

SFC WG
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2019

T. Ao
ZTE Corporation
G. Mirsky
ZTE Corp.
Z. Chen
China Telecom
K. Leung
Cisco System
March 10, 2019

SFC OAM for path consistency
draft-ao-sfc-oam-path-consistency-05

Abstract

Service Function Chain (SFC) defines an ordered set of service functions (SFs) to be applied to packets and/or frames and/or flows selected as a result of classification. SFC Operation, Administration and Maintenance can monitor the continuity of the SFC, i.e., that all elements of the SFC are reachable to each other in the downstream direction. But SFC OAM must support verification that the order of traversing these SFs corresponds to the state defined by the SFC control plane or orchestrator, the metric referred in this document as the path consistency of the SFC. This document defines a new SFC OAM method to support SFC consistency check, i.e. verification that all elements of the given SFC are being traversed in the expected order.

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 11, 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	2
2. Conventions used in this document	3
2.1. Terminology	3
2.2. Requirements Language	3
3. Consistency OAM: Theory of Operation	3
3.1. COAM packet	4
3.2. SFF Information Record TLV	4
3.3. SF Information Sub-TLV	5
3.4. SF Information Sub-TLV Construction	6
3.4.1. Multiple SFs as hops of SFP	6
3.4.2. Multiple SFs for load balance	7
4. Security Considerations	7
5. IANA Considerations	8
5.1. COAM Message Types	8
5.2. SFF Information Record TLV Type	8
5.3. SF Information Sub-TLV Type	8
5.4. SF Identifier Types	9
6. Acknowledgements	9
7. References	9
7.1. Normative References	9
7.2. Informational References	10
Authors' Addresses	10

1. Introduction

Service Function Chain (SFC) is a chain with a series of ordered Service Functions (SFs). Service Function Path (SFP) is a path of a SFC. SFC is described in detail in the SFC architecture document [RFC7665]. The SFs in the SFC are ordered and only when one SF processes traffic then it can be processed by the next SF. Changes in the order may cause errors. Sometimes, an SF uses the metadata

from its upstream SF process. That's why it's very important for the operator to make sure that the order of traversing the SFs is exactly as defined by the control plane or the orchestrator. This document refers to the correspondence between the state of the control plane and the SFP itself as the SFP consistency.

This document defines the method to check the path consistency of the SFP. It is an extension of the SFC Echo-request/Echo-reply specified in the [I-D.ietf-sfc-multi-layer-oam].

2. Conventions used in this document

2.1. Terminology

SFC(Service Function Chain): An ordered set of some abstract SFs.

SFF: Service Function Forwarder

SF: Service Function

OAM: Operation, Administration and Maintenance

SFP: Service Function Path

COAM(Consistency OAM): OAM that can be used to check the consistency of the Service Function Path.

2.2. Requirements Language

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. Consistency OAM: Theory of Operation

Consistency OAM(COAM) uses two functions: COAM Request and COAM Reply. Every SFF that receives the COAM Request MUST perform the following actions:

- o Collect information of traversed by the COAM Request packet SFs and send it to the ingress SFF as COAM Reply packet over IP network [I-D.ietf-sfc-multi-layer-oam];
- o Forward the COAM Request to next downstream SFF if the one exists.

As result, the ingress SFF collects information about all traversed SFFs and SFs, information of the actual path the COAM packet has traveled, so that we can verify the path consistency of the SFC. The mechanism for the SFP consistency verification is outside the scope of this document.

3.1. COAM packet

Consistency OAM introduces two new types of messages to the SFC Echo request/reply operation [I-D.ietf-sfc-multi-layer-oam] with the following values detailed in Section 5.1:

- o TBA1 - COAM Request
- o TBA2 - COAM Reply

Upon receiving the COAM Request, the SFF MUST respond with the COAM Reply. The SFF MUST include the SFs information, as described in Section 3.3 and Section 3.2.

The COAM packet is displayed in Figure 1.

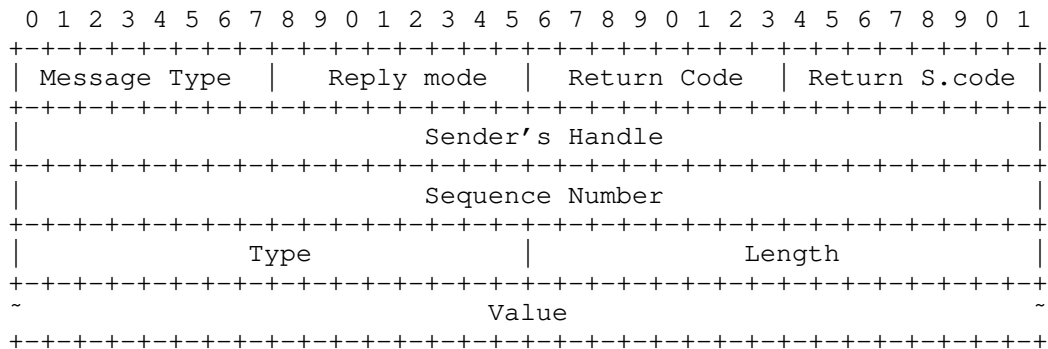


Figure 1: COAM Packet Header

3.2. SFF Information Record TLV

For COAM Request, the SFF MUST include the Information of SFs into the SF Information Record TLV in the COAM Reply message. Every SFF send back one COAM Reply Message with all the SFs that are attaching to the SFF along the SFP indicated by the COAM Request.

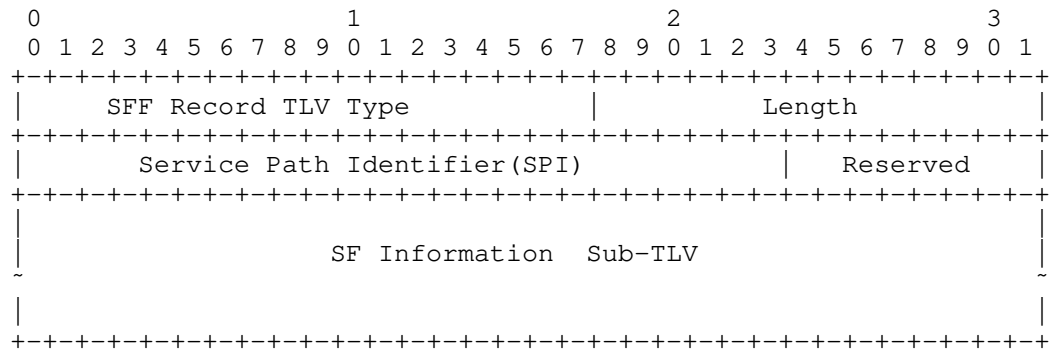


Figure 2: SFF Information Record TLV

Service Path Identifier (SPI): The identifier of SFP to which all the SFs in this TLV belong.

SF Information Sub-TLV: The Sub-TLV as defined in Figure 3.

3.3. SF Information Sub-TLV

Every SFF receiving COAM Request packet MUST include the SF characteristic data into the COAM Reply packet. The data format of each SF includes in a COAM Reply packet as SF Information sub-TLV that is displayed in Figure 3.

After the COAM traversed the SFP, all the information of the SFs on the SFP are collected from the TLVs with COAM Reply.

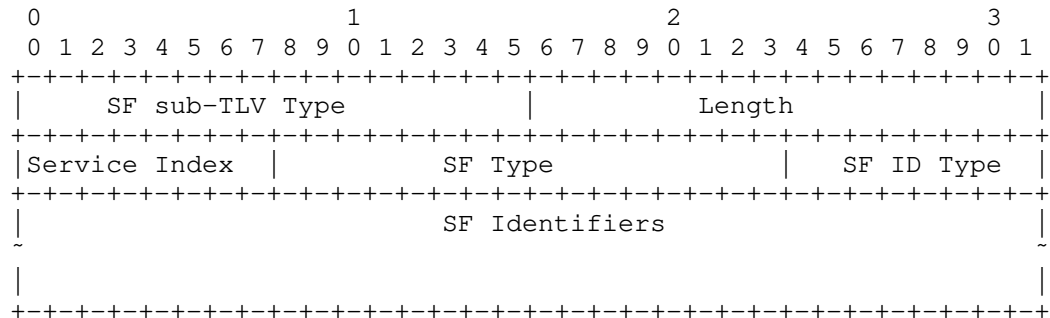


Figure 3: Service Function information sub-TLV

SF sub-TLV Type: Two octets long field. It indicates that the TLV is a SF TLV which contains the information of one SF.

Length: Two octets long field. The value of the field is the length of the data following the Length field counted in octets.

Service Index: Indicates the SF's position on the SFP.

SF Type: Two octets long field. It is defined in [I-D.ietf-bess-nsh-bgp-control-plane] and indicates the type of SF, e.g., Firewall, Deep Packet Inspection, WAN optimization controller, etc.

Reserved: For future use. MUST be zeroed on transmission and MUST be ignored on receipt.

SF ID Type: One octet long field with values defined as Section 5.4.

SF Identifier: An identifier of the SF. The length of the SF Identifier depends on the type of the SF ID Type. For example, if the SF Identifier is its IPv4 address, the SF Identifier should be 32 bits. SF ID Type and SF Identifier may be a list, indicating the list of the SFs are which are included in a load balance group.

3.4. SF Information Sub-TLV Construction

Each SFF in the SFP MUST send one and only one COAM Reply corresponding to the COAM Request. If there is only one SF attached to the SFF in such SFP, only one SF information sub-TLV is included in the on COAM Reply. If there are several SFs attached to the SFF in the SFP, SF Information Sub-TLV MUST be constructed as described below in either Section 3.4.1 and Section 3.4.2.

3.4.1. Multiple SFs as hops of SFP

Multiple SFs attached to one SFF are the hops of the SFP, the service indexes of these SFs are different. Service function types of these SFs could be different or be the same. Information about all SFs MAY be included in the COAM Reply message. Information about each SF MUST be listed as separate SF Information Sub-TLVs in the COAM Reply message.

An example of the COAM procedure for this case is shown in Figure 4. The Service Function Path(SPI=x) is SF1->SF2->SF4->SF3. The SF1, SF2 and SF3 are attached to SFF1, and SF4 is attached to SFF2. The COAM Request message is sent to the SFFs in the sequence of the SFP(SFF1->SFF2->SFF1). Every SFF(SFF1, SFF2) replies with the information of SFs belonging to the SFP. The SF information Sub-TLV in Figure 3 contains information for each SF(SF1, SF2, SF3 and SF4).

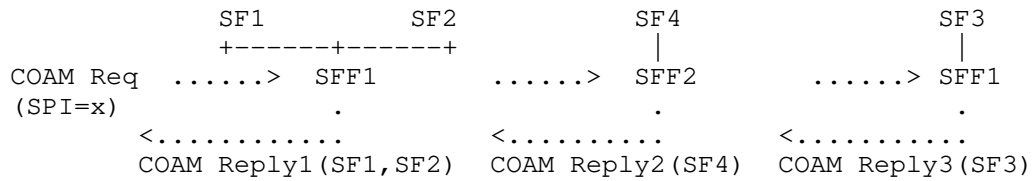


Figure 4: Example 1 for COAM Reply with multiple SFs

3.4.2. Multiple SFs for load balance

Multiple SFs may be attached to one SFF to balance the load, in other words, that means that the particular traffic flow will transmit only one of these SFs. These SFs have the same Service Function Type and Service Index. For this case, the SF identifiers and SF ID Type of all these SFs will be listed in the SF Identifiers field and SF ID Type in a single SF information sub-TLV of COAM Reply message. The number of these SFs can be calculated according to SF ID Type and the value of Length field of the sub-TLV.

An example of the COAM procedure for this case is shown in Figure 4. The Service Function Path(SFI=x) is SF1a/SF1b->SF2a/SF2b. The Service Functions SF1a and SF1b are attached to SFF1 which are load balance for each other, and the Service Functions SF2a and SF2b are attached to SFF2 which are load balance for each other as well. The COAM Request message is sent to the SFFs in the sequence of the SFP (i.e. SFF1->SFF2). Every SFF(SFF1,SFF2) replies with the information of SFs belonging to the SFP. The SF information Sub-TLV in Figure 3 contains information for all SFs at that hop.

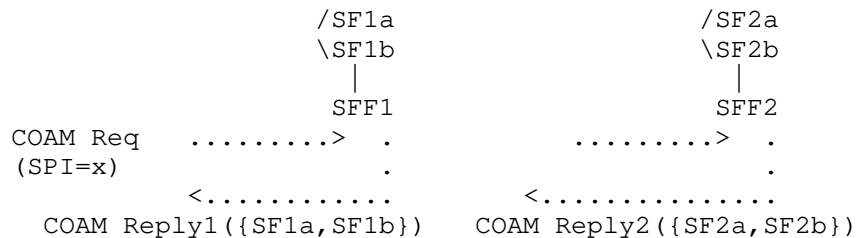


Figure 5: Example 2 for COAM Reply with multiple SFs

4. Security Considerations

Security considerations discussed in [RFC8300] and [I-D.ietf-sfc-multi-layer-oam] apply to this document.

Also, since Service Function sub-TLV discloses information about the SFP the spoofed COAM Request packet may be used to obtain network information, it is RECOMMENDED that implementations provide a means of checking the source addresses of COAM Request messages, specified in SFC Source TLV [I-D.ietf-sfc-multi-layer-oam], against an access list before accepting the message.

5. IANA Considerations

5.1. COAM Message Types

IANA is requested to assign values from its Message Types sub-registry in SFC Echo Request/Echo Reply Message Types registry as follows:

Value	Description	Reference
TBA1	SFP Consistency Echo Request	This document
TBA2	SFP Consistency Echo Reply	This document

Table 1: SFP Consistency Echo Request/Echo Reply Message Types

5.2. SFF Information Record TLV Type

IANA is requested to assign new type value from SFC OAM TLV Type registry as follows:

Value	Description	Reference
TBA3	SFF Information Record Type	This document

Table 2: SFF-Information Record

5.3. SF Information Sub-TLV Type

IANA is requested to assign new type value from SFC OAM TLV Type registry as follows:

Value	Description	Reference
TBA4	SF Information	This document

Table 3: SF-Information Sub-TLV Type

5.4. SF Identifier Types

IANA is requested to create in the registry SF Types the new sub-registry SF Identifier Types. All code points in the range 1 through 191 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC8126] and assign values as follows:

Value	Description	Reference
0	Reserved	This document
TBA6	IPv4	This document
TBA7	IPv6	This document
TBA8	MAC	This document
TBA8+1-191	Unassigned	IETF Review
192-251	Unassigned	First Come First Served
252-254	Unassigned	Private Use
255	Reserved	This document

Table 4: SF Identifier Type

6. Acknowledgements

Thanks to John Drake for his review and the reference to the work on BGP Control Plane for NSH SFC.

Thanks to Joel M. Halpern for his suggestion about the load balance scenario.

Thank to Dirk von Hugo for his useful comments.

7. References

7.1. Normative References

- [I-D.ietf-bess-nsh-bgp-control-plane]
Farrel, A., Drake, J., Rosen, E., Uttaro, J., and L. Jalil, "BGP Control Plane for NSH SFC", draft-ietf-bess-nsh-bgp-control-plane-09 (work in progress), March 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-02 (work in progress), March 2019.
- [I-D.ietf-sfc-nsh-tlv]
Quinn, P., Elzur, U., and S. Majee, "Network Service Header TLVs", draft-ietf-sfc-nsh-tlv-00 (work in progress), January 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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.

7.2. Informational 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>>.

Authors' Addresses

Ting Ao
ZTE Corporation
No.889, BiBo Road
Shanghai 201203
China

Phone: +86 21 68897642
Email: ao.ting@zte.com.cn

Greg Mirsky
ZTE Corp.
1900 McCarthy Blvd. #205
Milpitas, CA 95035
USA

Email: gregimirsky@gmail.com

Zhonghua Chen
China Telecom
No.1835, South PuDong Road
Shanghai 201203
China

Phone: +86 18918588897
Email: 18918588897@189.cn

Kent Leung
Cisco System
170 West Tasman Drive
San Jose, CA 95134
USA

Email: kleung@cisco.com

SFC WG
Internet-Draft
Intended status: Standards Track
Expires: 1 October 2021

G. Mirsky
ZTE Corp.
T. Ao
Individual contributor
Z. Chen
China Telecom
K. Leung
Cisco System
G. Mishra
Verizon Inc.
30 March 2021

SFC OAM for path consistency
draft-ao-sfc-oam-path-consistency-11

Abstract

Service Function Chain (SFC) defines an ordered set of service functions (SFs) to be applied to packets and/or frames and/or flows selected due to classification. SFC Operation, Administration and Maintenance can monitor the continuity of the SFC, i.e., that all SFC elements are reachable to each other in the downstream direction. But SFC OAM must support verification that the order of traversing these SFs corresponds to the state defined by the SFC control plane or orchestrator, the metric referred to in this document as the path consistency of the SFC. This document defines a new SFC active OAM method to support SFC consistency check, i.e., verification that all elements of the given SFC are being traversed in the expected order.

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 1 October 2021.

Copyright Notice

Copyright (c) 2021 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	3
2.1. Acronyms	3
2.2. Requirements Language	3
3. Consistency OAM: Theory of Operation	3
3.1. COAM packet	4
3.2. SFF Information Record TLV	5
3.3. SF Information Sub-TLV	6
3.4. SF Information Sub-TLV Construction	7
3.4.1. Multiple SFs as hops of SFP	7
3.4.2. Multiple SFs for load balance	8
4. Security Considerations	8
5. IANA Considerations	8
5.1. COAM Message Types	9
5.2. SFF Information Record TLV Type	9
5.3. SF Information Sub-TLV Type	9
5.4. SF Identifier Types	10
6. Acknowledgements	10
7. References	10
7.1. Normative References	10
7.2. Informational References	11
Authors' Addresses	12

1. Introduction

Service Function Chain (SFC) is a chain with a series of ordered Service Functions (SFs). Service Function Path (SFP) is a path of a SFC. SFC is described in detail in the SFC architecture document [RFC7665]. The SFs in the SFC are ordered, i.e., only when an SF processes traffic, then it can be processed by the next SF. Changes in the order are very likely to cause errors. That's why an operator needs to ensure that the order of traversing the SFs is as defined by

the control plane or the orchestrator. This document refers to the correlation between the state of the control plane and the SFP itself as the SFP consistency. The need to verify the consistency of the particular SFP, using a mechanism of an active OAM protocol, is noted in [RFC8924].

This document defines the method to check the path consistency of the SFP. It is an extension of the SFC Echo-request/Echo-reply specified in the [I-D.ietf-sfc-multi-layer-oam].

2. Conventions used in this document

2.1. Acronyms

SFC: Service Function Chain. An ordered set of some abstract SFs.

SFF: Service Function Forwarder

SF: Service Function

OAM: Operation, Administration and Maintenance

SFP: Service Function Path

COAM: Consistency OAM, OAM that can be used to check the consistency of the Service Function Path.

MAC: Message Authentication Code

2.2. Requirements Language

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. Consistency OAM: Theory of Operation

Consistency OAM (COAM) uses two functions: COAM Request and COAM Reply. Every SFF that receives the COAM Request MUST perform the following actions:

- * Collect information of the traversed by the COAM Request packet SFs and send it to the ingress SFF as COAM Reply packet over IP network [I-D.ietf-sfc-multi-layer-oam];

- * Forward the COAM Request to the next downstream SFF if the one exists.

As a result, the ingress SFF collects information about all traversed SFFs and SFs, information on the actual path the COAM packet has traveled. That information is used to verify the SFC's path consistency. The mechanism for the SFP consistency verification is outside the scope of this document.

3.1. COAM packet

Consistency OAM introduces two new types of messages to the SFC Echo Request/Reply operation defined in [I-D.ietf-sfc-multi-layer-oam] with the following values detailed in Section 5.1:

- * TBA1 - COAM Request
- * TBA2 - COAM Reply

Upon receiving the COAM Request, the SFF MUST respond with the COAM Reply. The SFF MUST include the SFs information, as described in Section 3.3 and Section 3.2.

The COAM packet, defined in [I-D.ietf-sfc-multi-layer-oam], is displayed in Figure 1.

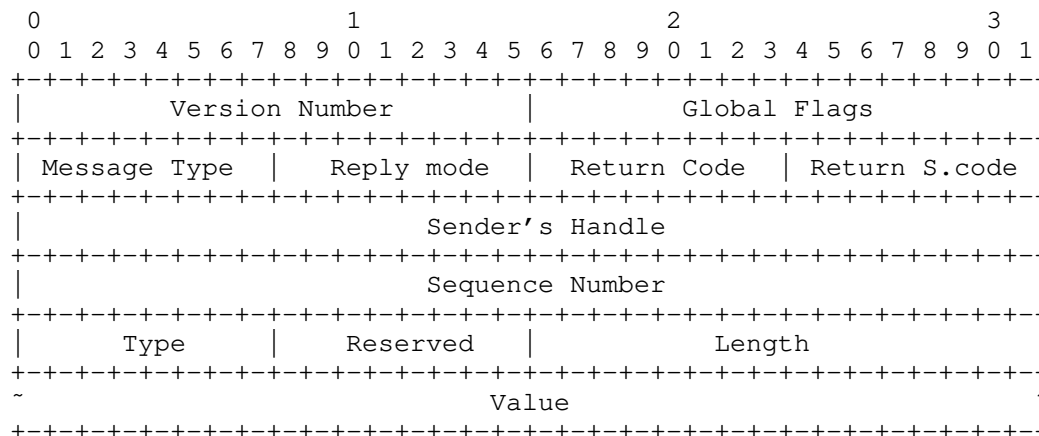


Figure 1: COAM Packet Header

The initiator of COAM Request MAY require the collected information in the COAM Reply be sent in the integrity-protected mode using the a Message Authentication Code (MAC) Context Header, defined in [I-D.ietf-sfc-nsh-integrity]. If the NSH of the received SFC Echo Reply includes the MAC Context Header, the authentication of the packet MUST be verified before using any data. If the verification fails, the receiver MUST stop processing the SFF Information Record TLV and notify an operator. Specification of the notification mechanism is outside the scope of this document.

3.2. SFF Information Record TLV

For COAM Request, the SFF MUST include the Information of SFs into the SF Information Record TLV in the COAM Reply message. Every SFF sends back a single COAM Reply Message, including information on all the SFs attached to the SFF on the SFP as requested in the COAM Request message.

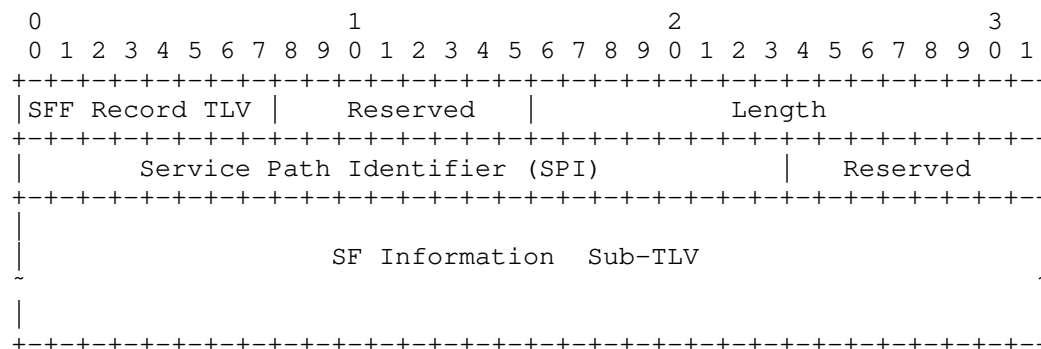


Figure 2: SFF Information Record TLV

SFF Information Record TLV is a variable-length TLV that includes the information of all SFFs mapped to the particular SFF instance for the specified SFP. Figure 2 presents the format of an SFC Echo Request/Reply TLV, where fields are defined as the following:

Reserved - one-octet-long field.

Service Path Identifier (SPI): The identifier of SFP to which all the SFs in this TLV belong.

SF Information Sub-TLV: The Sub-TLV is as defined in Figure 3.

3.3. SF Information Sub-TLV

Every SFF receiving COAM Request packet MUST include the SF characteristic data into the COAM Reply packet. The data format of an SF sub-TLV, included in a COAM Reply packet, is displayed in Figure 3.

After the COAM Request message traverses the SFP, all the information of the SFs on the SFP is collected from the TLVs included in COAM Reply messages.

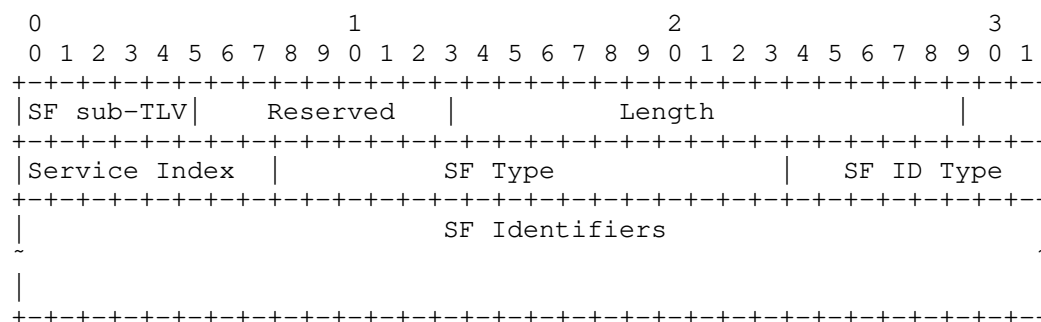


Figure 3: Service Function information sub-TLV

SF sub-TLV Type: Two octets long field. It indicates that the TLV is an SF TLV that contains the information of one SF.

Length: Two octets long field. The value of the field is the length of the data following the Length field counted in octets.

Service Index: Indicates the SF's position on the SFP.

SF Type: Two octets long field. It is defined in [I-D.ietf-bess-nsh-bgp-control-plane] and indicates the type of SF, e.g., Firewall, Deep Packet Inspection, WAN optimization controller, etc.

Reserved: For future use. MUST be zeroed on transmission and MUST be ignored on receipt.

SF ID Type: One octet-long field with values defined as Section 5.4.

SF Identifier: An identifier of the SF. The length of the SF Identifier depends on the type of the SF ID Type. For example, if the SF Identifier is its IPv4 address, the SF Identifier should be 32 bits. SF ID Type and SF Identifier may be a list, indicating the list of the SFs are which are included in a load balance group.

3.4. SF Information Sub-TLV Construction

Each SFF in the SFP MUST send one and only one COAM Reply corresponding to the COAM Request. If only one SF is attached to the SFF in such SFP, only one SF information sub-TLV is included in the COAM Reply. If several SFs attached to the SFF in the SFP, SF Information Sub-TLV MUST be constructed as described below in either Section 3.4.1 and Section 3.4.2.

3.4.1. Multiple SFs as hops of SFP

Multiple SFs attached to the same SFF are the hops of the SFP. The service indexes of these SFs are different. Service function types of these SFs could be different or be the same. Information about all SFs MAY be included in the COAM Reply message. Information about each SF MUST be listed as separate SF Information Sub-TLVs in the COAM Reply message.

An example of the COAM procedure for this case is shown in Figure 4. The Service Function Path(SPI=x) is SF1->SF2->SF4->SF3. The SF1, SF2 and SF3 are attached to SFF1, and SF4 is attached to SFF2. The COAM Request message is sent to the SFFs in the sequence of the SFP(SFF1->SFF2->SFF1). Every SFF(SFF1, SFF2) replies with the information of SFs belonging to the SFP. The SF information Sub-TLV in Figure 3 contains information for each SF (SF1, SF2, SF3, and SF4).

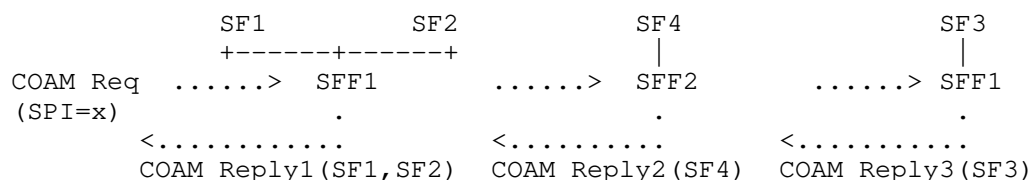


Figure 4: Example 1 for COAM Reply with multiple SFs

3.4.2. Multiple SFs for load balance

Multiple SFs may be attached to the same SFF to balance the load; in other words, that means that the particular traffic flow will traverse only one of these SFs. These SFs have the same Service Function Type and Service Index. For this case, the SF identifiers and SF ID Type of all these SFs will be listed in the SF Identifiers field and SF ID Type in a single SF information sub-TLV of COAM Reply message. The number of these SFs can be calculated according to SF ID Type and the value of the Length field of the sub-TLV.

An example of the COAM procedure for this case is shown in Figure 5. The Service Function Path (SPI=x) is SF1a/SF1b->SF2a/SF2b. The Service Functions SF1a and SF1b are attached to SFF1, which balances the load among them. The Service Functions SF2a and SF2b are attached to SFF2, which, in turn, balances its load between them. The COAM Request message is sent to the SFFs in the sequence of the SFP (i.e. SFF1->SFF2). Every SFF (SFF1, SFF2) replies with the information of SFs belonging to the SFP. The SF information Sub-TLV in Figure 3 contains information for all SFs at that hop.

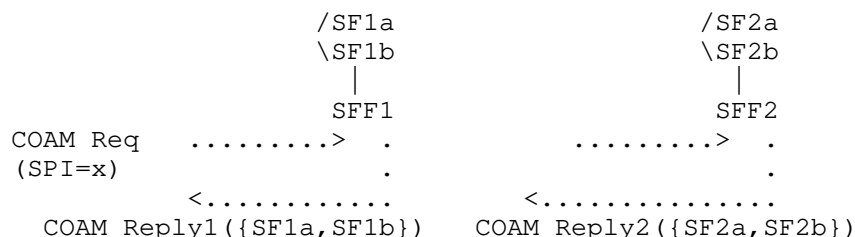


Figure 5: Example 2 for COAM Reply with multiple SFs

4. Security Considerations

Security considerations discussed in [RFC8300] and [I-D.ietf-sfc-multi-layer-oam] apply to this document.

Also, since Service Function sub-TLV discloses information about the SFP the spoofed COAM Request packet may be used to obtain network information, it is RECOMMENDED that implementations provide a means of checking the source addresses of COAM Request messages, specified in SFC Source TLV [I-D.ietf-sfc-multi-layer-oam], against an access list before accepting the message.

5. IANA Considerations

5.1. COAM Message Types

IANA is requested to assign values from its Message Types sub-registry in SFC Echo Request/Echo Reply Message Types registry as follows:

Value	Description	Reference
TBA1	SFP Consistency Echo Request	This document
TBA2	SFP Consistency Echo Reply	This document

Table 1: SFP Consistency Echo Request/Echo Reply Message Types

5.2. SFF Information Record TLV Type

IANA is requested to assign a new type value from SFC OAM TLV Type registry as follows:

Value	Description	Reference
TBA3	SFF Information Record Type	This document

Table 2: SFF-Information Record

5.3. SF Information Sub-TLV Type

IANA is requested to assign a new type value from SFC OAM TLV Type registry as follows:

Value	Description	Reference
TBA4	SF Information	This document

Table 3: SF-Information Sub-TLV Type

5.4. SF Identifier Types

IANA is requested to create in the registry SF Types the new sub-registry SF Identifier Types. All code points in the range 1 through 191 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC8126] and assign values as follows:

Value	Description	Reference
0	Reserved	This document
TBA6	IPv4	This document
TBA7	IPv6	This document
TBA8	MAC	This document
TBA8+1-191	Unassigned	IETF Review
192-251	Unassigned	First Come First Served
252-254	Unassigned	Private Use
255	Reserved	This document

Table 4: SF Identifier Type

6. Acknowledgements

The authors are thankful to John Drake for his review and the reference to the work on BGP Control Plane for NSH SFC. The authors express their appreciation to Joel M. Halpern for his suggestion about the load balancing scenario. The authors also thank Dirk von Hugo, for his useful comments.

7. References

7.1. Normative References

- [I-D.ietf-sfc-multi-layer-oam]
 Mirsky, G., Meng, W., Khasnabish, B., and C. Wang, "Active OAM for Service Function Chaining", Work in Progress, Internet-Draft, draft-ietf-sfc-multi-layer-oam-09, 11 February 2021, <<https://tools.ietf.org/html/draft-ietf-sfc-multi-layer-oam-09>>.

- [I-D.ietf-sfc-nsh-integrity]
Boucadair, M., Reddy, T., and D. Wing, "Integrity Protection for the Network Service Header (NSH) and Encryption of Sensitive Context Headers", Work in Progress, Internet-Draft, draft-ietf-sfc-nsh-integrity-05, 23 March 2021, <<https://tools.ietf.org/html/draft-ietf-sfc-nsh-integrity-05>>.
- [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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.

7.2. Informational References

- [I-D.ietf-bess-nsh-bgp-control-plane]
Farrel, A., Drake, J., Rosen, E., Uttaro, J., and L. Jalil, "BGP Control Plane for the Network Service Header in Service Function Chaining", Work in Progress, Internet-Draft, draft-ietf-bess-nsh-bgp-control-plane-18, 21 August 2020, <<https://tools.ietf.org/html/draft-ietf-bess-nsh-bgp-control-plane-18>>.
- [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>>.
- [RFC8924] Aldrin, S., Pignataro, C., Ed., Kumar, N., Ed., Krishnan, R., and A. Ghanwani, "Service Function Chaining (SFC) Operations, Administration, and Maintenance (OAM) Framework", RFC 8924, DOI 10.17487/RFC8924, October 2020, <<https://www.rfc-editor.org/info/rfc8924>>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com, gregory.mirsky@ztetx.com

Ting Ao
Individual contributor
No.889, BiBo Road
Shanghai
201203
China

Phone: +86 17721209283
Email: 18555817@qq.com

Zhonghua Chen
China Telecom
No.1835, South PuDong Road
Shanghai
201203
China

Phone: +86 18918588897
Email: 18918588897@189.cn

Kent Leung
Cisco System
170 West Tasman Drive
San Jose, CA 95134,
United States of America

Email: kleung@cisco.com

Gyan Mishra
Verizon Inc.

Email: gyan.s.mishra@verizon.com

SFC WG
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2019

T. Ao
ZTE Corporation
G. Mirsky
ZTE Corp.
Z. Chen
China Telecom
March 10, 2019

Controlled Return Path for Service Function Chain (SFC) OAM
draft-ao-sfc-oam-return-path-specified-03

Abstract

This document defines extensions to the Service Function Chain (SFC) Operation, Administration and Maintenance (OAM) that enable control of the Echo Reply return path by specifying it as Reverse Service Function Path. Enforcing the specific return path can be used to verify bidirectional connectivity of SFC and increase the robustness of SFC OAM.

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 11, 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	2
2. Conventions used in this document	3
2.1. Terminology	3
2.2. Requirements Language	3
3. Extension	3
4. SFC Reply Path TLV	4
5. Theory of Operation	5
5.1. Bi-directional SFC Case	5
6. Security Considerations	5
7. IANA Considerations	6
7.1. SFC Return Path Type	6
7.2. New Return Codes	6
8. References	6
8.1. Normative References	6
8.2. Informative References	7
Authors' Addresses	7

1. Introduction

While Service Function Chain (SFC) Echo Request, defined in [I-D.ietf-sfc-multi-layer-oam], always traverses the SFC it directed to, the corresponding Echo Reply is sent over IP network [I-D.ietf-sfc-multi-layer-oam]. There are scenarios when it is beneficial to direct the responder to use a path other than the IP network. This document defines extensions to the Service Function Chain (SFC) Operation, Administration and Maintenance (OAM) that enable control of the Echo Reply return path by specifying it as Reply Service Function Path. This document defines a new Type-Length-Value (TLV), Reply Service Function Path TLV, for Reply via Specified Path mode of SFC Echo Reply (Section 4).

The Reply Service Function Path TLV can provide an efficient mechanism to test SFCs, such as bidirectional and hybrid SFC, as these were defined in Section 2.2 [RFC7665]. For example, it allows an operator to test both directions of the bidirectional or hybrid SFP with a single SFC Echo Request/Echo Reply operation.

2. Conventions used in this document

2.1. Terminology

SF - Service Function

SFF - Service Function Forwarder

SFC - Service Function Chain, an ordered set of some abstract SFs.

SFP - Service Function Path

SPI - Service Path Index

OAM - Operation, Administration, and Maintenance

2.2. Requirements Language

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. Extension

Following reply modes had been defined in [I-D.ietf-sfc-multi-layer-oam]:

- o Do Not Reply
- o Reply via an IPv4/IPv6 UDP Packet
- o Reply via Application Level Control Channel
- o Reply via Specified Path

The Reply via Specified Path mode is intended to enforce the use of the particular return path specified in the included TLV. This mode may help to verify bidirectional continuity or increase the robustness of the monitoring of the SFC by selecting a more stable path. In the case of SFC, the sender of Echo Request instructs the destination SFF to send Echo Reply message along the SFP specified in the SFC Reply Path TLV as described in Section 4.

4. SFC Reply Path TLV

The SFC Reply Path TLV carries the information that sufficiently identifies the return SFP that the SFC Echo Reply message is expected to follow. The format of SFC Reply Path TLV is shown in Figure 1.

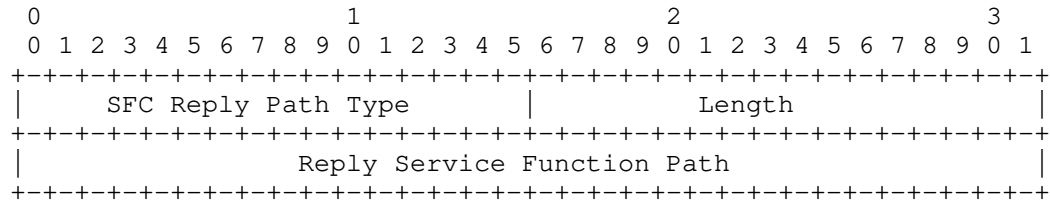


Figure 1: SFC Reply TLV Format

where:

- o Reply Path TLV Type: is two octets long, indicates the TLV that contains information about the SFC Reply path.
- o Length: is two octets long, MUST be equal to 4
- o Reply Service Function Path is used to describe the return path that an SFC Echo Reply is requested to follow.

The format of the Reply Service Function Path field displayed in Figure 2

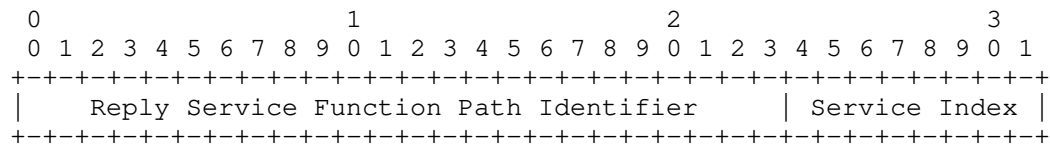


Figure 2: Reply Service Function Path Field Format

where:

- o Reply Service Function Path Identifier: SFP identifier for the path that the SFC Echo Reply message is requested to be sent over.
- o Service Index: used for forwarding in the reply SFP.

5. Theory of Operation

[RFC7110] defined mechanism to control return path for MPLS LSP Echo Reply. In case of SFC, the return path is a SFP along which SFC Echo Reply message MUST be transmitted. Hence, the SFC Reply Path TLV included in the SFC Echo Request message MUST sufficiently identify the SFP that the sender of the Echo Request message expects the receiver to use for the corresponding SFC Echo Reply.

When sending an Echo Request, the sender MUST set the value of Reply Mode field to "Reply via Specified Path", defined in [I-D.ietf-sfc-multi-layer-oam], and if the specified path is SFC path, the Request MUST include SFC Reply Path TLV. The SFC Reply Path TLV includes identifier of the reverse SFP and an appropriate Service Index.

Echo Reply is expected to be sent by the destination SFF of the SFP being tested or by the SFF at which SFC TTL expires as defined [RFC8300]. The processing described below equally applies in both cases and referred to as responding SFF.

If the Echo Request message with SFC Reply Path TLV, received by the responding SFF, has Reply Mode value of "Reply via Specified Path" but no SFC Reply Path TLV is present, then the responding SFF MUST send Echo Reply with Return Code set to "Reply Path TLV is missing" value (TBA2). If the responding SFF cannot find requested SFP it MUST send Echo Reply with Return Code set to "Reply SFP was not found" and include the SFC Reply Path TLV from the Echo Request message.

5.1. Bi-directional SFC Case

Ability to specify the return path to be used for Echo Reply is handy in bi-directional SFC. For bi-directional SFC, since the last SFF of the forward SFP may not co-locate with a classifier of the reverse SFP, it is assumed that the last SFF doesn't know the reply path of a SFC. So even for bi-directional SFC, a reverse SFP also need to be indicated in reply path TLV in echo request message.

6. Security Considerations

Security considerations discussed in [RFC8300] apply to this document.

In addition, the SFC Return Path extension, defined in this document, can be used for potential "proxying" attacks. For example, an echo request initiator may specify a return path that has a destination different from that of the initiator. But usually, such attacks will

not happen in an SFC domain where the initiators and receivers belong to the same domain, as specified in [RFC7665]. Even if the attack occurs, in order to prevent using the SFC Return Path extension for proxying any possible attacks, the return path SFP SHOULD have a path to reach the sender of the echo request, identified in SFC Source TLV [I-D.ietf-sfc-multi-layer-oam]. The receiver MAY drop the echo request when it cannot determine whether the return path SFP has the route to the initiator. That means, when sending echo request, the sender SHOULD choose a proper source address according to specified return path SFP to help the receiver to make the decision.

7. IANA Considerations

7.1. SFC Return Path Type

IANA is requested to assign from its SFC Echo Request/Echo Reply TLV registry new type as follows:

Value	Description	Reference
TBA1	SFC Reply Path Type	This document

Table 1: SFC Return Path Type

7.2. New Return Codes

IANA is requested to assign new return codes from the SFC Echo Request/Echo Reply Return Codes registry as following:

Value	Description	Reference
TBA2	Reply Path TLV is missing	This document
TBA3	Reply SFP was not found	This document

Table 2: SFC Echo Reply Return Codes

8. References

8.1. Normative References

[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-02 (work in progress), March 2019.

- [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

- [RFC7110] Chen, M., Cao, W., Ning, S., Jounay, F., and S. Delord, "Return Path Specified Label Switched Path (LSP) Ping", RFC 7110, DOI 10.17487/RFC7110, January 2014, <<https://www.rfc-editor.org/info/rfc7110>>.
- [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

Ting Ao
ZTE Corporation
No.889, BiBo Road
Shanghai 201203
China

Phone: +86 21 68897642
Email: ao.ting@zte.com.cn

Greg Mirsky
ZTE Corp.
1900 McCarthy Blvd. #205
Milpitas, CA 95035
USA

Email: gregimirsky@gmail.com

Zhonghua Chen
China Telecom
No.1835, South PuDong Road
Shanghai 201203
China

Phone: +86 18918588897
Email: 18918588897@189.cn

SFC WG
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2019

T. Ao
R. Chen
W. Wei
ZTE Corporation
March 10, 2019

YANG data model for SFC
draft-ao-sfc-yang-00

Abstract

This document is to define the YANG data model for SFC configuration.

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 11, 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	2
2. Design tree for SFC YANG data model	2
3. YANG data model for SFC configuration	3
4. Security Considerations	8
5. IANA Considerations	8
6. References	8
6.1. Normative References	8
6.2. Information References	8
Authors' Addresses	9

1. Introduction

YANG[RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document defines a YANG data model for the configuration of SFC which data plane has been defined in [RFC8300].

2. Design tree for SFC YANG data model

```

module: ietf-sfc
+--rw sfc-config
|
|   +--rw sfc-enable?    boolean
|   +--rw sfc-domain* [sfc-domain-id]
|   |   +--rw sfc-domain-id    uint32
|   |   +--rw ipv4-prefix?     inet:ipv4-prefix
|   |   +--rw ipv6-prefix?     inet:ipv6-prefix
|   |   +--rw sfc-sfp* [sfpid si]
|   |   |   +--rw sfpid                uint32
|   |   |   +--rw si                    uint16
|   |   |   +--rw metric?              uint16
|   |   +--rw (nexthop-trans-type)?
|   |   |   +--:(ipv4-nexthop)
|   |   |   |   +--rw nh-node-type?    sfp-nexthop-type
|   |   |   |   +--rw remote-ipv4?     inet:ipv4-address
|   |   |   +--:(ipv6-nexthop)
|   |   |   |   +--rw nh-node-type?    sfp-nexthop-type
|   |   |   |   +--rw remote-ipv6?     inet:ipv6-address
|   |   |   +--:(mac-nexthops)
|   |   |   |   +--rw nh-node-type?    sfp-nexthop-type
|   |   |   |   +--rw remote-mac?      yang:mac-address
|   |   +--:(vxlan-gpe-nexthop)
|   |   |   +--rw nh-node-type?    sfp-nexthop-type
|   |   |   +--rw remote-ip?        inet:ipv4-address
|   |   |   +--rw source-ip?        inet:ipv4-address
|   |   |   +--rw destination-ip?    inet:ipv4-address

```

```

|          |          +---rw vni                uint32
|          |          +---rw last-sff           boolean
+---ro sfc-state
  +---ro sfc-enable?          boolean
  +---ro sfc-domain * [sfc-domain-id]
    +---ro sfc-domain-id      uint32
    +---ro ipv4-prefix?       inet:ipv4-prefix
    +---ro ipv6-prefix?       inet:ipv6-prefix
    +---ro sfc-sfp-state
      +---ro sfc-sfp*[sfpid si]
        +---ro sfpid?         uint32
        +---ro si?            uint16
        +---ro metric?        uint16
        +---ro nexthop-trans-type? enumeration
          +---:(ipv4-nexthop)
            +---ro nx-node-type? node-type
            +---ro remote-ipv4?  inet:ipv4-address
          +---:(ipv6-nexthop)
            +---ro nx-node-type? node-type
            +---ro remote-ipv6?  inet:ipv6-address
          +---:(mac-nexthop)
            +---ro nx-node-type? node-type
            +---ro remote-mac?   yang:mac-address
          +---:(vxlan-gpe-nexthop)
            +---ro nx-node-type? node-type
            +---ro remote-ip?    inet:ipv4-address
            +---ro source-ip?    inet:ipv4-address
            +---ro destination-ip? inet:ipv4-address
            +---ro vni?          uint32
        +---ro last-sff?       boolean

```

3. YANG data model for SFC configuration

This container defines a YANG model to configurate of SFC. The SF Type listed in this YANG model is referenced by [I-D.ietf-sfc-use-case-mobility] and [I-D.ietf-sfc-dc-use-cases].

```

<CODEBEGINS> file "ietf-sfc@2019-03-10.yang"
module ietf-sfc {
  namespace "urn:ietf:params:xml:ns:yang:ietf-sfc";
  prefix "sfc";
  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-yang-types {
    prefix "yang";
  }

```

```
organization "IETF SFC Working Group";

contact
"WG Web:    <http://tools.ietf.org/wg/sfc/>
WG List:    <mailto:sfc@ietf.org>
WG Chair:Jim Guichard
            <mailto:james.n.guichard@huawei.com>
WG Chair:Joel M. Halpern
            <mailto:jmh@joelhalpern.com>

Editor: Ting Ao
            <mailto:ao.ting@zte.com.cn>
Editor: Ran Chen
            <mailto:chen.ran@zte.com.cn>
Editor: Wei Wei
            <mailto:wei.wei@zte.com.cn>
            ";

description
"The YANG module defines a generic configuration
model for SFC.";

revision 2019-03-07{
  description
    "Initial revision.";
  reference "RFC XXXX: YANG Data Model for SFC Protocol.";
}
/*Typedefs*/
typedef sfp-nextthop-type {

  type enumeration {
    enum sff {
      value 1 ;
    }

    enum sf-firewall {
      value 2 ;
    }

    enum sf-dpi {
      value 3 ;
    }

    enum sf-ids {
      value 4 ;
    }

    enum sf-edgefw {
      value 5 ;
    }

    enum sf-segfw {
      value 6 ;
    }
  }
}
```

```
        enum sf-appfw {
value 7 ;
        }
        enum sf-adc {
value 8 ;
        }
        enum sf-woc {
value 9 ;
        }
        enum sf-mon {
value 10 ;
        }
        enum sf-sgw {
value 11 ;
        }
        enum sf-pgw {
value 12 ;
        }
        enum sf-hss {
value 13 ;
        }
        enum sf-mme {
value 14 ;
        }
        enum sf-pcrf {
value 15 ;
        }
        enum sf-pcef {
value 16 ;
        }
        enum sf-tdf {
value 17 ;
        }
        enum sf-tssf {
value 18 ;
        }
        enum sf-tds {
value 19 ;
        }
        enum sf-pep {
value 20 ;
        }
        enum sf-ims {
value 21 ;
        }
        enum sf-li {
value 22 ;
        }
```

```
        enum sf-proxy {
            value 23;
        }
        description "The nexthop node type.";
    }

    container sfc-config {
        leaf sfc-enable {
            type boolean;
            default false ;
            description "Enable SFC." ;
        }
        list sfc-domain {
            key "sfc-domain-id";
            leaf sfc-domain-id {
                type uint32;
                description "The identifier of the sfc domain." ;
            }
            leaf ipv4-prefix {
                type inet:ipv4-prefix ;
                description "The IPv4 address of the sff.";
            }
            leaf ipv6-prefix {
                type inet:ipv6-prefix ;
                description "The IPv6 address of the sff.";
            }
        }
        list sfc-sfp {
            key "sfpid si";
            leaf sfpid {
                type uint32;
                description "The identifier of the SFP";
            }
            leaf si {
                type uint16;
                description "Service index.";
            }
            leaf metric {
                type uint16;
                description "Forwarding metric.";
            }
            choice nexthop-trans-type {
                case ipv4-nexthop {
                    leaf nh-node-type {
                        type sfp-nexthop-type ;
                        description "Nexthop node type.";
                    }
                    leaf remote-ipv4 {
```

```
        type inet:ipv4-address ;
        description "Remote IPv4 address.";
    }
    description "The configuration for SFP nexthop which enc
apsulation type is ethernet&ipv4.";
}

case ipv6-nexthop {
    leaf nh-node-type {
        type sfp-nexthop-type ;
        description "Nexthop node type." ;
    }
    leaf remote-ipv6 {
        type inet:ipv6-address ;
        description "Remote IPv6 address.";
    }
    description "The configuration for SFP nexthop which enc
apsulation type is ethernet&ipv6.";
}

case mac-nexthops {
    leaf nh-node-type {
        type sfp-nexthop-type ;
        description "Nexthop node type.";
    }
    leaf remote-mac {
        type yang:mac-address ;
        description "MAC address.";
    }
    description "The configuration for SFP nexthop which spe
cifies the MAC address." ;
}

case vxlan-gpe-nexthop {
    leaf nh-node-type {
        type sfp-nexthop-type ;
        description "Nexthop node type.";
    }
    leaf remote-ip {
        type inet:ip-address ;
        description "Remote IP address.";
    }
    leaf source-ip {
        description "The source IP address.";
        type inet:ipv4-address ;
    }
    leaf destination-ip {
        description "The destination address.";
        type inet:ipv4-address ;
    }
    leaf vni {
```

```

        type uint32;
        mandatory true;
        description "VNI value of the tunnel.";
    }
    description "The configuration for SFP nexthop is vxlan-
gpe." ;
}
description "The configuration for SFP nexthop." ;
}

leaf last-sff {
    type boolean ;
    default false ;
    description "This is the SFP terminal.";
}
}
}

<CODE ENDS>
```

4. Security Considerations

TBD.

5. IANA Considerations

TBD.

6. References

6.1. 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>>.
- [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>>.

6.2. Information References

- [I-D.ietf-bess-nsh-bgp-control-plane]
Farrel, A., Drake, J., Rosen, E., Uttaro, J., and L. Jalil, "BGP Control Plane for NSH SFC", draft-ietf-bess-nsh-bgp-control-plane-09 (work in progress), March 2019.

[I-D.ietf-sfc-dc-use-cases]

Kumar, S., Tufail, M., Majee, S., Captari, C., and S. Homma, "Service Function Chaining Use Cases In Data Centers", draft-ietf-sfc-dc-use-cases-06 (work in progress), February 2017.

[I-D.ietf-sfc-use-case-mobility]

Haeffner, W., Napper, J., Stiernerling, M., Lopez, D., and J. Uttaro, "Service Function Chaining Use Cases in Mobile Networks", draft-ietf-sfc-use-case-mobility-09 (work in progress), January 2019.

Authors' Addresses

Ting Ao
ZTE Corporation
No.889, BiBo Road
Shanghai 201203
China

Phone: +86 21 68897642
Email: ao.ting@zte.com.cn

Ran Chen
ZTE Corporation

Email: ran.chen@zte.com.cn

Wei Wei
ZTE Corporation

Email: wei.wei@zte.com.cn

SFC WG
Internet-Draft
Intended status: Standards Track
Expires: June 30, 2021

R. Chen
ZTE Corporation
X. Liu
Volta Networks
H. Chen
China Telecom
W. Wei
ZTE Corporation
T. Ao
individual
December 27, 2020

YANG data model for SFF
draft-ao-sfc-yang-03

Abstract

This document is to define the YANG data model for SFF configuration.

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 June 30, 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	2
2. Design tree for SFF YANG data model	2
3. YANG data model for SFF configuration	3
4. Security Considerations	10
5. IANA Considerations	10
6. References	10
6.1. Normative References	10
6.2. Information References	11
Authors' Addresses	11

1. Introduction

YANG[RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document defines a YANG data model for the configuration of SFF which data plane has been defined in [RFC8300].

2. Design tree for SFF YANG data model

```

module: ietf-sff
  +--rw sff
    +--rw sff* [sff-name]
      +--rw sfc-enable?   boolean
      +--rw sff-name      sff-name
      +--rw ip-mgmt-address?  inet:ip-address
      +--rw sff-locator* [name]
        +--rw sff-locator-name      sff-locator-name
        +--rw (sff-transport-locator)?
          +--:(ipv4)
            | +--rw remote-ipv4?      inet:ipv4-address
            +--:(ipv6)
              | +--rw remote-ipv6?      inet:ipv6-address
              +--:(mac)
                +--rw remote-mac?      yang:mac-address
          +--:(vxlan-gpe)
            | +--rw source-ip?      inet:ipv4-address
            | +--rw destination-ip?  inet:ipv4-address
            | +--rw vni              uint32
            +--:(mpls)
              +--rw mpls-label?      uint32

```

```

|   +--rw transport?                               identityref
+--rw connected-sf*[name]
|   +--rw sf-name                                   sf-name
|   +--rw sf-type                                   sf-type
|   +--rw sff-sf-locator*
|   |   +--rw sf-locator-name                       sf-locator-name
|   |   +--rw sff-locator-name                     sff-locator-name
|   +--rw sff-interfaces* [sff-interface]
|   |   +--rw sff-interface                         string
+--rw connected-sff* [name]
|   +--rw sff-name                                   sff-name
|   +--rw sff-sff-locator
|   |   +--rw (locator-type)
|   |   |   +--:(ipv4)
|   |   |   |   +--rw remote-ipv4?                 inet:ipv4-address
|   |   |   +--:(ipv6)
|   |   |   |   +--rw remote-ipv6?                 inet:ipv6-address
|   |   |   +--:(mac)
|   |   |   |   +--rw remote-mac?                   yang:mac-address
|   |   |   +--:(vxlan-gpe)
|   |   |   |   +--rw source-ip?                   inet:ipv4-address
|   |   |   |   +--rw destination-ip?              inet:ipv4-address
|   |   |   |   +--rw vni                           uint32
|   |   |   +--:(mpls)
|   |   |   |   +--rw mpls-label?                   uint32
|   |   +--rw transport?                           identityref
|   |   |   +--rw sff-interfaces* [sff-interface]
|   |   |   |   +--rw sff-interface                 string

```

3. YANG data model for SFF configuration

This container defines a YANG model to configurate of SFF. The SF Type listed in this YANG model is referenced by [I-D.ietf-sfc-use-case-mobility] and [I-D.ietf-sfc-dc-use-cases].

```
<CODEBEGINS> file "ietf-sff@2020-12-10.yang"
```

```

module ietf-sff {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-sff";
  prefix "sff";

  import ietf-inet-types {
    prefix "inet";
    reference "RFC6991: Common YANG Data Types";
  }

```

```
    }
import ietf-routing-types {
  prefix "rt-types";
  reference "RFC8294:Common YANG Data Types for the Routing Area";
}
import ietf-yang-types {
  prefix "yang";
  reference "RFC6991: Common YANG Data Types";
}
import ietf-interfaces {
  prefix "if";
  reference "RFC8343: A YANG Data Model for Interface Management";
}
organization "IETF SFC Working Group";

contact
"WG Web:  <http://tools.ietf.org/wg/sfc/>
WG List:  <mailto:sfc@ietf.org>
  WG Chair:Jim Guichard
    <mailto:james.n.guichard@huawei.com>
WG Chair:Joel M. Halpern
    <mailto:jmh@joelhalpern.com>
Editor: Ran Chen
    <mailto:150387479@qq.com>
  Editor: Xufeng Liu
    <mailto:xufeng.liu.ietf@gmail.com>
  Editor: Huanan Chen
    <mailto:chenhua6@chinatelecom.com>
Editor: Wei Wei
    <mailto:wei.wei26@zte.com.cn>
  Editor: Ting Ao
    <mailto:ao.ting@zte.com.cn>
";

description
"The YANG module defines a generic configuration model for SFF.
  Copyright (c) 2019 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).
  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.";

revision "2020-01-20"{
```

```
        description "Initial revision.";
        reference "RFC XXXX: YANG Data Model for SFC Protocol.";
    }

    revision "2020-02-06"{
        description "01 revision.";
        reference "RFC XXXX: YANG Data Model for SFC Protocol.";
    }

    revision "2020-12-10"{
        description "02 revision.";
        reference "RFC XXXX: YANG Data Model for SFC Protocol.";
    }

/* Typedef */
    typedef sff-name {
        type string;
        description "Service Function Forwarder Name";
    }

    typedef sff-locator-name {
        type string;
        description "Service Function Forwarder data-plane-locator name";
    }

    typedef sf-locator-name {
        type string;
        description
            "A unique name for SF data-plane-locator";
    }

    typedef sf-name {
        type string;
        description "Service Function Name";
    }

    typedef sf-type {
        type string;
        description "Service Function type Name";
    }

/* Identities */
    identity transport-type{
        description
            "Base identity from which specific transport types are derived.";
    }

    identity locator-transport-type {
        base "transport-type";
    }
```

```

        description
            "This identity is used as a base for all transport types";
    }
    identity ipv4 {
        base locator-transport-type;
        description
            "This identity represents IPv4 transport type.";
    }
    identity ipv6 {
        base locator-transport-type;
        description
            "This identity represents IPv6 transport-type.";
    }
    identity mac {
        base locator-transport-type;
        description
            "This identity represents sfp mac transport-type .";
    }
    identity vxlan-gpe {
        base locator-transport-type;
        description
            "This identity represents vxlan-gpe transport-type.";
    }
    identity mpls {
        base locator-transport-type;
        description
            "This identity represents mpls transport-type.";
    }
}

/*grouping*/
grouping locator-transport-type {
    description
        "This group presents configuration for the overlay data plane locator. This
        could be VXLAN-GRE, MPLS, IP, MAC, etc";
    choice transport-type {
        mandatory true;

        case ipv4 {
            description
                "The configuration for overlay data plane which encapsulation
                type is ethernet&ipv4.";
            leaf ipv4 {
                type inet:ipv4-address ;
                description "Data-plane IPv4 address.";
            }
            description "The configuration for overlay data plane which encapsulation
            type is ethernet&ipv4.";
        }

        case ipv6{

```

```

        description
            "The configuration for overlay data plane which encapsulation
type is ethernet&ipv6.";
        leaf ipv6 {
            type inet:ipv6-address ;
            description "IPv6 address.";
        }
    }

    case mac{
        description
            "The configuration for overlay data plane which encapsulation
type is mac.";
        leaf mac {
            type yang:mac-address ;
            description "MAC address.";
        }
    }

    case vxlan-gpe-nextthop {
        description
            "The configuration for overlay data plane which encapsulation
n type is vxlan-gpe.";
        leaf source-ip {
            description "The source IP address.";
            type inet:ipv4-address ;
        }
        leaf destination-ip {
            description "The destination address.";
            type inet:ipv4-address ;
        }
        leaf vni {
            type uint32;
            mandatory true;
            description "VNI value of the tunnel.";
        }
    }

    case mpls-nexthop{
        description "The configuration for overlay data plane which encapsula
tion type is mpls.";
        uses rt-types:mpls-label-stack;
        description "The collection of all possible data-plane
locators.";
    }
    leaf transport {
        type identityref {
            base locator-transport-type;
        }
        description
            "The encapsulation used to carry NSH packets";
    }

```

```
    }

    container sff {
      description
        "A service function forwarder is responsible for delivering traffic received from the SFC network forwarder to one or more connected service functions via information carried in the SFC encapsulation."
      ";
      leaf sfc-enable {
        type boolean;
        default false ;
        description "Enable SFC." ;
      }

      list sff {
        key "sff-name";
        description
          "a list of all SFF configurations in the domain.";
        leaf sff-name {
          type sff-name;
          description
            "The unique name of this service function forwarder.";
        }

        leaf ip-mgmt-address {
          type inet:ip-address;
          description
            "The IP and port used to configure this service-function-forwarder";
        }

        list sff-locator{
          key "sff-locator-name";
          leaf sff-locator-name {
            type sff-locator-name ;
            description "A list of all data-plane-locators of this SFF." ;
          }
          container sff-transport-locator{
            description
              "The overlay data plane locator used by this SFF. This could be VXLAN-GRE,MPLS,MAC,etc";
            uses locator-transport-type;
          }
          description
            "The list of sff data plane locator related informations.";
        }

        list connected-sf {
          key "sf-name";
          leaf sf-name {
            type sf-name;
          }
        }
      }
    }
  }
}
```



```

        description
            "The name of the service function.";
        }
        leaf sf-type {
            type sf-type;
        }
        description
            "Service Function type names such as firewall, dpi,etc";
    }
    container sff-sf-locator {
        description
            "SFF and SF data plane locators to use when sending packets from t
his SFF to the associated SF";
        leaf sf-locator-name {
            type sf-locator-name;
            description
                "The SF data plane locator to use when sending packets to the as
sociated service function";
        }
        leaf sff-locator-name {
            type sff-locator-name;
            description
                "The SFF data plane locator to use when sending
packets to the associated service function.";
        }
    }
    list sff-interfaces {
        key "sff-interface";
        leaf sff-interface {
            type string;
            description
                "An individual interface on the SFF connected to the SF";
        }
        description
            "A list of interfaces on the SFF which are connected to the SF";
    }
    description
        "A list of all Service Functions attached to this SFF.";
}

list connected-sff{
key "sff-name";
leaf sff-name {
    type sff-name;
    description
        "The name of the SFF connected to this SFF";
}
container sff-sff-locator {
    description
        "The SFF uses this data plane locator when sending
packets to the associated SFF";
}

```

```
        uses locator-transport-type;
    }
    list sff-interfaces {
        key "sff-interface";
        leaf sff-interface {
            type string;
            description
                "An individual SFF interface connected to this SFF";
        }
        description
            "A list of SFF interfaces connected to this SFF";
    }
    description
        "A list of all Service Function Forwarders connected to
        this SFF";
    }
    }
}

<CODE ENDS>
```

4. Security Considerations

TBD.

5. IANA Considerations

TBD.

6. References

6.1. 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>>.
- [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>>.

6.2. Information References

[I-D.ietf-sfc-dc-use-cases]

Kumar, S., Tufail, M., Majee, S., Captari, C., and S. Homma, "Service Function Chaining Use Cases In Data Centers", draft-ietf-sfc-dc-use-cases-06 (work in progress), February 2017.

[I-D.ietf-sfc-use-case-mobility]

Haeffner, W., Napper, J., Stiernerling, M., Lopez, D., and J. Uttaro, "Service Function Chaining Use Cases in Mobile Networks", draft-ietf-sfc-use-case-mobility-09 (work in progress), January 2019.

Authors' Addresses

Ran Chen
ZTE Corporation

Email: chen.ran@zte.com.cn

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Huanan Chen
China Telecom

Email: chenhua6@chinatelecom.com

Wei Wei
ZTE Corporation

Email: wei.wei26@zte.com.cn

Ting Ao
individual

Email: 18555817@qq.com

SFC
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2019

F. Brockners
S. Bhandari
V. Govindan
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JPMC
T. Mizrahi
Huawei Network.IO Innovation Lab
D. Mozes

P. Lapukhov
Facebook
R. Chang
Barefoot Networks
March 11, 2019

Network Service Header (NSH) Encapsulation for In-situ OAM (IOAM) Data
draft-ietf-sfc-ioam-nsh-01

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) 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 September 12, 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 (<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), as defined in [I-D.ietf-ippm-ioam-data], 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 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

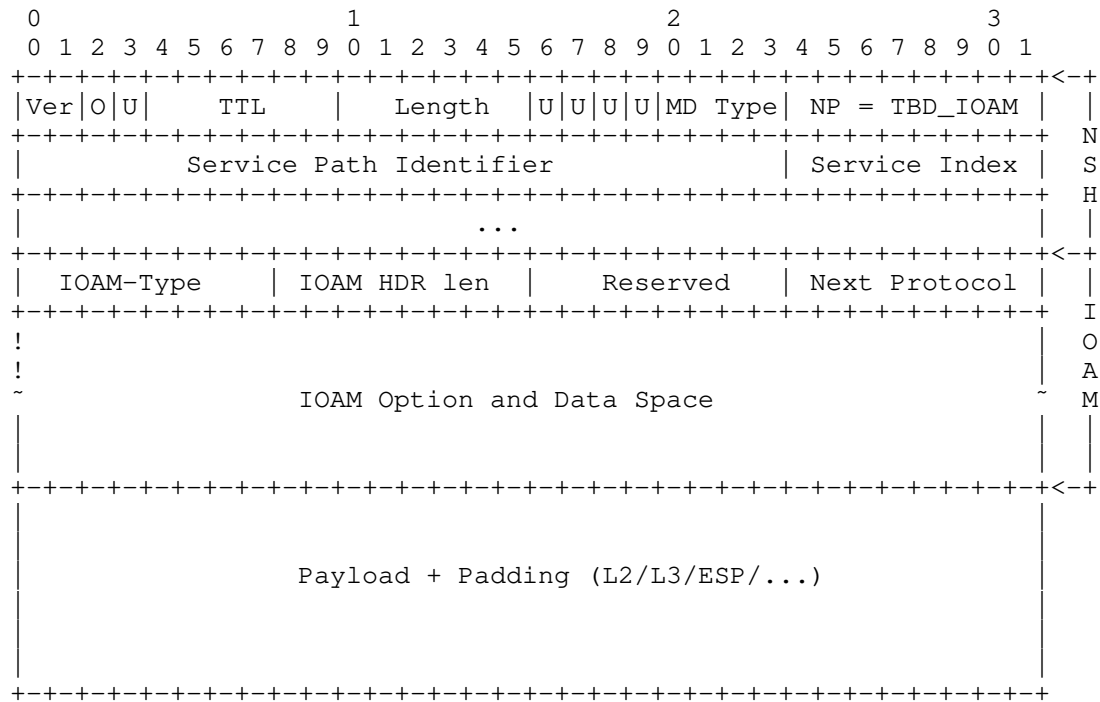
OAM: Operations, Administration, and Maintenance

SFC: Service Function Chaining

TLV: Type, Length, Value

3. IOAM data fields encapsulation in NSH

The NSH is defined in [RFC8300]. IOAM data fields are carried in NSH using a next protocol header which follows the NSH MD context headers. 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.



The NSH header and fields are defined in [RFC8300]. 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 Section 7.2 of [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 protocol. The semantics of this field are identical to the Next Protocol field in [RFC8300].

IOAM Option and Data Space: IOAM option header and data is present as specified by the IOAM-Type field, and is defined in Section 4 of [I-D.ietf-ippm-ioam-data].

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-04 (work in progress), October 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>>.
- [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/>>.
- [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

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

Email: shwethab@cisco.com

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
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
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

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

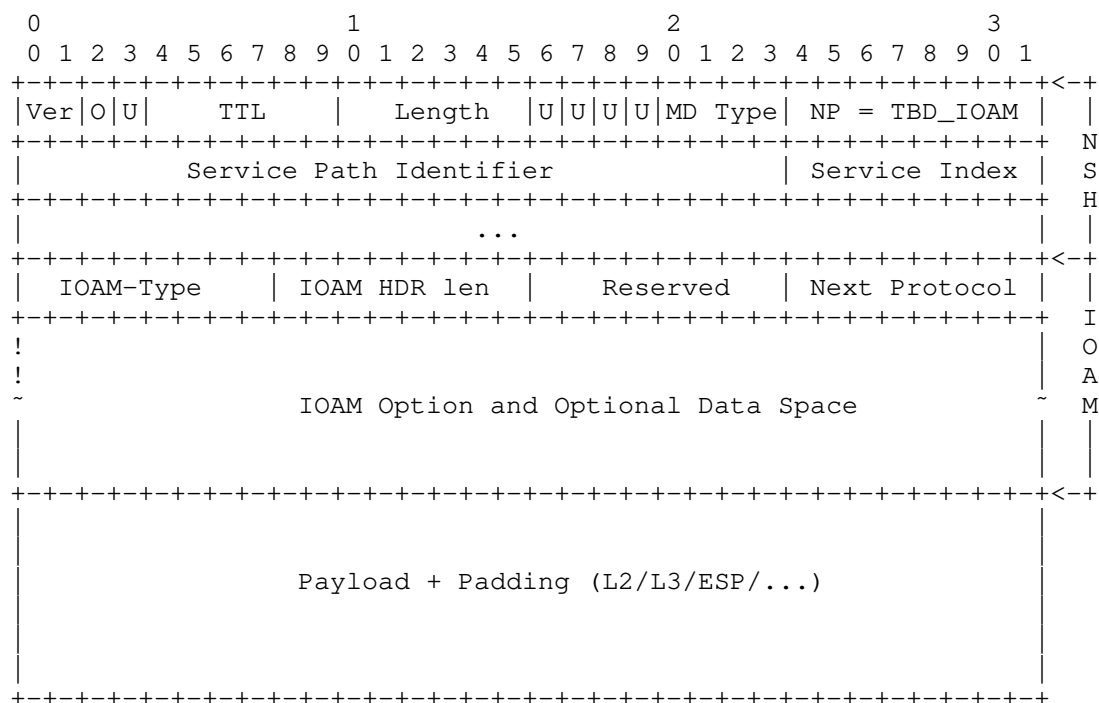
OAM: Operations, Administration, and Maintenance

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

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

SFC WG
Internet-Draft
Updates: 8300 (if approved)
Intended status: Standards Track
Expires: September 9, 2019

G. Mirsky
ZTE Corp.
W. Meng
ZTE Corporation
B. Khasnabish
Individual contributor
C. Wang
March 8, 2019

Active OAM for Service Function Chains in Networks
draft-ietf-sfc-multi-layer-oam-02

Abstract

A set of requirements for active Operation, Administration and Maintenance (OAM) of Service Function Chains (SFCs) in networks is presented. Based on these requirements an encapsulation of active OAM message in SFC and a mechanism to detect and localize defects described. Also, this document updates RFC 8300 in the definition of O (OAM) bit in the Network Service Header (NSH) and defines how the active OAM message identified in SFC 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 September 9, 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	2
2. Conventions	3
2.1. Requirements Language	3
2.2. Terminology	3
3. Requirements for Active OAM in SFC Network	4
4. Active OAM Identification in SFC NSH	5
5. Echo Request/Echo Reply for SFC in Networks	7
5.1. Return Codes	8
5.2. SFC Echo Request Transmission	9
5.3. SFC Echo Request Reception	9
5.4. SFC Echo Reply Transmission	10
5.5. SFC Echo Reply Reception	10
6. Security Considerations	11
7. Acknowledgments	12
8. IANA Considerations	12
8.1. SFC Active OAM Protocol	12
8.2. SFC Active OAM Message Type	12
8.3. SFC Echo Request/Echo Reply Parameters	13
8.4. SFC Echo Request/Echo Reply Message Types	13
8.5. SFC Echo Reply Modes	13
8.6. SFC Echo Return Codes	14
8.7. SFC TLV Type	14
8.8. SFC OAM UDP Port	15
9. References	16
9.1. Normative References	16
9.2. Informative References	16
Authors' Addresses	17

1. Introduction

[RFC7665] defines components necessary to implement Service Function Chain (SFC). These include a classifier which performs the classification of incoming packets. A Service Function Forwarder (SFF) is responsible for forwarding traffic to one or more connected Service Functions (SFs) according to the information carried in the SFC encapsulation. SFF also handles traffic coming back from the SF and transports the data packets to the next SFF. And the SFF serves as termination element of the Service Function Path (SFP). SF is responsible for the specific treatment of received packets.

Resulting from that SFC is constructed by a number of these components, there are different views from different levels of the SFC. One is the SFC, entirely abstract entity, which defines an ordered set of SFs that must be applied to packets selected as a result of classification. But SFC doesn't specify the exact mapping between SFFs and SFs. Thus there exists another semi-abstract entity referred to as SFP. SFP is the instantiation of the SFC in the network and provides a level of indirection between the entirely abstract SFC and a fully specified ordered list of SFFs and SFs identities that the packet will visit when it traverses the SFC. The latter entity is being referred to as Rendered Service Path (RSP). The main difference between SFP and RSP is that in the former the authority to select the SFF/SF has been delegated to the network.

This document defines how active Operation, Administration and Maintenance (OAM), per [RFC7799] definition of active OAM, identified in Network Service Header (NSH) SFC, lists requirements to improve the troubleshooting efficiency, and defines SFC Echo request and Echo reply that enables on-demand Continuity Check, Connectivity Verification among other operations over SFC in networks. Also, this document updates Section 2.2 of [RFC8300] in part of the definition of O bit in the (NSH).

2. Conventions

2.1. Requirements Language

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.

2.2. Terminology

Unless explicitly specified in this document, active OAM in SFC and SFC OAM are being used interchangeably.

e2e: End-to-End

FM: Fault Management

NSH: Network Service Header

OAM: Operations, Administration, and Maintenance

PRNG: Pseudorandom number generator

RDI: Remote Defect Indication

RSP: Rendered Service Path

SMI Structure of Management Information

SF: Service Function

SFC: Service Function Chain

SFF: Service Function Forwarder

SFP: Service Function Path

3. Requirements for Active OAM in SFC Network

To perform the OAM task of fault management (FM) in an SFC, that includes failure detection, defect characterization and localization, this document defines the set of requirements for active OAM mechanisms to be used on an SFC.

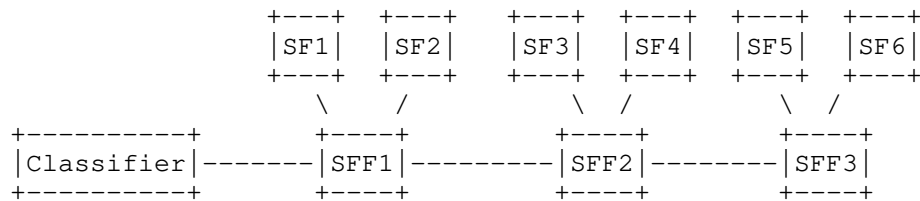


Figure 1: SFC reference model

In the example presented in Figure 1, the service SFP1 may be realized through two independent RSPs, RSP1(SF1--SF3--SF5) and RSP2(SF2--SF4--SF6). To perform end-to-end (e2e) FM SFC OAM:

REQ#1: Packets of active OAM in SFC SHOULD be fate sharing with data traffic, i.e., in-band with the monitored traffic follow the same RSP, in the forward direction from ingress toward egress endpoint(s) of the OAM test.

REQ#2: SFC OAM MUST support pro-active monitoring of any element in the SFC availability.

The egress, SFF3 in the example in Figure 1, is the entity that detects the failure of the SFC. It must be able to signal the new defect state to the ingress SFF1. Hence the following requirement:

REQ#3: SFC OAM MUST support Remote Defect Indication (RDI) notification by the egress to the ingress.

REQ#4: SFC OAM MUST support connectivity verification. Definition of the misconnection defect, entry and exit criteria are outside the scope of this document.

Once the SFF1 detects the defect objective of OAM switches from failure detection to defect characterization and localization.

REQ#5: SFC OAM MUST support fault localization of Loss of Continuity check in the SFC.

REQ#6: SFC OAM MUST support tracing an SFP to realize the RSP.

It is practical, as presented in Figure 1, that several SFs share the same SFF. In such case, SFP1 may be realized over two RSPs, RSP1(SF1--SF3--SF5) and RSP2(SF2--SF4--SF6).

REQ#7: SFC OAM MUST have the ability to discover and exercise all available RSPs in the transport network.

In the process of localizing the SFC failure, separating SFC OAM layers is an efficient approach. To achieve that continuity among SFFs that are part of the same SFP should be verified. Once SFFs reachability along the particular SFP has been confirmed task of defect localization may focus on SF reachability verification. Because reachability of SFFs has already verified, SFF local to the SF may be used as a source of the test packets.

REQ#8: SFC OAM MUST be able to trigger on-demand FM with responses being directed towards initiator of such proxy request.

4. Active OAM Identification in SFC NSH

The interpretation of O bit flag in the NSH header is defined in [RFC8300] as:

O bit: Setting this bit indicates an OAM packet.

This document updates the definition of O bit as follows:

O bit: Setting this bit indicates an OAM command and/or data in the NSH Context Header or packet payload

Active SFC OAM defined as a combination of OAM commands and/or data included in a message that immediately follows the NSH. To identify the active OAM message the value on the Next Protocol field MUST be

set to Active SFC OAM (TBA1) according to Section 8.1. 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 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.

Several active OAM protocols will be needed to address all the requirements listed in Section 3. Destination UDP port number may identify protocols if IP/UDP encapsulation used. But extra IP/UDP headers, especially in the case of IPv6, add noticeable overhead. This document defines Active OAM Header Figure 2 to demultiplex active OAM protocols on an SFC.

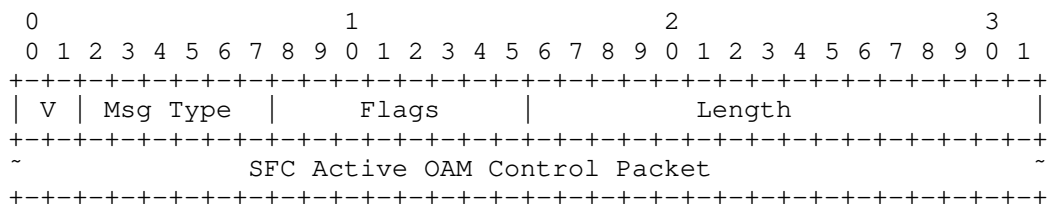


Figure 2: SFC Active OAM Header

V - two bits long field indicates the current version of the SFC active OAM header. The current value is 0.

Msg Type - six bits long field identifies OAM protocol, e.g., Echo Request/Reply or Bidirectional Forwarding Detection.

Flags - eight bits long field carries bit flags that define optional capability and thus processing of the SFC active OAM control packet, e.g., optional timestamping.

Length - two octets long field that is the length of the SFC active OAM control packet in octets.

5. Echo Request/Echo Reply for SFC in Networks

Echo Request/Reply is a well-known active OAM mechanism that is extensively used to detect inconsistencies between a state in control and the data planes, localize defects in the data plane. The format of the Echo request/Echo reply control packet is to support ping and traceroute functionality in SFC in networks Figure 3 resembles the format of MPLS LSP Ping [RFC8029] with some exceptions.

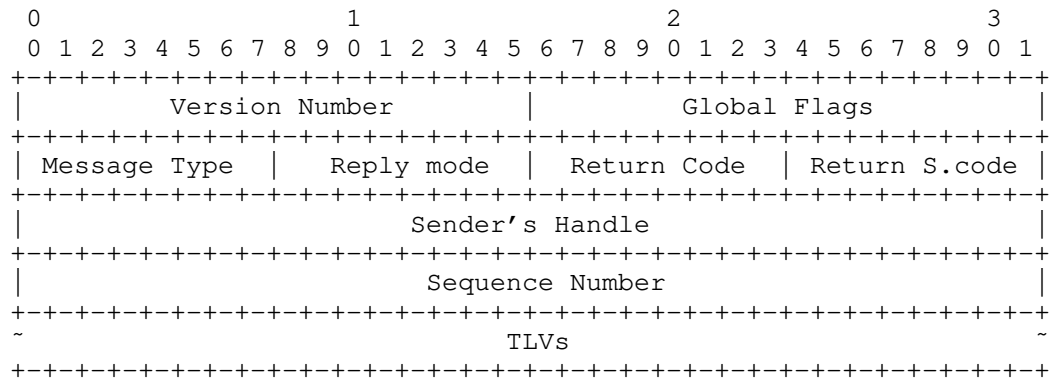


Figure 3: SFC Echo Request/Reply format

The interpretation of the fields is as follows:

The Version reflects the current version. The version number is to be incremented whenever a change is made that affects the ability of an implementation to parse or process control packet correctly.

The Global Flags is a bit vector field.

The Message Type field reflects the type of the packet. Value TBA3 identifies echo request and TBA4 - echo reply

The Reply Mode defines the type of the return path requested by the sender of the echo request.

Return Codes and Subcodes can be used to inform the sender about the result of processing its request.

The Sender's Handle is filled in by the sender and returned unchanged by the receiver in the echo reply. The sender MAY use a pseudo-random number generator (PRNG) to set the value of the Sender's Handle field. The value of the Sender's Handle field SHOULD NOT be changed in the course of the test session.

The Sequence Number is assigned by the sender and can be (for example) used to detect missed replies. The value of the Sequence Number field SHOULD be monotonically increasing in the course of the test session.

TLVs (Type-Length-Value tuples) have the two octets long Type field, two octets long Length field that is the length of the Value field in octets. Type values, see Section 8.7, less than 32768 identify mandatory TLVs that MUST either be supported by an implementation or result in the Return Code of 2 ("One or more of the TLVs was not understood") being sent in the echo response. Type values greater than or equal to 32768 identify optional TLVs that SHOULD be ignored if the implementation does not understand or support them. If a Type value for TLV or sub-TLV is in the range for Vendor Private Use, the Length MUST be at least 4, and the first four octets MUST be that vendor's the Structure of Management Information (SMI) [RFC1423] Private Enterprise Number, in network octet order. The rest of the Value field is private to the vendor.

5.1. Return Codes

The Return Code is set to zero by the sender of an echo request. The receiver of said echo request can set it to one of the values listed below in the corresponding echo reply that it generates.

Value	Meaning
-----	-----
0	No Return Code
1	Malformed echo request received
2	One or more of the TLVs was not understood

5.2. SFC Echo Request Transmission

SFC echo request control packet MUST use the appropriate encapsulation of the monitored SFP. If Network Service Header (NSH) is used, echo request MUST set O bit, as defined in [RFC8300]. SFC NSH MUST be immediately followed by the SFC Active OAM Header defined in Section 4. Message Type field in the SFC Active OAM Header MUST be set to SFC Echo Request/Echo Reply value (TBA2) per Section 8.2.

Value of the Reply Mode field MAY be set to:

- o Do Not Reply (TBA5) if one-way monitoring is desired. If the echo request is used to measure synthetic packet loss; the receiver may report loss measurement results to a remote node.
- o Reply via an IPv4/IPv6 UDP Packet (TBA6) value likely will be the most used.
- o Reply via Application Level Control Channel (TBA7) value if the SFP may have bi-directional paths.
- o Reply via Specified Path (TBA8) value to enforce the use of the particular return path specified in the included TLV to verify bi-directional continuity and also increase the robustness of the monitoring by selecting a more stable path.

5.3. SFC Echo Request Reception

Sending an SFC echo request to the control plane is triggered by one of the following packet processing exceptions: NSH TTL expiration, NSH Service Index (SI) expiration or the receiver is the terminal SFF for an SFP.

Firstly, the SFF that has received an SFC echo request verifies the general sanity of the received packet. If the packet is not well-formed, the receiver SFF SHOULD send an SFC echo reply with the Return Code set to "Malformed echo request received" and the Subcode set to zero. If there are any TLVs not marked as "Ignore" (i.e., if the TLV type is less than 32768, see Section 3) that SFF does not understand, the SFF SHOULD send an SFC echo reply with the Return Code set to "TLV not understood" and set the Subcode to zero. In the latter case, the SFF SHOULD include an Errored TLVs TLV that as sub-TLVs contains only the misunderstood TLVs. The header field's Sender's Handle, Sequence Number are not examined but are included in the SFC echo reply message.

5.4. SFC Echo Reply Transmission

The Reply Mode field directs whether and how the echo reply message should be sent. The sender of the echo request MAY use TLVs to request that the corresponding echo reply is transmitted over the specified path. Value TBA3 is referred to as "Do not reply" mode and suppresses transmission of echo reply packet. The default value (TBA6) for the Reply mode field requests the responder to send the echo reply packet out-of-band as IPv4 or IPv6 UDP packet.

Responder to the SFC echo request sends the echo reply over IP network if the Reply mode is Reply via an IPv4/IPv6 UDP Packet. Because SFC NSH does not identify the ingress of the SFP the echo request, the source ID MUST be included in the message and used as the IP destination address for IP/UDP encapsulation of the SFC echo reply. The sender of the SFC echo request MUST include SFC Source TLV Figure 4.

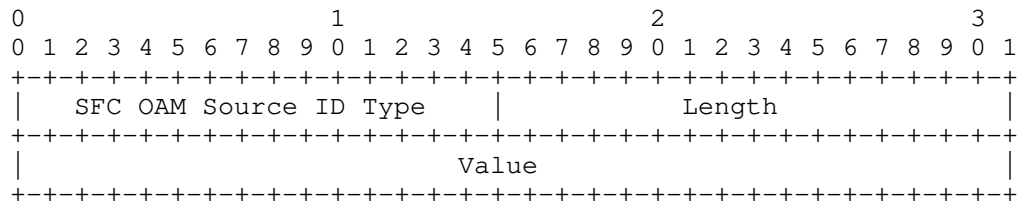


Figure 4: SFC Source TLV

where

SFC OAM Source Id Type is two octets in length and has the value of TBA9 Section 8.7.

Length is two octets long field, and the value equals the length of the Value field in octets.

Value field contains the IP address of the sender of the SFC OAM control message, IPv4 or IPv6.

The UDP destination port for SFC Echo Reply TBA10 will be allocated by IANA Section 8.8.

5.5. SFC Echo Reply Reception

An SFF SHOULD NOT accept SFC echo reply unless the received passes the following checks:

- o the received SFC echo reply is well-formed;
- o it has outstanding SFC echo request sent from the UDP port that matches destination UDP port number of the received packet;
- o if the matching to the echo request found, the value of Sender's Handle in the echo request sent is equal to the value of Sender's Handle in the echo reply received;
- o if all checks passed, the SFF checks if the Sequence Number in the echo request sent matches to the Sequence Number in the echo reply received.

6. Security Considerations

Overlay Echo Request/Reply operates within the domain of the overlay network and thus inherits any security considerations that apply to the use of that overlay technology and, consequently, underlay data plane. Also, the security needs for SFC echo request/reply are similar to those of ICMP ping [RFC0792], [RFC4443] and MPLS LSP ping [RFC8029].

There are at least three approaches of attacking a node in the overlay network using the mechanisms defined in the document. One is a Denial-of-Service attack, by sending SFC ping to overload an element of the SFC. The second may use spoofing, hijacking, replying, or otherwise tampering with SFC echo requests and/or replies to misrepresent, alter operator's view of the state of the SFC. The third is an unauthorized source using an SFC echo request/reply to obtain information about the SFC and/or its elements, e.g. SFF or SF.

It is RECOMMENDED that implementations throttle the SFC ping traffic going to the control plane to mitigate potential Denial-of-Service attacks.

Reply and spoofing attacks involving faking or replying SFC echo reply messages would have to match the Sender's Handle and Sequence Number of an outstanding SFC echo request message which is highly unlikely. Thus the non-matching reply would be discarded.

To protect against unauthorized sources trying to obtain information about the overlay and/or underlay an implementation MAY check that the source of the echo request is indeed part of the SFP.

7. Acknowledgments

Authors greatly appreciate thorough review and the most helpful comments from Dan Wing and Dirk von Hugo.

8. IANA Considerations

8.1. SFC Active OAM Protocol

IANA is requested to assign a new type from the SFC Next Protocol registry as follows:

Value	Description	Reference
TBA1	SFC Active OAM	This document

Table 1: SFC Active OAM Protocol

8.2. SFC Active OAM Message Type

IANA is requested to create a new registry called "SFC Active OAM Message Type". All code points in the range 1 through 32767 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC8126]. Remaining code points to be allocated according to the table Table 2:

Value	Description	Reference
0	Reserved	
1 - 32767	Reserved	IETF Consensus
32768 - 65530	Reserved	First Come First Served
65531 - 65534	Reserved	Private Use
65535	Reserved	

Table 2: SFC Active OAM Message Type

IANA is requested to assign new type from the SFC Active OAM Message Type registry as follows:

Value	Description	Reference
TBA2	SFC Echo Request/Echo Reply	This document

Table 3: SFC Echo Request/Echo Reply Type

8.3. SFC Echo Request/Echo Reply Parameters

IANA is requested to create new SFC Echo Request/Echo Reply Parameters registry.

8.4. SFC Echo Request/Echo Reply Message Types

IANA is requested to create in the SFC Echo Request/Echo Reply Parameters registry the new sub-registry Message Types. All code points in the range 1 through 191 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC8126] and assign values as follows:

Value	Description	Reference
0	Reserved	
TBA3	SFC Echo Request	This document
TBA4	SFC Echo Reply	This document
TBA4+1-191	Unassigned	IETF Review
192-251	Unassigned	First Come First Served
252-254	Unassigned	Private Use
255	Reserved	

Table 4: SFC Echo Request/Echo Reply Message Types

8.5. SFC Echo Reply Modes

IANA is requested to create in the SFC Echo Request/Echo Reply Parameters registry the new sub-registry Reply Modes All code points in the range 1 through 191 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC8126] and assign values as follows:

Value	Description	Reference
0	Reserved	
TBA5	Do Not Reply	This document
TBA6	Reply via an IPv4/IPv6 UDP Packet	This document
TBA7	Reply via Application Level Control Channel	This document
TBA8	Reply via Specified Path	This document
TBA8+1-191	Unassigned	IETF Review
192-251	Unassigned	First Come First Served
252-254	Unassigned	Private Use
255	Reserved	

Table 5: SFC Echo Reply Modes

8.6. SFC Echo Return Codes

IANA is requested to create in the SFC Echo Request/Echo Reply Parameters registry the new sub-registry Return Codes:

Value	Description	Reference
0-191	Unassigned	IETF Review
192-251	Unassigned	First Come First Served
252-254	Unassigned	Private Use
255	Reserved	

Table 6: SFC Echo Return Codes

Return Codes defined in this document are the following:

Value	Meaning
0	No Return Code
1	Malformed echo request received
2	One or more of the TLVs was not understood

8.7. SFC TLV Type

IANA is requested to create SFC OAM TLV Type registry. All code points in the range 1 through 32759 in this registry shall be allocated according to the "IETF Review" procedure as specified in

[RFC8126]. Code points in the range 32760 through 65279 in this registry shall be allocated according to the "First Come First Served" procedure as specified in [RFC8126]. Remaining code points are allocated according to the Table 7:

Value	Description	Reference
0	Reserved	This document
1- 32767	Mandatory TLV, unassigned	IETF Review
32768 - 65279	Optional TLV, unassigned	First Come First Served
65280 - 65519	Experimental	This document
65520 - 65534	Private Use	This document
65535	Reserved	This document

Table 7: SFC TLV Type Registry

This document defines the following new value in SFC OAM TLV Type registry:

Value	Description	Reference
TBA9	Source IP Address	This document

Table 8: SFC OAM Source IP Address Type

8.8. SFC OAM UDP Port

IANA is requested to allocate UDP port number according to

Service Name	Port Number	Transport Protocol	Description	Semantics Definition	Reference
SFC OAM	TBA10	UDP	SFC OAM	Section 5.4	This document

Table 9: SFC OAM Port

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>>.
- [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>>.

9.2. Informative References

- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<https://www.rfc-editor.org/info/rfc792>>.
- [RFC1423] Balenson, D., "Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers", RFC 1423, DOI 10.17487/RFC1423, February 1993, <<https://www.rfc-editor.org/info/rfc1423>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [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>>.
- [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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

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

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

Bhumip Khasnabish
Individual contributor

Email: vumipl@gmail.com

Cui Wang

Email: lindawangjoy@gmail.com

SFC WG
Internet-Draft
Intended status: Standards Track
Expires: 2 October 2022

G. Mirsky
Ericsson
W. Meng
ZTE Corporation
B. Khasnabish
Individual contributor
T. Ao
China Mobile
K. Leung
Cisco System
G. Mishra
Verizon Inc.
31 March 2022

Active OAM for Service Function Chaining
draft-ietf-sfc-multi-layer-oam-19

Abstract

A set of requirements for active Operation, Administration, and Maintenance (OAM) of Service Function Chains (SFCs) in a network is presented in this document. Based on these requirements, an encapsulation of active OAM messages in SFC and a mechanism to detect and localize defects are described.

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 2 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	3
2. Terminology and Conventions	4
2.1. Requirements Language	4
2.2. Acronyms	4
3. Requirements for Active OAM in SFC	5
4. Active OAM Identification in the NSH	7
5. Active SFC OAM Header	7
6. Echo Request/Echo Reply for SFC	8
6.1. Return Codes	10
6.2. Authentication in Echo Request/Reply	11
6.3. SFC Echo Request Transmission	11
6.3.1. Source TLV	12
6.4. SFC Echo Request Reception	13
6.4.1. Errored TLVs TLV	14
6.5. SFC Echo Reply Transmission	15
6.5.1. SFC Reply Path TLV	15
6.5.2. Theory of Operation	17
6.5.3. SFC Echo Reply Reception	18
6.5.4. Tracing an SFP	18
6.6. Verification of the SFP Consistency	19
6.6.1. SFP Consistency Verification packet	19
6.6.2. SFP Information Record TLV	19
6.6.3. SF Information Sub-TLV	20
6.6.4. SF Information Sub-TLV Construction	22
7. Security Considerations	23
8. Operational Considerations	24
9. Acknowledgments	25
10. IANA Considerations	25
10.1. SFC Active OAM Protocol	25
10.2. SFC Active OAM	25
10.2.1. SFC Active OAM Message Type	25
10.2.2. SFC Active OAM Header Flags	26
10.3. SFC Echo Request/Echo Reply Parameters	26
10.3.1. SFC Echo Request Flags	27
10.3.2. SFC Echo Request/Echo Reply Message Types	27
10.3.3. SFC Echo Reply Modes	28
10.3.4. SFC Echo Return Codes	29

10.4. SFC Active OAM TLV Type	30
10.5. SF Identifier Types	31
11. References	32
11.1. Normative References	32
11.2. Informative References	33
Contributors' Addresses	34
Authors' Addresses	35

1. Introduction

[RFC7665] defines data plane elements necessary to implement a Service Function Chaining (SFC). These include:

1. Classifiers that perform the classification of incoming packets. Such classification may result in associating a received packet to a service function chain.
2. Service Function Forwarders (SFFs) that are responsible for forwarding traffic to one or more connected Service Functions (SFs) according to the information carried in the SFC encapsulation and handling traffic coming back from the SFs and forwarding it to the next SFF.
3. SFs that are responsible for executing specific service treatment on received packets.

There are different views from different levels of the SFC. One is the service function chain, an entirely abstract view, which defines an ordered set of SFs that must be applied to packets selected based on classification rules. But service function chain doesn't specify the exact mapping between SFFs and SFs. Thus, another logical construct used in SFC is a Service Function Path (SFP). According to [RFC7665], SFP is the instantiation of the SFC in the network and provides a level of indirection between the entirely abstract SFCs and a fully specified ordered list of SFFs and SFs identities that the packet will visit when it traverses the SFC. The latter entity is referred to as Rendered Service Path (RSP). The main difference between SFP and RSP is that the former is the logical construct, while the latter is the realization of the SFP via the sequence of specific SFC data plane elements.

This document defines how active Operation, Administration and Maintenance (OAM), per [RFC7799] definition of active OAM, is identified when Network Service Header (NSH) is used as the SFC encapsulation. Following the analysis of SFC OAM in [RFC8924], this document applies and, when necessary, extends requirements listed in Section 4 of [RFC8924] for the use of active OAM in an SFP supporting fault management and performance monitoring. Active OAM tools,

conformant to the requirements listed in Section 3, improve, for example, troubleshooting efficiency and defect localization in SFP because they specifically address the architectural principles of NSH. For that purpose, SFC Echo Request and Echo Reply are specified in Section 6. This mechanism enables on-demand Continuity Check, Connectivity Verification, among other operations over SFC in networks, addresses functionalities discussed in Sections 4.1, 4.2, and 4.3 of [RFC8924]. SFC Echo Request and Echo Reply, defined in this document, can be used with encapsulations other than NSH, for example, using MPLS encapsulation, as described in [RFC8595]. The applicability of the SFC Echo Request/Reply mechanism in SFC encapsulations other than NSH is outside the scope of this document.

2. Terminology and Conventions

The terminology defined in [RFC7665] is used extensively throughout this document, and the reader is expected to be familiar with it.

In this document, SFC OAM refers to an active OAM [RFC7799] in an SFC architecture. In this document, "Echo Request/Reply" and "SFC Echo Request/Reply" are used interchangeably.

2.1. Requirements Language

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.

2.2. Acronyms

E2E: End-to-End

FM: Fault Management

NSH: Network Service Header

OAM: Operations, Administration, and Maintenance

RSP: Rendered Service Path

SF: Service Function

SFC: Service Function Chain

SFF: Service Function Forwarder

SFP: Service Function Path

MAC: Message Authentication Code

3. Requirements for Active OAM in SFC

As discussed in [RFC8924], SFC-specific means are needed to perform the OAM task of fault management (FM) in an SFC architecture, including failure detection, defect characterization, and localization. This document defines the set of requirements for active FM OAM mechanisms to be used in an SFC architecture.

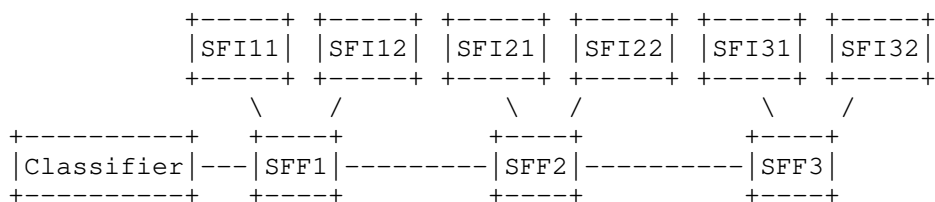


Figure 1: An Example of SFC Data Plane Architecture

The architecture example depicted in Figure 1 considers a service function chain that includes three distinct service functions. In this example, the SFP traverses SFF1, SFF2, and SFF3. Each SFF is connected to two instances of the same service function. End-to-end (E2E) SFC OAM has the Classifier as the ingress and SFF3 as its egress. Segment SFC OAM is between two elements that are part of the same SFP. Following are the requirements for an FM SFC OAM, whether with the E2E or segment scope:

REQ#1: Packets of active SFC OAM SHOULD be fate sharing with the monitored SFC data in the forward direction from ingress toward egress endpoint(s) of the OAM test.

The fate sharing, in the SFC environment, is achieved when a test packet traverses the same path and receives the same treatment in the underlay network layer as an SFC-encapsulated packet (e.g., NSH).

REQ#2: SFC OAM MUST support monitoring of the continuity of the SFP between any of its elements.

An SFC failure might be declared when several consecutive test packets are not received within a pre-determined time. For example, in the E2E FM SFC OAM case, the egress, SFF3, in the example in Figure 1, could be the entity that detects the SFP's failure by

monitoring a flow of periodic test packets. The ingress may be capable of recovering from the failure, e.g., using redundant SFC elements. Thus, it is beneficial for the egress to signal the new defect state to the ingress, which in this example is the Classifier. Hence the following requirement:

REQ#3: SFC OAM MUST support Remote Defect Indication notification by the egress to the ingress.

REQ#4: SFC OAM MUST support connectivity verification of the SFP. Definition of the misconnection defect, entry, and exit criteria are outside the scope of this document.

Once the SFF1 detects the defect, the objective of the SFC OAM changes from the detection of a defect to defect characterization and localization.

REQ#5: SFC OAM MUST support fault localization of the Loss of Continuity Check within an SFP.

REQ#6: SFC OAM MUST support an SFP tracing to discover the RSP.

In the example presented in Figure 1, two distinct instances of the same service function share the same SFF. In this example, the SFP can be realized over several RSPs that use different instances of SF of the same type. For instance, RSP1(SFI11--SFI21--SFI31) and RSP2(SFI12--SFI22--SFI32). Available RSPs can be discovered using the trace function discussed in Section 4.3 [RFC8924] or the procedure defined in Section 6.5.4.

REQ#7: SFC OAM MUST have the ability to discover and exercise all available RSPs in the network.

The SFC OAM layer model described in [RFC8924] offers an approach for defect localization within a service function chain. As the first step, the SFP's continuity for SFFs that are part of the same SFP could be verified. After the reachability of SFFs has already been verified, SFFs that serve an SF may be used as a test packet source. In such a case, SFF can act as a proxy for another element within the service function chain.

REQ#8: SFC OAM MUST be able to trigger on-demand FM with responses being directed towards the initiator of such proxy request.

4. Active OAM Identification in the NSH

Active SFC OAM combines OAM commands and/or data included in a message that immediately follows the NSH. To identify the active SFC OAM message, the "Next Protocol" field MUST be set to Active SFC OAM (TBA1) (Section 10.1). The O bit in NSH MUST be set, according to [I-D.ietf-sfc-oam-packet]. A case when the O bit is clear and the "Next Protocol" field value is set to Active SFC OAM (TBA1) is considered an erroneous combination. An implementation MUST report it. The notification mechanism is outside the scope of this specification. The packet SHOULD be dropped. An implementation MAY have control to enable the processing of the OAM payload.

5. Active SFC OAM Header

As demonstrated in Section 4 [RFC8924] and Section 3 of this document, SFC OAM is required to perform multiple tasks. Several active OAM protocols could be used to address all the requirements. When IP/UDP encapsulation of an SFC OAM control message is used, protocols can be demultiplexed using the destination UDP port number. But extra IP/UDP headers, especially in an IPv6 network, add noticeable overhead. This document defines Active OAM Header (Figure 2) to demultiplex active OAM protocols on an SFC.

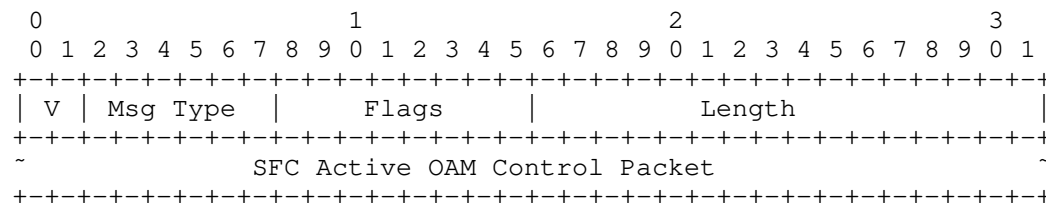


Figure 2: SFC Active OAM Header

V - two-bit-long field indicates the current version of the SFC active OAM header. The current value is 0. The version number is to be incremented whenever a change is made that affects the ability of an implementation to parse or process the SFC Active OAM header correctly. For example, if syntactic or semantic changes are made to any of the fixed fields.

Msg Type - six bits long field identifies OAM protocol, e.g., Echo Request/Reply.

Flags - eight bits long field carries bit flags that define optional capability and thus processing of the SFC active OAM control packet, e.g., optional timestamping. No flags are defined in this document, and therefore, the bit flags MUST be zeroed on transmission and ignored on receipt.

Length - two octets long field that is the length of the SFC active OAM control packet in octets.

6. Echo Request/Echo Reply for SFC

Echo Request/Reply is a well-known active OAM mechanism extensively used to verify a path's continuity, detect inconsistencies between a state in control and the data planes, and localize defects in the data plane. ICMP ([RFC0792] for IPv4 and [RFC4443] for IPv6 networks, respectively) and [RFC8029] are examples of broadly used active OAM protocols based on the Echo Request/Reply principle. The SFC Echo Request/Reply defined in this document addresses several requirements listed in Section 3. Specifically, it can be used to check the continuity of an SFP, trace an SFP, or localize the failure within an SFP. The SFC Echo Request/Reply control message format is presented in Figure 3.

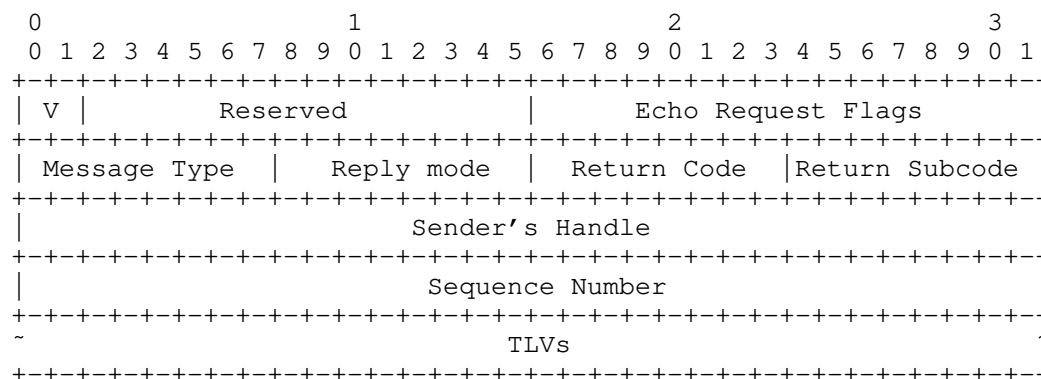


Figure 3: SFC Echo Request/Reply Format

The interpretation of the fields is as follows:

Version (V) is a two-bit field that indicates the current version of the SFC Echo Request/Reply. The current value is 0. The version number is to be incremented whenever a change is made that affects the ability of an implementation to parse or process the control packet correctly. If a packet presumed to carry an SFC Echo Request/Reply is received at an SFF, and the SFF does not understand the Version field value, the packet **MUST** be discarded, and the event **SHOULD** be logged.

Reserved - fourteen-bit field. It **MUST** be zeroed on transmission and ignored on receipt.

The Echo Request Flags is a two-octet bit vector field. Note that a flag defined in the Flags field of the SFC Active OAM header in Figure 2 has no implication of those defined in the Echo Request Flags field of an Echo Request/Reply message.

The Message Type is a one-octet field that reflects the packet type. Value 1 identifies Echo Request and 2 - Echo Reply.

The Reply Mode is a one-octet field. It defines the type of the return path requested by the sender of the Echo Request.

Return Codes and Subcodes are one-octet fields each. These can be used to inform the sender about the result of processing its request. Initial Return Code values are provided in Table 1. For all Return Code values defined in this document, the value of the Return Subcode field **MUST** be set to zero.

The Sender's Handle is a four-octet field. It **MUST** be filled in by the sender of the Echo Request and returned unchanged by the Echo Reply sender (if a reply mandated). The sender of the Echo Request **SHOULD** use a pseudo-random number generator to set the value of the Sender's Handle field.

The Sequence Number is a four-octet field, and it is assigned by the sender and can be, for example, used to detect missed replies. Initial Sequence Number **MUST** be randomly generated and then **SHOULD** be monotonically increasing in the course of the test session.

TLV is a variable-length construct. Multiple TLVs **MAY** be placed in an SFC Echo Request/Reply packet. None, one or more sub-TLVs may be enclosed in a TLV, subject to the semantics of the (outer) TLV. Figure 4 presents the format of an SFC Echo Request/Reply TLV, where fields are defined as follows:

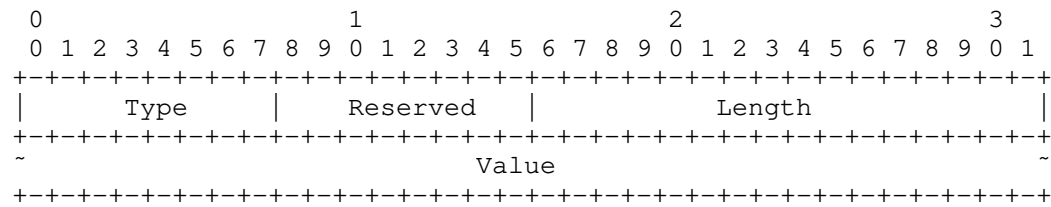


Figure 4: SFC Echo Request/Reply TLV Format

Type - a one-octet field that characterizes the interpretation of the Value field. The value of the Type field determines its interpretation and encoding. Type values allocated according to Section 10.4.

Reserved - a one-octet field. The field MUST be zeroed on transmission and ignored on receipt.

Length - a two-octet field equal to the Value field's length in octets.

Value - a variable-length field. The value of the Type field determines its interpretation and encoding.

6.1. Return Codes

The value of the Return Code field is set to zero by the sender of an Echo Request. The receiver of said Echo Request can set it to one of the values listed in Table 1 in the corresponding Echo Reply that it generates (in cases when the reply is requested).

Value	Description
0	No Return Code
1	Malformed Echo Request received
2	One or more of the TLVs was not understood
3	Authentication failed

Table 1: SFC Echo Return Codes

6.2. Authentication in Echo Request/Reply

Authentication can be used to protect the integrity of the information in SFC Echo Request and/or Echo Reply. In the [RFC9145] a variable-length Context Header has been defined to protect the integrity of the NSH and the payload. The header can also be used for the optional encryption of sensitive metadata. MAC#1 (Message Authentication Code) Context Header is more suitable for the integrity protection of active SFC OAM, particularly of the defined in this document SFC Echo Request and Echo Reply. On the other hand, using MAC#2 Context Header allows the detection of mishandling of the O-bit by a transient SFC element.

6.3. SFC Echo Request Transmission

SFC Echo Request control packet MUST use the appropriate underlay network encapsulation of the monitored SFP. If the NSH is used, Echo Request MUST set O bit, as defined in [I-D.ietf-sfc-oam-packet]. NSH MUST be immediately followed by the SFC Active OAM Header defined in Section 4. The Message Type field's value in the SFC Active OAM Header MUST be set to SFC Echo Request/Echo Reply value (1) per Section 10.2.1.

Value of the Reply Mode field MAY be set to:

- * Do Not Reply (1) if one-way monitoring is desired. If the Echo Request is used to measure synthetic packet loss, the receiver may report loss measurement results to a remote node. Note that ways of learning the identity of that node are outside the scope of this specification.
- * Reply via an IPv4/IPv6 UDP Packet (2) value likely will be the most used.
- * Reply via Application-Level Control Channel (3) value if the SFP may have bi-directional paths.
- * Reply via Specified Path (4) value to enforce the use of the particular return path specified in the included TLV to verify bi-directional continuity and also increase the robustness of the monitoring by selecting a more stable path. Section 6.5.1 provides an example of communicating an explicit path for the Echo Reply.

6.3.1. Source TLV

Responder to the SFC Echo Request encapsulates the SFC Echo Reply message in IP/UDP packet if the Reply mode is "Reply via an IPv4/IPv6 UDP Packet". Because the NSH does not identify the ingress node that generated the Echo Request, the source ID MUST be included in the message and used as the IP destination address and destination UDP port number of the SFC Echo Reply. The sender of the SFC Echo Request MUST include an SFC Source TLV (Figure 5).

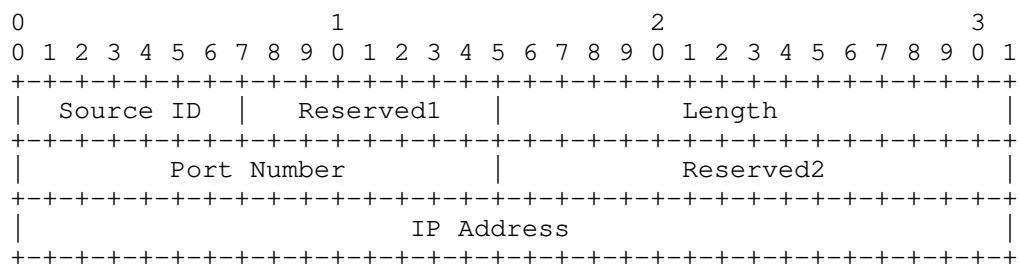


Figure 5: SFC Source TLV

where

Source ID Type is a one-octet field and has the value of 1
Section 10.4.

Reserved1 - one-octet field. The field MUST be zeroed on
transmission and ignored on receipt.

Length is a two-octet field, and the value equals the length of
the data following the Length field counted in octets. The value
of the Length field can be 8 or 20. If the value of the field is
neither, the Source TLV is considered to be malformed.

Port Number is a two-octet field. It contains the UDP port number
of the sender of the SFC OAM control message. The value of the
field MUST be used as the destination UDP port number in the IP/
UDP encapsulation of the SFC Echo Reply message.

Reserved2 is a two-octet field. The field MUST be zeroed on
transmit and ignored on receipt.

IP Address field contains the IP address of the sender of the SFC
OAM control message, IPv4 or IPv6. The value of the field MUST be
used as the destination IP address in the IP/UDP encapsulation of
the SFC Echo Reply message.

A single Source ID TLV for each address family, i.e., IPv4 and IPv6, MAY be present in an SFC Echo Request message. If the Source TLVs for both address families are present in an SFC Echo Request message, the SFF MUST NOT replicate an SFC Echo Reply but choose the destination IP address for the SFC Echo Reply based on the local policy. If more than one Source ID TLV per the address family is present, the receiver MUST use the first TLV and ignore the rest.

6.4. SFC Echo Request Reception

Punting received SFC Echo Request to the control plane is triggered by one of the following packet processing exceptions: NSH TTL expiration, NSH Service Index (SI) expiration, or the receiver is the terminal SFF for an SFP.

Firstly, if the SFC Echo Request is integrity-protected, the receiving SFF first MUST verify the authentication. Then the receiver SFF MUST validate the Source TLV, as defined in Section 6.3.1. Suppose the authentication validation has failed and the Source TLV is considered properly formatted. In that case, the SFF MUST send to the system identified in the Source TLV (see Section 6.5), according to a rate-limit control mechanism, an SFC Echo Reply with the Return Code set to "Authentication failed" and the Subcode set to zero. If the Source TLV is determined malformed, the received SFC Echo Request processing is stopped, the message is dropped, and the event SHOULD be logged, according to a rate-limiting control for logging. Then, the SFF that has received an SFC Echo Request verifies the rest of the received packet's general sanity. If the packet is not well-formed, the receiver SFF SHOULD send an SFC Echo Reply with the Return Code set to "Malformed Echo Request received" and the Subcode set to zero under the control of the rate-limiting mechanism to the system identified in the Source TLV (see Section 6.5). If there are any TLVs that the SFF does not understand, the SFF MUST send an SFC Echo Reply with the Return Code set to 2 ("One or more TLVs was not understood") and set the Subcode to zero. In the latter case, the SFF MAY include an Errored TLVs TLV (Section 6.4.1) that, as sub-TLVs, contains only the misunderstood TLVs. Sender's Handle and Sequence Number fields are not examined but are included in the SFC Echo Reply message. If the sanity check of the received Echo Request succeeded, then the SFF at the end of the SFP MUST set the Return Code value to 5 ("End of the SFP") and the Subcode set to zero. If the SFF is not at the end of the SFP and the TTL value is 1, the value of the Return Code MUST be set to 4 ("TTL Exceeded") and the Subcode set to zero. In all other cases, SFF MUST set the Return Code value to 0 ("No Return Code") and the Subcode set to zero.

6.4.1. Errored TLVs TLV

If the Return Code for the Echo Reply is determined as 2 ("One or more TLVs was not understood"), the Errored TLVs TLV might be included in an Echo Reply. The use of this TLV is meant to inform the sender of an Echo Request of TLVs either not supported by an implementation or parsed and found to be in error.

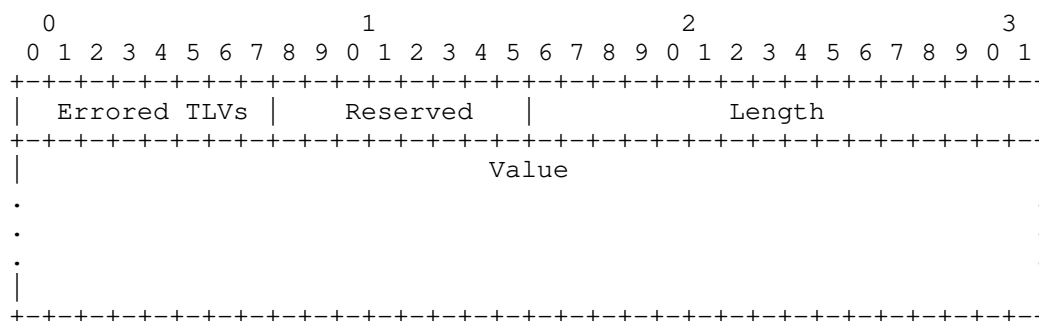


Figure 6: Errored TLVs TLV

where

The Errored TLVs Type MUST be set to 2 Section 10.4.

Reserved - one-octet field. The field MUST be zeroed on transmission and ignored on receipt.

Length - two-octet field equal to the length of the Value field in octets.

The Value field contains the TLVs, encoded as sub-TLVs (as shown in Figure 7), that were not understood or failed to be parsed correctly.

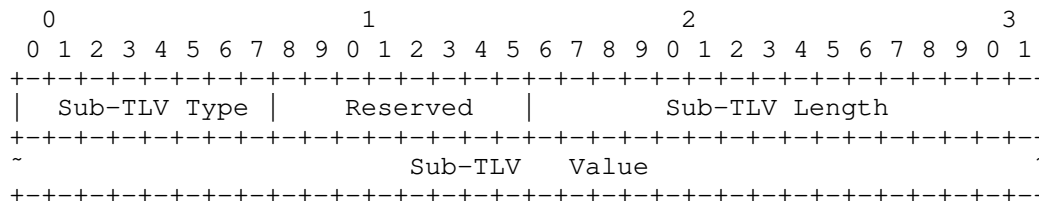


Figure 7: Not Understood or Failed TLV as Sub-TLV

where

The Sub-TLV's Type the copy of the first octet of the not understood or failed to be parced TLV.

Reserved - one-octet field. The field MUST be zeroed on transmission and ignored on receipt.

Sub-TLV Length - two-octet field equal to the value of the Length field of the errored TLV.

The Sub-TLV Value field contains data that follow the Legth field in the errored TLV.

6.5. SFC Echo Reply Transmission

The "Reply Mode" field directs whether and how the Echo Reply message should be sent. The Echo Request sender MAY use TLVs to request that the corresponding Echo Reply be transmitted over the specified path. Section 6.5.1 provides an example of a TLV that specifies the return path of the Echo Reply. Value 1 is the "Do not reply" mode and suppresses the Echo Reply packet transmission. The default value (2) for the Reply mode field requests the responder to send the Echo Reply packet out-of-band as IPv4 or IPv6 UDP packet.

6.5.1. SFC Reply Path TLV

While SFC Echo Request always traverses the SFP it is directed to by using NSH, the corresponding Echo Reply usually is sent without NSH. In some cases, an operator might choose to direct the responder to send the Echo Reply with NSH over a particular SFP. This section defines a new Type-Length-Value (TLV), Reply Service Function Path TLV, for Reply via Specified Path mode of SFC Echo Reply.

The Reply Service Function Path TLV can provide an efficient mechanism to test SFCs, such as bidirectional and hybrid SFC, as defined in Section 2.2 [RFC7665]. For example, it allows an operator to test both directions of the bidirectional or hybrid SFP with a single SFC Echo Request/Echo Reply operation.

The SFC Reply Path TLV carries the information that sufficiently identifies the return SFP that the SFC Echo Reply message is expected to follow. The format of SFC Reply Path TLV is shown in Figure 8.

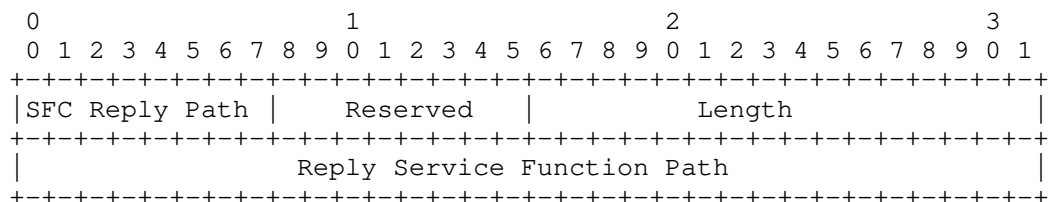


Figure 8: SFC Reply TLV Format

where:

- * SFC Reply Path Type: is a one-octet field, indicates the TLV that contains information about the SFC Reply path. IANA is requested to assign value 3,
- * Reserved - one-octet field. The field MUST be zeroed on transmission and ignored on receipt.
- * Length: is a two-octet field, MUST be equal to 4
- * Reply Service Function Path is used to describe the return path that an SFC Echo Reply is requested to follow.

The format of the Reply Service Function Path field displayed in Figure 9.

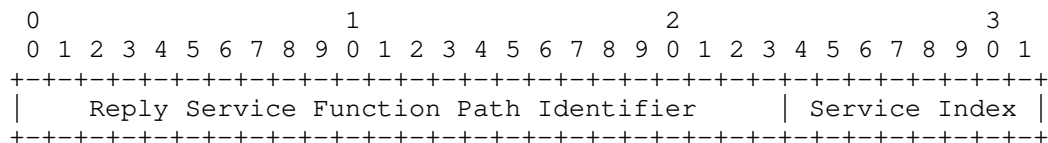


Figure 9: Reply Service Function Path Field Format

where:

- * Reply Service Function Path Identifier: SFP identifier for the path that the SFC Echo Reply message is requested to be sent over.
- * Service Index: the value for the Service Index field in the NSH of the SFC Echo Reply message.

6.5.2. Theory of Operation

[RFC7110] defined mechanism to control return path for MPLS LSP Echo Reply. In SFC's case, the return path is an SFP along which the SFC Echo Reply message MUST be transmitted. Hence, the SFC Reply Path TLV included in the SFC Echo Request message MUST sufficiently identify the SFP that the sender of the Echo Request message expects the receiver to use for the corresponding SFC Echo Reply.

When sending an Echo Request, the sender MUST set the value of Reply Mode field to "Reply via Specified Path", defined in Section 6.3, and if the specified path is an SFC path, the Request MUST include SFC Reply Path TLV. The SFC Reply Path TLV consists of the identifier of the reverse SFP and an appropriate Service Index.

If the NSH of the received SFC Echo Request includes the MAC Context Header, the packet's authentication MUST be verified before using any data. If the verification fails, the receiver MUST stop processing the SFC Return Path TLV and MUST send the SFC Echo Reply with the Return Codes value set to the value Authentication failed from the IANA's Return Codes sub-registry of the SFC Echo Request/Echo Reply Parameters registry.

The destination SFF of the SFP being tested or the SFF at which SFC TTL expired (as per [RFC8300]) may be sending the Echo Reply. The processing described below equally applies to both cases and is referred to as responding SFF.

If the Echo Request message with SFC Reply Path TLV, received by the responding SFF, has Reply Mode value of "Reply via Specified Path" but no SFC Reply Path TLV is present, then the responding SFF MUST send Echo Reply with Return Code set to 6 ("Reply Path TLV is missing"). If the responding SFF cannot find the requested SFP it MUST send Echo Reply with Return Code set to 7 ("Reply SFP was not found") and include the SFC Reply Path TLV from the Echo Request message.

Suppose the SFC Echo Request receiver cannot determine whether the specified return path SFP has the route to the initiator. In that case, it SHOULD set the value of the Return Codes field to 8 ("Unverifiable Reply Path"). The receiver MAY drop the Echo Request when it cannot determine whether SFP's return path has the route to the initiator. When sending Echo Request, the sender SHOULD choose a proper source address according to the specified return path SFP to help the receiver find the viable return path.

6.5.2.1. Bi-directional SFC Case

The ability to specify the return path for an Echo Reply might be used in the case of bi-directional SFC. The egress SFF of the forward SFP might not be co-located with a classifier of the reverse SFP, and thus the egress SFF has no information about the reverse path of an SFC. Because of that, even for bi-directional SFC, a reverse SFP needs to be indicated in a Reply Path TLV in the Echo Request message.

6.5.3. SFC Echo Reply Reception

An SFF SHOULD NOT accept SFC Echo Reply unless the received message passes the following checks:

- * the received SFC Echo Reply is well-formed;
- * if the matching to the Echo Request found, the value of the Sender's Handle in the Echo Request sent is equal to the value of Sender's Handle in the Echo Reply received;
- * if all checks passed, the SFF checks if the Sequence Number in the Echo Request sent matches to the Sequence Number in the Echo Reply received.

6.5.4. Tracing an SFP

SFC Echo Request/Reply can be used to isolate a defect detected in the SFP and trace an RSP. As with ICMP echo request/reply [RFC0792] and MPLS echo request/reply [RFC8029], this mode is referred to as "traceroute". In the traceroute mode, the sender transmits a sequence of SFC Echo Request messages starting with the NSH TTL value set to 1 and is incremented by 1 in each next Echo Request packet. The sender stops transmitting SFC Echo Request packets when the Return Code in the received Echo Reply equals 5 ("End of the SFP").

Suppose a specialized information element (e.g., IPv6 Flow Label [RFC6437] or Flow ID [I-D.ietf-sfc-nsh-tlv]) is used for distributing the load across Equal Cost Multi-Path or Link Aggregation Group paths. In that case, such an element MAY also be used for the SFC OAM traffic. Doing so is meant to induce the SFC Echo Request to follow the same RSP as the monitored flow.

6.6. Verification of the SFP Consistency

The consistency of an SFP can be verified by comparing the view of the SFP from the control or management plane with information collected from traversed by an SFC NSH Echo Request message. Every SFF that receives the Consistency Verification Request (CVReq) (specified in Section 6.6.1) MUST perform the following actions:

- * Collect information of the traversed by the CVReq packet SFs and send it to the ingress SFF as CVRep packet over IP network;
- * Forward the CVReq to the next downstream SFF if the one exists.

As a result, the ingress SFF collects information about all traversed SFFs and SFs, information on the actual path the CVReq packet has traveled. That information is used to verify the SFC's path consistency. The mechanism for the SFP consistency verification is outside the scope of this document.

6.6.1. SFP Consistency Verification packet

For the verification of an SFP consistency, two new types of messages to the SFC Echo Request/Reply operation defined in Section 6 with the following values detailed in Section 10.3.2:

- * 3 - SFP Consistency Verification Request
- * 4 - SFP Consistency Verification Reply

Upon receiving the CVReq, the SFF MUST respond with the Consistency Verification Reply (CVRep). The SFF MUST include the SFs information, as described in Section 6.6.3 and Section 6.6.2.

6.6.2. SFF Information Record TLV

For the received CVReq, an SFF is expected to include in the CVRep message the information about SFs that are mapped to that SFF. The SFF MUST include SFF Information Record TLV (Figure 10) in CVRep message. Every SFF sends back a single CVRep message, including information on all the SFs attached to the SFF on the SFP, as requested in the received CVReq message using the SF Information sub-TLV (Section 6.6.3).

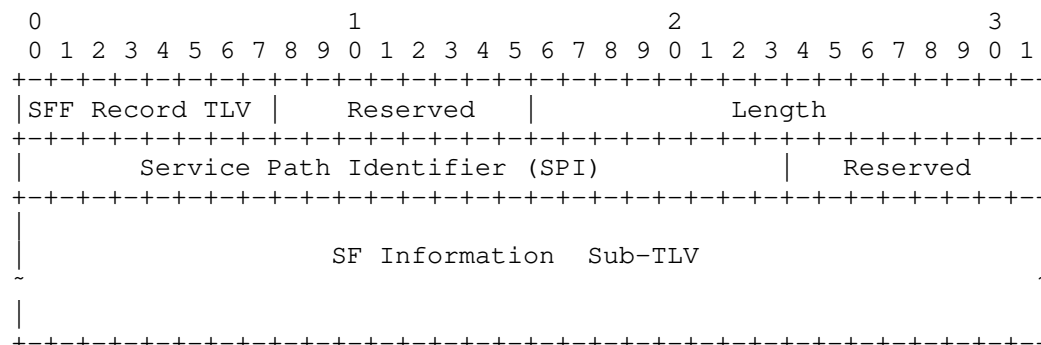


Figure 10: SFF Information Record TLV

The SFF Information Record TLV is a variable-length TLV that includes the information of all SFs mapped to the particular SFF instance for the specified SFP. Figure 10 presents the format of an SFF Information Record TLV, where fields are defined as the following:

SFF Record TLV - one-octet field. The value is (4) (Section 10.4).

Reserved - one-octet field. The field MUST be zeroed on transmission and ignored on receipt.

Service Path Identifier (SPI): The identifier of SFP to which all the SFs in this TLV belong.

SF Information Sub-TLV: The sub-TLV is as defined in Figure 11.

If the NSH of the received SFC Echo Reply includes the MAC Context Header [RFC9145], the authentication of the packet MUST be verified before using any data. If the verification fails, the receiver MUST stop processing the SFF Information Record TLV and notify an operator. The notification mechanism SHOULD include control of rate-limiting messages. Specification of the notification mechanism is outside the scope of this document.

6.6.3. SF Information Sub-TLV

Every SFF receiving CVReq packet MUST include the SF characteristic data into the CVRep packet. The format of an SF Information sub-TLV, included in a CVRep packet, is shown in Figure 11.

After the CVReq message traverses the SFP, all the information about the SFs on the SFP is available from the TLVs included in CVRep messages.

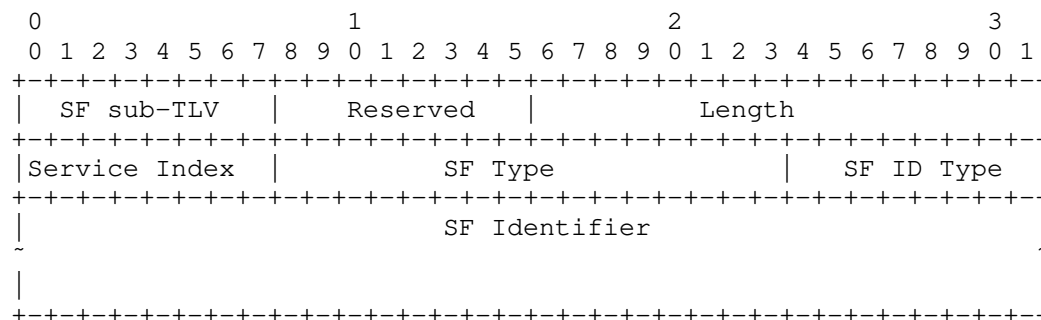


Figure 11: Service Function Information Sub-TLV

SF sub-TLV Type: Two-octets long field. The value is (5) (Section 10.4).

Reserved - one-octet field. The field MUST be zeroed on transmission and ignored on receipt.

Length - two-octet long field. The value of this field is the length of the data following the Length field counted in octets.

Service Index - indicates the SF's position on the SFP.

SF Type - two-octet field. It is defined in [RFC9015] and indicates the type of SF, e.g., Firewall, Deep Packet Inspection, WAN optimization controller, etc.

SF ID Type - one-octet field with values defined as Section 10.5.

SF Identifier - an identifier of the SF. The length of the SF Identifier depends on the type of the SF ID Type. For example, if the SF Identifier is its IPv4 address, the SF Identifier should be 32 bits.

6.6.4. SF Information Sub-TLV Construction

Each SFF in the SFP MUST send one and only one CVRep corresponding to the CVReq. If only one SF is attached to the SFF in such SFP, only one SF information sub-TLV is included in the CVRep. If several SFs attached to the SFF in the SFP, SF Information sub-TLV MUST be constructed as described below in either Section 6.6.4.1 and Section 6.6.4.2.

6.6.4.1. Multiple SFs as Hops of an SFP

Multiple SFs attached to the same SFF can be the hops of the SFP. The service indexes of these SFs on thatSFP will be different. Service function types of these SFs could be different or be the same. Information about all SFs MAY be included in the CVRep message. Information about each SF MUST be listed as separate SF Information sub-TLVs in the CVRep message.

An example of the SFP consistency verification procedure for this case is shown in Figure 12. The Service Function Path (SPI=x) is SF1->SF2->SF4->SF3. The SF1, SF2, and SF3 are attached to SFF1, and SF4 is attached to SFF2. The CVReq message is sent to the SFFs in the sequence of the SFP(SFF1->SFF2->SFF1). Every SFF(SFF1, SFF2) replies with the information of SFs belonging to the SFP. The SF information Sub-TLV in Figure 11 contains information for each SF (SF1, SF2, SF3, and SF4).

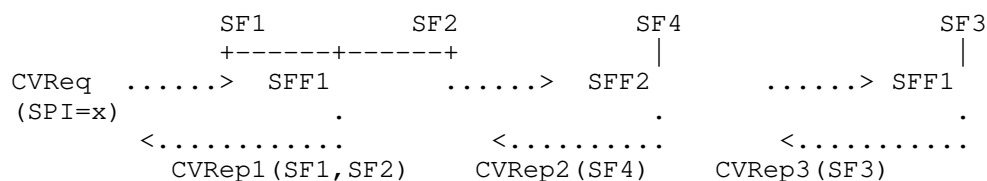


Figure 12: Example 1 for CVRep with multiple SFs

6.6.4.2. Multiple SFs for load balance

Multiple SFs may be attached to the same SFF to spread the load; in other words, that means that the particular traffic flow will traverse only one of these SFs. These SFs have the same Service Function Type and Service Index. For this case, the SF identifiers and SF ID Type of all these SFs will be listed in the SF Identifiers field and SF ID Type in a single SF information sub-TLV of the CVRep message. The number of these SFs can be calculated using the SF ID Type and the value of the Length field of the sub-TLV.

An example of the SFP consistency verification procedure for this case is shown in Figure 13. The Service Function Path (SPI=x) is SF1a/SF1b->SF2a/SF2b. The Service Functions SF1a and SF1b are attached to SFF1, which balances the load among them. The Service Functions SF2a and SF2b are attached to SFF2, which, in turn, balances its load between them. The CVReq message is sent to the SFFs in the sequence of the SFP (i.e. SFF1->SFF2). Every SFF (SFF1, SFF2) replies with the information of SFs belonging to the SFP. The SF information Sub-TLV in Figure 11 contains information for all SFs at that hop.

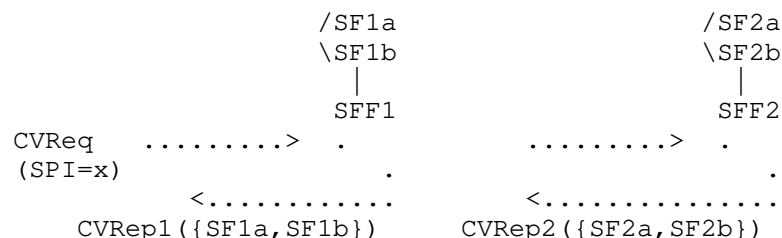


Figure 13: Example 2 for CVRep with multiple SFs

7. Security Considerations

When the integrity protection for SFC active OAM, and SFC Echo Request/Reply in particular, is required, using one of the Context Headers defined in [RFC9145] is RECOMMENDED. MAC#1 Context Header could be more suitable for active SFC OAM because it does not require re-calculation of the MAC when the value of the NSH Base Header's TTL field is changed. Integrity protection for SFC active OAM can also be achieved using mechanisms in the underlay data plane. For example, if the underlay is an IPv6 network, IP Authentication Header [RFC4302] or IP Encapsulating Security Payload Header [RFC4303] can be used to provide integrity protection. Confidentiality for the SFC Echo Request/Reply exchanges can be achieved using the IP Encapsulating Security Payload Header [RFC4303]. Also, the security needs for SFC Echo Request/Reply are similar to those of ICMP ping [RFC0792], [RFC4443] and MPLS LSP ping [RFC8029].

There are at least three approaches to attacking a node in the overlay network using the mechanisms defined in the document. One is a Denial-of-Service attack, sending SFC Echo Requests to overload an element of the SFC. The second may use spoofing, hijacking, replying, or otherwise tampering with SFC Echo Requests and/or replies to misrepresent, alter the operator's view of the state of the SFC. The third is an unauthorized source using an SFC Echo Request/Reply to obtain information about the SFC and/or its elements, e.g., SFFs and/or SFPs.

It is RECOMMENDED that implementations throttle the SFC ping traffic going to the control plane to mitigate potential Denial-of-Service attacks.

Reply and spoofing attacks involving faking or replying to SFC Echo Reply messages would have to match the Sender's Handle and Sequence Number of an outstanding SFC Echo Request message, which is highly unlikely for off-path attackers. A non-matching reply would be discarded.

To protect against unauthorized sources trying to obtain information about the overlay and/or underlay, an implementation MAY check that the source of the Echo Request is indeed part of the SFP.

Also, since the Service Function Information sub-TLV discloses information about the SFP, the spoofed CVReq packet may be used to obtain network information. Thus it is RECOMMENDED that implementations provide a means of checking the source addresses of CVReq messages, specified in SFC Source TLV Section 6.3.1, against an access list before accepting the message.

8. Operational Considerations

This section provides information about operational aspects of the SFC NSH Echo Request/Reply according to recommendations in [RFC5706].

SFC NSH Echo Request/Reply provides essential OAM functions for network operators. SFC NSH Echo Request/Reply is intended to detect and localize defects in an SFC. For example, by comparing results of the trace function in operational and failed states, an operator can locate the defect, e.g., the connection between SFF1 and SFF2 (Figure 1). Note that a more specific failure location can be determined using OAM tools in the underlay network. The mechanism defined in this document can be used on-demand or for periodic validation of an SFP or RSP. Because the protocol uses information in the SFC control plane, an operator must have the ability to control the frequency of transmitted Echo Request and Reply messages. A reasonably selected default interval between Echo Request control

packets can provide additional benefit for an operator. If the protocol is incrementally deployed in the NSH domain, SFC elements, e.g., Classifier or SFF, that don't support Active SFC OAM will discard protocol's packets. SFC NSH Echo Request/Reply also can be used in combination with the existing mechanisms discussed in [RFC8924], filling the gaps and extending their functionalities.

Management of the SFC NSH Echo Request/Reply protocol can be provided by a proprietary tool, e.g., command line interface, or based on a data model, structured or standardized.

9. Acknowledgments

The authors greatly appreciate the thorough review and the most helpful comments from Dan Wing, Dirk von Hugo, Mohamed Boucadair, Donald Eastlake, Carlos Pignataro, and Frank Brockners. The authors are thankful to John Drake for his review and the reference to the work on BGP Control Plane for NSH SFC. The authors express their appreciation to Joel M. Halpern for his suggestion about the load-balancing scenario.

10. IANA Considerations

10.1. SFC Active OAM Protocol

IANA is requested to assign a new type from the SFC Next Protocol registry as follows:

Value	Description	Reference
TBA1	SFC Active OAM	This document

Table 2: SFC Active OAM Protocol

10.2. SFC Active OAM

IANA is requested to create a new SFC Active OAM registry.

10.2.1. SFC Active OAM Message Type

IANA is requested to create in the SFC Active OAM registry a new sub-registry as follows:

Sub-registry Name: SFC Active OAM Message Type.

Assignment Policy:

2-32767 IETF Consensus

32768-65530 First Come First Served

Reference: [this document]

Value	Description	Reference
0	Reserved	This document
1	SFC Echo Request/Echo Reply	This document
2 - 32767	Unassigned	This document
32768 - 65530	Unassigned	This document
65531 - 65534	Unassigned	This document
65535	Reserved	This document

Table 3: SFC Active OAM Message Type

10.2.2. SFC Active OAM Header Flags

IANA is requested to create in the SFC Active OAM registry the new sub-registry SFC Active OAM Flags.

This sub-registry tracks the assignment of 8 flags in the Flags field of the SFC Active OAM Header. The flags are numbered from 0 (most significant bit, transmitted first) to 7.

New entries are assigned by Standards Action.

Bit Number	Description	Reference
7-0	Unassigned	This document

Table 4: SFC Active OAM Header Flags

10.3. SFC Echo Request/Echo Reply Parameters

IANA is requested to create a new SFC Echo Request/Echo Reply Parameters registry.

10.3.1. SFC Echo Request Flags

IANA is requested to create in the SFC Echo Request/Echo Reply Parameters registry the new sub-registry SFC Echo Request Flags.

This sub-registry tracks the assignment of 16 flags in the SFC Echo Request Flags field of the SFC Echo Request message. The flags are numbered from 0 (most significant bit, transmitted first) to 15.

New entries are assigned by Standards Action.

Bit Number	Description	Reference
15-0	Unassigned	This document

Table 5: SFC Echo Request Flags

10.3.2. SFC Echo Request/Echo Reply Message Types

IANA is requested to create in the SFC Echo Request/Echo Reply Parameters registry the new sub-registry as follows:

Sub-registry Name: Message Types

Assignment Policy:

5 - 175 IETF Consensus

176 - 239 First Come First Served

240 - 251 Experimental

252 - 254 Private Use

Reference: [this document]

Value	Description	Reference
0	Reserved	This document
1	SFC Echo Request	This document
2	SFC Echo Reply	This document
3	SFP Consistency Verification Request	This document
4	SFP Consistency Verification Reply	This document
5 - 175	Unassigned	This document
176 - 239	Unassigned	This document
240 - 251	Unassigned	This document
252 - 254	Unassigned	This document
255	Reserved	This document

Table 6: SFC Echo Request/Echo Reply Message Types

10.3.3. SFC Echo Reply Modes

IANA is requested to create in the SFC Echo Request/Echo Reply Parameters registry the new sub-registry as follows:

Sub-registry Name: Reply Mode

Assignment Policy:

8 - 175 IETF Consensus

176 - 239 First Come First Served

240 - 251 Experimental

252 - 254 Private Use

Reference: [this document]

Value	Description	Reference
0	Reserved	This document
1	Do Not Reply	This document
2	Reply via an IPv4/IPv6 UDP Packet	This document
3	Reply via Application-Level Control Channel	This document
4	Reply via Specified Path	This document
5	Reply via an IPv4/IPv6 UDP Packet with the data integrity protection	This document
6	Reply via Application-Level Control Channel with the data integrity protection	This document
7	Reply via Specified Path with the data integrity protection	This document
8 - 175	Unassigned	IETF Review
176 - 239	Unassigned	First Come First Served
240 - 251	Unassigned	Experimental
252 - 254	Unassigned	Private Use
255	Reserved	This document

Table 7: SFC Echo Reply Mode

10.3.4. SFC Echo Return Codes

IANA is requested to create in the SFC Echo Request/Echo Reply Parameters registry the new sub-registry as follows:

Sub-registry Name: Return Codes

Assignment Policy:

9 - 191 IETF Consensus

192 - 251 First Come First Served

252 - 254 Private Use

Reference: [this document]

Value	Description	Reference
0	No Return Code	This document
1	Malformed Echo Request received	This document
2	One or more of the TLVs was not understood	This document
3	Authentication failed	This document
4	TTL Exceeded	This document
5	End of the SFP	This document
6	Reply Path TLV is missing	This document
7	Reply SFP was not found	This document
8	Unverifiable Reply Path	This document
9 -191	Unassigned	This document
192-251	Unassigned	This document
252-254	Unassigned	This document
255	Reserved	

Table 8: SFC Echo Return Codes

10.4. SFC Active OAM TLV Type

IANA is requested to create the new registry as follows:

Registry Name: SFC Active OAM TLV Type

Assignment Policy:

6 -175 IETF Consensus

176 - 239 First Come First Served

240 - 251 Experimental

252 - 254 Private Use

Reference: [this document]

Value	Description	Reference
0	Reserved	This document
1	Source ID TLV	This document
2	Errored TLVs	This document
3	SFC Reply Path Type	This document
4	SFF Information Record Type	This document
5	SF Information	This document
6 - 175	Unassigned	This document
176 - 239	Unassigned	This document
240 - 251	Unassigned	This document
252 - 254	Unassigned	This document
255	Reserved	This document

Table 9: SFC Active OAM TLV Type Registry

10.5. SF Identifier Types

IANA is requested to create in the SF Types registry the new sub-registry as follows:

Registry Name: SF Identifier Types

Assignment Policy:

4 -191 IETF Consensus

192 - 251 First Come First Served

252 - 254 Private Use

Reference: [this document]

Value	Description	Reference
0	Reserved	This document
1	IPv4	This document
2	IPv6	This document
3	MAC	This document
4 -191	Unassigned	This document
192-251	Unassigned	This document
252-254	Unassigned	This document
255	Reserved	This document

Table 10: SF Identifier Type

11. References

11.1. Normative References

- [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-00, 25 March 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-sfc-oam-packet-00>>.
- [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>>.

11.2. Informative References

- [I-D.ietf-sfc-nsh-tlv] Wei, Y., Elzur, U., Majee, S., Pignataro, C., and D. E. Eastlake, "Network Service Header Metadata Type 2 Variable-Length Context Headers", Work in Progress, Internet-Draft, draft-ietf-sfc-nsh-tlv-14, 30 March 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-sfc-nsh-tlv-14>>.
- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<https://www.rfc-editor.org/info/rfc792>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, DOI 10.17487/RFC5706, November 2009, <<https://www.rfc-editor.org/info/rfc5706>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<https://www.rfc-editor.org/info/rfc6437>>.

- [RFC7110] Chen, M., Cao, W., Ning, S., Jounay, F., and S. Delord, "Return Path Specified Label Switched Path (LSP) Ping", RFC 7110, DOI 10.17487/RFC7110, January 2014, <<https://www.rfc-editor.org/info/rfc7110>>.
- [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>>.
- [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>>.
- [RFC8595] Farrel, A., Bryant, S., and J. Drake, "An MPLS-Based Forwarding Plane for Service Function Chaining", RFC 8595, DOI 10.17487/RFC8595, June 2019, <<https://www.rfc-editor.org/info/rfc8595>>.
- [RFC8924] Aldrin, S., Pignataro, C., Ed., Kumar, N., Ed., Krishnan, R., and A. Ghanwani, "Service Function Chaining (SFC) Operations, Administration, and Maintenance (OAM) Framework", RFC 8924, DOI 10.17487/RFC8924, October 2020, <<https://www.rfc-editor.org/info/rfc8924>>.
- [RFC9015] Farrel, A., Drake, J., Rosen, E., Uttaro, J., and L. Jalil, "BGP Control Plane for the Network Service Header in Service Function Chaining", RFC 9015, DOI 10.17487/RFC9015, June 2021, <<https://www.rfc-editor.org/info/rfc9015>>.
- [RFC9145] Boucadair, M., Reddy, K. T., and D. Wing, "Integrity Protection for the Network Service Header (NSH) and Encryption of Sensitive Context Headers", RFC 9145, DOI 10.17487/RFC9145, December 2021, <<https://www.rfc-editor.org/info/rfc9145>>.

Contributors' Addresses

Cui Wang
Individual contributor
Email: lindawangjoy@gmail.com

Zhonghua Chen
China Telecom
No.1835, South PuDong Road
Shanghai
201203
China
Phone: +86 18918588897
Email: chenzhongh@chinatelecom.cn

Authors' Addresses

Greg Mirsky
Ericsson
Email: gregimirsky@gmail.com

Wei Meng
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing,
China
Email: meng.wei2@zte.com.cn

Bhumip Khasnabish
Individual contributor
Email: vumipl@gmail.com

Ting Ao
China Mobile
No.889, BiBo Road
Shanghai
201203
China
Phone: +86 17721209283
Email: 18555817@qq.com

Kent Leung
Cisco System
170 West Tasman Drive
San Jose, CA 95134,
United States of America
Email: kleung@cisco.com

Gyan Mishra
Verizon Inc.
Email: gyan.s.mishra@verizon.com

INTERNET-DRAFT
Intended status: Proposed Standard

Donald Eastlake
Huawei
Bob Briscoe
Independent
Andrew Malis
Huawei
February 7, 2019

Expires: August 6, 2019

Explicit Congestion Notification (ECN) and Congestion Feedback
Using the Network Service Header (NSH)
<draft-ietf-sfc-nsh-ecn-support-00.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.

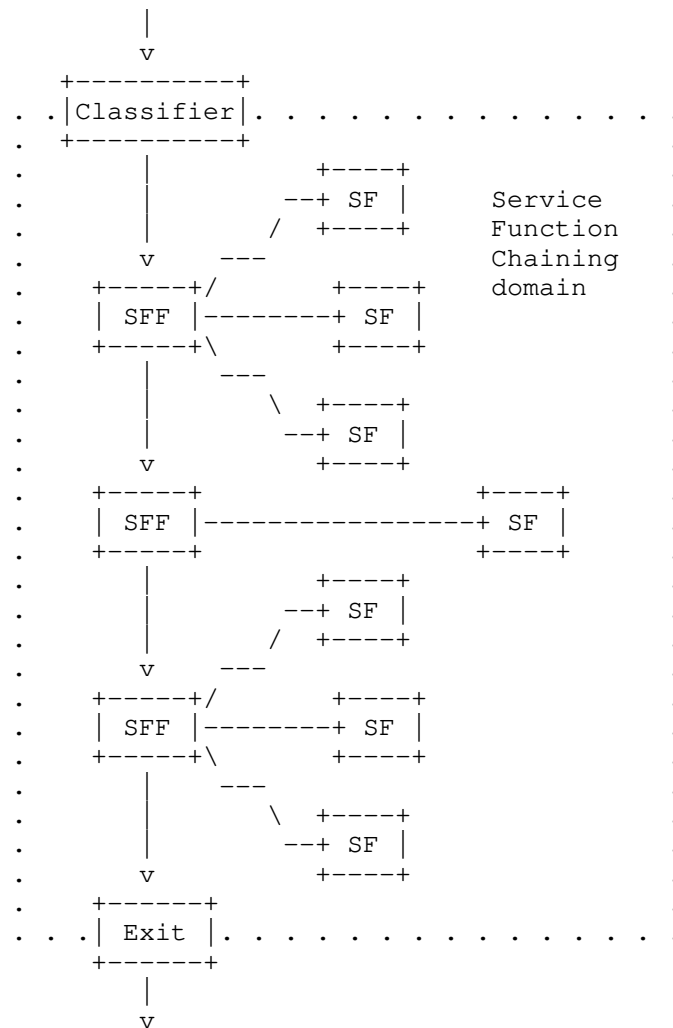


Figure 1. Example SFC Path Forwarding Nodes

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.

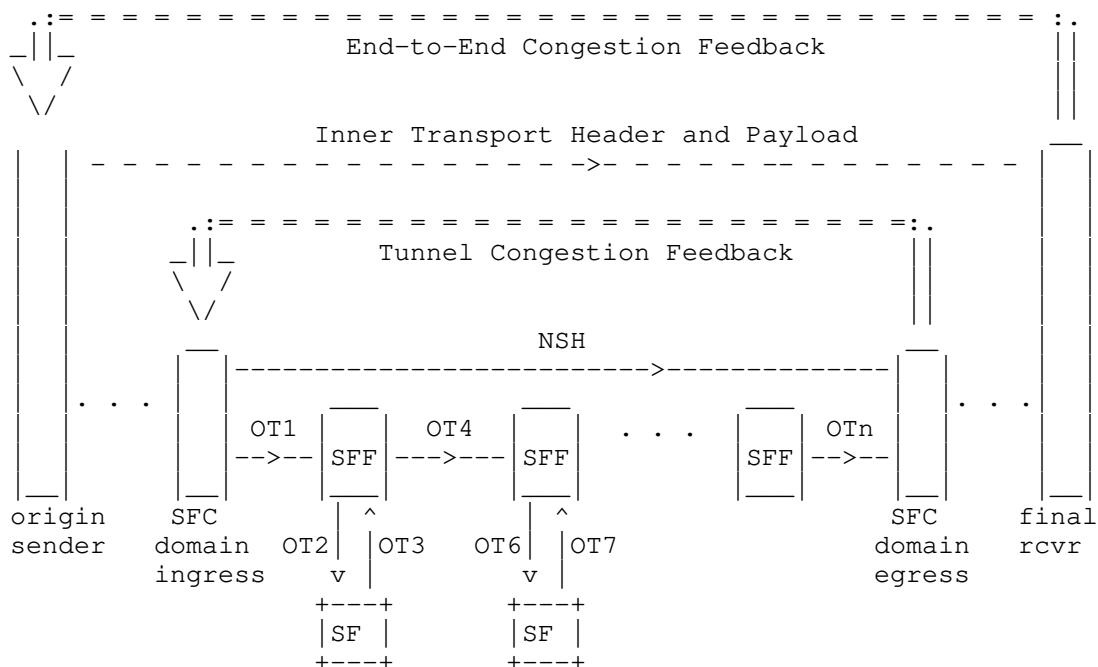


Figure 2: Congestion Feedback across an SFC Domain

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.

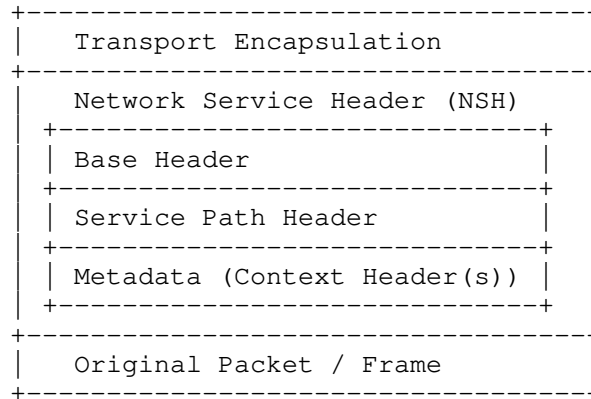


Figure 3. Data Encapsulation with the NSH

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.

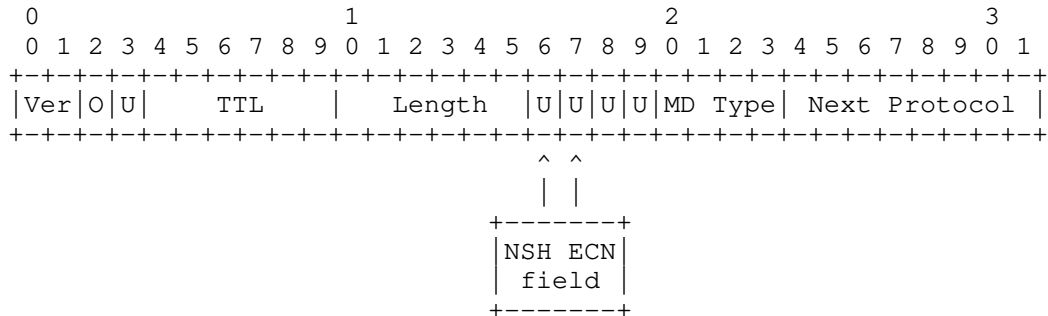


Figure 4: NSH Base Header

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 (also equal to departing Inner Header)	Departing NSH and Outer 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.

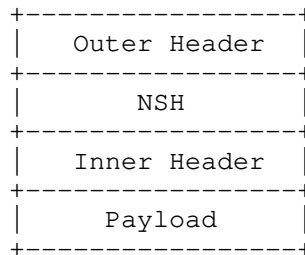


Figure 5. Packet in Transit

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 Inner 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.

INTERNET-DRAFT
Intended status: Proposed Standard

D. Eastlake
Futurewei Technologies
B. Briscoe
Independent
Y. Li
Huawei Technologies
A. Malis
Malis Consulting
X. Wei
Huawei Technologies
April 17, 2022

Expires: October 16, 2022

Explicit Congestion Notification (ECN) and Congestion Feedback
Using the Network Service Header (NSH) and IPFIX
<draft-ietf-sfc-nsh-ecn-support-09.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 information about congestion back 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) enabled domain through use of the Network Service Header (NSH, RFC 8300) and IP Flow Information Export (IPFIX, RFC 7011).

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

Table of Contents

1. Introduction.....	4
1.1 NSH Background.....	4
1.2 ECN Background.....	6
1.3 Tunnel Congestion Feedback Background.....	6
1.4 Conventions Used in This Document.....	8
2. The NSH ECN Field.....	10
3. ECN Support in the NSH.....	12
3.1 At The Ingress.....	13
3.2 At Transit Nodes.....	14
3.2.1 At NSH Transit Nodes.....	14
3.2.2 At an SF/Proxy.....	15
3.2.3 At Other Forwarding Nodes.....	15
3.3 At Exit/Egress.....	16
3.4 Congestion Statistics and the Conservation of Packets.....	16
4. Tunnel Congestion Feedback Support.....	18
4.1 Congestion Level Measurements.....	18
4.3 Congestion Information Delivery.....	19
4.3 IPFIX Extensions.....	21
4.3.1 nshServicePathID.....	21
4.3.2 tunnelEcnCeCeByteTotalCount.....	22
4.3.3 tunnelEcnEctNectBytetTotalCount.....	22
4.3.4 tunnelEcnCeNectByteTotalCount.....	22
4.3.5 tunnelEcnCeEctByteTotalCount.....	23
4.3.6 tunnelEcnEctEctByteTotalCount.....	23
4.3.7 tunnelEcnCEMarkedRatio.....	24
5. Example of Use.....	25
6. IANA Considerations.....	28
6.1 SFC NSH Header ECN Bits.....	28
6.2 IPFIX Information Element IDs.....	28
7. Security Considerations.....	30
8. Acknowledgements.....	30
Normative References.....	31
Informative References.....	32
Authors' Addresses.....	33

1. Introduction

Explicit Congestion Notification (ECN [RFC3168]) allows a forwarding element to notify downstream nodes of the onset of congestion without having to drop packets. Coupled with a means to feed information about congestion back 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]) enabled domain through use of the Network Service Header (NSH [RFC8300]) and IP Flow Information Export (IPFIX [RFC7011]).

This document 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 enabled domain implement ECN, some benefit is obtained even if some interior nodes do not implement ECN. Congestion at any interior bottleneck where ECN marking is not implemented will be unmanaged.

The following subsections 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 reasonable place, in a domain where traffic is NSH encapsulated, to accumulate congestion information.

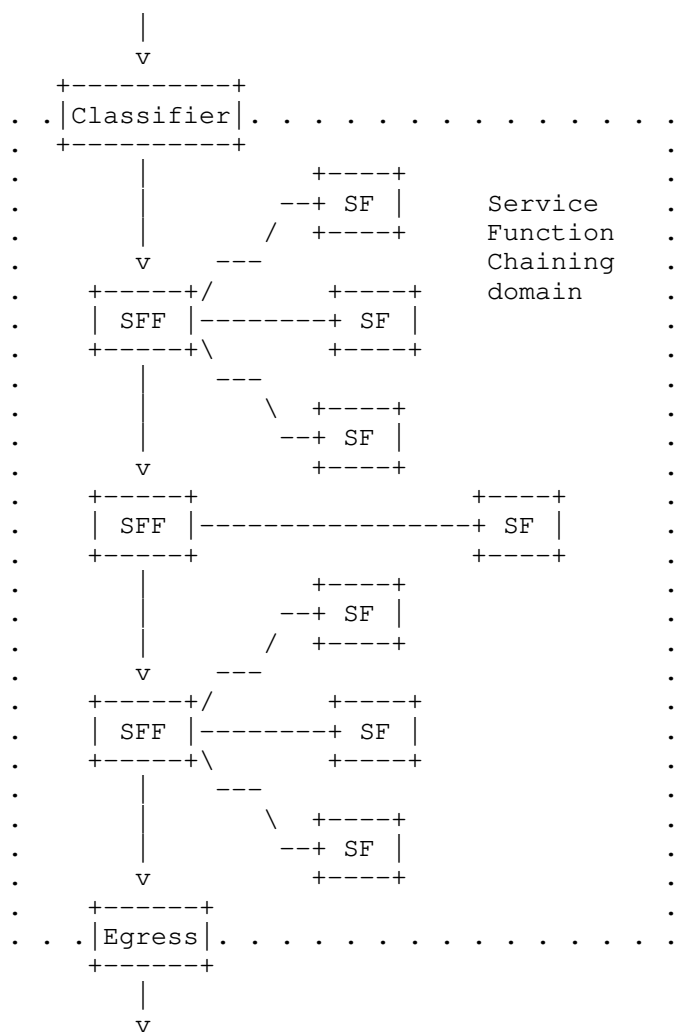


Figure 1. Example SFC Forwarding Nodes Path

Figure 1 shows an SFC enabled domain for the purpose of illustrating the use of the 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 firewalling or Network Address Translation (NAT) or load balancing, and then returns the traffic 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 (see Figure 3), the SFF decides on the next forwarding step, either adding a new outer

transport header or, if the SFF is the exit/egress, removing the NSH header. The outer transport headers added may be different in different regions of the SFC enabled domain. For example, IP could be used for some SFF-to-SFF communication and MPLS used for other SFF-to-SFF communication.

1.2 ECN Background

Explicit Congestion Notification (ECN [RFC3168]) allows a forwarding element (such as a router or a Service Function Forwarder (SFF) or Service Function (SF)) to notify downstream nodes 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

Tunnels are widely deployed in various networks including data center networks, enterprise network, and the public Internet. A tunnel consists of ingress, egress, and a set of intermediate nodes including routers. Tunnel Congestion Feedback (Section 4) is a building block for congestion mitigation methods. It supports feedback of congestion information from an egress node to an ingress node. This document treats the SFC enabled domain as a tunnel with the initial Classifier node being the ingress; however, the tunnel congestion feedback facilities specified in this document MAY be used in contexts other than SFC.

Any action by a tunnel ingress to reduce congestion needs to allow sufficient time for the end-to-end congestion control loop to respond first, otherwise the system could go unstable. For instance by the ingress taking a smoothed average of the level of congestion signaled by feedback from the tunnel egress or delaying any action for at least the worst case end-to-end round-trip time (for example, 200 milliseconds).

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 of some future traffic so that it avoids congested paths. Great care must be taken with this option to avoid (a) significant re-ordering of traffic in flows that it is desirable to keep in order (due to end-to-end requirement or due to a stateful SF) 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 to use a stable path per flow, sending only newly appearing flows on apparently uncongested paths.

Figure 2 shows an example path from an original sender to a final receiver passing through a chain of service functions between the ingress and egress of an SFC enabled domain. The path is also likely to pass through other network nodes outside the SFC enabled domain (not shown) before entering that domain and after leaving that domain.

Figure 2 shows typical congestion feedback that would be expected from the final receiver to the origin sender, which controls the load the origin sender directs to all elements on the path. The figure also shows the congestion feedback from the egress to the ingress of the SFC enabled domain that is described in this document, to control or balance load within that domain.

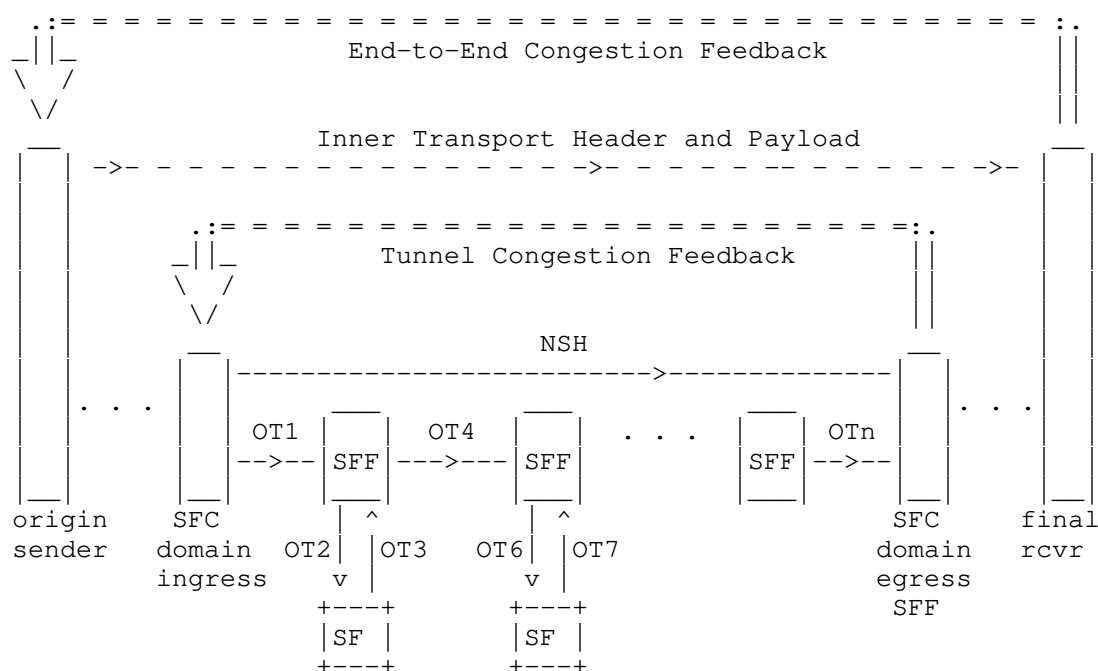


Figure 2. Congestion Feedback across an SFC enabled Domain

SFC enabled 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", "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.

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

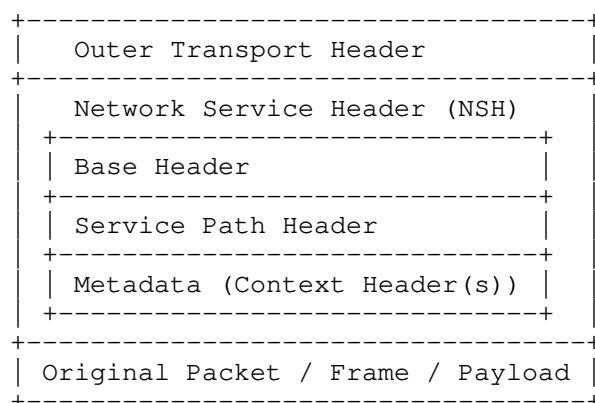


Figure 3. Data Encapsulation with the NSH

This document assigns two currently unused bits (indicated by "U") in the NSH Base Header (Section 2.2 of [RFC8300]) for the purpose of ECN indication as shown in Figure 4.

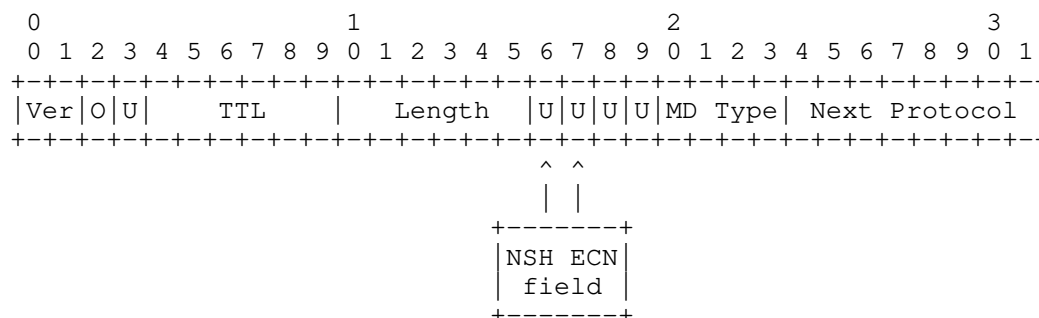


Figure 4. Updated NSH Base Header

RFC Editor NOTE: The above figure should be adjusted based on the bits actually 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 Section 23.1 of [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 ECN-unaware, 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 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 the full functionality mode of [RFC3168]).

The procedures in Section 3.2.1 ensure that each ingress of the transport links within the SFC enabled 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 enabled 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 Section 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 packet with an NSH, it MUST set the NSH ECN field using the "Normal mode" specified in [RFC6040] (e.g., 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 enabled domain are acting as an ECN-capable transport. This approach supports 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, for example MPLS [RFC5129], 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 (also equal to departing Inner Header)	Departing NSH and Outer 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 an NSH is being used to steer traffic.

3.2 At Transit Nodes

This section describes the 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.

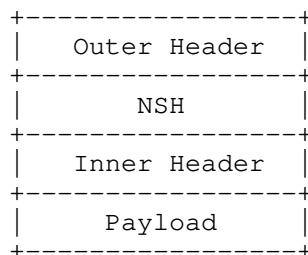


Figure 5. Packet in Transit

3.2.1 At NSH Transit Nodes

When a packet is received at an NSH based forwarding node such as an SFF, say N1, 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 previous 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 encapsulating node propagates the NSH ECN field to this outer encapsulation using the "Normal Mode" of ECN encapsulation [RFC6040] (the ECN field is copied). If it does not, then the encapsulating node 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 (except in the case where the SF terminates the packets path).

If the SF is NSH-aware but ECN-unaware, 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 SF that does not understand NSHs to the NSH as shown in Figure 6. 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.

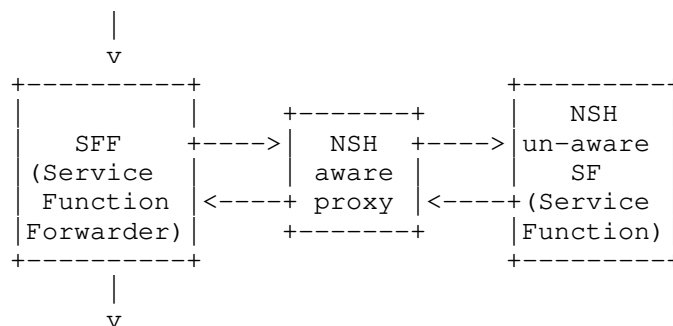


Figure 6. Proxy for NSH Un-aware SFF

If the proxy is ECN-aware, the proxy uses an AQM to indicate congestion within the proxy in the NSH that it returns to the SFF. The outer header used for the proxy-to-SF path uses Normal Mode. The outer header used for the proxy-to-SFF path uses Normal Mode based copying of 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 and implements 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 or label switched routers, might also

contain 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

At the SFC enabled domain egress node, first any actions are taken based on Congestion Experienced or other values of ECN marking, such as accumulating statistics to send back to the ingress (see Section 4) or for other uses. If the packet being carried inside the NSH is IP, when the NSH is removed the NSH ECN field MUST be combined with the IP ECN field as specified in Table 3 that was extracted from Section 3.2 of [RFC6040]. This requirement applies to all egress nodes for the domain in which NSH is being used to route traffic.

Arriving Inner 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 (Source [RFC6040])

All the egress nodes of the SFC enabled 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 Congestion Statistics and the Conservation of Packets

The SFC specification permits an SF to absorb packets and to generate new packets as well as simply processing and returning the packets it receives to an SFF. 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 (Section 4) can detect congestions in several ways. One way detects traffic loss by counting payload packets and bytes in at the ingress and counting them out at the egress. This does not work unless nodes conserve the number of

payload packets and/or bytes. Therefore, it will not be possible to accurately detect packet loss using this technique if traffic volume, as measured by the metric in used (packets or bytes), is not conserved by the service function chain processing that traffic.

Nonetheless, if a bottleneck supports ECN marking, it will be possible to detect the high level of CE markings that are associated with congestion at that bottleneck by looking at the ratio of CE-marked to non-CE-marked packets. However, it will not be possible to detect any congestion based on ECN marking, 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 at the egress may be useful for later analysis and MAY be used without the feedback mechanisms specified in this Section. However, if congestion information is not 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 this document, IPFIX messages from the egress to the ingress are used to communicate the extent of congestion between an ingress and egress based on ECN marking in the NSH. The ingress MUST be configured to know the relevant egress for a flow. The egress MUST be able to identify the relevant ingress for a packet based on the SPI, the Ingress Network Node Information Context Header [NSHTLV], or the like.

4.1 Congestion Level Measurements

The congestion level measurements are based on ECN marking in the NSH and packet drop. In particular congestion information includes at least one of cumulative bytes counts of packets with each type of outer/inner header ECN marking combination, the ratio of CE-marked packets to all packets, and the ratio of dropped packets to all packets.

If the congestion level is low enough, the packets are marked as CE instead of being dropped, and then it is easy to calculate congestion level according to the ratio of CE-marked packets. If the congestion level is so high that ECT packets will be dropped, then the packet loss ratio could be calculated by comparing total packets entering ingress and total packets arriving at egress over the same span of packets. If packet loss is detected for a flow that would preserve the number of packets in the absence of congestion, then it can be assumed that severe congestion has occurred in the tunnel.

The egress calculates the CE-marked packet ratio by counting packets with different ECN markings. The CE-marked packet ratio will be used as an indication of tunnel load level. It is assumed that nodes between the ingress and egress will not drop packets biased towards certain ECN codepoints, so calculating of CE-marked packet ratio is not affected by packet drop.

The calculation of the fraction of packets dropped is by comparing the traffic volumes between ingress and egress.

Faked ECN-Capable Transport (ECT) is used at the ingress to defer packet loss to the egress. The basic idea of faked ECT is that, when encapsulating packets, the ingress first marks the tunnel outer header according to [RFC6040], and then remarks the outer header of Not-ECT packets as ECT. (ECT(0) and ECT(1) are treated as the same.) In this case, the NSH is treated as the tunnel outer header because it will be present for the entire SFC enabled domain transit while transport headers may change. Thus, as transmitted by the ingress node, there will be one of three combinations of outer header ECN field and inner header ECN field as follows: CE|CE, ECT|N-ECT, and ECT|ECT (in the format of outer-ECN|inner-ECN); when decapsulating packets at the egress, [RFC6040] defined decapsulation behavior is used, and according to [RFC6040], the packets marked as CE|N-ECT will be dropped. Faked-ECT is used to shift some drops to the egress in order to allow the egress to calculate the CE-marked packet ratio more precisely.

The ingress encapsulates packets and marks their outer header according to faked ECT as described above. The ingress cumulatively counts packet bytes for three types of ECN combination (CE|CE, ECT|N-ECT, and ECT|ECT) and then the ingress regularly sends cumulative bytes counts message of each type of ECN combination to the egress.

When each message arrives at the egress, the following two steps occur: (1) the egress calculates the ratio of CE-marked packets; (2) the egress cumulatively counts packet bytes coming from the ingress and adds its own bytes counts of each type of ECN combination (CE|CE, ECT|N-ECT, CE|N-ECT, CE|ECT, and ECT|ECT) to the message for the ingress to calculate packet loss. The egress feeds back the CE-marked packet ratio, packet loss ratio, bytes counts information, and the like to the ingress as requested for evaluating congestion level in the tunnel.

The statistics can be at the granularity of all traffic from the ingress to the egress to learn about the overall congestion status of the path between the ingress and the egress or at the granularity of individual customer's traffic or a specific set of flows to learn about their congestion contribution.

For example, the tunnelEcnCEMarkedRatio field (specified below) indicates the fraction of traffic that has been marked in the ECN field of the NSH as Congestion Experienced (CE).

4.3 Congestion Information Delivery

As described above, the tunnel ingress sends a message containing cumulative byte counts of packets of each type of ECN marking to the tunnel egress, and the tunnel egress feeds back messages to the

ingress with at least one of the following: cumulative byte counts of packets of each type of ECN combination, the ratio of CE-marked packets to all packets, and the ratio of dropped packets to all packets. (It is possible for these messages to contribute to congestion.) This section specifies how the messages are conveyed.

IPFIX recommends, but does not require, use of SCTP [RFC4960] in partial reliability mode [RFC3758] for the transport of its messages. This mode allows loss of some packets, which is tolerable because IPFIX communicates cumulative statistics. IPFIX over SCTP over IP 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 enabled domain that block such direct IP connectivity. The NSH provides the general method of routing traffic within an SFC enabled domain so the encapsulation of the required IPFIX traffic in NSH MUST be implemented and, when IP connectivity is not available, IPFIX over NSH SHOULD be used along with configuration of appropriate SFC paths for the IPFIX over NSH traffic.

IPFIX messages could travel along the same path as network data traffic. In any case, an IPFIX message packet may get lost in case of network congestion. Even though the missing information could be recovered because of the use of cumulative counts, the message SHOULD be transmitted at a higher priority than users' traffic flows to improve the promptness of congestion information feedback.

The ingress node can do congestion management at different granularity which means both the overall aggregated inner tunnel congestion level and congestion level contributed by certain traffic flows could be measured for different congestion management purposes. For example, if the ingress only wants to limit congestion volume caused by certain traffic flows, such as UDP-based traffic, then congestion volume for that traffic can be fed back; or if the ingress is doing overall congestion management, the aggregated congestion volume can be fed back.

When sending IPFIX messages from ingress to egress, the ingress acts as IPFIX exporter and the egress acts as IPFIX collector; When feeding back congestion level information from egress to ingress, then the egress acts as IPFIX exporter and ingress acts as IPFIX collector.

The combination of congestion level measurement and congestion information delivery procedures are as following:

- o The ingress node determines the IPFIX template record to be used. The template record can be pre-configured or determined at runtime, the content of the template record will be determined according to the granularity of congestion management; if the

ingress wants to limit congestion volume contributed by specific traffic flows then the elements such as source IP address, destination IP address, flow ID, and CE-marked packet volume of the flows, etc., will be included in the template record.

- o Metering at the ingress measures traffic volume according to the template record chosen and then the measurement records are sent to the egress.
- o Metering on the egress measures congestion level information according to template record which SHOULD be the same as the template record sent by the ingress.
- o The egress sends its measurement records together with the measurement records of the ingress back to the ingress.

4.3 IPFIX Extensions

This section specifies the new IPFIX Information Elements needed. It conforms to [RFC7013].

4.3.1 nshServicePathID

In order to identify SFC flows, so that congestion can be measured and reported at that granularity, it is necessary for IPFIX to be able to classify traffic based on the Service Path Identifier field of the NSH [RFC8300]. Thus an NSH Service Path Identifier (nshServicePathID) IPFIX Information Element [RFC7012] is specified.

Name: nshServicePathID

Description: Network Service Header [RFC8300] Service Path Identifier. This is a 24-bit value which is left justified in the Information Element. The low order byte MUST be sent as zero and ignored on receipt.

Abstract Data Type: unsigned32

Data Type Semantics: identifier

ElementId: TBD0

Status: current

4.3.2 tunnelEcnCeCeByteTotalCount

Description: The total number of bytes of incoming packets with the CE|CE ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD1

Statuses: current

Units: bytes

4.3.3 tunnelEcnEctNectBytetTotalCount

Description: The total number of bytes of incoming packets with the ECT|N-ECT ECN marking combination (ECT(0) and ECT(1) are treated the same as each other) at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD2

Statuses: current

Units: bytes

4.3.4 tunnelEcnCeNectByteTotalCount

Description: The total number of bytes of incoming packets with the CE|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD3

Statuses: current

Units: bytes

4.3.5 tunnelEcnCeEctByteTotalCount

Description: The total number of bytes of incoming packets with the CE|ECT ECN marking combination (ECT(0) and ECT(1) are treated the same as each other) at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD4

Statuses: current

Units: bytes

4.3.6 tunnelEcnEctEctByteTotalCount

Description: The total number of bytes of incoming packets with the ECT|ECT ECN marking combination (ECT(0) and ECT(1) are treated the same as each other) at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD5

Statuses: current

Units: bytes

4.3.7 tunnelEcnCEMarkedRatio

Description: The ratio of packets that are CE-marked to packets that are not CE-marked at the Observation Point.

Abstract Data Type: float32

ElementId: TBD6

Statuses: current

5. Example of Use

This section provides an example of the solution described in this document.

First, IPFIX template records are exchanged between ingress and egress to negotiate the format of the data records to be exchanged. The example here is to measure the congestion level for the overall tunnel caused by all the traffic. After the negotiation is finished, the ingress sends in-band messages to the egress containing the number of each kind of ECN-marked packets (i.e., CE|CE, ECT|N-ECT and ECT|ECT) received before it sent the message.

After the egress receives the message, the egress calculates the CE-marked packet ratio and counts the number of different kinds of ECN-marking packets received before it received the message. Then the egress sends a feedback message containing the counts together with the information in the ingress's message back to the ingress.

Figures 7 to 10 below illustrate the example procedure between ingress and egress.

Set ID=2	Length=40
Template ID=256	Field Count=8
tunnelEcnCeCeByteTotalCount	Field Length=8
tunnelEcnEctNectByteTotalCount	Field Length=8
tunnelEcnEctEctByteTotalCount	Field Length=8
tunnelEcnCeNectByteTotalCount	Field Length=8
tunnelEcnCeEctByteTotalCount	Field Length=8
tunnelEcnCEMarkedRatio	Field Length=4

Figure 7. Template Record Sent From Egress to Ingress

Set ID=2	Length=28
Template ID=257	Field Count=3
tunnelEcnCeCeByteTotalCount	Field Length=8
tunnelEcnEctNectByteTotalCount	Field Length=8
tunnelEcnEctEctByteTotalCount	Field Length=8

Figure 8. Template Record Sent From Ingress to Egress

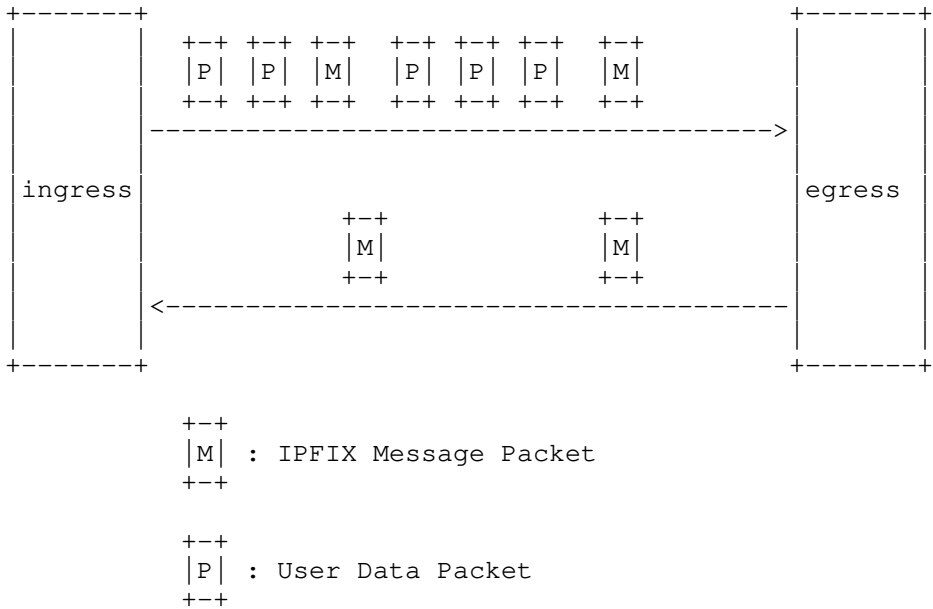


Figure 9. Traffic flow Between Ingress and Egress

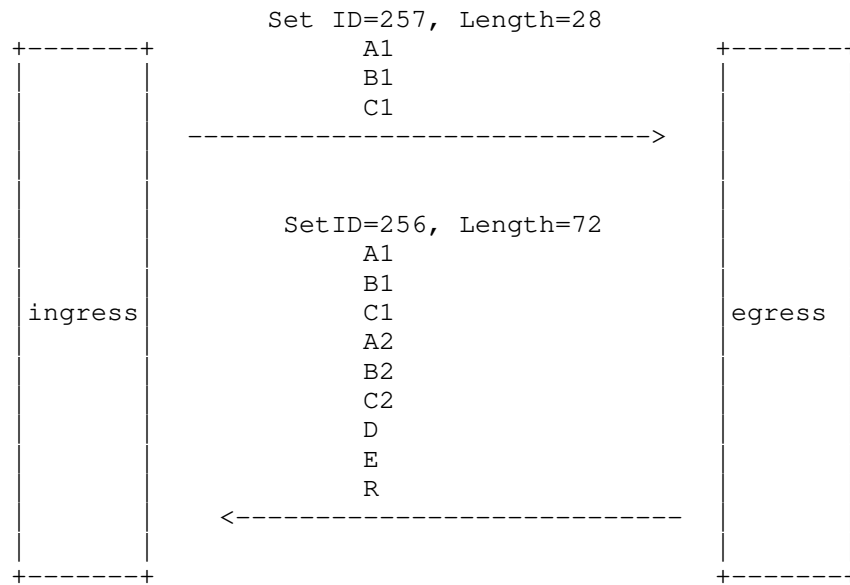


Figure 10. Messages Between Ingress and Egress

The following provides an example of how the tunnel congestion level can be calculated (see Figure 10):

The congestion Level could be divided into two categories: (1) slight congestion (no packets dropped); (2) serious congestion (packets are being dropped).

For slight congestion, the congestion level is indicated by the ratio of CE-marked packets:

$$R = \text{ce_marked_ratio} = \text{ce-marked} / \text{total_egress} ;$$

For serious congestion, the congestion level is indicated as the volume of traffic loss:

$$\text{total_ingress} = (A1 + B1 + C1)$$

$$\text{total_egress} = (A2 + B2 + C2 + D + E)$$

$$\text{volume_loss} = (\text{total_ingress} - \text{total_egress})$$

6. IANA Considerations

The following subsections provide IANA assignment considerations.

6.1 SFC NSH Header ECN Bits

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]

6.2 IPFIX Information Element IDs

IANA is requested to assign seven IPFIX Information Element IDs as follows:

ElementID: TBD0
Name: nshServicePathID
Data Type: unsigned32
Data Type Semantics: identifier
Status: current
Description: The Network Service Header [RFC8300] Service Path Identifier.

ElementID: TBD1
Name: tunnelEcnCeCePacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description: The total number of bytes of incoming packets with the CE|CE ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Units: octets

ElementID: TBD2
Name: tunnelEcnEctNectPacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description: The total number of bytes of incoming packets with the ECT|N-ECT ECN marking combination at the Observation Point since the Metering Process (re-)initialization for this

Observation Point.
Units: octets

ElementID: TBD3
Name: tunnelEcnCeNectPacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description: The total number of bytes of incoming packets with
the CE|N-ECT ECN marking combination at the Observation Point
since the Metering Process (re-)initialization for this
Observation Point.
Units: octets

ElementID: TBD4
Name: tunnelEcnCeEctPacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description: The total number of bytes of incoming packets with
the CE|ECT ECN marking combination at the Observation Point
since the Metering Process (re-)initialization for this
Observation Point.
Units: octets

ElementID: TBD5
Name: tunnelEcnEctEctPacketTotalCount
Data Type: unsigned64
Data Type Semantics: totalCounter
Status: current
Description: The total number of bytes of incoming packets with
the CE|ECT(0) ECN marking combination at the Observation Point
since the Metering Process (re-)initialization for this
Observation Point.
Units: octets

ElementID: TBD6
Name: tunnelEcnCEMarkedRatio
Data Type: float32
Status: current
Description: The ratio of CE-marked Packet at the Observation
Point.

7. Security Considerations

For general NSH security considerations, see [RFC8300].

For security considerations concerning ECN signaling tampering, see [RFC3168]. For security considerations concerning ECN and 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 tunnel endpoints (the ingress and egress for an SFC enabled domain) are assumed to be in the same administrative domain, so they will trust each other.

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

8. Acknowledgements

Most of the material on Tunnel Congestion Feedback was originally in draft-ietf-tsvwg-tunnel-congestion-feedback. After discussion with the authors of that draft, the authors of this draft, and the Chairs of the TSVWG and SFC Working Groups, the Tunnel Congestion Feedback draft was merged into this draft.

The authors wish to thank the following for their comments, suggestions, and reviews:

David Black, Mohamed Boucadair, Sami Boutros, Anthony Chan,
Lingli Deng, Liang Geng, Joel Halpern, Jake Holland,
John Kaippallimalil, Tal Mizrahi, Vincent Roca, Lei Zhu

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>>.
- [RFC3758] - Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <<https://www.rfc-editor.org/info/rfc3758>>.
- [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>>.
- [RFC7013] - Trammell, B. and B. Claise, "Guidelines for Authors and Reviewers of IP Flow Information Export (IPFIX) Information Elements", BCP 184, RFC 7013, DOI 10.17487/RFC7013, September 2013, <<https://www.rfc-editor.org/info/rfc7013>>.
- [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>>.

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>>.
- [RFC7012] - Claise, B., Ed., and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <<https://www.rfc-editor.org/info/rfc7012>>.
- [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.
- [NSHTLV] - Wei, Y., U. Elzur, S. Majee, C. Pignataro, and D. Eastlake, "Network Service Header Metadata Type 2 Variable-Length Context Headers" , draft-ietf-sfc-nsh-tlv, work in progress.

Authors' Addresses

Donald E. Eastlake, 3rd
Futurewei Technologies
2386 Panoramic Circle
Apopka, FL 32703 USA

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

Bob Briscoe
Independent
UK

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

Yizhou Li
Huawei Technologies
101 Software Avenue,
Nanjing 210012, P. R China

Phone: +86-25-56624584
EMail: liyizhou@huawei.com

Andrew G. Malis
Malis Consulting

Email: agmalis@gmail.com

Xinpeng Wei
Huawei Technologies
Beiqing Rd. Z-park No.156, Haidian District,
Beijing, 100095, P. R. China

EMail: weixinpeng@huawei.com

Copyright and IPR Provisions

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 (<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 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.

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 12, 2019

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

T. Mizrahi
Huawei Network.IO Innovation Lab
A. Aguado
Universidad Politecnica de Madrid
D. Lopez
Telefonica I+D
March 11, 2019

Proof of Transit
draft-ietf-sfc-proof-of-transit-02

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 September 12, 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 (<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	6
3.2. Solution Approach	7
3.2.1. Setup	7
3.2.2. In Transit	7
3.2.3. Verification	8
3.3. Illustrative Example	8
3.3.1. Baseline	8
3.3.1.1. Secret Shares	8
3.3.1.2. Lagrange Polynomials	9
3.3.1.3. LPC Computation	9
3.3.1.4. Reconstruction	9
3.3.1.5. Verification	10
3.3.2. Complete Solution	10
3.3.2.1. Random Polynomial	10
3.3.2.2. Reconstruction	10
3.3.2.3. Verification	11
3.3.3. Solution Deployment Considerations	11
3.4. Operational Aspects	12
3.5. Ordered POT (OPOT)	12
4. Sizing the Data for Proof of Transit	13
5. Node Configuration	14
5.1. Procedure	15
5.2. YANG Model	15

6. IANA Considerations	18
7. Manageability Considerations	18
8. Security Considerations	19
8.1. Proof of Transit	19
8.2. Cryptanalysis	20
8.3. Anti-Replay	20
8.4. Anti-Preplay	21
8.5. Tampering	21
8.6. Recycling	21
8.7. Redundant Nodes and Failover	22
8.8. Controller Operation	22
8.9. Verification Scope	22
8.9.1. Node Ordering	23
8.9.2. Stealth Nodes	23
9. Acknowledgements	23
10. References	23
10.1. Normative References	23
10.2. Informative References	24
Authors' Addresses	24

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 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.

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:

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 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" 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 POT information is encapsulated in packets as an IOAM Proof Of Transit Option. The details and format of the encapsulation and the POT Option format are specified in [I-D.ietf-ippm-ioam-data].

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 $POLY-3 = POLY-1 + POLY-2$. 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 SSSS algorithm [SSS]. The non-constant coefficients are used to generate the Lagrange Polynomial Constants (LPC). 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 $\text{CML} = \text{SECRET} + \text{RND}$. 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. 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. $\text{POLY-1}(x) = (3x^2 + 3x + 10) \bmod 53$. 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 = 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 $x_0=2$, $x_1=4$, $x_2=5$.

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

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

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

The three points above are the points on the curve which are considered the shares of the secret. They are assigned 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 Constant (LPC) would be $10/3$, -5 , $8/3$. 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

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

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

$$\text{SECRET} = (y_0 * \text{LPC}(10) + y_1 * \text{LPC}(11) + y_2 * \text{LPC}(12)) \bmod 53$$

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

$$\begin{aligned} \text{SECRET} &= (y_0 * \text{LPC}(10) + y_1 * \text{LPC}(11) + y_2 * \text{LPC}(12)) \bmod 53 = \bmod (28 * 21 \\ &+ 17 * 48 + 47 * 38) \bmod 53 = 3190 \bmod 53 = 10 \end{aligned}$$

One observes that the secret reconstruction can easily be performed cumulatively hop by hop, 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 8 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 assumes 64 octet packets

Table 1: Proof of transit data sizing

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 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. 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

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

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.

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.

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.

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). The application of symmetric masking for OPOT makes inter-node analysis less feasible.
- 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. 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.

8.3. 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.

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 preplay attacks. A mitigation mechanism may be included in future versions of the solution.

8.5. 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.

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).

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.

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.

If OPOT (Section 3.5) is applied using symmetric masking, the Controller will be required to perform 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.

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, 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.

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

[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-04 (work in progress), October 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>>.
- [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>.

10.2. Informative References

- [I-D.ietf-anima-autonomic-control-plane] Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-18 (work in progress), August 2018.

Authors' Addresses

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

Email: 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

Sashank Dara
Seconize
BANGALORE, Bangalore, KARNATAKA
INDIA

Email: sashank@seconize.co

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

Email: 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
Huawei Network.IO Innovation Lab
Israel

Email: tal.mizrahi.phd@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

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

```
CML = CML + (((Share(POLY-1) + Share(POLY-2)) * LPC) mod 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 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

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