## 1. Introduction

Operations, Administration, and Maintenance (OAM) protocols are used to detect, localize defects in the network, and monitor network performance. Some OAM functions, e.g., failure detection, work in the network proactively, while others, e.g., defect localization, usually performed on-demand. These tasks achieved by a combination of active, passive, and hybrid OAM methods, as defined in [RFC7799].

This document analyzes how the presence of Operations, Administration, and Maintenance (OAM) control command and/or special data, i.e., OAM packet, is identified in some overlay networks, and an impact the choice of identification may have on OAM functionality of active and hybrid OAM methods for the respective overlay network encapsulation.

## 2. Conventions used in this document

## 2.1. Terminology

AMM Alternate Marking method

BIER Bit Indexed Explicit Replication

DetNet Deterministic Networks

GUE Generic UDP Encapsulation

HTS Hybrid Two-step

NSH Network Service Header

NVO3 Network Virtualization Overlays

OAM Operations, Administration and Maintenance

SFC Service Function Chaining

TLV Type-Length-Value

VXLAN-GPE Generic Protocol Extension for VXLAN

ACH Associated Channed Header

Underlay Network or Underlay Layer: The network that provides connectivity between the DetNet nodes. MPLS network that provides LSP connectivity between DetNet nodes is an example of an underlay layer.

## 2.2. Keywords

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. A Control Channel in an Overlay Network

There's a need for a general control channel between the endpoints of an overlay network for OAM protocols that can be used for fault detection, diagnostics, maintenance, and other functions. Such a control tunnel is dedicated to carrying only control and management data between tunnel endpoints. In other words, the control channel of an overlay network SHOULD NOT carry the client's data. And the endpoint node SHOULD NOT forward a packet received over the control channel. The identification of the control channel might be using different methods. For example, Virtual Network Identifier might be used to identify the control channel in VXLAN and Geneve.

#### 4. Overlay Network Encapsulations

New overlay network encapsulations analyzed in two groups:

- o encapsulations that support optional meta-data;
- o fixed-size encapsulations.

##### 4.1. Encapsulations with Meta-data

Number of the new encapsulation protocols (e.g., Geneve [I-D.ietf-nvo3-geneve], GUE [I-D.ietf-intarea-gue], and SFC NSH [RFC8300]) support use of Type-Length-Value (TLV) encoding to include optional information into the header. The identification of OAM in these protocols is as the following:

Geneve:

O (1 bit): after the WGLC discussion, the interpretation of the O field has changed. The O field now identifies a control packet. This packet contains a control message. Control messages are sent between tunnel endpoints. Tunnel Endpoints MUST NOT forward the payload and transit devices MUST NOT attempt to interpret it. Since these are infrequent control messages, it is RECOMMENDED that tunnel endpoints direct these packets to a high priority control queue (for example, to direct the packet to a general purpose CPU from a forwarding ASIC or to separate out control traffic on a NIC). Transit devices MUST NOT alter forwarding behavior on the basis of this bit, such as ECMP link selection.

[I-D.mmbb-nvo3-geneve-oam] defines the Geneve encapsulation for active OAM. Initially, four options have been presented:

- + with IP/UDP header demultiplexing active OAM protocols, e.g., Fault Management and Performance Monitoring, can be done using the destination UDP port number.
- + demultiplex active OAM protocols by the value of the Protocol Type field in the Geneve header.
- + with using MPLS Generic Associated Channel Label [RFC5586] and Associated Channel Header (ACH) [RFC4385]. Active OAM protocols are demultiplexed using the value of the Channel Type field.

- + using the new EtherType to identify Geneve OAM and the ACH. Active OAM protocols will be demultiplexed based on the Channel Type field's value.

#### GUE:

C-bit provides the separate namespace to carry formatted data that are implicitly addressed to the decapsulator to monitor or control the state or behavior of a tunnel. The payload is interpreted as a control message with the type specified in the proto/ctype field. The format and contents of the control message are indicated by the type and can be variable length.

#### SFC NSH:

0 bit: Setting this bit indicates an OAM packet.

Common between Geneve and NSH is the use of the dedicated flag to identify the OAM packet and, at the same time, the presence of the field that identifies the protocol of the payload that immediately follows after the encapsulation header. [RFC8393] points out that if the value of that field interpreted as none, i.e., no payload follows the header, then OAM may be included in TLVs, thus creating an active OAM packet. The problem with this mechanism to support active OAM methods may be a limitation of the size of data that can be included in a TLV. For example, the maximum size of data in an NSH Meta-data Type 2, as defined in section 2.5.1 [RFC8300], is 512 octets. The maximum length of data in Geneve Option, per section 3.5 [I-D.ietf-nvo3-geneve], is 128 octets. Thus, using one TLV as active OAM packet, would not allow creating test packets of larger size, which is useful when measuring packet loss and latency with synthetic traffic as part of the service activation procedure.

[I-D.ietf-sfc-oam-framework] suggests that the 0 bit used to identify OAM packet and the Next Protocol field identifies the OAM function:

While the presence of OAM marker in the overlay header (e.g., 0 bit in the NSH header) indicates it as OAM packet, it is not sufficient to signal for which OAM function the packet is intended.

At the same time, some of in-situ OAM proposals, e.g., [I-D.ietf-sfc-ioam-nsh], suggest using TLV to communicate hybrid OAM commands and data. The proposed resolution of using the combination of 0 bit and the Next Protocol field:

... the O bit MUST NOT be set for regular customer traffic which also carries IOAM data and the O bit MUST be set for OAM packets which carry only IOAM data without any regular data payload.

implies that the O bit only identifies the active OAM packet and not set when hybrid OAM methods used.

#### 4.1.1. Available Solutions

One of the possible solutions for encapsulations with meta-data has been specified in [I-D.ietf-sfc-multi-layer-oam]:

To identify the active OAM message the value on the Next Protocol field MUST be set to Active SFC OAM. The rules of interpreting the values of O bit and the Next Protocol field are as follows:

- o O bit set and the Next Protocol value is not one of identifying active or hybrid OAM protocol (per [RFC7799] definitions), e.g., defined in this specification Active SFC OAM - a Fixed-Length Context Header or Variable-Length Context Header(s) contain OAM command or data and the type of payload determined by the Next Protocol field;
- o O bit set and the Next Protocol value is one of identifying active or hybrid OAM protocol - the payload that immediately follows SFC NSH contains OAM command or data;
- o O bit is clear - no OAM in a Fixed-Length Context Header or Variable-Length Context Header(s) and the payload determined by the value of the Next Protocol field;
- o O bit is clear, and the Next Protocol value is one of identifying active or hybrid OAM protocol MUST be identified and reported as the erroneous combination. An implementation MAY have control to enable processing of the OAM payload.

From the above-listed rules follows the recommendation to avoid the combination of OAM in a Fixed-Length Context Header or Variable-Length Context Header(s) and in the payload immediately following the SFC NSH because there is no unambiguous way to identify such combination using the O bit and the Next Protocol field.

#### 4.2. Fixed-size Encapsulations

Number of the new encapsulation protocols (e.g., VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe], BIER [RFC8296]) use fixed-size header. The identification of OAM in these protocols is as the following:



**VXLAN-GPE:**

OAM Flag Bit (O bit): The O bit is set to indicate that the packet is an OAM packet.

**BIER:**

OAM packet identified by the value of the Next Protocol field. IANA BIER Next Protocol Identifiers registry includes the identifier for OAM (5).

The use of a combination of OAM Flag Bit and the Next Protocol field in VXLAN-GPE requires clarification of the header interpretation when the OAM Flag Bit is set, and the value of the Next Protocol field is one of defined in section 3.2 of [I-D.ietf-nvo3-vxlan-gpe].

BIER encapsulation, defined in [RFC8296], identifies OAM message immediately following the BIER header by the value of the Next Protocol field.

**4.3. Source Information Availability**

Availability of the packet originator's source information is required for active two-way OAM, e.g., echo request/reply. In cases when the underlay network is IPv4/IPv6 the source information will be derived from the underlay. But when using MPLS underlay network encapsulation of an active OAM packet have to follow specific rules:

- o if available, use Sender ID in the overlay domain (example BFIR ID in BIER [RFC8296];
- o use IP/UDP encapsulation of an OAM packet in the overlay (similar to Section 4.3 [RFC8029]).

**4.4. On-path OAM**

In addition to active methods, OAM toolset may include methods that don't use specially constructed and injected in the network test packets. [RFC7799] defines OAM methods that are neither entirely active nor passive but are a combination of both as hybrid methods.

One of the examples of the hybrid OAM methods, in-situ OAM, mentioned in Section 4.1. Another example, Alternate Marking method (AMM) [RFC8321], enables on-path OAM functions, e.g., delay and loss measurements, using the data traffic. Because AMM impact on the network can be minimized, measured metrics can be correlated to the network conditions experienced by the specific service. Of all listed in Section 4, BIER allocated the field that may be used for

AMM, as discussed in [I-D.ietf-bier-pmmm-oam]. Applicability of AMM to other overlay protocols, i.e., SFC NSH discussed in [I-D.mirsky-sfc-pmamm], Geneve [I-D.fmm-nvo3-pm-alt-mark], and in IPv6 networks [I-D.fioccola-v6ops-ipv6-alt-mark], been actively discussed.

Hybrid Two-step (HTS), defined in [I-D.mirsky-ippm-hybrid-two-step], provides on-path collection and transport of the telemetry information. HTS enables accurate and consistent measurements by separating the measurement action from the transporting data while ensuring that the follow-up packet that carries the telemetry information does follow the data packet that had triggered the measurement.

## 5. Conclusions

OAM control commands and data may be present as part of the overlay encapsulation header or as a payload that follows the overlay network header. The recommendations:

- o OAM in the overlay header, if supported by the overlay network, identified by the dedicated flag. Use of this method as active OAM is possible, but functionality is limited.
- o OAM that follows the overlay header identified as payload type, e.g., by the value of the Next Protocol field.

## 6. IANA Considerations

This document does not propose any IANA consideration. This section may be removed.

## 7. Security Considerations

This document lists the OAM requirements for a DetNet domain and does not raise any security concerns or issues in addition to ones common to networking.

## 8. Acknowledgment

TBD

## 9. References

### 9.1. 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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 9.2. Informational References

- [I-D.fioccola-v6ops-ipv6-alt-mark]  
Fioccola, G., Velde, G., Cociglio, M., and P. Muley, "IPv6 Performance Measurement with Alternate Marking Method", draft-fioccola-v6ops-ipv6-alt-mark-01 (work in progress), June 2018.
- [I-D.fmm-nvo3-pm-alt-mark]  
Fioccola, G., Mirsky, G., and T. Mizrahi, "Performance Measurement (PM) with Alternate Marking in Network Virtualization Overlays (NVO3)", draft-fmm-nvo3-pm-alt-mark-03 (work in progress), October 2018.
- [I-D.ietf-bier-pmmm-oam]  
Mirsky, G., Zheng, L., Chen, M., and G. Fioccola, "Performance Measurement (PM) with Marking Method in Bit Index Explicit Replication (BIER) Layer", draft-ietf-bier-pmmm-oam-07 (work in progress), January 2020.
- [I-D.ietf-intarea-gue]  
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-09 (work in progress), October 2019.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-14 (work in progress), September 2019.
- [I-D.ietf-nvo3-vxlan-gpe]  
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-09 (work in progress), December 2019.

- [I-D.ietf-sfc-ioam-nsh]  
Brockners, F. and S. Bhandari, "Network Service Header (NSH) Encapsulation for In-situ OAM (IOAM) Data", draft-ietf-sfc-ioam-nsh-02 (work in progress), September 2019.
- [I-D.ietf-sfc-multi-layer-oam]  
Mirsky, G., Meng, W., Khasnabish, B., and C. Wang, "Active OAM for Service Function Chains in Networks", draft-ietf-sfc-multi-layer-oam-04 (work in progress), November 2019.
- [I-D.ietf-sfc-oam-framework]  
Aldrin, S., Pignataro, C., Kumar, N., Krishnan, R., and A. Ghanwani, "Service Function Chaining (SFC) Operations, Administration and Maintenance (OAM) Framework", draft-ietf-sfc-oam-framework-11 (work in progress), September 2019.
- [I-D.mirsky-ippm-hybrid-two-step]  
Mirsky, G., Lingqiang, W., and G. Zhui, "Hybrid Two-Step Performance Measurement Method", draft-mirsky-ippm-hybrid-two-step-04 (work in progress), October 2019.
- [I-D.mirsky-sfc-pmamm]  
Mirsky, G., Fioccola, G., and T. Mizrahi, "Performance Measurement (PM) with Alternate Marking Method in Service Function Chaining (SFC) Domain", draft-mirsky-sfc-pmamm-09 (work in progress), December 2019.
- [I-D.mmbb-nvo3-geneve-oam]  
Mirsky, G., Xiao, M., Boutros, S., and D. Black, "OAM for use in GENEVE", draft-mmbb-nvo3-geneve-oam-01 (work in progress), January 2020.
- [RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <<https://www.rfc-editor.org/info/rfc4385>>.
- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <<https://www.rfc-editor.org/info/rfc5586>>.
- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.

- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.
- [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>>.
- [RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <<https://www.rfc-editor.org/info/rfc8321>>.
- [RFC8393] Farrel, A. and J. Drake, "Operating the Network Service Header (NSH) with Next Protocol "None"", RFC 8393, DOI 10.17487/RFC8393, May 2018, <<https://www.rfc-editor.org/info/rfc8393>>.

## Author's Address

Greg Mirsky  
ZTE Corp.

Email: [gregimirsky@gmail.com](mailto:gregimirsky@gmail.com)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: December 29, 2018

D. Trossen  
D. Purkayastha  
A. Rahman  
InterDigital Communications, LLC  
June 27, 2018

Name-Based Service Function Forwarder (nSFF) component within SFC  
framework  
draft-trossen-sfc-name-based-sff-00

## Abstract

Many stringent requirements are imposed on today's network, such as low latency, high availability and reliability in order to support several use cases such as IoT, Gaming, Content distribution, Robotics etc.

Adoption of cloud and fog technology at the edge of the network allows operator to deploy a single "Service Function" to multiple "Execution locations". The decision to steer traffic to a specific location may change frequently based on load, proximity etc. Under the current SFC framework, steering traffic dynamically to the different execution end points require a specific 're-chaining', i.e., a change in the service function path reflecting the different IP endpoints to be used for the new execution points. In order to address this, we discuss separating the logical Service Function Path from the specific execution end points. This can be done by identifying the Service Functions using a name rather than a routable IP endpoint (or Layer 2 address). This draft describes the necessary extensions to the various concepts and methods of SFC, specifically the SFF, in order to handle such name based relationships.

## 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 29, 2018.

#### Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

|                                                              |    |
|--------------------------------------------------------------|----|
| 1. Introduction . . . . .                                    | 2  |
| 2. Example use case: 5G control plane services . . . . .     | 4  |
| 3. Background . . . . .                                      | 5  |
| 3.1. Basic Definitions . . . . .                             | 5  |
| 3.2. Chaining of Service Functions . . . . .                 | 7  |
| 3.3. Operations for Traversing SFCs . . . . .                | 8  |
| 3.4. Operations inside an SFF . . . . .                      | 8  |
| 4. Challenges with current framework . . . . .               | 11 |
| 5. Name based operation in SFF . . . . .                     | 12 |
| 5.1. General Idea . . . . .                                  | 12 |
| 5.2. Name-Based Service Function Path (nSFP) . . . . .       | 12 |
| 5.3. Name Based Network Locator Map (nNLM) . . . . .         | 14 |
| 5.4. Name-based Service Function Forwarder (nSFF) . . . . .  | 16 |
| 5.4.1. Lifecycle of a Packet . . . . .                       | 17 |
| 5.4.2. Determining Suitable Forwarding Information . . . . . | 21 |
| 5.4.3. Layered View of Realizing nSFF . . . . .              | 23 |
| 6. IANA Considerations . . . . .                             | 24 |
| 7. Security Considerations . . . . .                         | 24 |
| 8. Informative References . . . . .                          | 24 |
| Authors' Addresses . . . . .                                 | 25 |

#### 1. Introduction

The requirements on today's networks are very diverse, enabling multiple use cases such as IoT, Content Distribution, Gaming, Network functions such as Cloud RAN. Every use case imposes certain requirements on the network. These requirements vary from one extreme to other and often they are in a divergent direction. Network operator and service providers are pushing many functions

towards the edge of the network in order to be closer to the users. This reduces latency and backhaul traffic, as user request can be processed locally.

It becomes more challenging for the network when user mobility as well as non-deterministic availability of compute and storage resources are considered. The impact is felt most at the edge of the network because as users move, their point of attachment changes frequently, which results in (at least partially) relocating the service as well as the service endpoint. Furthermore, network functions are pushed more and more towards the edge, where compute and storage resources are constrained and availability is non-deterministic.

In such a dynamic network environment, the capability to dynamically compose new services from available services as well as move a service instance in response to user mobility or resource availability is desirable.

The Service Function Chaining (SFC) framework [RFC7665] allows network operators as well as service providers to compose new services by chaining individual "Service Functions". Such chains are expressed through explicit relationships of functional components (the service functions), realized through their direct Layer 2 (e.g., MAC address) or Layer 3 (e.g., IP address) relationship as defined through next hop information that is being defined by the network operator, see Section 3 for more background on SFC.

A dynamic service environment as the one outlined above, i.e. utilizing edge-based services, requires that service end points are created and recreated frequently. Within the SFC framework, it means to reconfigure the existing chain through information based on the new relationships, causing overhead in a number of components, specifically the orchestrator that initiates the initial service function chain and any possible reconfiguration.

This document describes how such changes can be handled without involving the initiation of new and reconfigured SFCs by embedding the necessary functionality directly into the Service Function Forwarder. In other words, the SFF described here allows for keeping an existing SFC intact, as described by its service function path (SFP), while enabling the selection of an appropriate service function endpoint(s) during the traversal of packets through the SFC.



## 2. Example use case: 5G control plane services

As stated above, we can formulate the problem of keeping a service function chain intact while providing selection of the specific service function endpoints that are traversed at any point in time. We exemplify the need for such a solution through a use case stemming from the current 3GPP Rel 16 work on Service Based Architecture (SBA) [3GPP\_SBA], [3GPP\_SBA\_ENHANCEMENT]. In this work, a change in designing mobile network control planes is motivated by replacing the traditional network function interfaces with a fully service-based one. HTTP was chosen as the application layer protocol for exchanging suitable service requests [3GPP\_SBA]. With this in mind, the exchange between, say the session management function (SMF) and the authentication management function (AMF) in a 5G control plane is being described as a set of web service like requests which are in turn embedded into HTTP requests. Hence, interactions in a 5G control plane can be modelled based on service function chains where the relationship is between the specific (IP-based) service function endpoints that implement the necessary service endpoints in the SMF and AMF.

Motivating the move from a network function model (in pre-Rel 15 systems of 3GPP) to a service-based model is the proliferation of data center operations for mobile network control plane services. In other words, typical IT-based methods to service provisioning, in particular that of virtualization of entire compute resources, are envisioned to being used in future operations of mobile networks. Hence, operators of such future mobile network desire to virtualize service function endpoints and direct (control plane) traffic to the most appropriate current service instance. 'Appropriate' here can be defined by topological or geographical proximity of the service initiator to the service function endpoint. Alternatively, network or service instance compute load can be used to direct a request to a more appropriate (in this case less loaded) instance to reduce possible latency of the overall request. Such data centre centric operation is extended with the trend towards regionalization of load through a 'regional office' approach, where micro data centres provide virtualizable resources that can be used in the service execution, creating a larger degree of freedom when choosing the 'most appropriate' service endpoint for a particular incoming service request.

While the move to a service-based model aligns well with the framework of SFC, choosing the most appropriate service instance at runtime requires so-called 're-chaining' of the SFC since the relationships in said SFC are defined through Layer 2 or 3 identifiers, which in turn are likely to be different if the chosen

service instances reside in different parts of the network (e.g., in a regional data center).

### 3. Background

#### 3.1. Basic Definitions

We first start with few basic definitions, according to [RFC7665]:

- o A Service Function (SF) is responsible for specific treatment of received packets. A Service Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). As a logical component, a service function can be realized as a virtual element or be embedded in a physical network element.
- o The ordered series of Service Functions is defined through the Service Function Chain (SFC).
- o Service Function Path (SFP) defines the specific execution of an SFC in specific Service Function instances. The SFP reflects the mechanism used by service chaining to express the result of applying more granular policy and operational constraints to the abstract requirements of a service chain (SFC).
- o The SFP is identified by the Service Path Identifier (SPI) and the Service index (SI).
- o A Network Locator Map exists at each SFF that translates the SPI and SI information into a Next Hop information within the SFC domain.

| SPI | SI  | Next Hop(s)        | Transport Encapsulation |
|-----|-----|--------------------|-------------------------|
| 10  | 255 | 192.0.2.1          | VXLAN-gpe               |
| 10  | 254 | 198.51.100.10      | GRE                     |
| 10  | 251 | 198.51.100.15      | GRE                     |
| 40  | 251 | 198.51.100.15      | GRE                     |
| 50  | 200 | 01:23:45:67:89:ab  | Ethernet                |
| 15  | 212 | Null (end of path) | None                    |

Figure 1: Network Locator Map in SFC

- o Network Service Header (NSH) [RFC8300] is a distinct identifiable plane that can be used across all transports to create a service chain and exchange metadata along the chain. A Network Service Header (NSH) contains service path information and optionally metadata that are added to a packet or frame and used to create a service plane. A control plane is required in order to exchange NSH values with participating nodes, and to provision the same nodes with requisite information such as service path ID to overlay mapping

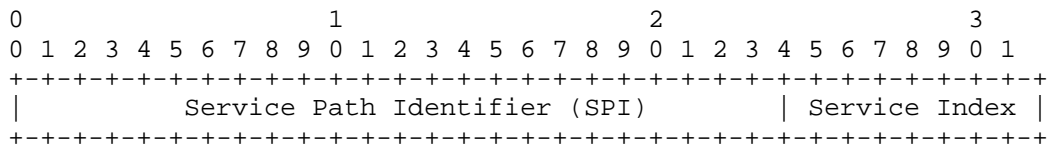


Figure 2: NSH Service Path Header

- o Service Classifier / Service Classification is responsible for directing incoming packets from a user or another network towards an identified SFP, therefore forwarding the packet to the respective service functions specified in the SFP. For this, the service classifier adds the NSH header, which contains the respective (SPI, SI) to identify the Service Function Path. Additional context information such as UserID, ApplicationID maybe added. Finally, the service classifier determines the suitable transport and adds transport encapsulation for delivery to the SFF.

- o Service Function Forwarder (SFF at the router) executes the following functions:
  - \* Remove Transport encapsulation
  - \* Inspect NSH header to find SPI/SI, determine the next SF1, which is the Firewall function in our example above.
  - \* Forward the Packet and NSH encapsulation to appropriate SF, here SF1 in our example

### 3.2. Chaining of Service Functions

When a traffic flow is forwarded over a service chain, packets in the traffic flow are processed by the various service function instances, with each service function instance applying a service function (e.g., firewall, network access translation (NAT), deep packet inspection (DPI), etc.) prior to forwarding the packets to the next network node. It is a Service layer concept and can possibly work over any Virtual network layer and an Underlay network, possibly IP and any Layer 2 technology. At the service layer, Service Functions are identified using a path identifier and an index. Eventually this index is translated to an IP address (or MAC address) of the host where the service function is running.

As an example, a user's request to access a video server from his home over a fixed network, may flow through several service functions over a service chain, as deployed by network operator. The CPE accepts the request, pass it on through a firewall, video optimizer and video server. These service functions define a Service Function Chain. The following diagram in Figure 3 shows the example Service Function Chain described here.

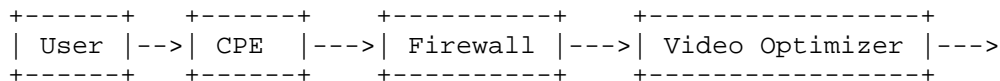


Figure 3: Mapping SFC onto Service Function Execution Points along a Service Function Path

[RFC7665] describes an architecture for the specification, creation, and ongoing maintenance of Service Function Chains (SFCs). It includes architectural concepts, principles, and components used in the construction of composite services through deployment of SFCs.

### 3.3. Operations for Traversing SFCs

The following diagram in Figure 4 fits the architectural concepts and operation of SFC to the use case described above. How packets are encapsulated and de-capsulated, as it traverses through the service functions, is also shown.

The CPE may act as a Service Classifier. The Service Function Forwarder (SFF) is the router or gateways in the network. SF1 represents the Firewall function. SF2 represents Video Optimization function and Video server function.

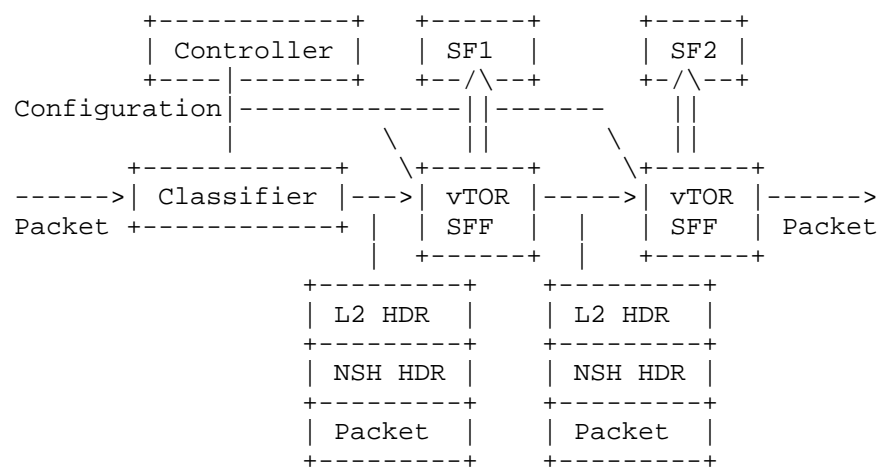


Figure 4: Processing of an SFC Packet along an SFP

### 3.4. Operations inside an SFF

The following diagram in Figure 5 describes the functions inside an SFF.

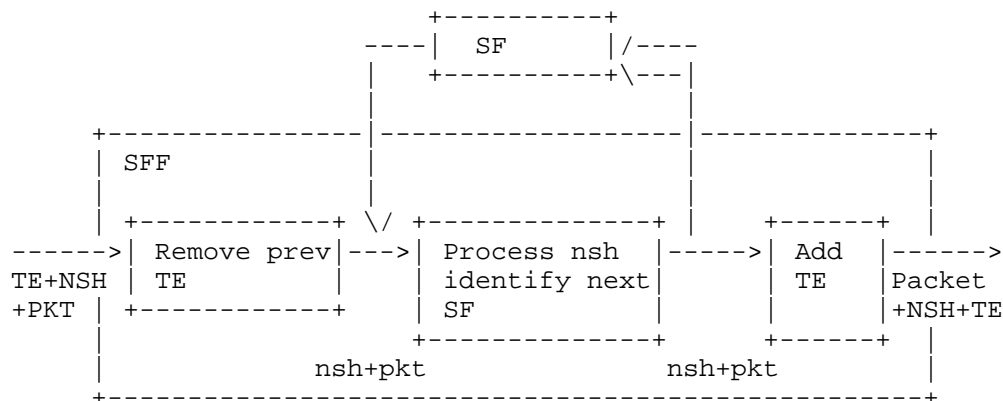


Figure 5: Processing an incoming SFC Packet in an SFF

Service Function Forwarder (SFF at the router) executes the following functions:

- o Remove TE (RTE): SFF first removes the Transport Encapsulation (TE) and forwards the Packet with NSH header to the next sub-function
- o Process NSH (PNSH):
  - \* Based on the NSH header, a possibly local service function is identified as next SF. If it is reachable from the same SFF, the SFF then forwards the packet. The packet is processed by service function. It is returned to SFF with updated NSH header. SFF processes the NSH header again to identify next SF (e.g., the Video Optimization and Delivery function in Figure 3. This SF is not reachable by the same SFF, it needs to be sent to the next SFF.
  - \* The "Process NSH (PNSH)" sub-function may process the following information: NSH header, Service Function Graph, Network locator map available at SFF
  - \* The SPI (Service Path ID) and SI (Service Index) from NSH header may be used to identify the current position in the Service Function Graph [RFC7665]. From that, PNSH function determines the next Service Function, which needs to be reached.

- \* As an example, let us examine the following Service Function Graphs available at the SFF node. Each graph may be identified/associated with a SPI.

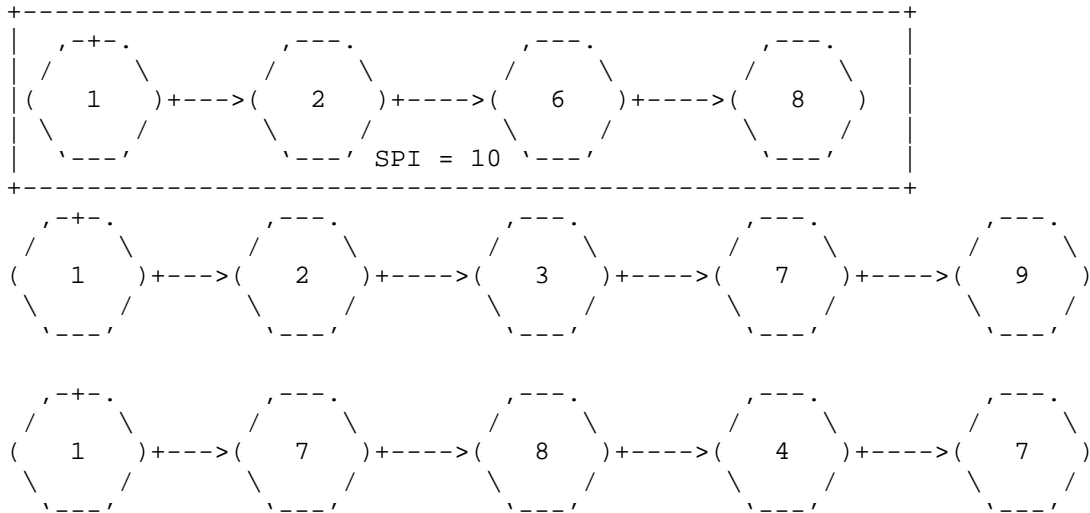


Figure 6: Example of mapping SFCs onto SPI/SI Information

- \* Let us assume that PNSH determines from NSH header that SPI = 10 and SI = 254. SPI value of 10 indicates that the first graph (inside dotted line) in the above diagram is referenced. The SI value indicates the location in the graph. Generally at the beginning of the graph, SI is set to 255 and decremented by 1 as the packet traverses each node. In this case, e.g. a value of 254 indicates that 1 nodes (e.g. Firewall function) have been traversed and the next SF to reach is 2 (e.g. the Video Optimizer / Video delivery function).
- \* Next SPI may be used as an index in the Network Locator Map [RFC8300] (see Figure 1) to identify the next hop .
- \* SPI = 10 and SI = 254 indicates next hop is 192.51.100.10. It indicates to reach the next SF, the packet needs to reach the host with the address and use GRE as Layer 2 encapsulation.
- o Add TE (ATE): The third sub-function is to identify the Transport encapsulation (TE) for the Packet and NSH. The transport encapsulation is added to the packet and NSH. SFF puts the encapsulated packet in the network to be forwarded to the next SF.

Service functions (SF, SF1, ....SFn) act on an incoming packet based on the NSH context information. E.g. SF1, i.e. Firewall function apply policy based on UserID and decides if the packet should be forwarded or not. The SF then decrements SI, add the NSH back into packet and forwards to SFF. The local SFF then looks into NSH and forwards in a similar way to next service function.

The SFC architecture [RFC7665] does not mandate how the SFP is compiled or how SFCs are defined in general. For the compilation of SFC, it is envisioned that the network operator identifies the Service Functions (SFs), that a packet need to traverse, to compose or offer a specific service. Other alternatives are the specification of SFCs through standardization bodies, reflecting a standardized exchange that represent a standardized service that can be deployed in operator network.

It is envisioned that the network operator or the service provider defines the Service Function Path for a given SFC. The specific SFP can be decided based on policy, network deployment, services to be offered, how a packet related to an user or a service needs to be processed or served. Through the selection of the specific SFP for a given SFC, the network operator has full control over how to realize the given SFC.

For the distribution of the SFC forwarding information, i.e., the Network Locator Map, a controller is envisioned to provide the necessary information to the SFC nodes (i.e., the classifier and SFF). The configuration information and provisioning may be done using Netconf, CLI and it includes "Network Locator Map". This configuration information may be provided to the SFC nodes during initialization, periodic refresh or when required. Realizations of such 'controller' could be a Northbound SDN controller application.

It is this management level distribution of configuration information, such as the Next Hop information to the SFF nodes, that impede flexibility by requiring any change to, for instance, Next Hop information to be distributed to the suitable SFF node in cases of changing the specific Service Function instance which should be used for the next transactions traversing this SFF node. In the following chapters, we formalize this problem and provide a solution for it.

#### 4. Challenges with current framework

As outlined in Section 3, the Service Function Path defines an ordered sequence of specific Service Functions instances being used for the interaction between initiator and service functions along the SFP. These service functions are addressed by IP addresses and



defined as next hop information in the network locator maps of traversing SFF nodes.

As outlined in the use case, however, the service provider may want to provision SFC nodes based on dynamically spun up service function instances so that these (now virtualized) service functions can be reached in the SFC domain using the SFC underlay layer.

Following the original model of SFC, any change in a specific execution points for a specific Service Function along the SFP will require a change of the SFP information (since the new service function execution points likely carries different IP or L2 address information) and possibly even the Next Hop information in SFFs along the SFP. In case the availability of new service function instances is dynamic (e.g., through the use of container-based virtualization techniques), the current model and realization of SFC reduces the flexibility of the service providers and increases the management complexity incurred by the frequent changes of (service) forwarding information in the respective SFF nodes. This is because any change of the SFP (and possibly next hop info) will need to go through suitable management cycles.

The next section outlines a solution to address the issue, allowing for keeping SFC information (represented in its SFP) intact while addressing the flexibility desire of the service provider.

## 5. Name based operation in SFF

### 5.1. General Idea

The general idea is two-pronged. Firstly, we elevate the definition of a Service Function Path onto the level of 'name-based interactions' rather than limiting SFPs to Layer 3 or Layer 2 information only. Secondly, we extend the operations of the SFF to allow for forwarding decisions that take into account such name-based interaction while remaining backward compatible to the current SFC architecture [RFC7665]. In the following sections, we outline these two components of our solution.

### 5.2. Name-Based Service Function Path (nSFP)

In the existing SFC framework [2], as outlined in Section 3, the SFP information encapsulates forwarding information based on Layer 2 or 3 information, i.e., MAC or IP addresses. We introduce the notion of a 'name-based service function path (nSFP)'.

In today's networking terms, any identifier can be treated as a name. The realization of "Name based SFP" through extended SFF operations

(see Section 5.4) is illustrated using HTTP transactions. Here, URIs are being used to name for a Service Function along the nSFP. It is to be noted that the Name based SFP approach is not restricted to HTTP (as the protocol) and URIs (as next hop identifier within the SFP). Other identifiers such as an IP address itself can also be used and are interpreted as a 'name' in the nSFP. With this, our notion of the nSFP goes beyond the initial proposals made in [7], which limited the notion of a 'name' to a URL (uniform resource locator), commonly used in the addressing of HTTP requests. In other words, IP addresses as well as fully qualified domain names forming complex URIs (uniform resource identifiers), such as `www.foo.com/service_name1`, are all captured by the notion of 'name' in this draft.

Generally, nSFPs are defined as an ordered sequence of the "name" of Service Functions (SF) and a typical name-based Service Function Path may look like: `192.168.x.x -> www.foo.com -> www.foo2.com/service1 -> www.foo2.com/service2`

Our use case in Section 2 can then be represented as an ordered named sequence. An example for a session initiation that involves an authentication procedure, this could look like `192.168.x.x -> smf.3gpp.org/session_initiate -> amf.3gpp.org/auth -> smf.3gpp.org/session_complete -> 192.168.x.x`

In accordance with our use case in Section 2, any of these named services can potentially be realized through more than one replicated SF instances. This leads to make dynamic decision on where to send packets along the SAME service function path information, being provided during the execution of the SFC. Through elevating the SFP onto the notion of name-based interactions, the SFP will remain the same even if those specific execution points change for a specific service interaction.

The following diagram describes this name-based SFP concept and the resulting mapping of those named interactions onto (possibly) replicated instances.

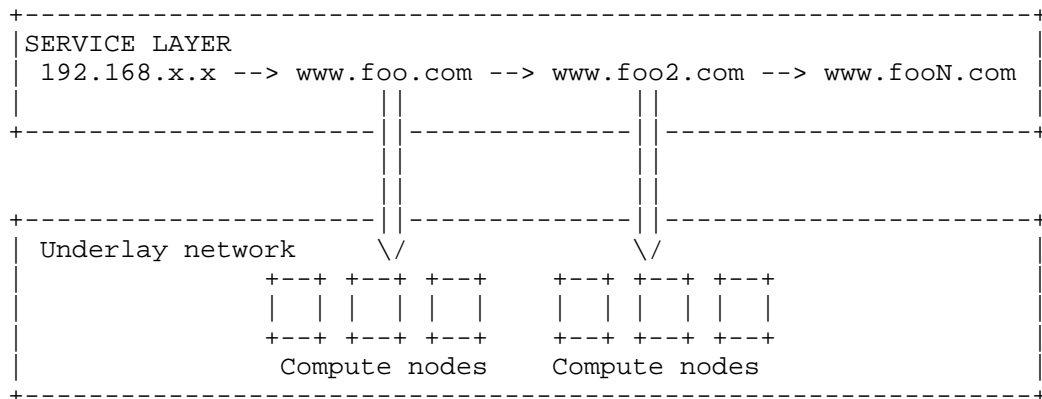


Figure 7: Mapping SFC onto Service Function Execution Points along a Service Function Path based on Virtualized Service Function Instance

### 5.3. Name Based Network Locator Map (nNLM)

In order to forward a name-based SFC, we need to extend the network locator map as originally defined in [RFC8300] with the ability to consider name relations based on URIs as well as high-level transport protocols such as HTTP for mean of packet forwarding.

The extended Network Locator Map or name-based Network Locator Map (nNLM) is shown in Figure 8 as an example for www.foo.com being part of the nSFP. Such extended nNLM is stored at each SFF throughout the SFC domain with suitable information populated to the nNLM during the configuration phase.

| SPI | SI  | Next Hop(s)        | Transport Encapsulation |
|-----|-----|--------------------|-------------------------|
| 10  | 255 | 192.0.2.1          | VXLAN-gpe               |
| 10  | 254 | 198.51.100.10      | GRE                     |
| 10  | 253 | www.foo.com        | HTTP                    |
| 40  | 251 | 198.51.100.15      | GRE                     |
| 50  | 200 | 01:23:45:67:89:ab  | Ethernet                |
| 15  | 212 | Null (end of path) | None                    |

Figure 8: Name-based Network Locator Map

Alternatively, the extended network locator map may be defined with implicit name information rather than explicit URIs as in Figure 8. In the example of Figure 9 below, the next hop is represented as a generic HTTP service without a specific URI being identified in the extended network locator map. In this scenario, the SFF forwards the packet based on parsing the HTTP request in order to identify the host name or URI. It retrieves the URI and may apply policy information to determine the destination host/service.

| SPI | SI  | Next Hop(s)        | Transport Encapsulation |
|-----|-----|--------------------|-------------------------|
| 10  | 255 | 192.0.2.1          | VXLAN-gpe               |
| 10  | 254 | 198.51.100.10      | GRE                     |
| 10  | 253 | HTTP Service       | HTTP                    |
| 40  | 251 | 198.51.100.15      | GRE                     |
| 50  | 200 | 01:23:45:67:89:ab  | Ethernet                |
| 15  | 212 | Null (end of path) | None                    |

Figure 9: Name-based Network Locator Map with Implicit Name information

#### 5.4. Name-based Service Function Forwarder (nSFF)

While [I-D.purkayastha-sfc-service-indirection] outlined the realization of forwarding packets in URL-based interaction through HTTP via a specific function (called Service Request Routing in [I-D.purkayastha-sfc-service-indirection]), it is desirable to extend the SFF of the SFC underlay in order to handle nSFPs transparently and without the need to insert a special (SRR) service function into the nSFP. Such extended name-based SFF would then be responsible for forwarding a packet in the SFC domain as per the definition of the (extended) nSFP.

In our exemplary realization for an extended SFF, the solution described in this document uses HTTP transactions, i.e., HTTP as an example bearer protocol for the next hop information. The URI in the HTTP transaction are the names in our nSFP information, which will be used for name based forwarding, while the HTTP requests are encoded in plain text, e.g.,

```
GET / HTTP/1.1 Host: www.foo.com Accept-Language: en
```

Following our reasoning so far, HTTP requests (and more specifically the plain text encoded requests above) are the equivalent of Packets that enter the SFC domain. The handling of any intermediate layers such as TCP, IP is left to the realization of the (extended) SFF operations towards the next (named) hop. For this, we will first outline the general lifecycle of an SFC packet in the following

subsection, followed by two examples for determining next hop information in Section 5.4.2, finalized by a layered view on the realization of the nSFF in Section 5.4.3.

#### 5.4.1. Lifecycle of a Packet

With reference to the SFC architecture, the lifecycle of packet is shown in Figure 10 below, following the steps outlined in Section 3:

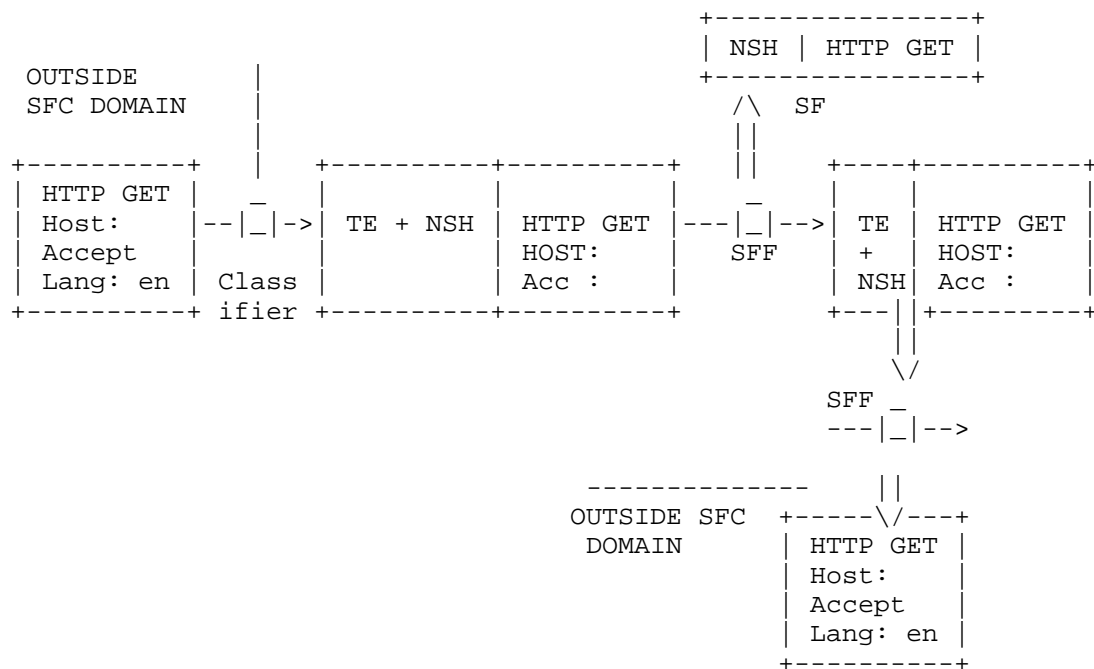


Figure 10: Lifecycle of an SFC Packet traversing SFC Domain

It is also possible that within an SFC domain there may be mix of routing decisions at the SFF. Conceptually, all routing is done by interpreting the SFP information as 'names'. The routing decisions themselves are either based on well-known, e.g., IP routing methods, or they utilize methods such as those outlined in [EuCNC] (for IP-addressed SFC packets) or [H2020POINT] (for URI-addressed SFC packets).

In the proposed solution, the respective operations shown in Figure 5 of Section 3.4 are realized by the name-based SFF, called nSFF in the following. The following diagram in Figure 11 describes the traversal between two service function instances of an nSFP-based

transactions in an example chain of: 192.168.x.x -> SF1 (www.foo.com)  
-> SF2 (www.foo2.com) -> SF3 -> ...

According to the lifecycle in Figure 10 and based on our example chain above, the following figure shows an example traffic flow in an nSFP with nSFF and nNLM.

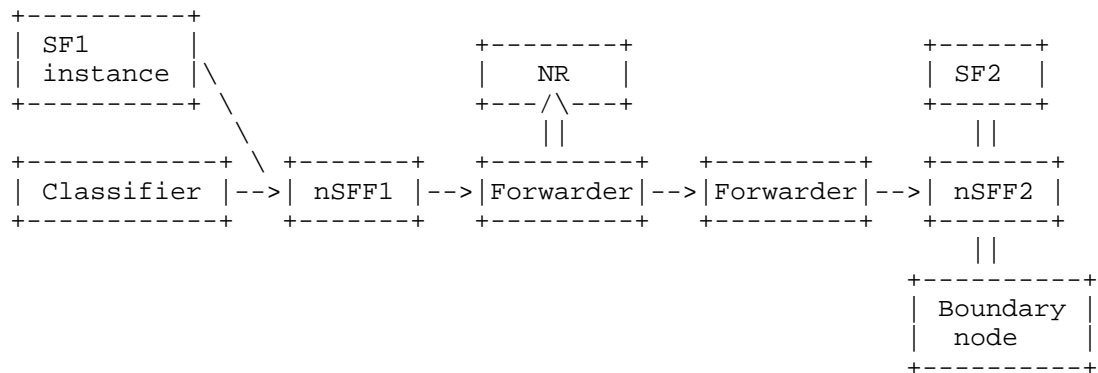


Figure 11: Operations at extended SFF

Traffic originates from a Classifier or another SFF on the left. Traffic is processed by the incoming nSFF1 (on the left side) through the following steps. The operations described in Figure 4 are extended and the traffic exits at nSFF2:

- o Step 1: At nSFF1 the nNLM in Figure 12 is assumed

| SPI | SI  | Next Hop(s)        | Transport Encapsulation |
|-----|-----|--------------------|-------------------------|
| 10  | 255 | 192.0.2.1          | VXLAN-gpe               |
| 10  | 254 | 198.51.100.10      | GRE                     |
| 10  | 253 | www.foo.com        | HTTP                    |
| 10  | 252 | www.foo2.com       | HTTP                    |
| 40  | 251 | 198.51.100.15      | GRE                     |
| 50  | 200 | 01:23:45:67:89:ab  | Ethernet                |
| 15  | 212 | Null (end of path) | None                    |

Figure 12: nNLM at SFF1

- o Step 2: nSFF1 removes the previous transport encapsulation (TE) for any traffic originating from another SFF or classifier (traffic from an SF instance does not carry any TE and is therefore directly processed at the nSFF).
- o Step 3: nSFF1 then processes the NSH information to identify the next SF at the nSFP level by mapping the NSH information to the appropriate entry in its nNLM (see Figure 12) based on the provided SPI/SI information in the NSH (see Section 3) in order to determine the name-based identifier of the next hop SF. With such nNLM in mind, the nSFF searches the map for SPI = 10 and SI = 253. It identifies the next hop as = www.foo.com and HTTP as the transport. Given the next hop resides locally, the SFC packet is forwarded to the SF1 instance of www.foo.com (see Figure 11).
- o Step 4: The SF1 instance then processes the received SFC packet according to its service semantics and modifies the NSH by setting SPI = 10, SI = 252 for forwarding the packet along the SFP. It then forwards the SFC packet to its local nSFF, i.e., nSFF1.
- o Step 5: nSFF1 processes the NSH of the SFC packet again, now with the NSH modified (SPI = 10, SI = 252) by the SF1 instance. It retrieves the next hop information from its nNLM in Figure 12, to be www.foo2.com. Due to this SF not being locally available, the nSFF consults any locally available information regarding the routing/forwarding information towards a suitable nSFF that can serve this next hop.



- o Step 6: If such information exists, the Packet (plus the NSH information) is marked to be sent towards the nSFF serving the next hop based on such information in step 8.
- o Step 7: If such information does not exist, nSFF1 consults the Name Resolver (NR) to determine the suitable routing/forwarding information towards the identified nSFF serving the next hop of the SFP. For future SFC packets towards this next hop, such resolved information may be locally cached, avoiding to contact the Name Resolver for every SFC packet forwarding. The packet is now marked to be sent via the network described in next step 8.
- o Step 8: Utilizing the forwarding information determined in steps 6 or 7, nSFF1 adds the suitable transport encapsulation (TE) for the SFC packet before forwarding via the forwarders in the network towards the next nSFF i.e. nSFF2.
- o Step 9: When the packet (+NSH+TE) arrives at the outgoing nSFF2, i.e., the nSFF serving the identified next hop of the SFP, it removes the TE and processes the NSH to identify the next hop information. At nSFF2 the nNLM in Figure 13 is assumed. Based on the nNLM in Figure 13 and NSH information where SPI = 10 and SI = 252, nSFF2 identifies the next SF as www.foo2.com.

| SPI | SI  | Next Hop(s)        | Transport Encapsulation |
|-----|-----|--------------------|-------------------------|
| 10  | 252 | www.foo2.com       | HTTP                    |
| 40  | 251 | 198.51.100.15      | GRE                     |
| 50  | 200 | 01:23:45:67:89:ab  | Ethernet                |
| 15  | 212 | Null (end of path) | None                    |

Figure 13: nNLM at SFF2

- o Step 10: If the next hop is locally registered at the nSFF, it forwards the packet (+NSH) to the service function instance, using suitable IP/MAC methods for doing so.
- o Step 11: Otherwise, the outgoing nSFF adds a new TE information to the packet and forwards the packet (+NSH+TE) to the next SFF or boundary node, as shown in Figure 11.

#### 5.4.2. Determining Suitable Forwarding Information

The extended operations of the nSFF rely on determining suitable forwarding information in step 3, 5, 6 and 7 as well as adding this forwarding information as traffic encapsulation in steps 8 and 10 of the previous section. We describe details of the extended nSFF operations in the following sections.

##### 5.4.2.1. DNS-based Resolution

The SFF-Name Resolver (NR) communication in Figure 11, i.e., step 7, may be realized through a DNS lookup towards the next hop information derived from the (extended) network locator map.

Hence, the NR is the first DNS resolver configured at the nSFF for any such lookup. The NR will return the suitable IP address of the SF instance provided in the lookup. The NR may also propagate any received lookup in case a NR-local setup cannot be done. Examples for the protocol used between nSFF and NR are the available DNS protocol.

Upon receiving the NR response, the nSFF uses IP-based communication to send the SFC packet to the next hop, i.e., IP is being used for the traffic encapsulation of steps 8 and 10, while the forwarders in Figure 11 are standard IP routers.

For registration of the local SF instances at each nSFF, methods such as management consoles (mapping IP addresses onto DNS names) or DNS registrations (with nSFF intercepting such registrations) can be used to build an nSFF-local knowledge of all locally available service functions at the level of a named function.

One problem arises, however, with caching NR (i.e., Name Resolver) responses in step 5 since this requires dealing with stale DNS entries when such DNS entries might change due to new instances of service functions becoming available

##### 5.4.2.2. IP/HTTP-over-ICN

The methods in [EuCNC], [H2020POINT] describe steps to interpret IP or HTTP level communication from an incoming service instance (residing on, for instance, an IP-based device such as an UE or a server or a network component) as ICN (information-centric networking) based communication.

Here, the NR in Figure 11 is implemented by the functions of Rendezvous (RV) and Topology Management (TM) in [EuCNC],

[H2020POINT]. The nSFF in Figure 11 is implemented by the methods in the Network Attachment Point (NAP) in [EuCNC], [H2020POINT].

Hence, after having removed any transport encapsulation of the incoming SFC packet (in case of receiving the packet from a classifier) and the determination of an SFC packet as being sent to another SFF (see steps of the nSFF described earlier), the nSFF publishes a suitable ICN packet with the HTTP or IP packet (depending on the next hop information in the nNLM) as well as the NSH as payload.

If no suitable local forwarding information exists at the nSFF for this publication, a path computation request is sent to the NR (i.e., the combination of RV and TM in [EuCNC], [H2020POINT] ) and suitable forwarding information is returned to the nSFF. This forwarding information is used to publish the SFC packet (i.e., either an IP packet or an HTTP plain text request, depending on the next hop information in the nNLM) to the suitable nNLM by using this forwarding information as traffic encapsulation towards the next nSFF. As the next nSFF for the specific next hop information, said nSFF has previously subscribed to the named SFP information, e.g., www.foo.com or the next hop IP address. It will therefore receive the forwarded SFC packet + NSH, published in the previous step.

For the forwarding information provided by the NR in the initial publication, solutions such as those in [Reed2016] can be used. Here, the IPv6 source and destination addresses are used to realize a so-called path-based forwarding from the incoming to the outgoing nSFF. The forwarders in Figure 11 are realized via SDN (software-defined networking) switches, implementing an AND/CMP operation based on arbitrary wildcard matching over the IPv6 source and destination addresses, as outlined in [Reed2016] . As described in step 8 of the extended nSFF operations, this forwarding information is used as traffic encapsulation. With the forwarding information utilizing existing IPv6 information, IP headers are utilized as TE in this case. The next hop nSFF (see Figure 11) will restore the IP header of the packet with the relevant IP information used to forward the SFC packet to SF2 or it will create a suitable TE information to forward the information to another nSFF or boundary node.

Alternatively, the solution in [I-D.purkayastha-bier-multicast-http] suggests using a so-called BIER (Binary Indexed Explicit Replication) underlay. Here, the nSFF would be realized at the ingress to the BIER underlay, injecting the SFC packet (plus the NSH) header with BIER-based traffic encapsulation into the BIER underlay with each of the forwarders in Figure 11 being realized as a so-called Bit-Forwarding Router (BFR) [RFC8279].

Various registration methods can be used to bind next hop information to a specific nSFF, such as a management console, DNS interception, or a local registration interface. All of those methods eventually result in a suitable subscription of the outgoing nSFF to the next hop information, this subscription being sent to the NR, which in turns allows the NR to determine suitable forwarding from an incoming nSFF to said outgoing (next hop) nSFF.

#### 5.4.3. Layered View of Realizing nSFF

In Figure 14 below, we present a layered view on realizing the nSFF with a focus on the layers that are present at the incoming and outgoing nSFF based on well known protocols being used but not limiting to those only. For the sake of simplicity, we here assume SDN being used as a Layer 2 technology although the implementation approach equally applies to a BIER-based approach. We also show only a sub-chain from one SF to another, leaving out the reception from another SFF at the incoming SFF.

Here we assume the use of HTTP as the bearer protocol over the operator (transport) network. As indicated in Section 5.2, one could also realize named services that would normally run over a TCP-based connection only, such as a BitTorrent type of protocol with named chunks of content.

As also shown in the figure, IP-level SFCs can also be realized via this implementation approach, utilizing methods such as those in [EuCNC] by interpreting IP addresses as names. With this in mind, the HTTP, TCP or IP level transactions originating from SF1 on the left of Figure 12 are terminated at the incoming nSFF and then mapped onto the Layer 2 exchange over the transport network, while the right nSFF restores the suitable transaction and forwards the SFC packet to SF2.

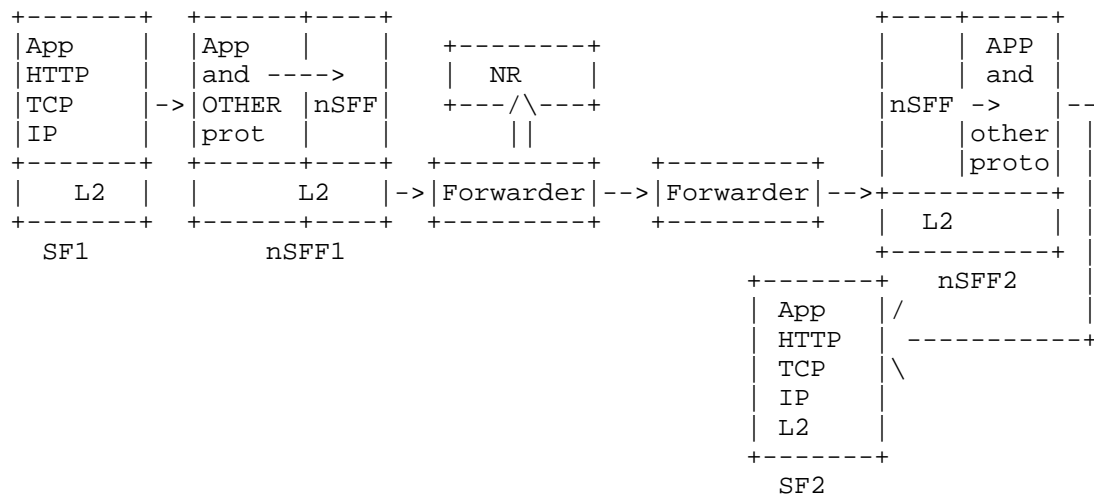


Figure 14: Layered View of Realizing a Service Router as Part of extended SFF

## 6. IANA Considerations

This document requests no IANA actions.

## 7. Security Considerations

TBD.

## 8. Informative References

### [\_3GPP\_SBA]

3GPP, "Technical Realization of Service Based Architecture", 3GPP TS 29.500 0.4.0, January 2018, <<http://www.3gpp.org/ftp/Specs/html-info/29500.htm>>.

### [\_3GPP\_SBA\_ENHANCEMENT]

3GPP, "New SID for Enhancements to the Service-Based 5G System Architecture", 3GPP S2-182904 , February 2018, <[http://www.3gpp.org/ftp/tsg\\_sa/WG2\\_Arch/TSGS2\\_126\\_Montreal/Docs/S2-182904.zip](http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_126_Montreal/Docs/S2-182904.zip)>.

### [EuCNC]

Trossen, D. and et al., "POINT: IP Over ICN - The Better IP?", European Conference on Networks and Communications (EuCNC), , 2015.

## [H2020POINT]

EU, "H2020 POINT project, First Platform Design", D 3.1, 2015, <[https://www.point-h2020.eu/wp-content/uploads/2015/12/POINT\\_D3.1-First\\_Platform\\_Design\\_v1.01.pdf](https://www.point-h2020.eu/wp-content/uploads/2015/12/POINT_D3.1-First_Platform_Design_v1.01.pdf)>.

## [I-D.purkayastha-bier-multicast-http]

Purkayastha, D., Rahman, A., and D. Trossen, "Multicast HTTP using BIER", draft-purkayastha-bier-multicast-http-00 (work in progress), March 2018.

## [I-D.purkayastha-sfc-service-indirection]

Purkayastha, D., Rahman, A., Trossen, D., Despotovic, Z., and R. Khalili, "Alternative Handling of Dynamic Chaining and Service Indirection", draft-purkayastha-sfc-service-indirection-02 (work in progress), March 2018.

## [Reed2016]

Reed, M., Al-Naday, M., Thomas, N., Trossen, D., and S. Spirou, "Stateless multicast switching in software defined networks", ICC 2016, 2016.

[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>>.

[RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <<https://www.rfc-editor.org/info/rfc8279>>.

[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>>.

## Authors' Addresses

Dirk Trossen  
InterDigital Communications, LLC  
64 Great Eastern Street, 1st Floor  
London EC2A 3QR  
United Kingdom

Email: [Dirk.Trossen@InterDigital.com](mailto:Dirk.Trossen@InterDigital.com)  
URI: <http://www.InterDigital.com/>

Debashish Purkayastha  
InterDigital Communications, LLC  
Conshohocken  
USA

Email: Debashish.Purkayastha@InterDigital.com

Akbar Rahman  
InterDigital Communications, LLC  
Montreal  
Canada

Email: Akbar.Rahman@InterDigital.com

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: November 27, 2019

D. Trossen  
InterDigital Europe, Ltd  
D. Purkayastha  
A. Rahman  
InterDigital Communications, LLC  
May 26, 2019

Name-Based Service Function Forwarder (nSFF) component within SFC  
framework  
draft-trossen-sfc-name-based-sff-07

## Abstract

Adoption of cloud and fog technology allows operators to deploy a single "Service Function" to multiple "Execution locations". The decision to steer traffic to a specific location may change frequently based on load, proximity etc. Under the current SFC framework, steering traffic dynamically to the different execution end points require a specific 're-chaining', i.e., a change in the service function path reflecting the different IP endpoints to be used for the new execution points. This procedure may be complex and take time. In order to simplify re-chaining and reduce the time to complete the procedure, we discuss separating the logical Service Function Path from the specific execution end points. This can be done by identifying the Service Functions using a name rather than a routable IP endpoint (or Layer 2 address). This document describes the necessary extensions, additional functions and protocol details in SFF (Service Function Forwarder) to handle name based relationships.

This document presents InterDigital's approach to name-based service function chaining. It does not represent IETF consensus and is presented here so that the SFC community may benefit from considering this mechanism and the possibility of its use in the edge data centers.

## 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 November 27, 2019.

#### Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

|                                                             |    |
|-------------------------------------------------------------|----|
| 1. Introduction . . . . .                                   | 3  |
| 2. Terminology . . . . .                                    | 4  |
| 3. Example use case: 5G control plane services . . . . .    | 4  |
| 4. Background . . . . .                                     | 6  |
| 4.1. Relevant part of SFC architecture . . . . .            | 6  |
| 4.2. Challenges with current framework . . . . .            | 7  |
| 5. Name based operation in SFF . . . . .                    | 8  |
| 5.1. General Idea . . . . .                                 | 8  |
| 5.2. Name-Based Service Function Path (nSFP) . . . . .      | 8  |
| 5.3. Name Based Network Locator Map (nNLM) . . . . .        | 10 |
| 5.4. Name-based Service Function Forwarder (nSFF) . . . . . | 12 |
| 5.5. High Level Architecture . . . . .                      | 13 |
| 5.6. Operational Steps . . . . .                            | 14 |
| 6. nSFF Forwarding Operations . . . . .                     | 16 |
| 6.1. nSFF Protocol Layers . . . . .                         | 16 |
| 6.2. nSFF Operations . . . . .                              | 18 |
| 6.2.1. Forwarding between nSFFs and nSFF-NR . . . . .       | 18 |
| 6.2.2. SF Registration . . . . .                            | 21 |
| 6.2.3. Local SF Forwarding . . . . .                        | 22 |
| 6.2.4. Handling of HTTP responses . . . . .                 | 23 |
| 6.2.5. Remote SF Forwarding . . . . .                       | 23 |
| 7. IANA Considerations . . . . .                            | 27 |
| 8. Security Considerations . . . . .                        | 27 |
| 9. Acknowledgement . . . . .                                | 27 |

|                                        |    |
|----------------------------------------|----|
| 10. References . . . . .               | 27 |
| 10.1. Normative References . . . . .   | 27 |
| 10.2. Informative References . . . . . | 28 |
| Authors' Addresses . . . . .           | 29 |

## 1. Introduction

The requirements on today's networks are very diverse, enabling multiple use cases such as IoT, Content Distribution, Gaming and Network functions such as Cloud RAN and 5G control planes based on a service-based architecture. These services are deployed, provisioned and managed using Cloud based techniques as seen in the IT world. Virtualization of compute and storage resources is at the heart of providing (often web) services to end users with the ability to quickly provisioning such virtualized service endpoints through, e.g., container based techniques. This creates a dynamicity with the capability to dynamically compose new services from available services as well as move a service instance in response to user mobility or resource availability where desirable. When moving from a pure 'distant cloud' model to one of localized micro data centers with regional, metro or even street level, often called 'edge' data centers, such virtualized service instances can be instantiated in topologically different locations with the overall 'distant' data center now being transformed into a network of distributed ones. The reaction of content providers, like Facebook, Google, Netflix and others, are not just relying on deploying content server at the ingress of the customer network. Instead the trend is towards deploying multiple POPs within the customer network, those POPs being connected through proprietary mechanisms [Schlinker2017] to push content.

The Service Function Chaining (SFC) framework [RFC7665] allows network operators as well as service providers to compose new services by chaining individual "Service Functions (SFs)". Such chains are expressed through explicit relationships of functional components (the service functions), realized through their direct Layer 2 (e.g., MAC address) or Layer 3 (e.g., IP address) relationship as defined through next hop information that is being defined by the network operator, see Section 4 for more background on SFC.

In a dynamic service environment of distributed data centers as the one outlined above, with the ability to create and recreate service endpoints frequently, the SFC framework requires to reconfigure the existing chain through information based on the new relationships, causing overhead in a number of components, specifically the orchestrator that initiates the initial service function chain and any possible reconfiguration.

This document describes how such changes can be handled without involving the initiation of new and reconfigured SFCs by lifting the chaining relationship from Layer 2 and 3 information to that of service function 'names', such as names for instance being expressed as URIs. In order to transparently support such named relationships, we propose to embed the necessary functionality directly into the Service Function Forwarder (SFF), as described in [RFC7665]). With that, the SFF described in this document allows for keeping an existing SFC intact, as described by its service function path (SFP), while enabling the selection of an appropriate service function endpoint(s) during the traversal of packets through the SFC. This document is an Independent Submission to the RFC Editor. It is not an output of the IETF SFC WG.

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. Example use case: 5G control plane services

We exemplify the need for chaining service functions at the level of a service name through a use case stemming from the current 3GPP Rel 16 work on Service Based Architecture (SBA) [\_3GPP\_SBA], [\_3GPP\_SBA\_ENHANCEMENT]. In this work, mobile network control planes are proposed to be realized by replacing the traditional network function interfaces with a fully service-based one. HTTP was chosen as the application layer protocol for exchanging suitable service requests [\_3GPP\_SBA]. With this in mind, the exchange between, say the 3GPP (Rel. 15) defined Session Management Function (SMF) and the Access and Mobility management Function (AMF) in a 5G control plane is being described as a set of web service like requests which are in turn embedded into HTTP requests. Hence, interactions in a 5G control plane can be modelled based on service function chains where the relationship is between the specific (IP-based) service function endpoints that implement the necessary service endpoints in the SMF and AMF. The service functions are exposed through URIs with work ongoing to define the used naming conventions for such URIs.

This move from a network function model (in pre-Rel 15 systems of 3GPP) to a service-based model is motivated through the proliferation of data center operations for mobile network control plane services. In other words, typical IT-based methods to service provisioning, in particular that of virtualization of entire compute resources, are envisioned to being used in future operations of mobile networks.

Hence, operators of such future mobile networks desire to virtualize service function endpoints and direct (control plane) traffic to the most appropriate current service instance in the most appropriate (local) data centre, such data centre envisioned as being interconnected through a software-defined wide area network (SD-WAN). 'Appropriate' here can be defined by topological or geographical proximity of the service initiator to the service function endpoint. Alternatively, network or service instance compute load can be used to direct a request to a more appropriate (in this case less loaded) instance to reduce possible latency of the overall request. Such data center centric operation is extended with the trend towards regionalization of load through a 'regional office' approach, where micro data centers provide virtualizable resources that can be used in the service execution, creating a larger degree of freedom when choosing the 'most appropriate' service endpoint for a particular incoming service request.

While the move to a service-based model aligns well with the framework of SFC, choosing the most appropriate service instance at runtime requires so-called 're-chaining' of the SFC since the relationships in said SFC are defined through Layer 2 or 3 identifiers, which in turn are likely to be different if the chosen service instances reside in different parts of the network (e.g., in a regional data center).

Hence, when a traffic flow is forwarded over a service chain expressed as an SFC-compliant Service Function Path (SFP), packets in the traffic flow are processed by the various service function instances, with each service function instance applying a service function prior to forwarding the packets to the next network node. It is a Service layer concept and can possibly work over any Virtual network layer and an Underlay network, possibly IP or any Layer 2 technology. At the service layer, Service Functions are identified using a path identifier and an index. Eventually this index is translated to an IP address (or MAC address) of the host where the service function is running. Because of this, any change of service function instance is likely to require a change of the path information since either IP address (in the case of changing the execution from one data centre to another) or MAC address will change due to the newly selected service function instance.

Returning to our 5G Control plane example, a user's connection request to access an application server in the internet may start with signaling in the Control Plane to setup user plane bearers. The connection request may flow through service functions over a service chain in the Control plane, as deployed by network operator. Typical SFs in a 5G control plane may include "RAN termination / processing", "Slice Selection Function", "AMF" and "SMF". A Network Slice is a

complete logical network including Radio Access Network (RAN) and Core Network (CN). Distinct RAN and Core Network Slices may exist. A device may access multiple Network Slices simultaneously through a single RAN. The device may provide Network Slice Selection Assistance Information (NSSAI) parameters to the network to help it select a RAN and a Core network part of a slice instance. Part of the control plane, the Common Control Network Function (CCNF), the Network Slice Selection Function (NSSF) is in charge of selecting core Network Slice instances. The Classifier, as described in SFC architecture, may reside in the user terminal or at the eNB. These service functions can be configured to be part of a Service Function Chain. We can also say that some of the configurations of the Service Function Path may change at the execution time. For example, the SMF may be relocated as user moves and a new SMF may be included in the Service Function Path based on user location. The following diagram in Figure 1 shows the example Service Function Chain described here.

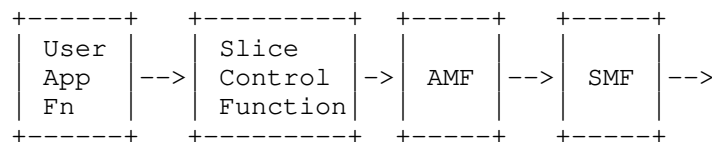


Figure 1: Mapping SFC onto Service Function Execution Points along a Service Function Path

#### 4. Background

[RFC7665] describes an architecture for the specification, creation and ongoing maintenance of Service Function Chains (SFCs). It includes architectural concepts, principles, and components used in the construction of composite services through deployment of SFCs. In the following, we outline the parts of this SFC architecture relevant for our proposed extension, followed by the challenges with this current framework in the light of our example use case.

##### 4.1. Relevant part of SFC architecture

SFC Architecture, as defined in [RFC7665], describes architectural components such as Service Function (SF), Classifier, and Service Function Forwarder (SFF). It describes the Service Function Path (SFP) as the logical path of an SFC. Forwarding traffic along such SFP is the responsibility of the SFF. For this, the SFFs in a

network maintain the requisite SFP forwarding information. Such SFP forwarding information is associated with a service path identifier (SPI) that is used to uniquely identify an SFP. The service forwarding state is represented by the Service Index (SI) and enables an SFF to identify which SFs of a given SFP should be applied, and in what order. The SFF also has information that allows it to forward packets to the next SFF after applying local service functions.

The operational steps to forward traffic are then as follows: Traffic arrives at an SFF from the network. The SFF determines the appropriate SF the traffic should be forwarded to via information contained in the SFC encapsulation. After SF processing, the traffic is returned to the SFF, and, if needed, is forwarded to another SF associated with that SFP. If there is another non-local hop (i.e., to an SF with a different SFP) in the SFP, the SFF further encapsulates the traffic in the appropriate network transport protocol and delivers it to the network for delivery to the next SFF along the path. Related to this forwarding responsibility, an SFF should be able to interact with metadata.

#### 4.2. Challenges with current framework

As outlined in previous section, the Service Function Path defines an ordered sequence of specific Service Functions instances being used for the interaction between initiator and service functions along the SFP. These service functions are addressed by IP (or any L2/MAC) addresses and defined as next hop information in the network locator maps of traversing SFF nodes.

As outlined in our use case, however, the service provider may want to provision SFC nodes based on dynamically spun up service function instances so that these (now virtualized) service functions can be reached in the SFC domain using the SFC underlay layer.

Following the original model of SFC, any change in a specific execution point for a specific Service Function along the SFP will require a change of the SFP information (since the new service function execution point likely carries different IP or L2 address information) and possibly even the Next Hop information in SFFs along the SFP. In case the availability of new service function instances is rather dynamic (e.g., through the use of container-based virtualization techniques), the current model and realization of SFC could lead to reducing the flexibility of service providers and increasing the management complexity incurred by the frequent changes of (service) forwarding information in the respective SFF nodes. This is because any change of the SFP (and possibly next hop info) will need to go through suitable management cycles.

To address these challenges through a suitable solution, we identify the following requirements:

- o Relations between Service Execution Points MUST be abstracted so that, from an SFP point of view, the Logical Path never changes.
- o Deriving the Service Execution Points from the abstract SFP SHOULD be fast and incur minimum delay.
- o Identification of the Service Execution Points SHOULD not use a combination of Layer 2 or Layer 3 mechanisms.

The next section outlines a solution to address the issue, allowing for keeping SFC information (represented in its SFP) intact while addressing the desired flexibility of the service provider.

## 5. Name based operation in SFF

### 5.1. General Idea

The general idea is two-pronged. Firstly, we elevate the definition of a Service Function Path onto the level of 'name-based interactions' rather than limiting SFPs to Layer 3 or Layer 2 information only. Secondly, we extend the operations of the SFF to allow for forwarding decisions that take into account such name-based interaction while remaining backward compatible to the current SFC architecture, as defined in [RFC7665]. In the following sections, we outline these two components of our solution.

If the next hop information in the Network Locator Map (NLM) is described using L2/L3 identifier, the name-based SFF (nSFF) may operate as described for [traditional] SFF, as defined in [RFC7665]. On the other hand, if the next hop information in the NLM is described as a name, then the nSFF operates as described in the following sections.

In the following sections, we outline the two components of our solution.

### 5.2. Name-Based Service Function Path (nSFP)

The existing SFC framework is defined in [RFC7665]. Section 4 outlines that the SFP information is representing path information based on Layer 2 or 3 information, i.e., MAC or IP addresses, causing the aforementioned frequent adaptations in cases of execution point changes. Instead, we introduce the notion of a "name-based service function path (nSFP)".

In today's networking terms, any identifier can be treated as a name but we will illustrate the realization of a "Name based SFP" through extended SFF operations (see Section 6) based on URIs as names and HTTP as the protocol of exchanging information. Here, URIs are being used to name for a Service Function along the nSFP. It is to be noted that the Name based SFP approach is not restricted to HTTP (as the protocol) and URIs (as next hop identifier within the SFP). Other identifiers such as an IP address itself can also be used and are interpreted as a 'name' in the nSFP. IP addresses as well as fully qualified domain names forming complex URIs (uniform resource identifiers), such as `www.example.com/service_name1`, are all captured by the notion of 'name' in this document.

Generally, nSFPs are defined as an ordered sequence of the "name" of Service Functions (SF) and a typical name-based Service Function Path may look like: `192.0.x.x -> www.example.com -> www.example2.com/service1 -> www.example2.com/service2`.

Our use case in Section 3 can then be represented as an ordered named sequence. An example for a session initiation that involves an authentication procedure, this could look like `192.0.x.x -> smf.example.org/session_initiate -> amf.example.org/auth -> smf.example.org/session_complete -> 192.0.x.x`. [Note that this example is only a conceptual one, since the exact nature of any future SBA-based exchange of 5G control plane functions is yet to be defined by standardization bodies such as 3GPP].

In accordance with our use case in Section 3, any of these named services can potentially be realized through more than one replicated SF instances. This leads to make dynamic decision on where to send packets along the SAME service function path information, being provided during the execution of the SFC. Through elevating the SFP onto the notion of name-based interactions, the SFP will remain the same even if those specific execution points change for a specific service interaction.

The following diagram in Figure 2, describes this name-based SFP concept and the resulting mapping of those named interactions onto (possibly) replicated instances.



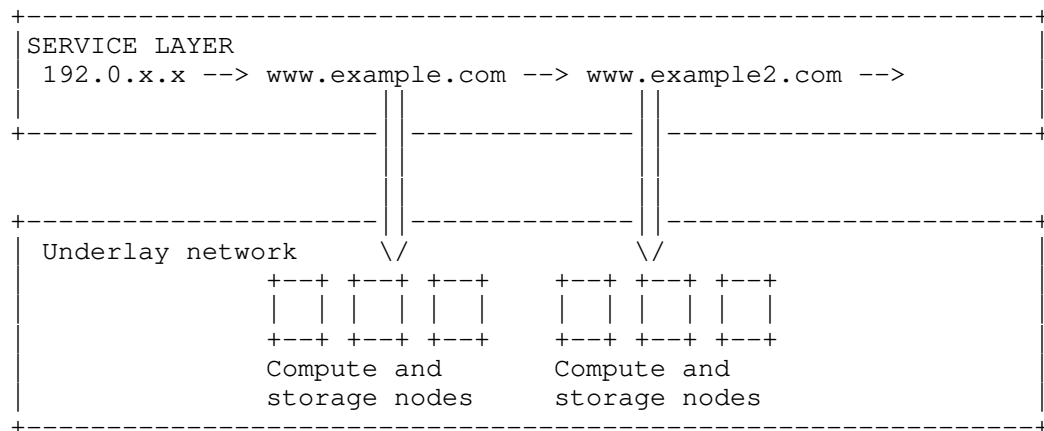


Figure 2: Mapping SFC onto Service Function Execution Points along a Service Function Path based on Virtualized Service Function Instance

### 5.3. Name Based Network Locator Map (nNLM)

In order to forward a packet within a name-based SFP, we need to extend the network locator map as defined in [RFC8300] with the ability to consider name relations based on URIs as well as high-level transport protocols such as HTTP for means of SFC packet forwarding. Another example for SFC packet forwarding could be that of CoAP.

The extended Network Locator Map or name-based Network Locator Map (nNLM) is shown in Figure 3 as an example for `www.example.com` being part of the nSFP. Such extended nNLM is stored at each SFF throughout the SFC domain with suitable information populated to the nNLM during the configuration phase.

| SPI | SI  | Next Hop(s)        | Transport Encapsulation (TE) |
|-----|-----|--------------------|------------------------------|
| 10  | 255 | 192.0.2.1          | VXLAN-gpe                    |
| 10  | 254 | 198.51.100.10      | GRE                          |
| 10  | 253 | www.example.com    | HTTP                         |
| 40  | 251 | 198.51.100.15      | GRE                          |
| 50  | 200 | 01:23:45:67:89:ab  | Ethernet                     |
| 15  | 212 | Null (end of path) | None                         |

Figure 3: Name-based Network Locator Map

Alternatively, the extended network locator map may be defined with implicit name information rather than explicit URIs as in Figure 3. In the example of Figure 4 below, the next hop is represented as a generic HTTP service without a specific URI being identified in the extended network locator map. In this scenario, the SFF forwards the packet based on parsing the HTTP request in order to identify the host name or URI. It retrieves the URI and may apply policy information to determine the destination host/service.

| SPI | SI  | Next Hop(s)        | Transport Encapsulation (TE) |
|-----|-----|--------------------|------------------------------|
| 10  | 255 | 192.0.2.1          | VXLAN-gpe                    |
| 10  | 254 | 198.51.100.10      | GRE                          |
| 10  | 253 | HTTP Service       | HTTP                         |
| 40  | 251 | 198.51.100.15      | GRE                          |
| 50  | 200 | 01:23:45:67:89:ab  | Ethernet                     |
| 15  | 212 | Null (end of path) | None                         |

Figure 4: Name-based Network Locator Map with Implicit Name information

#### 5.4. Name-based Service Function Forwarder (nSFF)

It is desirable to extend the SFF of the SFC underlay to handle nSFPs transparently and without the need to insert any service function into the nSFP. Such extended name-based SFF would then be responsible for forwarding a packet in the SFC domain as per the definition of the (extended) nSFP.

In our exemplary realization for an extended SFF, the solution described in this document uses HTTP as the protocol of forwarding SFC packets to the next (name-based) hop in the nSFP. The URI in the HTTP transaction are the names in our nSFP information, which will be used for name based forwarding.

Following our reasoning so far, HTTP requests (and more specifically the plain text encoded requests above) are the equivalent of Packets that enter the SFC domain. In the existing SFC framework, typically an IP payload is assumed to be a packet entering the SFC domain. This packet is forwarded to destination nodes using the L2 encapsulation. Any layer 2 network can be used as an underlay network. This notion is now extended to packets being possibly part of a entire higher layer application, such as HTTP requests. The handling of any intermediate layers such as TCP, IP is left to the realization of the (extended) SFF operations towards the next (named) hop. For this, we will first outline the general lifecycle of an SFC packet in the following subsection, followed by two examples for

determining next hop information in Section 6.2.3, finalized by a layered view on the realization of the nSFF in Section 6.2.4.

### 5.5. High Level Architecture

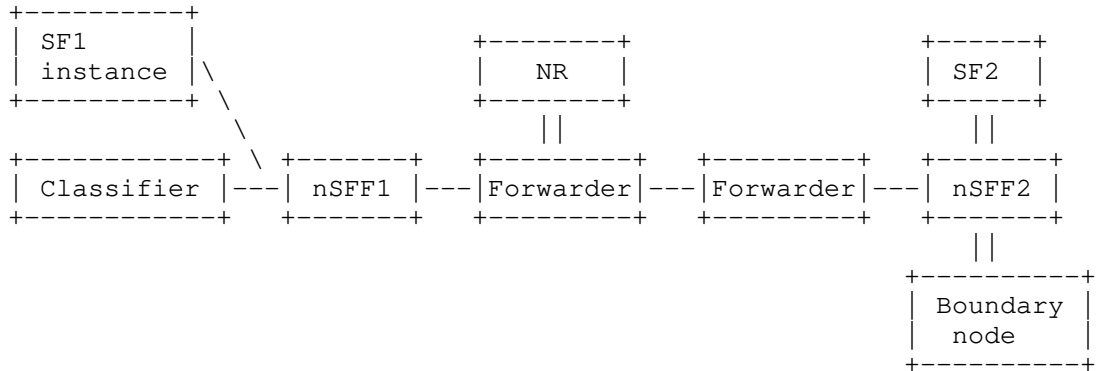


Figure 5: High-level architecture

The high-level architecture for name based operation shown in Figure 5 is very similar to the SFC architecture, as described in [RFC7665]. Two new functions are introduced, as shown in the above diagram, namely the name-based Service Function Forwarder (nSFF) and the Name Resolver (NR).

nSFF (name-based Service Function Forwarder) is an extension of the existing SFF and is capable of processing SFC packets based on name-based network locator map (nNLM) information, determining the next SF, where the packet should be forwarded and the required transport encapsulation. Like standard SFF operation, it adds transport encapsulation to the SFC packet and forwards it.

The Name Resolver is a new functional component, capable of identifying the execution end points, where a "named SF" is running, triggered by suitable resolution requests sent by the nSFF. Though this is similar to DNS function, but it is not same. It does not use DNS protocols or data records. A new procedure to determine the suitable routing/forwarding information towards the Nsff (name-based SFF) serving the next hop of the SFP (Service Function Path) is used. The details is described later.

The other functional components such as Classifier, SF are same as described in SFC architecture, as defined in [RFC7665], while the Forwarders shown in the above diagram are traditional Layer 2 switches.

### 5.6. Operational Steps

In the proposed solution, the operations are realized by the name-based SFF, called nSFF. We utilize the high-level architecture in Figure 5 to describe the traversal between two service function instances of an nSFP-based transactions in an example chain of : 192.0.x.x -> SF1 (www.example.com) -> SF2 (www.example2.com) -> SF3 -> ... Service Function 3 (SF3) is assumed to be a classical Service Function, hence existing SFC mechanisms can be used to reach it and will not be considered in this example.

According to the SFC lifecycle, as defined in [RFC7665], based on our example chain above, the traffic originates from a Classifier or another SFF on the left. The traffic is processed by the incoming nSFF1 (on the left side) through the following steps. The traffic exits at nSFF2.

- o Step 1: At nSFF1 the following nNLM is assumed

| SPI | SI  | Next Hop(s)        | Transport Encapsulation(TE) |
|-----|-----|--------------------|-----------------------------|
| 10  | 255 | 192.0.2.1          | VXLAN-gpe                   |
| 10  | 254 | 198.51.100.10      | GRE                         |
| 10  | 253 | www.example.com    | HTTP                        |
| 10  | 252 | www.example2.com   | HTTP                        |
| 40  | 251 | 198.51.100.15      | GRE                         |
| 50  | 200 | 01:23:45:67:89:ab  | Ethernet                    |
| 15  | 212 | Null (end of path) | None                        |

Figure 6: nNLM at nSFF1

- o Step 2: nSFF1 removes the previous transport encapsulation (TE) for any traffic originating from another SFF or classifier

(traffic from an SF instance does not carry any TE and is therefore directly processed at the nSFF).

- o Step 3: nSFF1 then processes the Network Service Header (NSH) information, as defined in [RFC8300], to identify the next SF at the nSFP level by mapping the NSH information to the appropriate entry in its nNLM (see Figure 6) based on the provided SPI/SI information in the NSH (see Section 4) in order to determine the name-based identifier of the next hop SF. With such nNLM in mind, the nSFF searches the map for SPI = 10 and SI = 253. It identifies the next hop as = www.example.com and HTTP as the protocol to be used. Given the next hop resides locally, the SFC packet is forwarded to the SF1 instance of www.example.com. Note that the next hop could also be identified from the provided HTTP request, if the next hop information was identified as a generic HTTP service, as defined in Section 5.3.
- o Step 4: The SF1 instance then processes the received SFC packet according to its service semantics and modifies the NSH by setting SPI = 10, SI = 252 for forwarding the packet along the SFP. It then forwards the SFC packet to its local nSFF, i.e., nSFF1.
- o Step 5: nSFF1 processes the NSH of the SFC packet again, now with the NSH modified (SPI = 10, SI = 252) by the SF1 instance. It retrieves the next hop information from its nNLM in Figure 6, to be www.example2.com. Due to this SF not being locally available, the nSFF consults any locally available information regarding routing/forwarding towards a suitable nSFF that can serve this next hop.
- o Step 6: If such information exists, the Packet (plus the NSH information) is marked to be sent towards the nSFF serving the next hop based on such information in step 8.
- o Step 7: If such information does not exist, nSFF1 consults the Name Resolver (NR) to determine the suitable routing/forwarding information towards the identified nSFF serving the next hop of the SFP. For future SFC packets towards this next hop, such resolved information may be locally cached, avoiding to contact the Name Resolver for every SFC packet forwarding. The packet is now marked to be sent via the network in step 8.
- o Step 8: Utilizing the forwarding information determined in steps 6 or 7, nSFF1 adds the suitable transport encapsulation (TE) for the SFC packet before forwarding via the forwarders in the network towards the next nSFF22.

- o Step 9: When the Packet (+NSH+TE) arrives at the outgoing nSFF2, i.e., the nSFF serving the identified next hop of the SFP, removes the TE and processes the NSH to identify the next hop information. At nSFF2 the nNLM in Figure 7 is assumed. Based on this nNLM and NSH information where SPI = 10 and SI = 252, nSFF2 identifies the next SF as www.example2.com.

| SPI | SI  | Next Hop(s)        | Transport Encapsulation (TE) |
|-----|-----|--------------------|------------------------------|
| 10  | 252 | www.example2.com   | HTTP                         |
| 40  | 251 | 198.51.100.15      | GRE                          |
| 50  | 200 | 01:23:45:67:89:ab  | Ethernet                     |
| 15  | 212 | Null (end of path) | None                         |

Figure 7: nNLM at SFF2

- o Step 10: If the next hop is locally registered at the nSFF, it forwards the packet (+NSH) to the service function instance, using suitable IP/MAC methods for doing so.
- o Step 11: Otherwise, the outgoing nSFF adds a new TE information to the packet and forwards the packet (+NSH+TE) to the next SFF or boundary node, as shown in Figure 7.

## 6. nSFF Forwarding Operations

This section outlines the realization of various nSFF forwarding operations in Section 5.6. Although the operations in Section 5 utilize the notion of name-based transactions in general, we exemplify the operations here in Section 5 specifically for HTTP-based transactions to ground our description into a specific protocol for such name-based transaction. We will refer to the various steps in each of the following sub-sections.

### 6.1. nSFF Protocol Layers

Figure 8 shows the protocol layers, based on the high-level architecture in Figure 5.

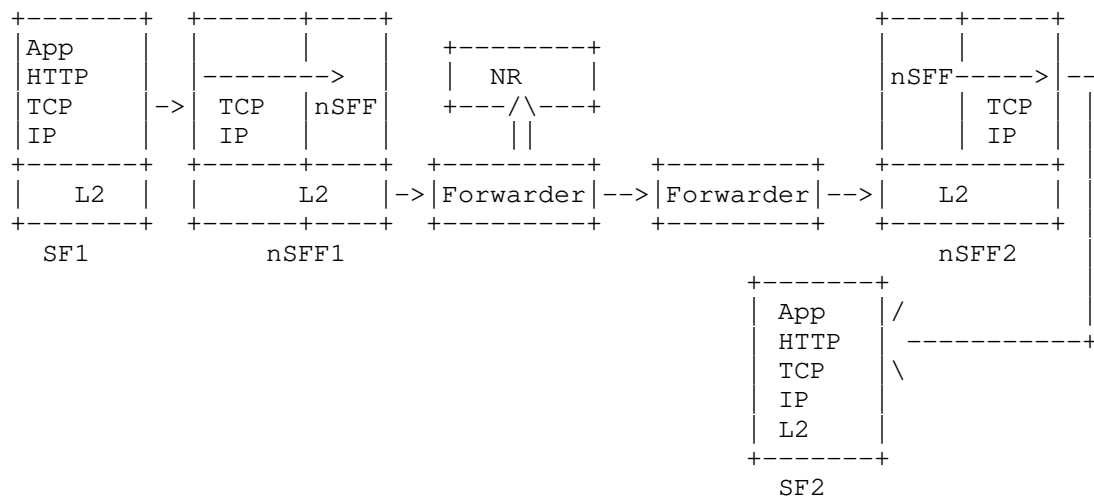


Figure 8: Protocol layers

The nSFF component here is shown as implementing a full incoming/outgoing TCP/IP protocol stack towards the local service functions, while implementing the nSFF-NR and nSFF-nSFF protocols based on the descriptions in Section 6.2.3.

For the exchange of HTTP-based service function transactions, the nSFF terminates incoming TCP connections from as well as outgoing TCP connections to local SFs, e.g., the TCP connection from SF1 terminates at nSFF1, and nSFF1 may store the connection information, such as socket information. It also maintains the mapping information for the HTTP request such as originating SF, destination SF and socket ID. nSFF1 may implement sending keep-alive messages over the socket to maintain the connection to SF1. Upon arrival of an HTTP request from SF1, nSFF1 extracts the HTTP Request and forwards it towards the next node, as outlined in Section 6.2. Any returning response is mapped onto the suitable open socket (for the original request) and send towards SF1.

At the outgoing nSFF2, the destination SF2/Host is identified from the HTTP request message. If no TCP connection exists to the SF2, a new TCP connection is opened towards the destination SF2 and the HTTP request is sent over said TCP connection. The nSFF2 may also save the TCP connection information (such as socket information) and maintain the mapping of the socket information to the destination SF2. When an HTTP response is received from SF2 over the TCP connection, nSFF2 extracts the HTTP response, which is forwarded to



the next node. nSFF2 may maintain the TCP connection through keep-alive messages.

## 6.2. nSFF Operations

In this section, we present three key aspects of operations for the realization of the steps in Section 5.6, namely (i) the registration of local SFs (for step 3 in Section 5.6), (ii) the forwarding of SFC packets to and from local SFs (for step 3 and 4 as well as 10 in Section 5.6), (iii) the forwarding to a remote SF (for steps 5, 6 and 7 in Section 5.6) and to the NR as well as (iv) for the lookup of a suitable remote SF (for step 7 in Section 5.6). We also cover aspects of maintaining local lookup information for reducing lookup latency and others issues.

### 6.2.1. Forwarding between nSFFs and nSFF-NR

Forwarding between the distributed nSFFs as well as between nSFF and NR is realized over the operator network via a path-based approach. A path-based approach utilizes path information provided by the source of the packet for forwarding said packet in the network. This is similar to segment routing albeit differing in the type of information provided for such source-based forwarding, as described in this section. In this approach, the forwarding information to a remote nSFF or the NR is defined as a 'path identifier' (pathID) of a defined length where said "Length" field indicates the full pathID length. The payload of the packet is defined by the various operations outlined in the following sub-sections, resulting in an overall packet being transmitted. With this, the generic forwarding format (GFF) for transport over the operator network is defined in Figure 9 with the length field defining the length of the pathID provided.

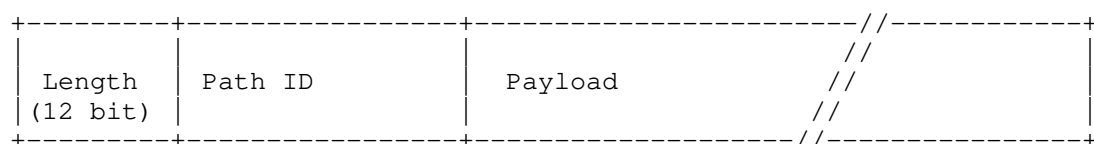


Figure 9: Generic Forwarding Format (GFF)

- o Length (12 bits): Defines the length of the pathID, i.e., up to 4096 bits

- o Path ID (): Variable length field, Bit field derived from IPv6 source and destination address

For the pathID information, solutions such as those in [Reed2016] can be used. Here, the IPv6 source and destination addresses are used to realize a so-called path-based forwarding from the incoming to the outgoing nSFF or the NR. The forwarders in Figure 8 are realized via SDN (software-defined networking) switches, implementing an AND/CMP operation based on arbitrary wildcard matching over the IPv6 source and destination addresses, as outlined in [Reed2016]. Note that in the case of using IPv6 address information for path-based forwarding, the step of removing the transport encapsulation at the outgoing nSFF in Figure 8 is realized by utilizing the provided (existing) IP header (which was used for the purpose of the path-based forwarding in [Reed2016]) for the purpose of next hop forwarding, such as that of IP-based routing. As described in step 8 of the extended nSFF operations, this forwarding information is used as traffic encapsulation. With the forwarding information utilizing existing IPv6 information, IP headers are utilized as TE in this case. The next hop nSFF (see Figure 8) will restore the IP header of the packet with the relevant IP information used to forward the SFC packet to SF2 or it will create a suitable TE (Transport Encapsulation) information to forward the information to another nSFF or boundary node. Forwarding operations at the intermediary forwarders, i.e., SDN switches, examine the pathID information through a flow matching rule in which a specific switch-local output port is represented through the specific assigned bit position in the pathID. Upon a positive match in said rule, the packet is forwarded on said output port.

Alternatively, the solution in [I-D.ietf-bier-multicast-http-response] suggests using a so-called BIER (Binary Indexed Explicit Replication) underlay. Here, the nSFF would be realized at the ingress to the BIER underlay, injecting the SFC packet (plus the NSH) header with BIER-based traffic encapsulation into the BIER underlay with each of the forwarders in Figure 8 being realized as a so-called Bit-Forwarding Router (BFR) [RFC8279].

#### 6.2.1.1. Transport Protocol Considerations

Given that the proposed solution operates at the 'named transaction' level, particularly for HTTP transactions, forwarding between nSFFs and/or NR SHOULD be implemented via a transport protocol between nSFFs and/or NR in order to provide reliability, segmentation of large GFF packets, and flow control, with the GFF in Figure 9 being the basic forwarding format for this.

Note that the nSFFs act as TCP proxies at ingress and egress, thus terminating incoming and initiating outgoing HTTP sessions to SFs.

Figure 10 shows the packet format being used for the transmission of data, being adapted from the TCP header. Segmentation of large transactions into single transport protocol packets is realized through maintaining a 'Sequence number'. A 'Checksum' is calculated over a single data packet with the ones-complement TCP checksum calculation being used. The 'Window Size' field indicates the current maximum number of transport packets that are allowed in-flight by the egress nSFF. A data packet is sent without 'Data' field to indicate the end of (e.g., HTTP) transaction.

Note that in order to support future named transactions based on other application protocols, such as CoAP, future versions of the transport protocol MAY introduce a 'Type' field that indicates the type of application protocol being used between SF and nSFF with 'Type' 0x01 proposed for HTTP. This is being left for future study.

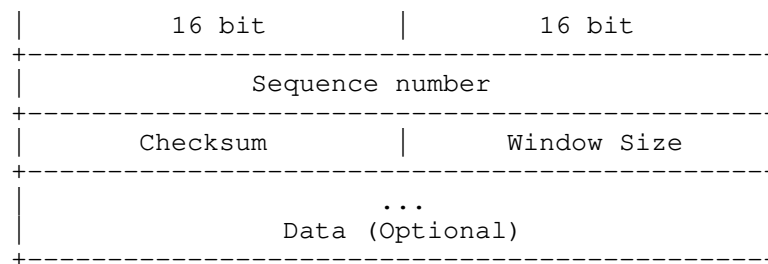


Figure 10: Transport protocol data packet format

Given the path-based forwarding being used between nSFFs, the transport protocol between nSFFs utilizes negative acknowledgements from the egress nSFF towards the ingress nSFF. The transport protocol NACK packet carries the number of NACKs as well as the specific sequence numbers being indicated as lost in the 'NACK number' field(s), as shown in Figure 11.

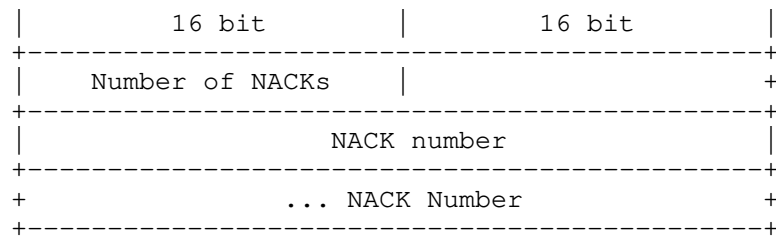


Figure 11: Transport protocol NACK packet format

If the indicated number of NACKs in a received NACK packet is non-zero, the ingress nSFF will retransmit all sequence numbers signalled in the packet, while decreasing its congestion window size for future transmissions.

If the indicated number of NACKs in a received NACK packet is zero, it will indicate the current congestion window as being successfully (and completely) being transmitted, increasing the congestion window size if smaller than the advertised 'Window Size' in Figure 10.

The maintenance of the congestion window is subject to realization at the ingress nSFF and left for further study in nSFF realizations.

#### 6.2.2. SF Registration

As outlined in step 3 and 10 of Section 5.6, the nSFF needs to determine if the SF derived from the nNLM is locally reachable or whether the packet needs forwarding to a remote SFF. For this, a registration mechanism is provided for such local SF with the local nSFF. Two mechanisms can be used for this:

1. SF-initiated: We assume that the SF registers its FQDN to the local nSFF. As local mechanisms, we foresee that either a REST-based interface over the link-local link or configuration of the nSFF (through configuration files or management consoles) can be utilized. Such local registration event leads to the nSFF to register the given FQDN with the NR in combination with a system-unique nSFF identifier that is being used for path computation purposes in the NR. For the registration, the packet format in Figure 12 is used (inserted as the payload in the GFF of Figure 9 with the pathID towards the NR).



Figure 12: Registration packet format

- o R/D: 1 bit length (0 for Register, 1 for De-register)
- o Hash(FQDN): 16 bit length for a hash over the FQDN of the SF
- o nSFF\_ID: 8 bit for a system-unique identifier for the SFF related to the SF.

We assume that the pathID towards the NR is known to the nSFF through configuration means.

The NR maintains an internal table that associates the hash(FQDN), the nSFF\_id information as well as the pathID information being used for communication between nSFF and NR. The nSFF locally maintains a mapping of registered FQDNs to IP addresses, for the latter using link-local private IP addresses.

2. Orchestration-based: in this mechanism, we assume that SFC to be orchestrated and the chain being provided through an orchestration template with FQDN information associated to a compute/storage resource that is being deployed by the orchestrator. We also assume knowledge at the orchestrator of the resource topology. Based on this, the orchestrator can now use the same REST-based protocol defined in option 1 to instruct the NR to register the given FQDN, as provided in the template, at the nSFF it has identified as being the locally servicing nSFF, provided as the system-unique nSFF identifier.

### 6.2.3. Local SF Forwarding

There are two cases of local SF forwarding, namely the SF sending an SFC packet to the local nSFF (incoming requests) or the nSFF sending a packet to the SF (outgoing requests) as part of steps 3 and 10 in Section 5.6. In the following, we outline the operation for HTTP as an example named transaction.

As shown in Figure 8, incoming HTTP requests from SFs are extracted by terminating the incoming TCP connection at their local nSFFs at the TCP level. The nSFF MUST maintain a mapping of open TCP sockets

to HTTP requests (utilizing the URI of the request) for HTTP response association.

For outgoing HTTP requests, the nSFF utilizes the maintained mapping of locally registered FQDNs to link-local IP addresses (see Section 6.2.2 option 1). Hence, upon receiving an SFC packet from a remote nSFF (in step 9 of Section 5.6), the nSFF determines the local existence of the SF through the registration mechanisms in Section 6.2.2. If said SF does exist locally, the HTTP (+NSH) packet, after stripping the TE, is sent to the local SF as step 10 in Section 5.6 via a TCP-level connection. Outgoing nSFF SHOULD keep TCP connections open to local SFs for improving SFC packet delivery in subsequent transactions.

#### 6.2.4. Handling of HTTP responses

When executing step 3 and 10 in Section 5.6, the SFC packet will be delivered to the locally registered next hop. As part of the HTTP protocol, responses to the HTTP request will need to be delivered on the return path to the originating nSFF (i.e., the previous hop). For this, the nSFF maintains a list of link-local connection information, e.g., sockets to the local SF and the pathID on which the request was received. Once receiving the response, nSFF consults the table to determine the pathID of the original request, forming a suitable GFF-based packet to be returned to the previous nSFF.

When receiving the HTTP response at the previous nSFF, the nSFF consults the table of (locally) open sockets to determine the suitable local SF connection, mapping the received HTTP response URI to the stored request URI. Utilizing the found socket, the HTTP response is forwarded to the locally registered SF.

#### 6.2.5. Remote SF Forwarding

In steps 5, 6, 7, and 8 of Section 5.6, an SFC packet is forwarded to a remote nSFF based on the nNLM information for the next hop of the nSFF. Section 6.2.5.1 handles the case of suitable forwarding information to the remote nSFF not existing, therefore consulting the NR to obtain suitable information, while Section 6.2.5.2 describes the maintenance of forwarding information at the local nSFF, while Section 6.2.5.3 describes the update of stale forwarding information. Note that the forwarding described in Section 6.2.1 is used for the actual forwarding to the various nSFF components. Ultimately, Section 6.2.5.4 describes the forwarding to the remote nSFF via the forwarder network.

#### 6.2.5.1. Remote SF Discovery

The nSFF communicates with the NR for two purposes, namely the registration and discovery of FQDNs. The packet format for the former was shown in Figure 10 in Section 6.2.2, while Figure 13 outlines the packet format for the discovery request.

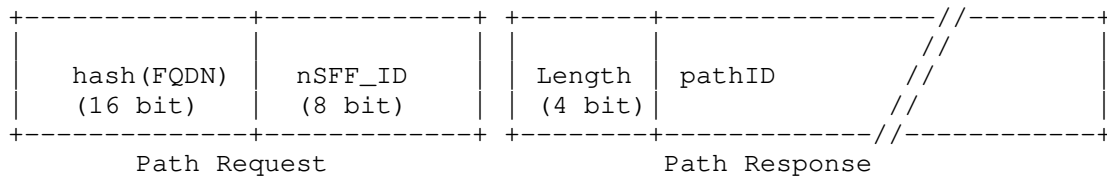


Figure 13: Discovery packet format

For Path Request:

- o Hash(FQDN): 16 bit length for a hash over the FQDN of the SF
- o nSFF\_ID: 8 bit for a system-unique identifier for the SFF related to the SF

For Path Response:

- o Length (4 bits): Defines the length of the pathID
- o Path ID (): Variable length field, Bit field derived from IPv6 source and destination address

A path to a specific FQDN is requested by sending a hash of the FQDN to the NR together with its nSFF\_id, receiving as a response a pathID with a length identifier. The NR SHOULD maintain a table of discovery requests that map discovered (hash of) FQDN to the nSFF\_id that requested it and the pathID that is being calculated as a result of the discovery request.

The discovery request for an FQDN that has not previously been served at the nSFF (or for an FQDN whose pathID information has been flushed as a result of the update operations in Section 6.2.5.3), results in an initial latency incurred by this discovery through the NR, while any SFC packet sent over the same SFP in a subsequent transaction will utilize the nSFF local mapping table. Such initial latency can be avoided by pre-populating the FQDN-pathID mapping proactively as part of the overall orchestration procedure, e.g., alongside the distribution of the nNLM information to the nSFF.

#### 6.2.5.2. Maintaining Forwarding Information at Local nSFF

Each nSFF MUST maintain an internal table that maps the (hash of the) FQDN information to a suitable pathID information. As outlined in step 7 of Section 5.6, if a suitable entry does not exist for a given FQDN, the pathID information is requested with the operations in Section 6.2.5.1 and the suitable entry is locally created upon receiving a reply with the forwarding operation being executed as described in Section 6.2.1.

If such entry does exist (i.e., step 6 of Section 5.6) the pathID is locally retrieved and used for the forwarding operation in Section 6.2.1.

#### 6.2.5.3. Updating Forwarding Information at nSFF

The forwarding information maintained at each nSFF (see Section 6.2.5.2) might need to be updated for three reasons:

- o An existing SF is no longer reachable: In this case, the nSFF with which the SF is locally registered, de-registers the SF explicitly at the NR by sending the packet in Figure 10 with the hashed FQDN and the R/D bit set to 1 (for de-register).
- o Another SF instance has become reachable in the network (and therefore might provide a better alternative to the existing SF): in this case, the NR has received another packet with format defined in Figure 11 but a different nSFF\_id value.
- o Links along paths might no longer be reachable: the NR might use suitable southbound interface to transport networks to detect link failures, which it associates to the appropriate pathID bit position.

For this purpose, the packet format in Figure 14 is sent from the NR to all affected nSFFs, using the generic format in Figure 9.

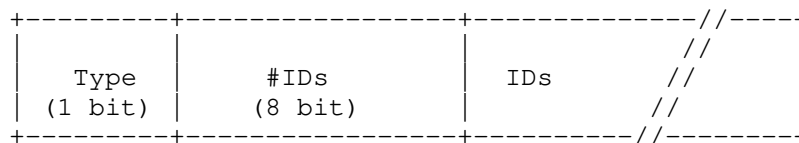


Figure 14: Path update format

- o Type: 1 bit length (0 for Nsff ID, 1 for Link ID)



- o #IDs: 8 bit length for number of IDs in the list
- o IDs: List of IDs (Nsff ID or Link ID)

The pathID to the affected nSFFs is computed as the binary OR over all pathIDs to those nSFF\_ids affected where the pathID information to the affected nSFF\_id values is determined from the NR-local table maintained in the registration/deregistration operation of Section 6.2.2.

The pathID may include the type of information being updated (e.g., node identifiers of leaf nodes or link identifiers for removed links). The node identifier itself may be a special identifier to signal "ALL NODES" as being affected. The node identifier may signal changes to the network that are substantial (e.g., parallel link failures). The node identifier may trigger (e.g., recommend) purging of the entire path table (e.g., rather than the selective removal of a few nodes only).

It will include the information according to the type. The included information may also be related to the type and length information for the number of identifiers being provided.

In case 1 and 2, the Type bit is set to 1 (type nSFF\_id) and the affected nSFFs are determined by those nSFFs that have previously sent SF discovery requests, utilizing the optional table mapping previously registered FQDNs to nSFF\_id values. If no table mapping the (hash of) FQDN to nSFF\_id is maintained, the update is sent to all nSFFs. Upon receiving the path update at the affected nSFF, all appropriate nSFF-local mapping entries to pathIDs for the hash(FQDN) identifiers provided will be removed, leading to a new NR discovery request at the next remote nSFF forwarding to the appropriate FQDN.

In case 3, the Type bit is set to 0 (type linkID) and the affected nSFFs are determined by those nSFFs whose discovery requests have previously resulted in pathIDs which include the affected link, utilizing the optional table mapping previously registered FQDNs to pathID values (see Section 6.2.5.1). Upon receiving the node identifier information in the path update, the affected nSFF will check its internal table that maps FQDNs to pathIDs to determine those pathIDs affected by the link problems and remove path information that includes the received node identifier(s). For this, the pathID entries of said table are checked against the linkID values provided in the ID entry of the path update through a binary AND/CMP operation to check the inclusion of the link in the pathIDs to the FQDNs. If any pathID is affected, the FQDN-pathID entry is removed, leading to a new NR discovery request at the next remote nSFF forwarding to the appropriate FQDN.

#### 6.2.5.4. Forwarding to remote nSFF

Once step 5, 6, and 7 in Section 5.6 are being executed, step 8 finally sends the SFC packet to the remote nSFF, utilizing the pathID returned in the discovery request (Section 6.2.5.1) or retrieved from the local pathID mapping table. The SFC packet is placed in the payload of the generic forwarding format in Figure 9 together with the pathID and the nSFF eventually executes the forwarding operations in Section 6.2.1.

### 7. IANA Considerations

This document requests no IANA actions.

### 8. Security Considerations

The operations in Sections 5 and 6 describes the forwarding of SFC packets between named SFs based on URIs exchanged in HTTP messages. For security considerations, TLS is sufficient between originating node and Nsff, Nsff to Nsff, Nsff to destination. TLS handshake allows to determine the FQDN, which in turn is enough for the service routing decision. Supporting TLS also allows the possibility of HTTPS based transactions.

### 9. Acknowledgement

The authors would like to thank Dirk von Hugo and Andrew Malis for their reviews and valuable comments. We would also like to thank Joel Halpern, the chair of the SFC WG, and Adrian Farrel for guiding us through the IETF Independent Submission Editor (ISE) path.

### 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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <<https://www.rfc-editor.org/info/rfc8279>>.
- [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

- [\_3GPP\_SBA]  
3GPP, "Technical Realization of Service Based Architecture", 3GPP TS 29.500 0.4.0, January 2018, <<http://www.3gpp.org/ftp/Specs/html-info/29500.htm>>.
- [\_3GPP\_SBA\_ENHANCEMENT]  
3GPP, "New SID for Enhancements to the Service-Based 5G System Architecture", 3GPP S2-182904 , February 2018, <[http://www.3gpp.org/ftp/tsg\\_sa/WG2\\_Arch/TSGS2\\_126\\_Montreal/Docs/S2-182904.zip](http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_126_Montreal/Docs/S2-182904.zip)>.
- [I-D.ietf-bier-multicast-http-response]  
Purkayastha, D., Rahman, A., Trossen, D., and T. Eckert, "Applicability of BIER Multicast Overlay for Adaptive Streaming Services", draft-ietf-bier-multicast-http-response-00 (work in progress), February 2019.
- [Reed2016]  
Reed, M., Al-Naday, M., Thomas, N., Trossen, D., and S. Spirou, "Stateless multicast switching in software defined networks", ICC 2016, 2016, <<https://arxiv.org/pdf/1511.06069.pdf>>.
- [Schlinker2017]  
Schlinker, B., Kim, H., Cui, T., Katz-Bassett, E., Madhyastha, Harsha., Cunha, I., Quinn, J., Hassan, S., Lapukhov, P., and H. Zeng, "Engineering Egress with Edge Fabric, Steering Oceans of Content to the World", ACM SIGCOMM 2017, 2017, <<https://research.fb.com/wp-content/uploads/2017/08/sigcomm17-final177-2billion.pdf>>.

Authors' Addresses

Dirk Trossen  
InterDigital Europe, Ltd  
64 Great Eastern Street, 1st Floor  
London EC2A 3QR  
United Kingdom

Email: Dirk.Trossen@InterDigital.com

Debashish Purkayastha  
InterDigital Communications, LLC  
1001 E Hector St  
Conshohocken  
USA

Email: Debashish.Purkayastha@InterDigital.com

Akbar Rahman  
InterDigital Communications, LLC  
1000 Sherbrooke Street West  
Montreal  
Canada

Email: Akbar.Rahman@InterDigital.com

SPRING  
Internet-Draft  
Intended status: Standards Track  
Expires: September 6, 2018

F. Clad, Ed.  
Cisco Systems, Inc.  
X. Xu, Ed.  
Alibaba  
C. Filsfils  
Cisco Systems, Inc.  
D. Bernier  
Bell Canada  
C. Li  
Huawei  
B. Decraene  
Orange  
S. Ma  
Juniper  
C. Yadlapalli  
AT&T  
W. Henderickx  
Nokia  
S. Salsano  
Universita di Roma "Tor Vergata"  
March 5, 2018

Segment Routing for Service Chaining  
draft-xuclad-spring-sr-service-chaining-01

Abstract

This document defines data plane functionality required to implement service segments and achieve service chaining in SR-enabled MPLS and IP networks, as described in the Segment Routing architecture.

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

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 September 6, 2018.

#### Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

|                                                                  |    |
|------------------------------------------------------------------|----|
| 1. Introduction . . . . .                                        | 3  |
| 2. Terminology . . . . .                                         | 4  |
| 3. Classification and steering . . . . .                         | 5  |
| 4. Service Functions . . . . .                                   | 5  |
| 4.1. SR-aware SFs . . . . .                                      | 6  |
| 4.2. SR-unaware SFs . . . . .                                    | 6  |
| 5. Service function chaining . . . . .                           | 7  |
| 5.1. SR-MPLS data plane . . . . .                                | 8  |
| 5.1.1. Encoding SFP Information by an MPLS Label Stack . . . . . | 8  |
| 5.1.2. Encoding SFC Information by an MPLS Label Stack . . . . . | 11 |
| 5.2. SRv6 data plane . . . . .                                   | 14 |
| 5.2.1. Encoding SFP Information by an SRv6 SRH . . . . .         | 14 |
| 5.2.2. Encoding SFC Information by an IPv6 SRH . . . . .         | 16 |
| 6. SR proxy behaviors . . . . .                                  | 17 |
| 6.1. Static SR proxy . . . . .                                   | 20 |
| 6.1.1. SR-MPLS pseudocode . . . . .                              | 21 |
| 6.1.2. SRv6 pseudocode . . . . .                                 | 22 |
| 6.2. Dynamic SR proxy . . . . .                                  | 25 |
| 6.2.1. SR-MPLS pseudocode . . . . .                              | 25 |
| 6.2.2. SRv6 pseudocode . . . . .                                 | 26 |
| 6.3. Shared memory SR proxy . . . . .                            | 26 |
| 6.4. Masquerading SR proxy . . . . .                             | 27 |
| 6.4.1. SRv6 masquerading proxy pseudocode . . . . .              | 28 |
| 6.4.2. Variant 1: Destination NAT . . . . .                      | 28 |

|                                        |    |
|----------------------------------------|----|
| 6.4.3. Variant 2: Caching . . . . .    | 29 |
| 7. Metadata . . . . .                  | 29 |
| 7.1. MPLS data plane . . . . .         | 29 |
| 7.2. IPv6 data plane . . . . .         | 29 |
| 7.2.1. SRH TLV objects . . . . .       | 29 |
| 7.2.2. SRH tag . . . . .               | 30 |
| 8. Implementation status . . . . .     | 30 |
| 9. Related works . . . . .             | 31 |
| 10. IANA Considerations . . . . .      | 31 |
| 11. Security Considerations . . . . .  | 31 |
| 12. Acknowledgements . . . . .         | 32 |
| 13. Contributors . . . . .             | 32 |
| 14. References . . . . .               | 32 |
| 14.1. Normative References . . . . .   | 32 |
| 14.2. Informative References . . . . . | 32 |
| Authors' Addresses . . . . .           | 34 |

## 1. Introduction

Segment Routing (SR) is an architecture based on the source routing paradigm that seeks the right balance between distributed intelligence and centralized programmability. SR can be used with an MPLS or an IPv6 data plane to steer packets through an ordered list of instructions, called segments. These segments may encode simple routing instructions for forwarding packets along a specific network path, or rich behaviors to support use-cases such as Service Function Chaining (SFC).

In the context of SFC, each Service Function (SF), running either on a physical appliance or in a virtual environment, is associated with a segment, which can then be used in a segment list to steer packets through the SF. Such service segments may be combined together in a segment list to achieve SFC, but also with other types of segments as defined in [I-D.ietf-spring-segment-routing]. SR thus provides a fully integrated solution for SFC, overlay and underlay optimization. Furthermore, the IPv6 dataplane natively supports metadata transportation as part of the SR information attached to the packets.

This document describes how SR enables SFC in a simple and scalable manner, from the segment association to the SF up to the traffic classification and steering into the service chain. Several SR proxy behaviors are also defined to support SR SFC through legacy, SR-unaware, SFs in various circumstances.

The definition of an SR Policy and the steering of traffic into an SR Policy is outside the scope of this document. These aspects are covered in [I-D.filsfils-spring-segment-routing-policy].

The definition of control plane components, such as service segment discovery, is outside the scope of this data plane document. BGP extensions to support SR-based SFC are proposed in [I-D.dawra-idr-bgp-sr-service-chaining].

Familiarity with the following IETF documents is assumed:

- o Segment Routing Architecture [I-D.ietf-spring-segment-routing]
- o Segment Routing with MPLS data plane [I-D.ietf-spring-segment-routing-mpls]
- o Segment Routing Traffic Engineering Policy [I-D.filsfils-spring-segment-routing-policy]
- o Segment Routing Header [I-D.ietf-6man-segment-routing-header]
- o SRv6 Network Programming [I-D.filsfils-spring-srv6-network-programming]
- o SR-MPLS over IP [I-D.xu-mpls-sr-over-ip]
- o Service Function Chaining Architecture [RFC7665]

## 2. Terminology

This document leverages the terminology introduced in [I-D.ietf-spring-segment-routing], [I-D.filsfils-spring-segment-routing-policy] and [RFC7665]. It also introduces the following new terminology.

SR-aware SF: Service Function fully capable of processing SR traffic

SR-unaware SF: Service Function unable to process SR traffic or behaving incorrectly for such traffic

SR proxy: Proxy handling the SR processing on behalf of an SR-unaware SF

Service Segment: Segment associated with an SF, either directly or via an SR proxy

SR SFC policy: SR policy, as defined in [I-D.filsfils-spring-segment-routing-policy], that includes at least one Service Segment. An SR SFC policy may also contain other types of segments, such as VPN or TE segments.



### 3. Classification and steering

Classification and steering mechanisms are defined in section 8 of [I-D.filsfils-spring-segment-routing-policy] and are independent from the purpose of the SR policy. From a headend perspective, there is no difference whether a policy contains IGP, BGP, peering, VPN and service segments, or any combination of these.

As documented in the above reference, traffic is classified when entering an SR domain. The SR policy head-end may, depending on its capabilities, classify the packets on a per-destination basis, via simple FIB entries, or apply more complex policy routing rules requiring to look deeper into the packet. These rules are expected to support basic policy routing such as 5-tuple matching. In addition, the IPv6 SRH tag field defined in [I-D.ietf-6man-segment-routing-header] can be used to identify and classify packets sharing the same set of properties. Classified traffic is then steered into the appropriate SR policy, which is associated with a weighted set of segment lists.

SR traffic can be re-classified by an SR endpoint along the original SR policy (e.g., DPI service) or a transit node intercepting the traffic. This node is the head-end of a new SR policy that is imposed onto the packet, either as a stack of MPLS labels or as an IPv6 and SRH encapsulation.

### 4. Service Functions

A Service Function (SF) may be a physical appliance running on dedicated hardware, a virtualized service inside an isolated environment such as a VM, container or namespace, or any process running on a compute element. An SF may also comprise multiple sub-components running in different processes or containers. Unless otherwise stated, this document does not make any assumption on the type or execution environment of an SF.

SR enables SFC by assigning a segment identifier, or SID, to each SF and sequencing these service SIDs in a segment list. A service SID may be of local significance or directly reachable from anywhere in the routing domain. The latter is realized with SR-MPLS by assigning a SID from the global label block ([I-D.ietf-spring-segment-routing-mpls]), or with SRv6 by advertising the SID locator in the routing protocol ([I-D.filsfils-spring-srv6-network-programming]). It is up to the network operator to define the scope and reachability of each service SID. This decision can be based on various considerations such as infrastructure dynamicity, available control plane or orchestration system capabilities.

This document categorizes SFs in two types, depending on whether they are able to behave properly in the presence of SR information or not. These are respectively named SR-aware and SR-unaware SFs. An SR-aware SF can process the SR information in the packets it receives. This means being able to identify the active segment as a local instruction and move forward in the segment list, but also that the SF own behavior is not hindered due to the presence of SR information. For example, an SR-aware firewall filtering SRv6 traffic based on its final destination must retrieve that information from the last entry in the SRH rather than the Destination Address field of the IPv6 header. Any SF that does not meet these criteria is considered as SR-unaware.

#### 4.1. SR-aware SFs

An SR-aware SF is associated with a locally instantiated service segment, which is used to steer traffic through it.

If the SF is configured to intercept all the packets passing through the appliance, the underlying routing system only has to implement a default SR endpoint behavior (SR-MPLS node segment or SRv6 End function), and the corresponding SID will be used to steer traffic through the SF.

If the SF requires the packets to be directed to a specific virtual interface, networking queue or process, a dedicated SR behavior may be required to steer the packets to the appropriate location. The definition of such SF-specific functions is out of the scope of this document.

An SRv6-aware SF may also retrieve, store or modify information in the SRH TLVs.

#### 4.2. SR-unaware SFs

An SR-unaware SF is not able to process the SR information in the traffic that it receives. It may either drop the traffic or take erroneous decisions due to the unrecognized routing information. In order to include such SFs in an SR SC policy, it is thus required to remove the SR information as well as any other encapsulation header before the SF receives the packet, or to alter it in such a way that the SF can correctly process the packet.

In this document, we define the concept of an SR proxy as an entity, separate from the SF, that performs these modifications and handle the SR processing on behalf of an SF. The SR proxy can run as a separate process on the SF appliance, on a virtual switch or router

on the compute node or on a remote host. In this document, we only assume that the proxy is connected to the SF via a layer-2 link.

An SR-unaware SF is associated with a service segment instantiated on the SR proxy, which is used to steer traffic through the SF. Section 6 describes several SR proxy behaviors to handle the encapsulation headers and SR information under various circumstances.

## 5. Service function chaining

When applying a particular Service Function Chain (SFC) [RFC7665] to the traffic selected by a service classifier, the traffic need to be steered through an ordered set of Service Functions (SF) in the network. This ordered set of SFs in the network indicates the Service Function Path (SFP) associated with the above SFC. In order to steer the selected traffic through the required ordered list of SFs, the service classifier needs to attach information to the packet specifying exactly which Service Function Forwarders (SFFs) and which SFs are to be visited by traffic, the SFC, or the partially specified SFP which is in between the former two extremes.

The SR source routing mechanisms can be used to steer traffic through an ordered set of devices (i.e., an explicit path) and instruct those nodes to execute specific operations on the packet.

This section describes how to leverage SR to realize a transport-independent service function chaining by encoding the service function path information or service function chain information as an MPLS label stack or an IPv6 SRH.

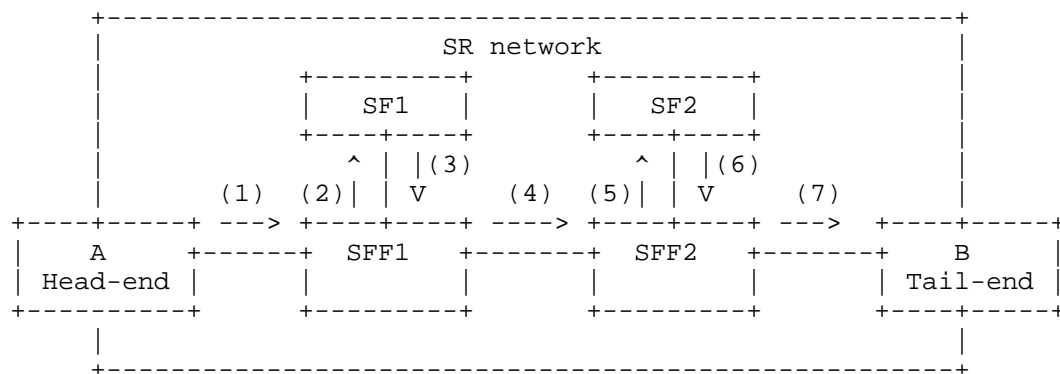


Figure 1: Service Function Chaining in SR networks

As shown in Figure 1, SFF1 and SFF2 are two SR-capable nodes. They are also SFFs, each with one SF attached. In addition, they have

allocated and advertised segments for their locally attached SFs. For example, SFF1 allocates and advertises a SID (i.e., S(SF1)) for SF1 while SFF2 allocates and advertises a SID (i.e., S(SF2)) for SF2. These SIDs, which are used to indicate SFs, are referred to as service segments, while the SFFs are identified by either node or adjacency segments depending on how strictly the network path needs to be specified. In this example, we assume that the traffic is steered to both SFFs using their node segments S(SFF1) and S(SFF2), respectively.

Now assume that a given traffic flow is steered in an SR policy instantiated on node A with an endpoint B, hereafter referred to as the SR policy head-end and tail-end respectively, and associated with particular SFC requirements (i.e., SF1-> SF2). From an SR policy perspective, SFC is only a particular case of traffic engineering where the SR path includes service functions. An SR-SFC policy inherits all the properties of SR-TE policies as defined in [I-D.filsfils-spring-segment-routing-policy]. Section 5.1 and Section 5.2 describe approaches of leveraging the SR-MPLS and SRv6 mechanisms to realize stateless service function chaining. The complete SFP and SFC information is encoded within an MPLS label stack or an IPv6 SRH carried by the packets, so that no per-chain state is required at the intermediate hops. Since the encoding of the partially specified SFP is just a simple combination of the encoding of the SFP and the encoding of the SFC, this document would not describe how to encode the partially specified SFP anymore.

## 5.1. SR-MPLS data plane

### 5.1.1. Encoding SFP Information by an MPLS Label Stack

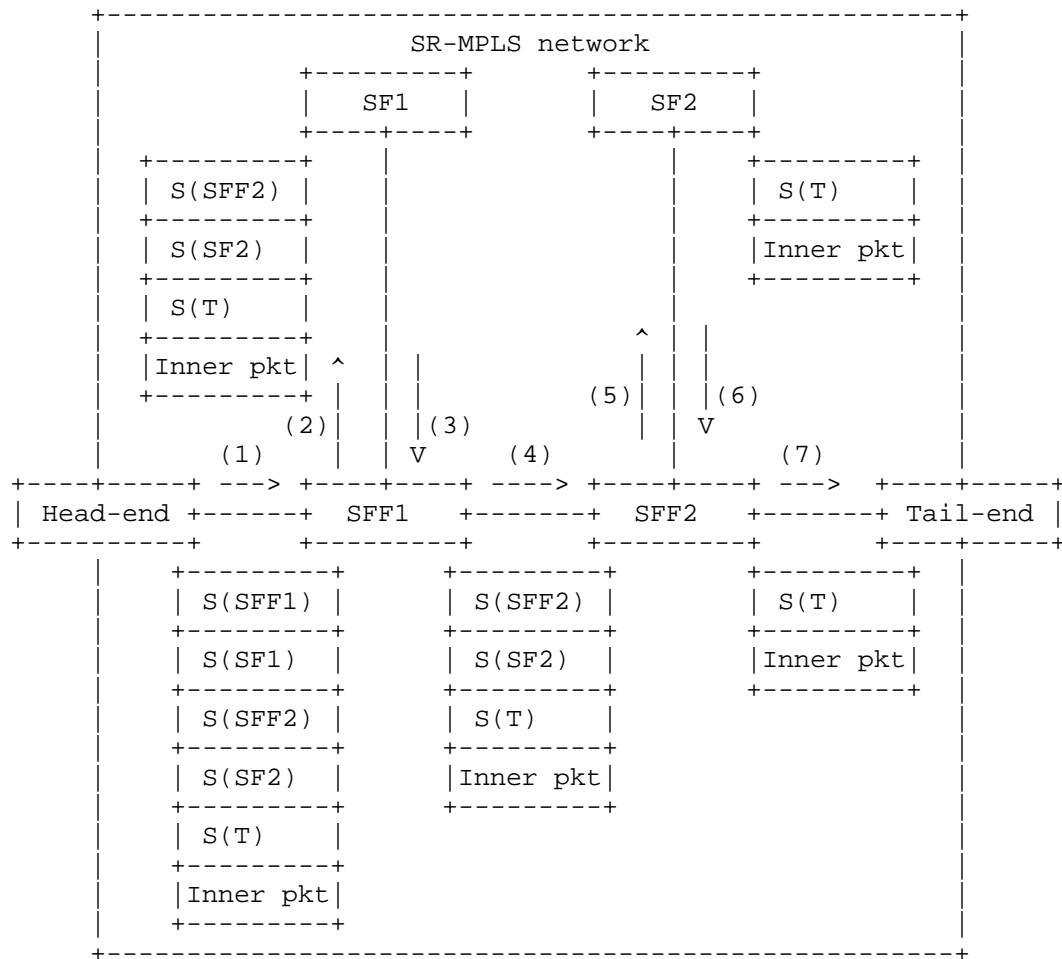


Figure 2: Packet walk in MPLS underlay

As shown in Figure 2, the head-end, acting as a service classifier, determines that the selected packet needs to travel through an SFC (SF1->SF2) and steers this packet into the appropriate SR policy as described in [I-D.filsfils-spring-segment-routing-policy]. As a result, the packet is encapsulated with an MPLS label stack containing the segment list <SFF1, SF1, SFF2, SF2, T>. This segment list encodes in a stateless manner the SFP corresponding to the above SFC as an MPLS label stack where each service segment is a local MPLS label allocated from SFFs' label spaces. To some extent, the MPLS label stack here could be looked as a specific implementation of the SFC encapsulation used for containing the SFP information [RFC7665], which does not require the SFF to maintain per-chain state.

When the encapsulated packet arrives at SFF1, SFF1 knows how to send the packet to SF1 based on the top label (i.e., S(SF1)) of the received MPLS packet. We first consider the case where SF1 is an SR-aware SF, i.e., it understands how to process a packet with a pre-pended SR-MPLS label stack. In this case the packet would be sent to SF1 by SFF1 with the label stack S(SFF2)->S(SF2). SF1 would perform the required service function on the received MPLS packet where the payload type is determined using the first nibble of the MPLS payload. After the MPLS packet is returned from SF1, SFF1 would send it to SFF2 according to the top label (i.e., S(SFF2)).

If SF1 is an SR-unaware SF, i.e. one that is unable to process the MPLS label stack, the remaining MPLS label stack (i.e., S(SFF2)->S(SF2)) MUST be stripped from the packet before sending the packet to SF1. When the packet is returned from SF1, SFF1 would re-impose the MPLS label stack which had been previously stripped and then send the packet to SFF2 according to the current top label (i.e., S(SFF2)). Proxy mechanisms to support SR-unaware SFs are proposed in section 6 of this document.

When the encapsulated packet arrives at SFF2, SFF2 would perform the similar action to that described above.

By leveraging the SR-MPLS data plane, [I-D.xu-mpls-sr-over-ip] describes a source routing instruction which works across both IPv4 and IPv6 underlays in addition to the MPLS underlay. As shown in Figure 3, if there is no MPLS LSP towards the next node segment (i.e., the next SFF identified by the current top label), the corresponding IP-based tunnel for MPLS (e.g., MPLS-in-IP/GRE tunnel [RFC4023], MPLS-in-UDP tunnel [RFC7510] or MPLS-in-L2TPv3 tunnel [RFC4817]) would be used.

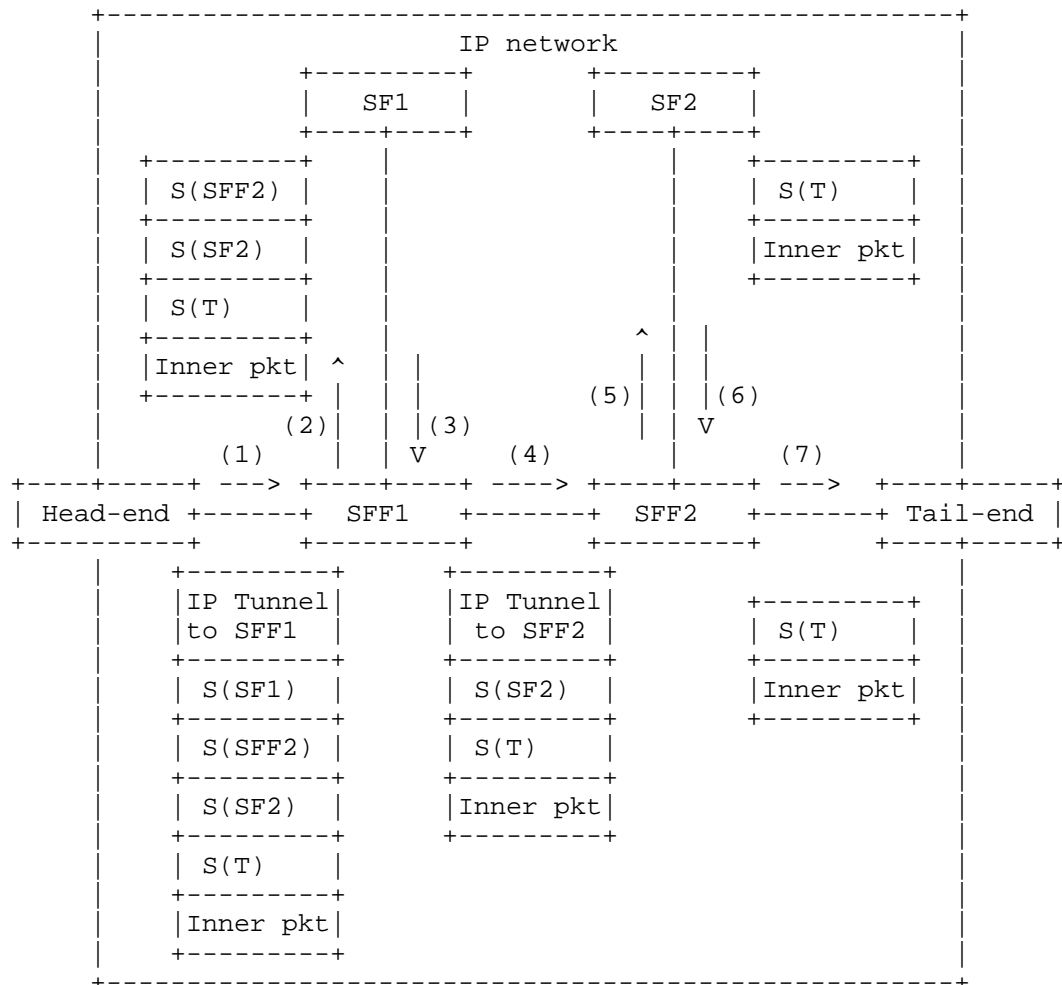


Figure 3: Packet walk in IP underlay

Since the transport (i.e., the underlay) could be IPv4, IPv6 or even MPLS networks, the above approach of encoding the SFP information by an MPLS label stack is fully transport-independent which is one of the major requirements for the SFC encapsulation [RFC7665].

#### 5.1.2. Encoding SFC Information by an MPLS Label Stack

The head-end, acting as a service classifier, determines that the selected packet needs to travel through an SFC (SF1->SF2) and steers this packet into the appropriate SR policy as described in [I-D.filsfils-spring-segment-routing-policy]. This results in the

packet being encapsulated with an MPLS label stack containing the segment list <SF1, SF2, T>, which encodes that SFC. Those SF labels MUST be domain-wide unique MPLS labels. Since it is known to the service classifier that SFF1 is attached with an instance of SF1, the service classifier would therefore send the MPLS encapsulated packet through either an MPLS LSP tunnel or an IP-based tunnel towards SFF1 (as shown in Figure 4 and Figure 5 respectively). When the MPLS encapsulated packet arrives at SFF1, SFF1 would know which SF should be performed according to the current top label (i.e., S(SF1)). Similarly, SFF1 would send the packet returned from SF1 to SFF2 through either an MPLS LSP tunnel or an IP-based tunnel towards SFF2 since it's known to SFF1 that SFF2 is attached with an instance of SF2. When the encapsulated packet arrives at SFF2, SFF2 would do the similar action as what has been done by SFF1. Since the transport (i.e., the underlay) could be IPv4, IPv6 or even MPLS networks, the above approach of encoding the SFC information by an MPLS label stack is fully transport-independent which is one of the major requirements for the SFC encapsulation [RFC7665].



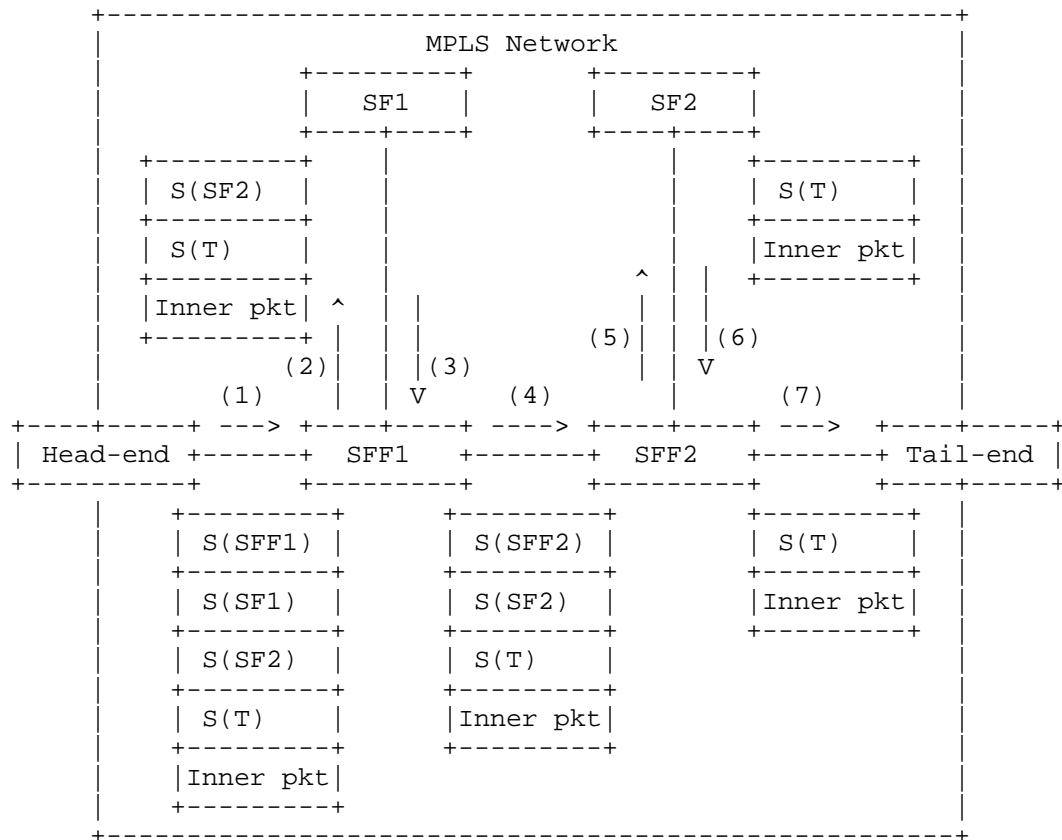


Figure 4: Packet walk in MPLS underlay

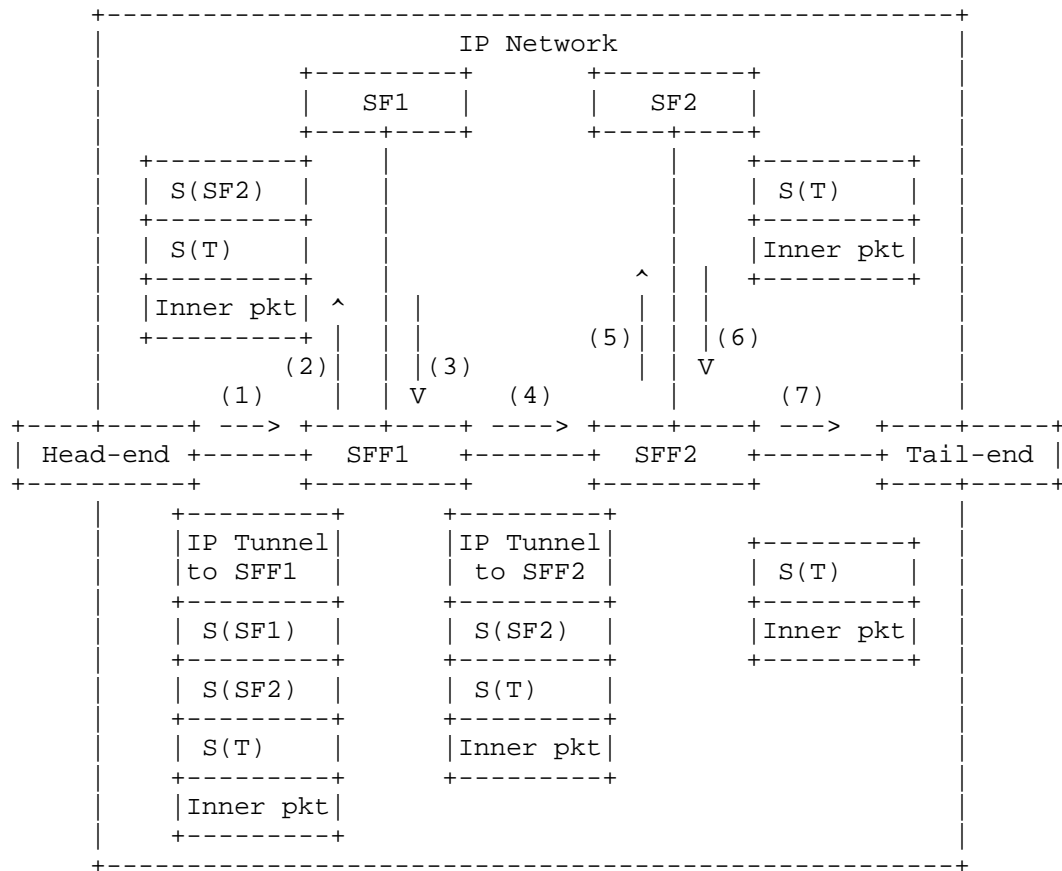


Figure 5: Packet walk in IP underlay

## 5.2. SRv6 data plane

### 5.2.1. Encoding SFP Information by an SRv6 SRH

Figure 6: Packet walk in SRv6 network

As shown in Figure 6, the head-end, acting as a service classifier, determines that the selected packet needs to travel through an SFC (SF1->SF2) and steers this packet into the appropriate SR policy as described in [I-D.filsfils-spring-segment-routing-policy]. As a result, the packet is encapsulated with an IPv6 header and an SRH containing the segment list <SFF1:SF1, SFF2:SF2, T>. The intermediate segments in this list leverage the SRv6 locator-function concept introduced in [I-D.filsfils-spring-srv6-network-programming] to encode both the SFF and the SF in a single IPv6 SID. The traffic is steered via regular IPv6 forwarding up to the SFF represented in the locator part of the SID and then passed to the SF identified by the SID function. This SRH thus indicates in a stateless manner the SFP corresponding to the above SFC. To some extent, the SRH here could be looked as a specific implementation of the SFC encapsulation used for containing the SFP information [RFC7665], which does not require the SFF to maintain per-chain state.

When the encapsulated packet arrives at SFF1, SFF1 knows how to send the packet to the SF based on the active segment. We first consider the case where SF1 is an SR-aware SF, i.e., it understands how to process an IPv6 encapsulated packet with an SRH. In this case the packet is sent to SF1 by SFF1 with the IP and SR headers (H,SFF2:SF2)(T,SFF2:SF2,SFF1:SF1;SL=1). SF1 performs the required

service function on the received packet, where the payload is determined based on the Next Header field value of last extension header and/or the active segment. After the packet is returned from SF1, SFF1 simply forwards it to SFF2 according to the IPv6 destination address.

If SF1 is an SR-unaware SF, i.e. one that is unable to process IPv6 encapsulated packets with an SRH, the encapsulation headers (i.e., outer IPv6 with any extension header) MUST be stripped from the packet before it is sent to SF1. When the packet is returned from SF1, SFF1 would re-encapsulate the packet with the IPv6 and SR headers that had been previously stripped and then send the packet to SFF2 according to the IPv6 destination address. Proxy mechanisms to support SR-unaware SFs are proposed in section 6 of this document.

When the encapsulated packet arrives at SFF2, SFF2 would perform the similar action to that described above.

#### 5.2.2. Encoding SFC Information by an IPv6 SRH

The head-end, acting as a service classifier, determines that the selected packet needs to travel through an SFC (SF1->SF2) and steers this packet into the appropriate SR policy. This results in the packet being encapsulated with an IPv6 header and an SRH containing the segment list <A1:SF1, A2:SF2, T>. In this case, the locator parts A1 and A2 of the intermediate service segments are anycast prefixes advertised by several SFFs attached to SF1 and SF2, respectively. The policy head-end may thus let the traffic be steered to the closest instance of each SF or add intermediate segments to select a particular SF instance. Furthermore, since it is known to the head-end that SFF1 is attached to an instance of SF1, the encapsulated packet may be sent to SFF1 through an MPLS LSP or an IP-based tunnel. Similar tunneling can then be performed between SFF1 and SFF1, and between SFF2 and the tail-end, as illustrated on Figure 7. Since the transport (i.e., the underlay) could be IPv4, IPv6 or even MPLS, the above approach of encoding the SFC information by an IPv6 SRH is fully transport-independent which is one of the major requirements for the SFC encapsulation [RFC7665].

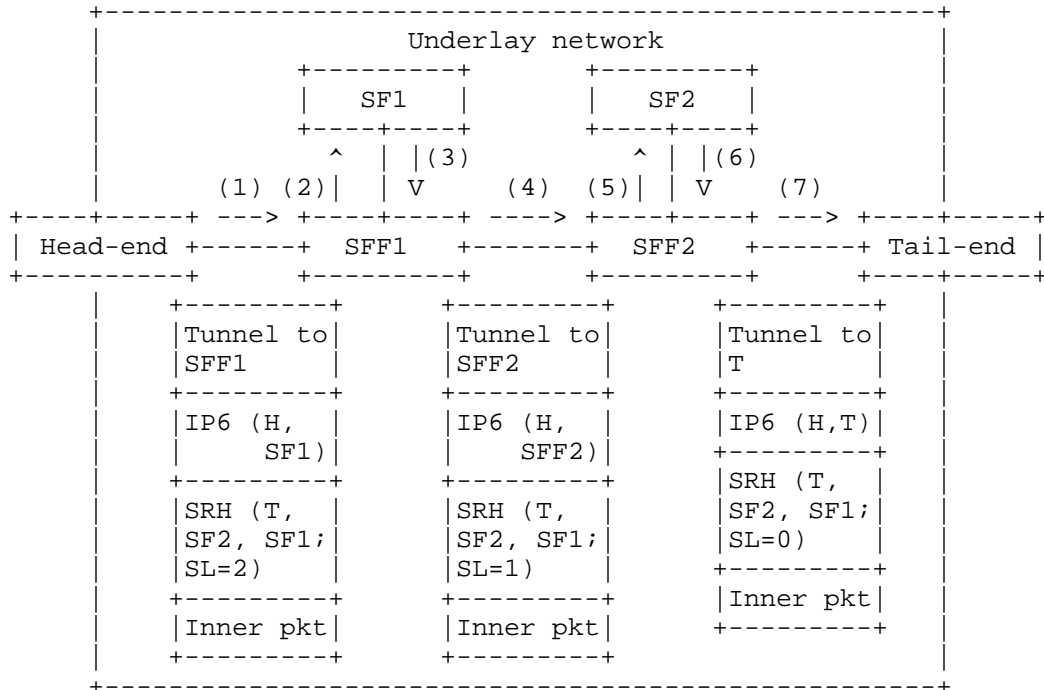


Figure 7: Packet walk in underlay network

## 6. SR proxy behaviors

This section describes several SR proxy behaviors designed to enable SR SFC through SR-unaware SFs. A system implementing one of these functions may handle the SR processing on behalf of an SR-unaware SF and allows the SF to properly process the traffic that is steered through it.

An SF may be located at any hop in an SR policy, including the last segment. However, the SR proxy behaviors defined in this section are dedicated to supporting SR-unaware SFs at intermediate hops in the segment list. In case an SR-unaware SF is at the last segment, it is sufficient to ensure that the SR information is ignored (IPv6 routing extension header with Segments Left equal to 0) or removed before the packet reaches the SF (MPLS PHP, SRv6 End.D or PSP).

As illustrated on Figure 8, the generic behavior of an SR proxy has two parts. The first part is in charge of passing traffic from the network to the SF. It intercepts the SR traffic destined for the SF via a locally instantiated service segment, modifies it in such a way that it appears as non-SR traffic to the SF, then sends it out on a

given interface, IFACE-OUT, connected to the SF. The second part receives the traffic coming back from the SF on IFACE-IN, restores the SR information and forwards it according to the next segment in the list. IFACE-OUT and IFACE-IN are respectively the proxy interface used for sending traffic to the SF and the proxy interface that receives the traffic coming back from the SF. These can be physical interfaces or sub-interfaces (VLANs) and, unless otherwise stated, IFACE-OUT and IFACE-IN can represent the same interface.

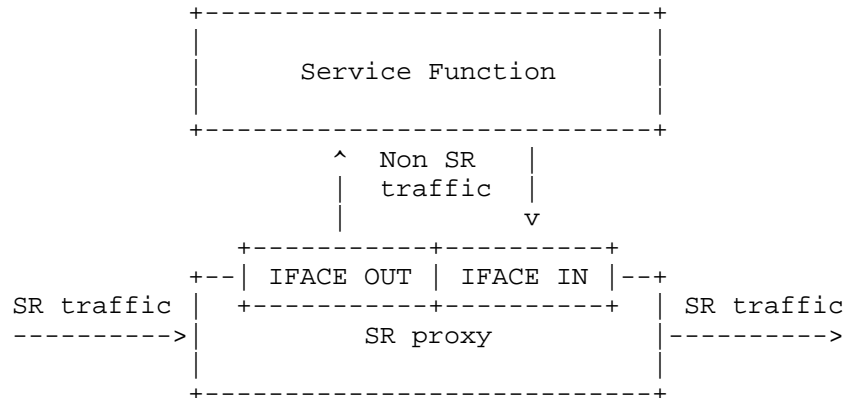


Figure 8: Generic SR proxy

In the next subsections, the following SR proxy mechanisms are defined:

- o Static proxy
- o Dynamic proxy
- o Shared-memory proxy
- o Masquerading proxy

Each mechanism has its own characteristics and constraints, which are summarized in the below table. It is up to the operator to select the best one based on the proxy node capabilities, the SF behavior and the traffic type. It is also possible to use different proxy mechanisms within the same service chain.

|                              |                      | S<br>t<br>a<br>t<br>i<br>c | D<br>y<br>n<br>a<br>m<br>i<br>c | S<br>h<br>a<br>r<br>e<br>d<br>m<br>e<br>m<br>. | M<br>a<br>s<br>q<br>u<br>e<br>r<br>a<br>d<br>i<br>n<br>g |
|------------------------------|----------------------|----------------------------|---------------------------------|------------------------------------------------|----------------------------------------------------------|
| SR flavors                   | SR-MPLS              | Y                          | Y                               | Y                                              | -                                                        |
|                              | SRv6 insertion       | P                          | P                               | P                                              | Y                                                        |
|                              | SRv6 encapsulation   | Y                          | Y                               | Y                                              | -                                                        |
| Inner header                 | Ethernet             | Y                          | Y                               | Y                                              | -                                                        |
|                              | IPv4                 | Y                          | Y                               | Y                                              | -                                                        |
|                              | IPv6                 | Y                          | Y                               | Y                                              | -                                                        |
| Chain agnostic configuration |                      | N                          | N                               | Y                                              | Y                                                        |
| Transparent to chain changes |                      | N                          | Y                               | Y                                              | Y                                                        |
| SF support                   | DA modification      | Y                          | Y                               | Y                                              | NAT                                                      |
|                              | Payload modification | Y                          | Y                               | Y                                              | Y                                                        |
|                              | Packet generation    | Y                          | Y                               | cache                                          | cache                                                    |
|                              | Packet deletion      | Y                          | Y                               | Y                                              | Y                                                        |
|                              | Transport endpoint   | Y                          | Y                               | cache                                          | cache                                                    |

Figure 9: SR proxy summary

Note: The use of a shared memory proxy requires both the SF and the proxy to be running on the same node.

### 6.1. Static SR proxy

The static proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware SF. This proxy thus receives SR traffic that is formed of an MPLS label stack or an IPv6 header on top of an inner packet, which can be Ethernet, IPv4 or IPv6.

A static SR proxy segment is associated with the following mandatory parameters:

- o INNER-TYPE: Inner packet type
- o S-ADDR: Ethernet or IP address of the SF (only for inner type IPv4 and IPv6)
- o IFACE-OUT: Local interface for sending traffic towards the SF
- o IFACE-IN: Local interface receiving the traffic coming back from the SF
- o CACHE: SR information to be attached on the traffic coming back from the SF, including at least
  - \* CACHE.SA: IPv6 source address (SRv6 only)
  - \* CACHE.LIST: Segment list expressed as MPLS labels or IPv6 address

A static SR proxy segment is thus defined for a specific SF, inner packet type and cached SR information. It is also bound to a pair of directed interfaces on the proxy. These may be both directions of a single interface, or opposite directions of two different interfaces. The latter is recommended in case the SF is to be used as part of a bi-directional SR SC policy. If the proxy and the SF both support 802.1Q, IFACE-OUT and IFACE-IN can also represent sub-interfaces.

The first part of this behavior is triggered when the proxy node receives a packet whose active segment matches a segment associated with the static proxy behavior. It removes the SR information from the packet then sends it on a specific interface towards the associated SF. This SR information corresponds to the full label stack for SR-MPLS or to the encapsulation IPv6 header with any attached extension header in the case of SRv6.

The second part is an inbound policy attached to the proxy interface receiving the traffic returning from the SF, IFACE-IN. This policy attaches to the incoming traffic the cached SR information associated



with the SR proxy segment. If the proxy segment uses the SR-MPLS data plane, CACHE contains a stack of labels to be pushed on top the packets. With the SRv6 data plane, CACHE is defined as a source address, an active segment and an optional SRH (tag, segments left, segment list and metadata). The proxy encapsulates the packets with an IPv6 header that has the source address, the active segment as destination address and the SRH as a routing extension header. After the SR information has been attached, the packets are forwarded according to the active segment, which is represented by the top MPLS label or the IPv6 Destination Address.

In this scenario, there are no restrictions on the operations that can be performed by the SF on the stream of packets. It may operate at all protocol layers, terminate transport layer connections, generate new packets and initiate transport layer connections. This behavior may also be used to integrate an IPv4-only SF into an SRv6 policy. However, a static SR proxy segment can be used in only one service chain at a time. As opposed to most other segment types, a static SR proxy segment is bound to a unique list of segments, which represents a directed SR SC policy. This is due to the cached SR information being defined in the segment configuration. This limitation only prevents multiple segment lists from using the same static SR proxy segment at the same time, but a single segment list can be shared by any number of traffic flows. Besides, since the returning traffic from the SF is re-classified based on the incoming interface, an interface can be used as receiving interface (IFACE-IN) only for a single SR proxy segment at a time. In the case of a bi-directional SR SC policy, a different SR proxy segment and receiving interface are required for the return direction.

#### 6.1.1. SR-MPLS pseudocode

##### 6.1.1.1. Static proxy for inner type Ethernet

Upon receiving an MPLS packet with top label L, where L is an MPLS L2 static proxy segment, a node N does:

1. IF payload type is Ethernet THEN
2.     Pop all labels
3.     Forward the exposed frame on IFACE-OUT
4. ELSE
5.     Drop the packet

Upon receiving on IFACE-IN an Ethernet frame with a destination address different than the interface address, a node N does:

1. Push labels in CACHE on top of the frame Ethernet header
2. Lookup the top label and proceed accordingly

The receiving interface must be configured in promiscuous mode in order to accept those Ethernet frames.

#### 6.1.1.2. Static proxy for inner type IPv4

Upon receiving an MPLS packet with top label L, where L is an MPLS IPv4 static proxy segment, a node N does:

1. IF payload type is IPv4 THEN
2.     Pop all labels
3.     Forward the exposed packet on IFACE-OUT towards S-ADDR
4. ELSE
5.     Drop the packet

Upon receiving a non link-local IPv4 packet on IFACE-IN, a node N does:

1. Decrement TTL and update checksum
2. Push labels in CACHE on top of the packet IPv4 header
3. Lookup the top label and proceed accordingly

#### 6.1.1.3. Static proxy for inner type IPv6

Upon receiving an MPLS packet with top label L, where L is an MPLS IPv6 static proxy segment, a node N does:

1. IF payload type is IPv6 THEN
2.     Pop all labels
3.     Forward the exposed packet on IFACE-OUT towards S-ADDR
4. ELSE
5.     Drop the packet

Upon receiving a non link-local IPv6 packet on IFACE-IN, a node N does:

1. Decrement Hop Limit
2. Push labels in CACHE on top of the packet IPv6 header
3. Lookup the top label and proceed accordingly

#### 6.1.2. SRv6 pseudocode

##### 6.1.2.1. Static proxy for inner type Ethernet

Upon receiving an IPv6 packet destined for S, where S is an IPv6 static proxy segment for Ethernet traffic, a node N does:

1. IF ENH == 59 THEN ;; Ref1
2.     Remove the (outer) IPv6 header and its extension headers
3.     Forward the exposed frame on IFACE-OUT
4. ELSE
5.     Drop the packet

Ref1: 59 refers to "no next header" as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving on IFACE-IN an Ethernet frame with a destination address different than the interface address, a node N does:

1. Retrieve CACHE entry matching IFACE-IN and traffic type
2. Push SRH with CACHE.LIST on top of the Ethernet header ;; Ref2
3. Push IPv6 header with
  - SA = CACHE.SA
  - DA = CACHE.LIST[0] ;; Ref3
  - Next Header = 43 ;; Ref4
4. Set outer payload length and flow label
5. Lookup outer DA in appropriate table and proceed accordingly

Ref2: Unless otherwise specified, the segments in CACHE.LIST should be encoded in reversed order, Segment Left and Last Entry values should be set of the length of CACHE.LIST minus 1, and Next Header should be set to 59.

Ref3: CACHE.LIST[0] represents the first IPv6 SID in CACHE.LIST.

Ref4: If CACHE.LIST contains a single entry, the SRH can be omitted and the Next Header value must be set to 59.

The receiving interface must be configured in promiscuous mode in order to accept those Ethernet frames.

#### 6.1.2.2. Static proxy for inner type IPv4

Upon receiving an IPv6 packet destined for S, where S is an IPv6 static proxy segment for IPv4 traffic, a node N does:

1. IF ENH == 4 THEN ;; Ref1
2.     Remove the (outer) IPv6 header and its extension headers
3.     Forward the exposed packet on IFACE-OUT towards S-ADDR
4. ELSE
5.     Drop the packet

Ref1: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving a non link-local IPv4 packet on IFACE-IN, a node N does:

1. Decrement TTL and update checksum
2. IF CACHE.SRH THEN ;; Ref2
3.     Push CACHE.SRH on top of the existing IPv4 header
4.     Set NH value of the pushed SRH to 4
5. Push outer IPv6 header with SA, DA and traffic class from CACHE
6. Set outer payload length and flow label
7. Set NH value to 43 if an SRH was added, or 4 otherwise
8. Lookup outer DA in appropriate table and proceed accordingly

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

#### 6.1.2.3. Static proxy for inner type IPv6

Upon receiving an IPv6 packet destined for S, where S is an IPv6 static proxy segment for IPv6 traffic, a node N does:

1. IF ENH == 41 THEN ;; Ref1
2.     Remove the (outer) IPv6 header and its extension headers
3.     Forward the exposed packet on IFACE-OUT towards S-ADDR
4. ELSE
5.     Drop the packet

Ref1: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N does:

1. Decrement Hop Limit
2. IF CACHE.SRH THEN ;; Ref2
3.     Push CACHE.SRH on top of the existing IPv6 header
4.     Set NH value of the pushed SRH to 41
5. Push outer IPv6 header with SA, DA and traffic class from CACHE
6. Set outer payload length and flow label
7. Set NH value to 43 if an SRH was added, or 41 otherwise
8. Lookup outer DA in appropriate table and proceed accordingly

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

## 6.2. Dynamic SR proxy

The dynamic proxy is an improvement over the static proxy that dynamically learns the SR information before removing it from the incoming traffic. The same information can then be re-attached to the traffic returning from the SF. As opposed to the static SR proxy, no CACHE information needs to be configured. Instead, the dynamic SR proxy relies on a local caching mechanism on the node instantiating this segment. Therefore, a dynamic proxy segment cannot be the last segment in an SR SC policy. As mentioned at the beginning of Section 6, a different SR behavior should be used if the SF is meant to be the final destination of an SR SC policy.

Upon receiving a packet whose active segment matches a dynamic SR proxy function, the proxy node pops the top MPLS label or applies the SRv6 End behavior, then compares the updated SR information with the cache entry for the current segment. If the cache is empty or different, it is updated with the new SR information. The SR information is then removed and the inner packet is sent towards the SF.

The cache entry is not mapped to any particular packet, but instead to an SR SC policy identified by the receiving interface (IFACE-IN). Any non-link-local IP packet or non-local Ethernet frame received on that interface will be re-encapsulated with the cached headers as described in Section 6.1. The SF may thus drop, modify or generate new packets without affecting the proxy.

### 6.2.1. SR-MPLS pseudocode

The dynamic proxy SR-MPLS pseudocode is obtained by inserting the following instructions between lines 1 and 2 of the static SR-MPLS pseudocode.

```
1.  IF top label S bit is 0 THEN
2.      Pop top label
3.      IF C(IFACE-IN) different from remaining labels THEN  ;; Ref1
4.          Copy all remaining labels into C(IFACE-IN)      ;; Ref2
5.  ELSE
6.      Drop the packet
```

Ref1: A TTL margin can be configured for the top label stack entry to prevent constant cache updates when multiple equal-cost paths with different hop counts are used towards the SR proxy node. In that case, a TTL difference smaller than the configured margin should not trigger a cache update (provided that the labels are the same).

Ref2: C(IFACE-IN) represents the cache entry associated to the dynamic SR proxy segment. It is identified with IFACE-IN in order to efficiently retrieve the right SR information when a packet arrives on this interface.

In addition, the inbound policy should check that C(IFACE-IN) has been defined before attempting to restore the MPLS label stack, and drop the packet otherwise.

#### 6.2.2. SRv6 pseudocode

The dynamic proxy SRv6 pseudocode is obtained by inserting the following instructions between lines 1 and 2 of the static proxy SRv6 pseudocode.

```
1.  IF NH=SRH & SL > 0 THEN
2.      Decrement SL and update the IPv6 DA with SRH[SL]
3.      IF C(IFACE-IN) different from IPv6 encaps THEN      ;; Ref1
4.          Copy the IPv6 encaps into C(IFACE-IN)          ;; Ref2
5.  ELSE
6.      Drop the packet
```

Ref1: "IPv6 encaps" represents the IPv6 header and any attached extension header.

Ref2: C(IFACE-IN) represents the cache entry associated to the dynamic SR proxy segment. It is identified with IFACE-IN in order to efficiently retrieve the right SR information when a packet arrives on this interface.

In addition, the inbound policy should check that C(IFACE-IN) has been defined before attempting to restore the IPv6 encapsulation, and drop the packet otherwise.

#### 6.3. Shared memory SR proxy

The shared memory proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware SF. This proxy behavior leverages a shared-memory interface with the SF in order to hide the SR information from an SR-unaware SF while keeping it attached to the packet. We assume in this case that the proxy and the SF are running on the same compute node. A typical scenario is an SR-capable vrouter running on a container host and forwarding traffic to virtual SFs isolated within their respective container.

More details will be added in a future revision of this document.

#### 6.4. Masquerading SR proxy

The masquerading proxy is an SR endpoint behavior for processing SRv6 traffic on behalf of an SR-unaware SF. This proxy thus receives SR traffic that is formed of an IPv6 header and an SRH on top of an inner payload. The masquerading behavior is independent from the inner payload type. Hence, the inner payload can be of any type but it is usually expected to be a transport layer packet, such as TCP or UDP.

A masquerading SR proxy segment is associated with the following mandatory parameters:

- o S-ADDR: Ethernet or IPv6 address of the SF
- o IFACE-OUT: Local interface for sending traffic towards the SF
- o IFACE-IN: Local interface receiving the traffic coming back from the SF

A masquerading SR proxy segment is thus defined for a specific SF and bound to a pair of directed interfaces or sub-interfaces on the proxy. As opposed to the static and dynamic SR proxies, a masquerading segment can be present at the same time in any number of SR SC policies and the same interfaces can be bound to multiple masquerading proxy segments. The only restriction is that a masquerading proxy segment cannot be the last segment in an SR SC policy.

The first part of the masquerading behavior is triggered when the proxy node receives an IPv6 packet whose Destination Address matches a masquerading proxy segment. The proxy inspects the IPv6 extension headers and substitutes the Destination Address with the last segment in the SRH attached to the IPv6 header, which represents the final destination of the IPv6 packet. The packet is then sent out towards the SF.

The SF receives an IPv6 packet whose source and destination addresses are respectively the original source and final destination. It does not attempt to inspect the SRH, as RFC8200 specifies that routing extension headers are not examined or processed by transit nodes. Instead, the SF simply forwards the packet based on its current Destination Address. In this scenario, we assume that the SF can only inspect, drop or perform limited changes to the packets. For example, Intrusion Detection Systems, Deep Packet Inspectors and non-NAT Firewalls are among the SFs that can be supported by a masquerading SR proxy. Variants of the masquerading behavior are

defined in Section 6.4.2 and Section 6.4.3 to support a wider range of SFs.

The second part of the masquerading behavior, also called de-masquerading, is an inbound policy attached to the proxy interface receiving the traffic returning from the SF, IFACE-IN. This policy inspects the incoming traffic and triggers a regular SRv6 endpoint processing (End) on any IPv6 packet that contains an SRH. This processing occurs before any lookup on the packet Destination Address is performed and it is sufficient to restore the right active segment as the Destination Address of the IPv6 packet.

#### 6.4.1. SRv6 masquerading proxy pseudocode

Masquerading: Upon receiving a packet destined for S, where S is an IPv6 masquerading proxy segment, a node N processes it as follows.

1. IF NH=SRH & SL > 0 THEN
2.     Update the IPv6 DA with SRH[0]
3.     Forward the packet on IFACE-OUT
4. ELSE
5.     Drop the packet

De-masquerading: Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N processes it as follows.

1. IF NH=SRH & SL > 0 THEN
2.     Decrement SL
3.     Update the IPv6 DA with SRH[SL] ; ; Ref1
4.     Lookup DA in appropriate table and proceed accordingly

Ref2: This pseudocode can be augmented to support the Penultimate Segment Popping (PSP) endpoint flavor. The exact pseudocode modification are provided in [I-D.filsfils-spring-srv6-network-programming].

#### 6.4.2. Variant 1: Destination NAT

SFs modifying the destination address in the packets they process, such as NATs, can be supported by a masquerading proxy with the following modification to the de-masquerading pseudocode.

De-masquerading - NAT: Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N processes it as follows.



1. IF NH=SRH & SL > 0 THEN
2.     Update SRH[0] with the IPv6 DA
3.     Decrement SL
4.     Update the IPv6 DA with SRH[SL]
5.     Lookup DA in appropriate table and proceed accordingly

#### 6.4.3. Variant 2: Caching

SFs generating packets or acting as endpoints for transport connections can be supported by adding a dynamic caching mechanism similar to the one described in Section 6.2.

More details will be added in a future revision of this document.

### 7. Metadata

#### 7.1. MPLS data plane

Since the MPLS encapsulation has no explicit protocol identifier field to indicate the protocol type of the MPLS payload, how to indicate the presence of metadata (i.e., the NSH which is only used as a metadata container) in an MPLS packet is a potential issue to be addressed. One possible way to address the above issue is: SFFs allocate two different labels for a given SF, one indicates the presence of NSH while the other indicates the absence of NSH. This approach has no change to the current MPLS architecture but it would require more than one label binding for a given SF. Another possible way is to introduce a protocol identifier field within the MPLS packet as described in [I-D.xu-mpls-payload-protocol-identifier].

More details about how to contain metadata within an MPLS packet would be considered in the future version of this draft.

#### 7.2. IPv6 data plane

##### 7.2.1. SRH TLV objects

The IPv6 SRH TLV objects are designed to carry all sorts of metadata. In particular, [I-D.ietf-6man-segment-routing-header] defines the NSH carrier TLV as a container for NSH metadata.

TLV objects can be imposed by the ingress edge router that steers the traffic into the SR SC policy.

An SR-aware SF may impose, modify or remove any TLV object attached to the first SRH, either by directly modifying the packet headers or via a control channel between the SF and its forwarding plane.

An SR-aware SF that re-classifies the traffic and steers it into a new SR SC policy (e.g. DPI) may attach any TLV object to the new SRH.

Metadata imposition and handling will be further discussed in a future version of this document.

#### 7.2.2. SRH tag

The SRH tag identifies a packet as part of a group or class of packets [I-D.ietf-6man-segment-routing-header].

In an SFC context, this field can be used to encode basic metadata in the SRH.

### 8. Implementation status

The static SR proxy is available for SR-MPLS and SRv6 on various Cisco hardware and software platforms. Furthermore, the following proxies are available on open-source software.

|                  |                                     | VPP         | Linux       |
|------------------|-------------------------------------|-------------|-------------|
| M<br>P<br>L<br>S | Static proxy                        | Available   | In progress |
|                  | Dynamic proxy                       | In progress | In progress |
|                  | Shared memory proxy                 | In progress | In progress |
| S<br>R<br>v<br>6 | Static proxy                        | Available   | In progress |
|                  | Dynamic proxy - Inner type Ethernet | In progress | In progress |
|                  | Dynamic proxy - Inner type IPv4     | Available   | Available   |
|                  | Dynamic proxy - Inner type IPv6     | Available   | Available   |
|                  | Shared memory proxy                 | In progress | In progress |
|                  | Masquerading proxy                  | Available   | Available   |
|                  | Masquerading proxy - NAT variant    | In progress | In progress |
|                  | Masquerading proxy - Cache variant  | In progress | In progress |

Open-source implementation status table

## 9. Related works

The Segment Routing solution addresses a wide problem that covers both topological and service chaining policies. The topological and service instructions can be either deployed in isolation or in combination. SR has thus a wider applicability than the architecture defined in [RFC7665]. Furthermore, the inherent property of SR is a stateless network fabric. In SR, there is no state within the fabric to recognize a flow and associate it with a policy. State is only present at the ingress edge of the SR domain, where the policy is encoded into the packets. This is completely different from other proposals such as [RFC8300] and the MPLS label swapping mechanism described in [I-D.farrel-mpls-sfc], which rely on state configured at every hop of the service chain.

## 10. IANA Considerations

This I-D requests the IANA to allocate, within the "SRv6 Endpoint Types" sub-registry belonging to the top-level "Segment-routing with IPv6 dataplane (SRv6) Parameters" registry, the following allocations:

| Value/Range | Hex | Endpoint function                   | Reference |
|-------------|-----|-------------------------------------|-----------|
| TBA         | TBA | End.AN - SR-aware function (native) | [This.ID] |
| TBA         | TBA | End.AS - Static proxy               | [This.ID] |
| TBA         | TBA | End.AD - Dynamic proxy              | [This.ID] |
| TBA         | TBA | End.AM - Masquerading proxy         | [This.ID] |

Table 1: SRv6 SFC Endpoint Types

## 11. Security Considerations

The security requirements and mechanisms described in [I-D.ietf-spring-segment-routing] and [I-D.ietf-6man-segment-routing-header] also apply to this document.

Furthermore, it is fundamental to the SFC design that the classifier is a trusted resource which determines the processing that the packet will be subject to, including for example the firewall. Where an SF is not SR-aware the packet may exist as an IP packet, however this is an intrinsic part of the SFC design which needs to define how a packet is protected in that environment. Where a tunnel is used to link two non-MPLS domains, the tunnel design needs to specify how it is secured.

Thus the security vulnerabilities are addressed in the underlying technologies used by this design, which itself does not introduce any new security vulnerabilities.

## 12. Acknowledgements

The authors would like to thank Loa Andersson, Andrew G. Malis, Adrian Farrel, Alexander Vainshtein and Joel M. Halpern for their valuable comments and suggestions on the document.

## 13. Contributors

P. Camarillo (Cisco), B. Peirens (Proximus), D. Steinberg (Steinberg Consulting), A. AbdelSalam (Gran Sasso Science Institute), G. Dawra (Cisco), S. Bryant (Huawei), H. Assarpour (Broadcom), H. Shah (Ciena), L. Contreras (Telefonica I+D), J. Tantsura (Individual), M. Vigoureux (Nokia) and J. Bhattacharya (Cisco) substantially contributed to the content of this document.

## 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, <<https://www.rfc-editor.org/info/rfc2119>>.

### 14.2. Informative References

[I-D.dawra-idr-bgp-sr-service-chaining]  
Dawra, G., Filsfils, C., daniel.bernier@bell.ca, d., Uttaro, J., Decraene, B., Elmalky, H., Xu, X., Clad, F., and K. Talaulikar, "BGP Control Plane Extensions for Segment Routing based Service Chaining", draft-dawra-idr-bgp-sr-service-chaining-02 (work in progress), January 2018.

[I-D.farrel-mpls-sfc]  
Farrel, A., Bryant, S., and J. Drake, "An MPLS-Based Forwarding Plane for Service Function Chaining", draft-farrel-mpls-sfc-04 (work in progress), March 2018.

[I-D.filsfils-spring-segment-routing-policy]

Filsfils, C., Sivabalan, S., Raza, K., Liste, J., Clad, F., Talaulikar, K., Ali, Z., Hegde, S., daniel.voyer@bell.ca, d., Lin, S., bogdanov@google.com, b., Krol, P., Horneffer, M., Steinberg, D., Decraene, B., Litkowski, S., and P. Mattes, "Segment Routing Policy for Traffic Engineering", draft-filsfils-spring-segment-routing-policy-05 (work in progress), February 2018.

[I-D.filsfils-spring-srv6-network-programming]

Filsfils, C., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., Sharif, M., Ayyangar, A., Mynam, S., Henderickx, W., Bashandy, A., Raza, K., Dukes, D., Clad, F., and P. Camarillo, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-03 (work in progress), December 2017.

[I-D.ietf-6man-segment-routing-header]

Previdi, S., Filsfils, C., Raza, K., Dukes, D., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-08 (work in progress), January 2018.

[I-D.ietf-spring-segment-routing]

Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[I-D.ietf-spring-segment-routing-mpls]

Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-12 (work in progress), February 2018.

[I-D.xu-mpls-payload-protocol-identifier]

Xu, X., Assarpour, H., and S. Ma, "MPLS Payload Protocol Identifier", draft-xu-mpls-payload-protocol-identifier-04 (work in progress), January 2018.

- [I-D.xu-mpls-sr-over-ip]  
Xu, X., Bryant, S., Farrel, A., Bashandy, A., Henderickx, W., and Z. Li, "SR-MPLS over IP", draft-xu-mpls-sr-over-ip-00 (work in progress), February 2018.
- [RFC4023] Worster, T., Rekhter, Y., and E. Rosen, Ed., "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005, <<https://www.rfc-editor.org/info/rfc4023>>.
- [RFC4817] Townsley, M., Pignataro, C., Wainner, S., Seely, T., and J. Young, "Encapsulation of MPLS over Layer 2 Tunneling Protocol Version 3", RFC 4817, DOI 10.17487/RFC4817, March 2007, <<https://www.rfc-editor.org/info/rfc4817>>.
- [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <<https://www.rfc-editor.org/info/rfc7510>>.
- [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>>.

#### Authors' Addresses

Francois Clad (editor)  
Cisco Systems, Inc.  
France

Email: [fclad@cisco.com](mailto:fclad@cisco.com)

Xiaohu Xu (editor)  
Alibaba

Email: [xiaohu.xxh@alibaba-inc.com](mailto:xiaohu.xxh@alibaba-inc.com)

Clarence Filsfils  
Cisco Systems, Inc.  
Belgium

Email: cf@cisco.com

Daniel Bernier  
Bell Canada  
Canada

Email: daniel.bernier@bell.ca

Cheng Li  
Huawei

Email: chenglil3@huawei.com

Bruno Decraene  
Orange  
France

Email: bruno.decraene@orange.com

Shaowen Ma  
Juniper

Email: mashaowen@gmail.com

Chaitanya Yadlapalli  
AT&T  
USA

Email: cy098d@att.com

Wim Henderickx  
Nokia  
Belgium

Email: wim.henderickx@nokia.com

Stefano Salsano  
Universita di Roma "Tor Vergata"  
Italy

Email: stefano.salsano@uniroma2.it