SFC WG
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2019

T. Ao
ZTE Corporation
G. Mirsky
ZTE Corp.
Z. Chen
China Telecom
Oct 16, 2018

SFC OAM for path consistency
draft-ao-sfc-oam-path-consistency-03

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, i.e. verification that all elements of the given SFC are being traversed in the expected order.

Table of Contents

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 traffic is processed by one SF then it should be processed by the next SF, otherwise errors may occur. Sometimes, a SF needs to use 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 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.wang-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 path consistency.

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 uses two functions: COAM Request and COAM Reply. The SFF, that is ingress of the SFP, transmits COAM Request packet. Every intermediate SFF that receives the COAM Request MUST perform the following actions:

- 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.wang-sfc-multi-layer-oam];

- 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.wang-sfc-multi-layer-oam] with the following values Section 5.1:

- TBA1 - COAM Request
- TBA2 - COAM Reply

An SFF, upon receiving the Consistency OAM Request, MUST include the corresponding SFs information, Section 3.2, into the Value field of the COAM Reply packet.

The COAM packet is displayed in Figure 1.

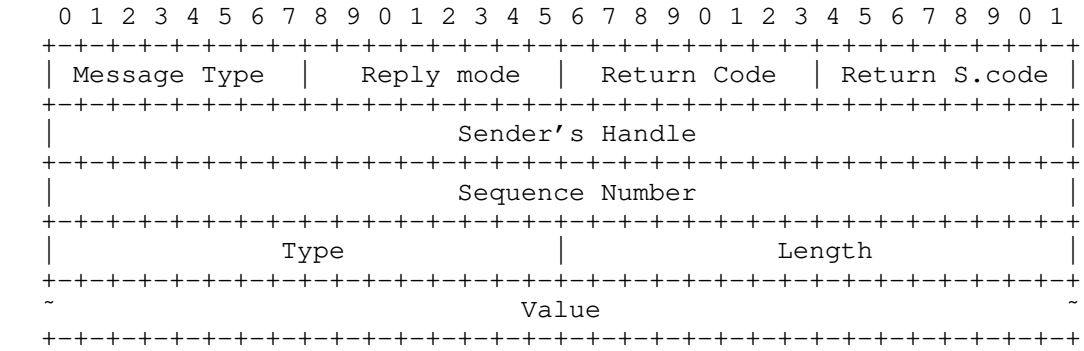


Figure 1: COAM Packet Header

3.2. SF information Sub-TLV

Every SFF receiving COAM Request packet MUST include the SF characteristic data into the COAM Reply packet. The per SF data included in COAM Reply packet as SF Information sub-TLV that is displayed in Figure 2.

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

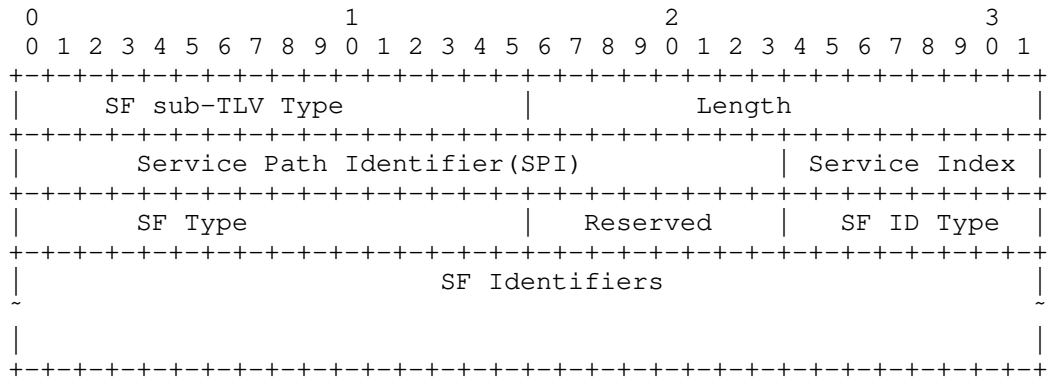


Figure 2: Service Function information sub-TLV

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

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

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

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

SF Type: is 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: is 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.

3.3. SF information Sub-TLV construction

For each SFF in the SFP, it should send one COAM Reply corresponding to each one 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 is constructed as the following two cases.

1. Multiple SFs as hops of SFP:

Multiple SFs attached to one SFF are the several hops of the SFP, the service indexes of these SFs are different. Service function types of these SFs could be different or be same. All these SFs information are included in one COAM Reply message, every SF information should be listed as separate SF information sub-TLVs in COAM Reply message.

2. Multiple SFs for load balance:

Multiple SFs are attached to one SFF for load balance, that means only one SF will be transmitted for one traffic flow. These SFs have the same Service Function Type, Service Index. For this case, the SF identifiers of all these SFs will be listed in the SF Identifiers field 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.

In the draft [I-D.ietf-sfc-nsh-tlv], a Flow ID is introduced which can be used for load balance. SFF will distribute the SFC traffic flow to a given SF according to the Flow ID. So for a NSH COAM message carrying a Flow ID TLV, the SFF along the SFC could detect the COAM message and then send back the COAM Reply message with the corresponding SF information according to the Flow ID in the COAM.

4. Security Considerations

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

In addition, since Service Function sub-TLV discloses information about the RSP 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.wang-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 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.

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-04 (work in progress), July 2018.

[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.

[I-D.wang-sfc-multi-layer-oam]
Mirsky, G., Meng, W., Khasnabish, B., and C. Wang, "Active OAM for Service Function Chains in Networks", draft-wang-sfc-multi-layer-oam-12 (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>.

[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

SFC WG
Internet-Draft
Intended status: Standards Track
Expires: December 5, 2020

G. Mirsky
ZTE Corp.
T. Ao
Individual contributor
Z. Chen
China Telecom
K. Leung
Cisco System

June 3, 2020

SFC OAM for path consistency
draft-ao-sfc-oam-path-consistency-08

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 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.

Table of Contents

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 for the ability to verify the consistency of the particular SFP, using a mechanism of an active OAM protocol, is noted in [I-D.ietf-sfc-oam-framework].

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:

- 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];

- Forward the COAM Request to 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 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 [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 is displayed in Figure 1.

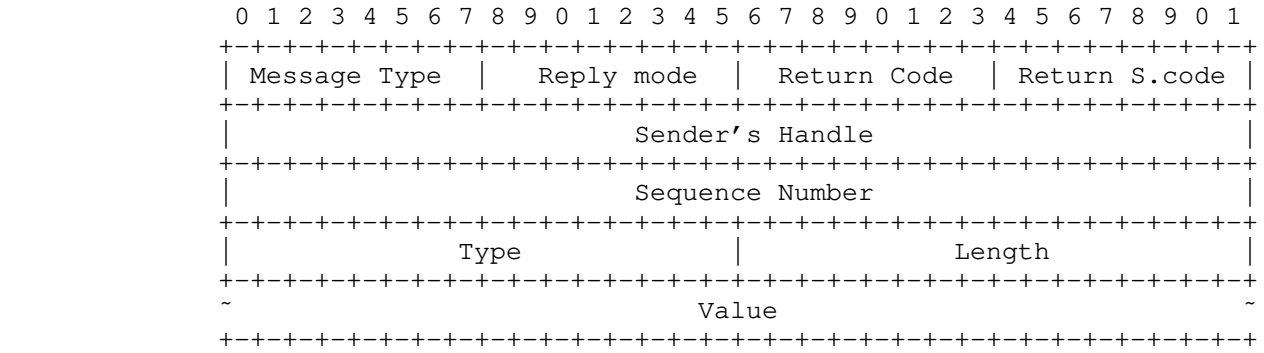


Figure 1: COAM Packet Header

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 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.

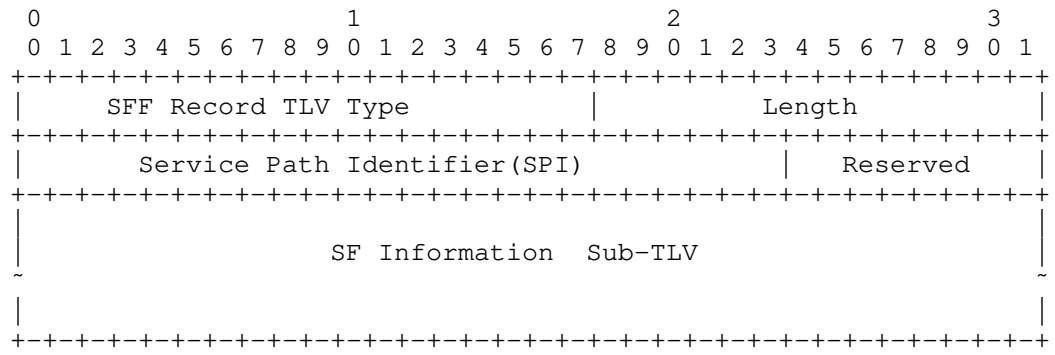


Figure 2: SFF Information Record TLV

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 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.

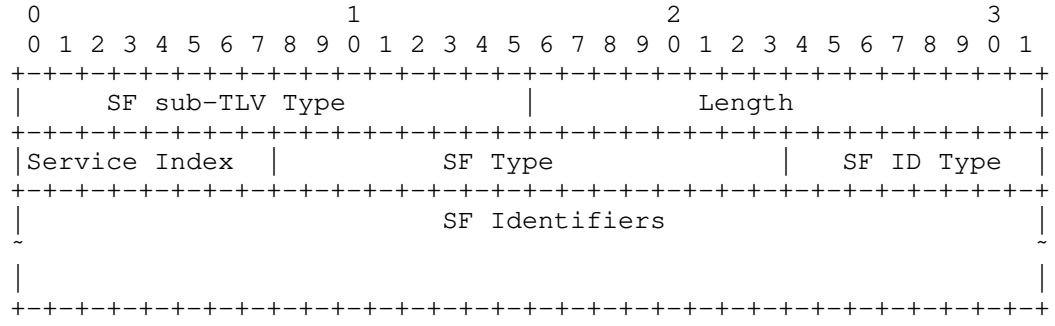


Figure 3: Service Function information sub-TLV

SF sub-TLV Type: Two octets long field. It indicates that the TLV is an SF TLV 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).

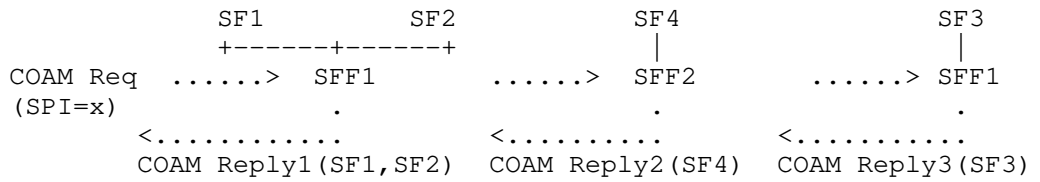


Figure 4: Example 1 for COAM Reply with multiple SFs

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 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 4. 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 also 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.

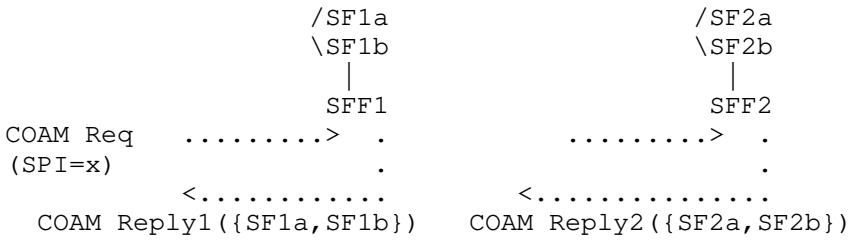


Figure 5: Example 2 for COAM Reply with multiple SFs

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 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

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 balance 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 Chains in Networks", draft-ietf-sfc-multi-layer-oam-06 (work in progress), June 2020.

[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", draft-ietf-bess-nsh-bgp-control-plane-14 (work in progress), June 2020.

[I-D.ietf-sfc-oam-framework] Aldrin, S., Pignataro, C., Nainar, N., Krishnan, R., and A. Ghanwani, "Service Function Chaining (SFC) Operations, Administration and Maintenance (OAM) Framework", draft-ietf-sfc-oam-framework-15 (work in progress), May 2020.

[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

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

Email: gregimirsky@gmail.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
USA

Email: kleung@cisco.com

SFC WG
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2019

T. Ao
ZTE Corporation
G. Mirsky
ZTE Corp.
Z. Chen
China Telecom

October 16, 2018

Controlled Return Path for Service Function Chain (SFC) OAM
draft-ao-sfc-oam-return-path-specified-02

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.

Table of Contents

1. Introduction

While Service Function Chain (SFC) Echo Request, defined in [I-D.wang-sfc-multi-layer-oam], always traverses the SFC it directed to, the corresponding Echo Reply is sent over IP network [I-D.wang-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.wang-sfc-multi-layer-oam]:

- Do Not Reply
- Reply via an IPv4/IPv6 UDP Packet
- Reply via Application Level Control Channel
- 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 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.

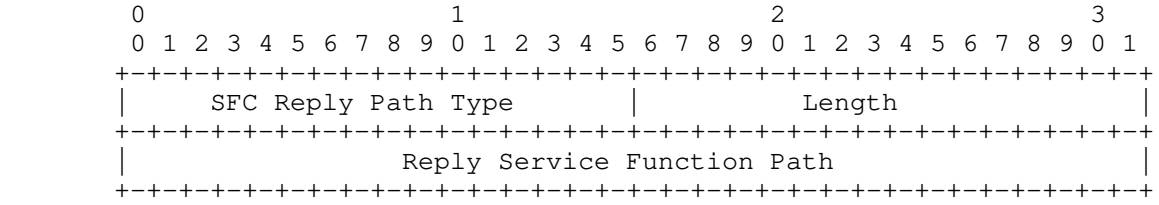


Figure 1: SFC Reply TLV Format

where:

- Reply Path TLV Type: is two octets long, indicates the TLV that contains information about the SFC Reply path.
- Length: is two octets long, 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 2

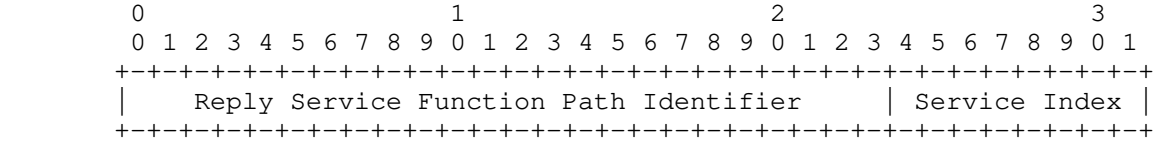


Figure 2: Reply Service Function Path Field Format

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: 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.wang-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 [I-D.ietf-sfc-nsh]. 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 [I-D.ietf-sfc-nsh] 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.wang-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-nsh]
Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", draft-ietf-sfc-nsh-28 (work in progress), November 2017.

[I-D.wang-sfc-multi-layer-oam]
Mirsky, G., Meng, W., Khasnabish, B., and C. Wang, "Active OAM for Service Function Chains in Networks", draft-wang-sfc-multi-layer-oam-12 (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>.

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: December 5, 2020

G. Mirsky
ZTE Corp.
T. Ao
Individual contributor
Z. Chen
China Telecom

June 3, 2020

Controlled Return Path for Service Function Chain (SFC) OAM
draft-ao-sfc-oam-return-path-specified-06

Abstract

This document defines an extension to the Service Function Chain (SFC) Operation, Administration and Maintenance (OAM) that enables control of the Echo Reply return path directing it over a Reverse Service Function Path. Enforcing the specific return path can be used to verify the 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 December 5, 2020.

Table of Contents

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 extends Service Function Chain (SFC) Operation, Administration and Maintenance (OAM) by enabling control of the Echo Reply return path to be directed over a Reply Service Function Path (SFP). Such extension is based on the analysis of SFC OAM, active OAM protocols in particular, provided in [I-D.ietf-sfc-oam-framework]. 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 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

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

- Do Not Reply
- Reply via an IPv4/IPv6 UDP Packet
- Reply via Application Level Control Channel
- 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 verify bidirectional continuity or increase SFC monitoring's robustness 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.

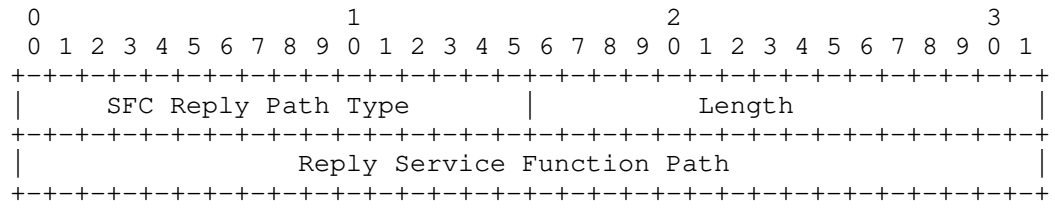


Figure 1: SFC Reply TLV Format

where:

- Reply Path TLV Type: is two octets long, indicates the TLV that contains information about the SFC Reply path.
- Length: is two octets long, 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 2

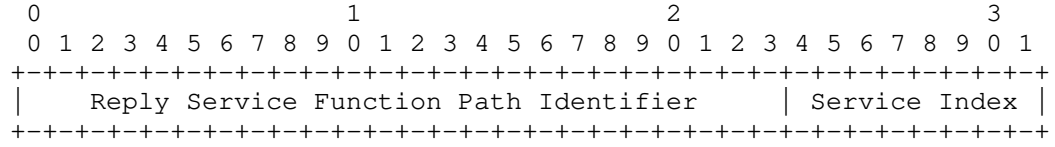


Figure 2: Reply Service Function Path Field Format

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.

5. Theory of Operation

[RFC7110] defined mechanism to control return path for MPLS LSP Echo Reply. In case of SFC, the return path is an 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 the 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 to 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

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 may be not 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. Security Considerations

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

The SFC Return Path extension, defined in this document, can be used for potential "proxying" attacks. For example, an initiator of the Echo Request 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, 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-06 (work in progress), June 2020.

[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

[I-D.ietf-sfc-oam-framework]
Aldrin, S., Pignataro, C., Nainar, N., Krishnan, R., and A. Ghanwani, "Service Function Chaining (SFC) Operations, Administration and Maintenance (OAM) Framework", draft-ietf-sfc-oam-framework-15 (work in progress), May 2020.

[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

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

Email: gregimirsky@gmail.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

INTERNET-DRAFT
Intended Status: Standard Track

Sami Boutros, Ed.
VMware
Dharma Rajan
Philip Kippen
Pierluigi Rolando
VMware

Expires: March 18, 2019

September 14, 2018

Geneve applicability for service function chaining
draft-boutros-nvo3-geneve-applicability-for-sfc-02

Abstract

This document describes the applicability of using Generic Network Virtualization Encapsulation (Geneve), to carry the service function path (SFP) information, and the network service header (NSH) encapsulation. The SFP information will be carried in Geneve option TLV(s).

Table of Contents

1. Introduction

The Service Function Chaining (SFC) Architecture [rfc7665] defines a service function chain (SFC) as (1) the instantiation of an ordered set of service functions and (2) the subsequent "steering" of traffic through them.

SFC defines a Service Function Path (SFP) as the exact set of service function forwarders (SFF)/service functions (SF)s the packet will visit when it actually traverses the network.

An optimized SFP helps to build an efficient Service function chain (SFC) that can be used to steer traffic based on classification rules, and metadata information to provide services for Network Function Virtualization (NFV). Metadata are typically passed between service functions and Service function forwarders SFF(s) along a service function path.

In a Network Virtualization Overlays (NVO3) domain, Network Virtualization Edges (NVE)s can be implemented on hypervisors hosting virtual network functions (VNF)s implementing service functions, or on physical routers connected to service function appliances. NVO3 domain uses tunneling and encapsulation protocols such as Geneve to provide connectivity for tenants workloads and service function running in its domain. NVEs in an NVO3 domain are typically controlled by a centralized network virtualization authority NVA.

[RFC8300] defines a new encapsulation protocol, network service header (NSH) to encode the SFP and the metadata.

1.1 Requirement for SFC in NVO3 domain

The requirement is to provide service function chaining in an NVO3 domain without the need to implement yet another control plane for service topology.

1.2 Proposed solution for SFC in NVO3 domain

This document specifies the applicability of using Generic Network Virtualization Encapsulation (Geneve), to carry the service function path (SFP) information, and the network service header (NSH) encapsulation.

The SFP will be implemented using a new Geneve Service Function List (SFL) option for use strictly between Network Virtualization Edges (NVEs) performing the service forwarding function (SFF) in the same Network Virtualization Overlay over Layer 3 NVO3 domain. The next protocol in the Geneve Header will be the NSH EtherType, 0x894F. The

NSH encapsulation will include the Service Path Identifier (SPI) and the Service Index (SI). The NSH SI will serve as an index to the VNF hop to visit in the SFL.

In the absence of the SFL we would need a service topology control plane. The Geneve overlay will encap the NSH encapsulation and the next protocol on Geneve will be the NSH Ethertype.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Abbreviations

NVO3 Network Virtualization Overlays over Layer 3

OAM Operations, Administration, and Maintenance

TLV Type, Length, and Value

VNI Virtual Network Identifier

NVE Network Virtualization Edge

NVA Network Virtualization Authority

NIC Network interface card

VTEP Virtual Tunnel End Point

Transit device Underlay network devices between NVE(s).

Service Function (SF): Defined in [RFC7665].

Service Function Chain (SFC): Defined in [RFC7665].

Service Function Forwarder (SFF): Defined in [RFC7665].

Service Function Path (SFP): Defined in [RFC7665].

Metadata: Defined in [[draft-ietf-sfc-nsh]

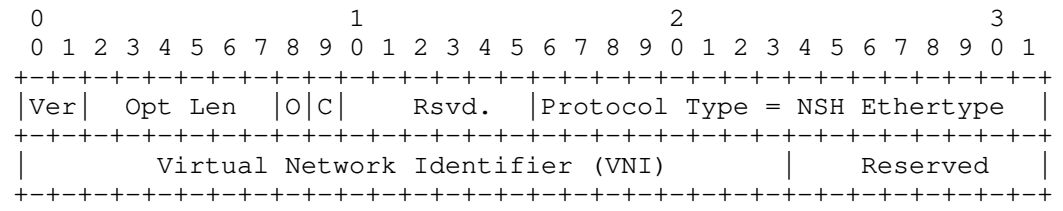
NFV: Network function virtualization.

VNF: Virtual network function

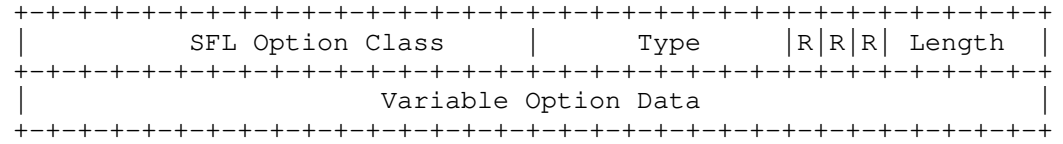
4. Geneve Option TLV(s)

4.1 Geneve Service Function List (SFL) Option TLV

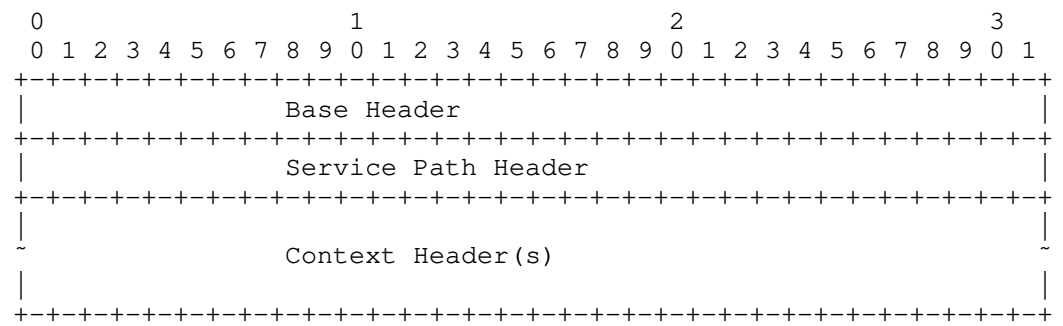
Geneve Header:



Geneve Option Header:



Followed by the NSH encapsulation which is composed of a 4-byte Base Header, a 4-byte Service Path Header, and optional Context Headers.



SFL Option Class = To be assigned by IANA

Type = To be assigned by IANA

'C' bit set, indicating endpoints must drop if they do not recognize this option)

Length = variable.

Variable option data:

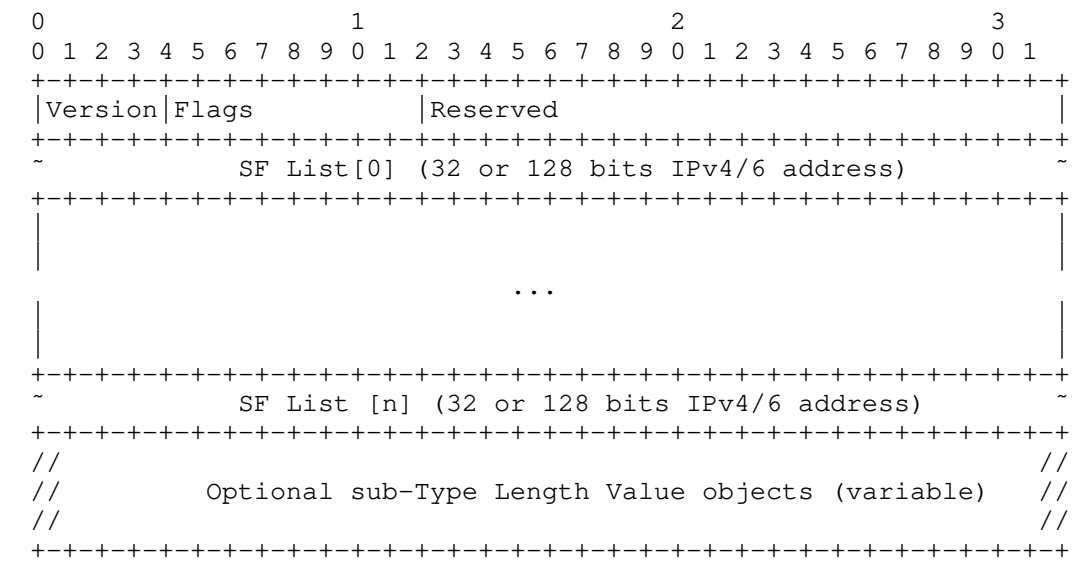


Figure 1: Service Function List (SFL) Option TLV.

Reserved: 12 bits. SHOULD be unset on transmission and MUST be ignored on receipt.

Flags:

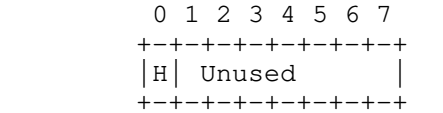


Figure 2: SFL flags

H-flag: HMAC flag. If set, the HMAC sub-TLV is present and is encoded as the last sub-TLV.

SF List[n]: 32 or 128 bits IPv4/6 addresses representing the nth service function ip address in the List.

The SF List is encoded starting from the last hop of the path. I.e., the first element of the list (SF List [0]) contains the last service function of the path while the last element of the SF List (SF List[n]) contains the first service function in the path.

HMAC sub-TLV is optional and contains the HMAC information. The

HMAC sub-TLV has the following format:

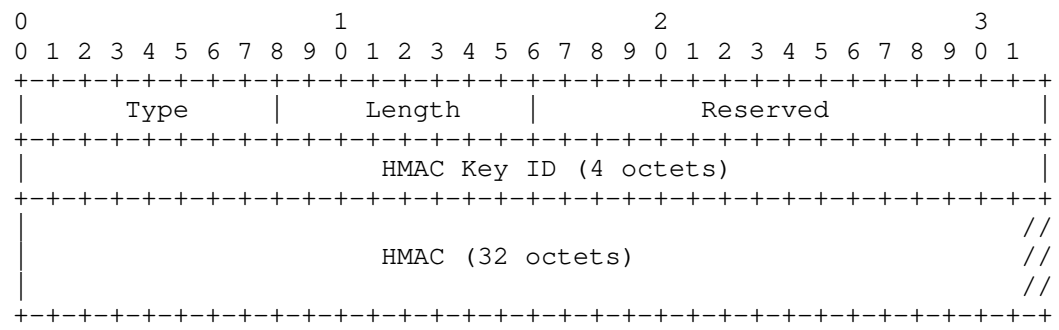


Figure 3: SFL HMAC sub-TLV.

Type: to be assigned by IANA (suggested value 1).
Length: 38.
Reserved: 2 octets. SHOULD be unset on transmission and MUST be ignored on receipt.
HMAC Key ID: 4 octets.
HMAC: 32 octets.
HMAC and HMAC Key ID usage is described in Operation section.

The Following applies to the HMAC TLV:

When present, the HMAC sub-TLV MUST be encoded as the last sub-TLV

If the HMAC sub-TLV is present, the H-Flag (Figure 2) MUST be set.

When the H-flag is set, the NVE inspecting the Geneve Service Function List Option TLV MUST find the HMAC sub-TLV in the last 38 octets of the option TLV.

5.. Operation

The mechanisms described in this section should work with both ipv4 and ipv6 for both customer inner payload and Geneve tunnel packets.

5.1 Operation at Ingress

A Source NVE acting as a service function classifier and a service function forwarder can be any node in an NVO3 domain, originating based on a classification policy for some customer inner payload an IP Geneve tunnel packet with the service function list (SFL) option TLV. The service functions in the SFL represent the IP addresses of the service functions that the inner customer packets needs to be inspected by. A controller can program the ingress NVE node to classify traffic and identify a service function paths i.e the set of

service functions in the path. The mechanism through which an SFL is derived by a controller or any other mechanisms is outside of the scope of this document.

The ingress NVE node fills in the list of service functions in the path, to the Geneve Service Function List option TLV, putting the first service function ip address as the last element in the list and the last service function ip address as the first element, setting of the NSH service index to the first element. The ingress NVE node, then, resolves the service first function ip address, to the NVE virtual tunnel endpoint node hosting or directly connected to the service function.

The Geneve tunnel destination is then set to the NVE tunnel endpoint hosting the first service function, and the service index is decremented to n-1 (where n is the number of elements in the SFL), and set on the SFL option TLV. An NSH metadata can also be set on the packet by the NVE ingress node.

The Geneve packet is sent out towards the first NVE.

HMAC optional sub-TLV may be set too.

5.2 Operation at each NVE along the service function path

The NVE node along the service function path corresponding to the Geneve tunnel destination of the packet, receives the packet, perform the service function forwarder function and identifies the SFL option, and locates the service function in the list based on the service index.

The Geneve tunnel header and option TLV(s) will be stripped and the packet will be delivered to the service function or virtual network function (VNF). The NVE maintains state related to the association of the SFL option TLV and the NSH service path identifier. The packet passed to the service function encaped with the NSH header and NSH context, if the SF is NSH aware, other encapsulations like vlan or q-in-q encap may be used to pass the metadata and NSH SPI to the SF too.

When the packet comes back from the service function along with the service path identifier (SPI) context, based on SPI on the packet the NVE acting as the SFF will be able to locate the SFL option TLV.

If the metadata context indicate (1) that some service functions need to be bypassed the NVE should bypass in the SFL the service functions to be skipped and update the NSH service index accordingly. (2) A new

classification need to be performed on the packet, in that case the NVE can re-classify the packet or sent it to an NVE node capable of classification.

The NVE node, then, resolves the next service function ip address, to the NVE virtual tunnel endpoint node hosting or directly connected to the service function.

The NVE then sets the Geneve tunnel destination to the next NVE tunnel endpoint, and the NSH service index is decremented by 1 and set on the NSH Header, along with other NSH metadata option TLV.

The Geneve ip packet is sent out towards the next NVE.

5.3 Operation at Egress

At the last NVE node along the service function path, the NVE locates the service function in the SFL option TLV based on the NSH service index. The service index received at the last NVE node will be set to 1.

The Geneve tunnel header and option TLV(s) will be stripped and the packet will be delivered to the service function. The NVE maintains state related to the association of the SFL option TLV and the NSH service path identifier. The packet passed to the service function encaped with the NSH header and NSH context, if the SF is NSH aware, other encapsulations like vlan or q-in-q encap may be used to pass the metadata and NSH SPI to the SF too.

When the packet comes back from the service function, based on NSH SPI on the packet or based the NVE will be able to locate the SFL option TLV.

Given that the service index will be set to 1, the last NVE will now deliver the packet to the NVE hosting or directly connected to the inner packet destination.

A packet received with a service function index of 0 MUST be dropped.

6. Security Considerations

Only NVE(s) that are the destinations of the Geneve tunnel packet will be inspecting the List of Service Function next hops Option. A Source routing option has some well-known security issues as described in [RFC4942] and [RFC5095].

The main use case for the use of the Geneve List of Service Function next hops Option will be within a single NVO3 administrative domain

where only trusted NVE nodes are enabled and configured participate, this is the same model as in [RFC6554].

NVE nodes MUST ignore the Geneve List of Service Function next hops Option created by outsiders based on NVA or trusted control plane information.

There is a need to prevent non-participating NVE node from using the Geneve Service Function List option TLV, as described in [draft-ietf-6man-segment-routing-header], we will use a security sub-TLV in the Service Function List option TLV, the security sub-TLV will be based on a key-hashed message authentication code (HMAC).

HMAC sub-TLV will contain:

HMAC Key-id, 32 bits wide;

HMAC, 256 bits wide (optional, exists only if HMAC Key-id is not 0).

The HMAC field is the output of the HMAC computation (per RFC 2104 [RFC2104]) using a pre-shared key identified by HMAC Key-id and of the text which consists of the concatenation of:

The source IPv4/IPv6 Geneve tunnel address

Version and Flags

HMAC Key-id.

All addresses in the List.

The purpose of the HMAC optional sub-TLV is to verify the validity, the integrity and the authorization of the Geneve Service Function List option TLV itself.

The HMAC optional sub-TLV is located at the end of the Geneve Service Function List option TLV.

The HMAC Key-id field serves as an index to the right combination of pre-shared key and hash algorithm and except that a value of 0 means that there is no HMAC field.

The HMAC Selection of a hash algorithm and Pre-shared key management will follow the procedures described in [draft-ietf-6man-segment-routing-header] section 6.2.

7. Management Considerations

The Source NVE can receive its information through any form of north bound Orchestrator. These could be from any open networking automation platform (ONAP) or others. The ingress to egress tunnel is built and managed by the service function classifier and service function forwarder by each node in an NVO3 domain. Error handling, is handled by the classifier reporting to north bound management systems.

8. Acknowledgements

The authors would like to acknowledge Jim Guichard for his feedback and valuable comments to this document.

9. IANA Considerations

This document makes the following registrations in the "Geneve Option Class" registry maintained by IANA:

Suggested Value	Description	Reference
XX	Geneve List of Service Function next hops	This document

In addition, this document request IANA to create and maintain a new Registry: "Geneve List of Service Function next hops Type-Value Objects".

The following code-point are requested from the registry:

Registry: Geneve List of Service Function next hops Type-Value Objects

Suggested Value	Description	Reference
1	HMAC TLV	This document

10. References

10.1 Normative References

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

10.2 Informative References

[Geneve] "Generic Network Virtualization Encapsulation", [I-D.ietf-

nvo3-geneve]

[RFC8300] Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", RFC 8300, January 2018, <http://www.rfc-editor.org/info/rfc8300>.

[RFC4942] Davies, E., Krishnan, S., and P. Savola, "IPv6 Transition/Co-existence Security Considerations", RFC 4942, DOI 10.17487/RFC4942, September 2007, <http://www.rfc-editor.org/info/rfc4942>.

[RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <http://www.rfc-editor.org/info/rfc6554>.

[draft-ietf-6man-segment-routing-header] Previdi, S., et all, "IPv6 Segment Routing Header (SRH)", July 20, 2017, draft-ietf-6man-segment-routing-header-07

[RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <http://www.rfc-editor.org/info/rfc5095>.

[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

Sami Boutros
VMware
Email: sboutros@vmware.com

Dharma Rajan
VMware
Email: drajan@vmware.com

Philip Kippen
VMware
Email: pkippen@vmware.com

Pierluigi Rolando
VMware
Email: prolando@vmware.com

INTERNET-DRAFT
Intended Status: Standard Track

Sami Boutros, Ed.
VMware
Dharma Rajan
Philip Kippen
Pierluigi Rolando
VMware
Jim Guichard
Huawei
Sam Aldrin
Google

Expires: March 19, 2020 September 16, 2019

Geneve applicability for service function chaining
draft-boutros-nvo3-geneve-applicability-for-sfc-04

Abstract

This document describes the applicability of using Generic Network Virtualization Encapsulation (Geneve), to carry the service function path (SFP) information, and the network service header (NSH) encapsulation. The SFP information will be carried in Geneve option TLV(s).

Table of Contents

1. Introduction

The Service Function Chaining (SFC) Architecture [rfc7665] defines a service function chain (SFC) as (1) the instantiation of an ordered set of service functions and (2) the subsequent "steering" of traffic through them.

SFC defines a Service Function Path (SFP) as the exact set of service function forwarders (SFF)/service functions (SF)s the packet will visit when it actually traverses the network.

An optimized SFP helps to build an efficient Service function chain (SFC) that can be used to steer traffic based on classification rules, and metadata information to provide services for Network Function Virtualization (NFV). Metadata are typically passed between service functions and Service function forwarders SFF(s) along a service function path.

In a Network Virtualization Overlays (NVO3) domain, Network Virtualization Edges (NVE)s can be implemented on hypervisors hosting virtual network functions VNF(s) or cloud native functions CNF(s) implementing service functions, or on CNFs on bare metal servers or on physical routers connected to service function appliances. NVO3 domain uses tunneling and encapsulation protocols such as Geneve to provide connectivity for tenants workloads and service function running in its domain. NVEs in an NVO3 domain are typically controlled by a centralized network virtualization authority NVA.

[RFC8300] defines a new encapsulation protocol, network service header (NSH) to encode the SFP and the metadata.

1.1 Requirement for SFC in NVO3 domain

The requirement is to provide service function chaining in an NVO3 domain without the need to implement yet another control plane for service topology.

1.2 Proposed solution for SFC in NVO3 domain

This document specifies the applicability of using Generic Network Virtualization Encapsulation (Geneve), to carry the service function path (SFP) information, and the network service header (NSH) encapsulation.

The SFP will be implemented using a new Geneve Service Function List (SFL) option for use strictly between Network Virtualization Edges (NVEs) performing the service forwarding function (SFF) in the same Network Virtualization Overlay over Layer 3 NVO3 domain. The next

protocol in the Geneve Header will be the NSH EtherType, 0x894F. The NSH encapsulation will include the Service Path Identifier (SPI) and the Service Index (SI). The NSH SI will serve as an index to the VNF/CNF hop to visit in the SFL.

In the absence of the SFL we would need a service topology control plane. The Geneve overlay will encap the NSH encapsulation and the next protocol on Geneve will be the NSH Ethertype.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Abbreviations

NVO3 Network Virtualization Overlays over Layer 3

OAM Operations, Administration, and Maintenance

TLV Type, Length, and Value

VNI Virtual Network Identifier

NVE Network Virtualization Edge

NVA Network Virtualization Authority

NIC Network interface card

VTEP Virtual Tunnel End Point

Transit device Underlay network devices between NVE(s).

Service Function (SF): Defined in [RFC7665].

Service Function Chain (SFC): Defined in [RFC7665].

Service Function Forwarder (SFF): Defined in [RFC7665].

Service Function Path (SFP): Defined in [RFC7665].

Metadata: Defined in [[draft-ietf-sfc-nsh]

NFV: Network function virtualization.

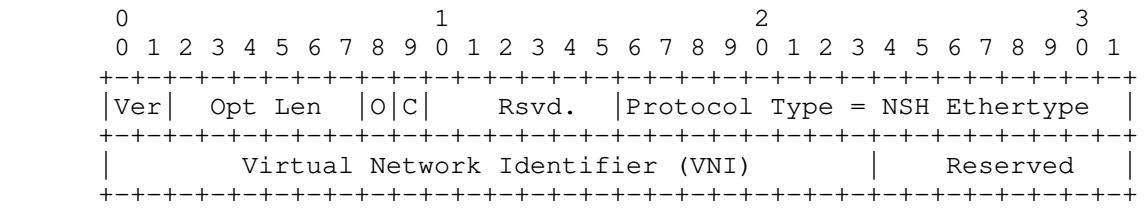
VNF: Virtual network function

CNF: Cloud native function

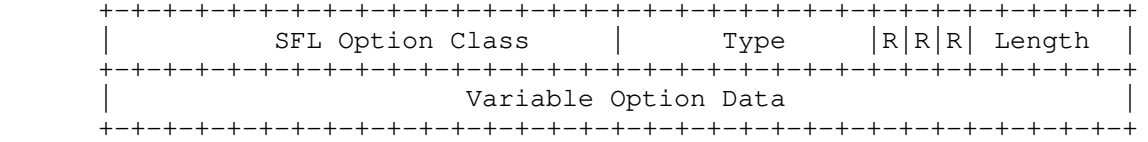
4. Geneve Option TLV(s)

4.1 Geneve Service Function List (SFL) Option TLV

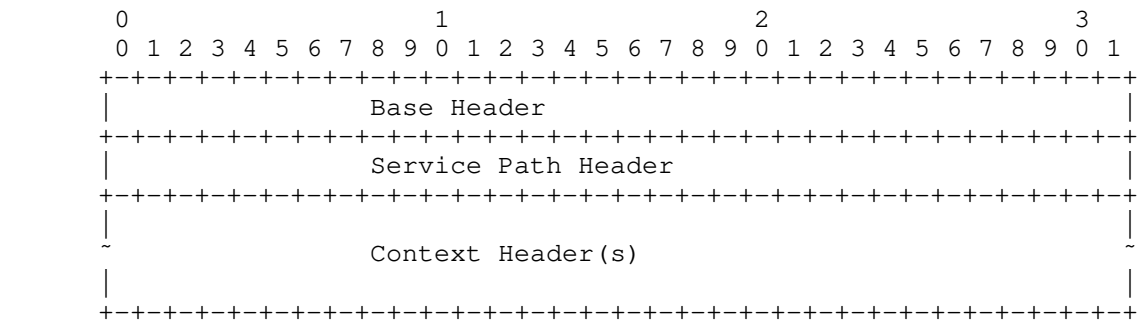
Geneve Header:



Geneve Option Header:



Followed by the NSH encapsulation which is composed of a 4-byte Base Header, a 4-byte Service Path Header, and optional Context Headers.



SFL Option Class = To be assigned by IANA

Type = To be assigned by IANA

'C' bit set, indicating endpoints must drop if they do not recognize this option)

Length = variable.

Variable option data:

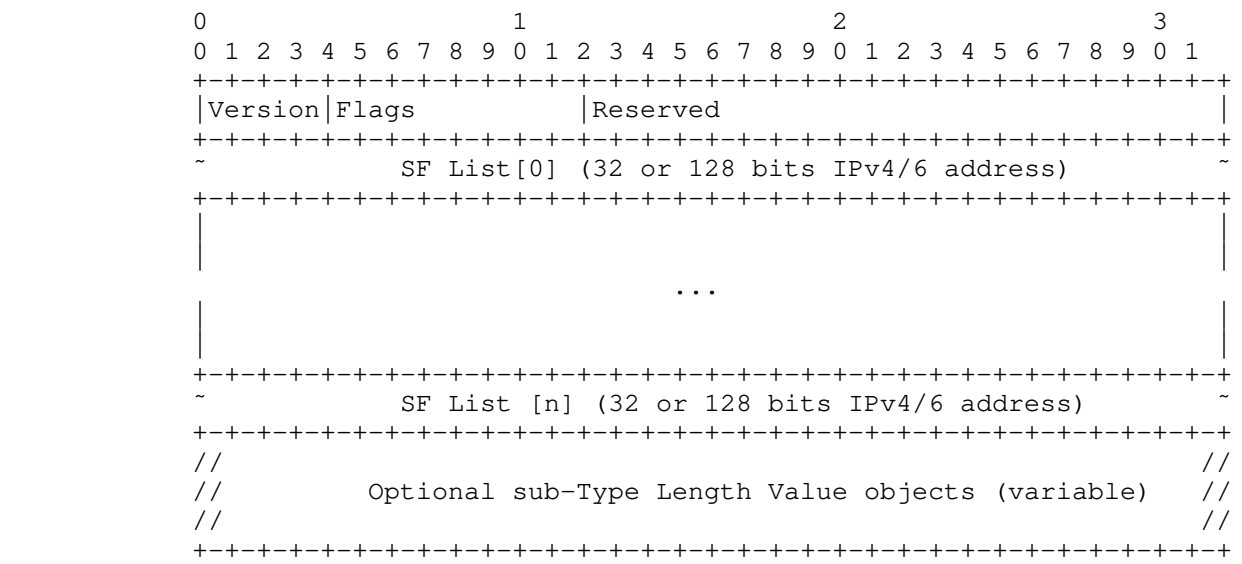


Figure 1: Service Function List (SFL) Option TLV.

Reserved: 12 bits. SHOULD be unset on transmission and MUST be ignored on receipt.

Flags:

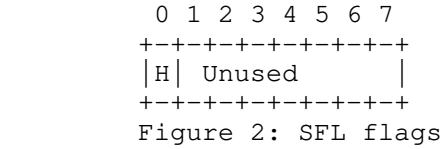


Figure 2: SFL flags

H-flag: HMAC flag. If set, the HMAC sub-TLV is present and is encoded as the last sub-TLV.

SF List[n]: 32 or 128 bits IPv4/6 addresses representing the nth service function ip address in the List.

The SF List is encoded starting from the last hop of the path. I.e., the first element of the list (SF List [0]) contains the last service function of the path while the last element of the SF List (SF List[n]) contains the first service function in the path.

HMAC sub-TLV is optional and contains the HMAC information. The

HMAC sub-TLV has the following format:

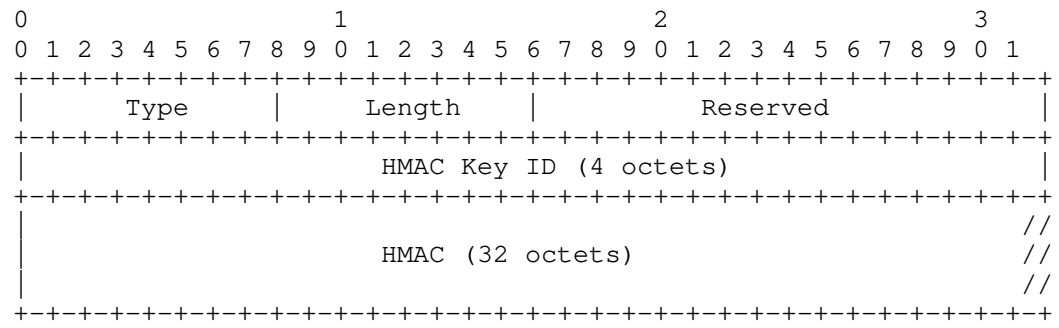


Figure 3: SFL HMAC sub-TLV.

Type: to be assigned by IANA (suggested value 1).
Length: 38.
Reserved: 2 octets. SHOULD be unset on transmission and MUST be ignored on receipt.
HMAC Key ID: 4 octets.
HMAC: 32 octets.
HMAC and HMAC Key ID usage is described in Operation section.

The Following applies to the HMAC TLV:

When present, the HMAC sub-TLV MUST be encoded as the last sub-TLV

If the HMAC sub-TLV is present, the H-Flag (Figure 2) MUST be set.

When the H-flag is set, the NVE inspecting the Geneve Service Function List Option TLV MUST find the HMAC sub-TLV in the last 38 octets of the option TLV.

5.. Operation

The mechanisms described in this section should work with both ipv4 and ipv6 for both customer inner payload and Geneve tunnel packets.

5.1 Operation at Ingress

A Source NVE acting as a service function classifier and a service function forwarder can be any node in an NVO3 domain, originating based on a classification policy for some customer inner payload an IP Geneve tunnel packet with the service function list (SFL) option TLV. The service functions in the SFL represent the IP addresses of the service functions that the inner customer packets needs to be inspected by. A controller can program the ingress NVE node to classify traffic and identify a service function paths i.e the set of

service functions in the path. The mechanism through which an SFL is derived by a controller or any other mechanisms is outside of the scope of this document.

The ingress NVE node fills in the list of service functions in the path, to the Geneve Service Function List option TLV, putting the first service function ip address as the last element in the list and the last service function ip address as the first element, setting of the NSH service index to the first element. The ingress NVE node, then, resolves the service first function ip address, to the NVE virtual tunnel endpoint node hosting or directly connected to the service function.

The Geneve tunnel destination is then set to the NVE tunnel endpoint hosting the first service function, and the service index is decremented to n-1 (where n is the number of elements in the SFL), and set on the SFL option TLV. An NSH metadata can also be set on the packet by the NVE ingress node.

The Geneve packet is sent out towards the first NVE.

HMAC optional sub-TLV may be set too.

5.2 Operation at each NVE along the service function path

The NVE node along the service function path corresponding to the Geneve tunnel destination of the packet, receives the packet, perform the service function forwarder function and identifies the SFL option, and locates the service function in the list based on the service index.

The Geneve tunnel header and option TLV(s) will be stripped and the packet will be delivered to the service function or virtual network function VNF or CNF. The NVE maintains state related to the association of the SFL option TLV and the NSH service path identifier. The packet passed to the service function encaped with the NSH header and NSH context, if the SF is NSH aware, other encapsulations like vlan or q- in-q encap may be used to pass the metadata and NSH SPI to the SF too.

When the packet comes back from the service function along with the service path identifier (SPI) context, based on SPI on the packet the NVE acting as the SFF will be able to locate the SFL option TLV.

If the metadata context indicate (1) that some service functions need to be bypassed the NVE should bypass in the SFL the service functions to be skipped and update the NSH service index accordingly. (2) A new

classification need to be performed on the packet, in that case the NVE can re-classify the packet or sent it to an NVE node capable of classification.

The NVE node, then, resolves the next service function ip address, to the NVE virtual tunnel endpoint node hosting or directly connected to the service function.

The NVE then sets the Geneve tunnel destination to the next NVE tunnel endpoint, and the NSH service index is decremented by 1 and set on the NSH Header, along with other NSH metadata option TLV.

The Geneve ip packet is sent out towards the next NVE.

5.3 Operation at Egress

At the last NVE node along the service function path, the NVE locates the service function in the SFL option TLV based on the NSH service index. The service index received at the last NVE node will be set to 1.

The Geneve tunnel header and option TLV(s) will be stripped and the packet will be delivered to the service function. The NVE maintains state related to the association of the SFL option TLV and the NSH service path identifier. The packet passed to the service function encaped with the NSH header and NSH context, if the SF is NSH aware, other encapsulations like vlan or q-in-q encap may be used to pass the metadata and NSH SPI to the SF too.

When the packet comes back from the service function, based on NSH SPI on the packet or based the NVE will be able to locate the SFL option TLV.

Given that the service index will be set to 1, the last NVE will now deliver the packet to the NVE hosting or directly connected to the inner packet destination.

A packet received with a service function index of 0 MUST be dropped.

6. Security Considerations

Only NVE(s) that are the destinations of the Geneve tunnel packet will be inspecting the List of Service Function next hops Option. A Source routing option has some well-known security issues as described in [RFC4942] and [RFC5095].

The main use case for the use of the Geneve List of Service Function next hops Option will be within a single NVO3 administrative domain

where only trusted NVE nodes are enabled and configured participate, this is the same model as in [RFC6554].

NVE nodes MUST ignore the Geneve List of Service Function next hops Option created by outsiders based on NVA or trusted control plane information.

There is a need to prevent non-participating NVE node from using the Geneve Service Function List option TLV, as described in [draft-ietf-6man-segment-routing-header], we will use a security sub-TLV in the Service Function List option TLV, the security sub-TLV will be based on a key-hashed message authentication code (HMAC).

HMAC sub-TLV will contain:

HMAC Key-id, 32 bits wide;

HMAC, 256 bits wide (optional, exists only if HMAC Key-id is not 0).

The HMAC field is the output of the HMAC computation (per RFC 2104 [RFC2104]) using a pre-shared key identified by HMAC Key-id and of the text which consists of the concatenation of:

The source IPv4/IPv6 Geneve tunnel address

Version and Flags

HMAC Key-id.

All addresses in the List.

The purpose of the HMAC optional sub-TLV is to verify the validity, the integrity and the authorization of the Geneve Service Function List option TLV itself.

The HMAC optional sub-TLV is located at the end of the Geneve Service Function List option TLV.

The HMAC Key-id field serves as an index to the right combination of pre-shared key and hash algorithm and except that a value of 0 means that there is no HMAC field.

The HMAC Selection of a hash algorithm and Pre-shared key management will follow the procedures described in [draft-ietf-6man-segment-routing-header] section 6.2.

7. Management Considerations

The Source NVE can receive its information through any form of north bound Orchestrator. These could be from any open networking automation platform (ONAP) or others. The ingress to egress tunnel is built and managed by the service function classifier and service function forwarder by each node in an NVO3 domain. Error handling, is handled by the classifier reporting to north bound management systems.

8. Acknowledgements

The authors would like to acknowledge Jim Guichard for his feedback and valuable comments to this document.

9. IANA Considerations

This document makes the following registrations in the "Geneve Option Class" registry maintained by IANA:

Suggested Value	Description	Reference
XX	Geneve List of Service Function next hops	This document

In addition, this document request IANA to create and maintain a new Registry: "Geneve List of Service Function next hops Type-Value Objects".

The following code-point are requested from the registry:

Registry: Geneve List of Service Function next hops Type-Value Objects

Suggested Value	Description	Reference
1	HMAC TLV	This document

10. References

10.1 Normative References

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

10.2 Informative References

[Geneve] "Generic Network Virtualization Encapsulation", [I-D.ietf-

nvo3-geneve]

[RFC8300] Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", RFC 8300, January 2018, <http://www.rfc-editor.org/info/rfc8300>.

[RFC4942] Davies, E., Krishnan, S., and P. Savola, "IPv6 Transition/Co-existence Security Considerations", RFC 4942, DOI 10.17487/RFC4942, September 2007, <http://www.rfc-editor.org/info/rfc4942>.

[RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <http://www.rfc-editor.org/info/rfc6554>.

[draft-ietf-6man-segment-routing-header] Previdi, S., et all, "IPv6 Segment Routing Header (SRH)", July 20, 2017, draft-ietf-6man-segment-routing-header-07

[RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <http://www.rfc-editor.org/info/rfc5095>.

[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

Sami Boutros
VMware
Email: boutross@vmware.com

Dharma Rajan
VMware
Email: drajan@vmware.com

Philip Kippen
VMware
Email: pkippen@vmware.com

Pierluigi Rolando
VMware
Email: prolando@vmware.com

Jim Guichard
Huawei
Email: james.n.guichard@huawei.com

Sam Aldrin
Google
Email:aldrin.ietf@gmail.com

INTERNET-DRAFT
Intended status: Proposed Standard

Donald Eastlake
Huawei
Bob Briscoe
Independent
Andrew Malis
Huawei

Expires: January 1, 2019
July 2, 2018

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

Abstract

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

Status of This Memo

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

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html. The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Table of Contents

1. Introduction

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

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

1.1 NSH Background

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

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

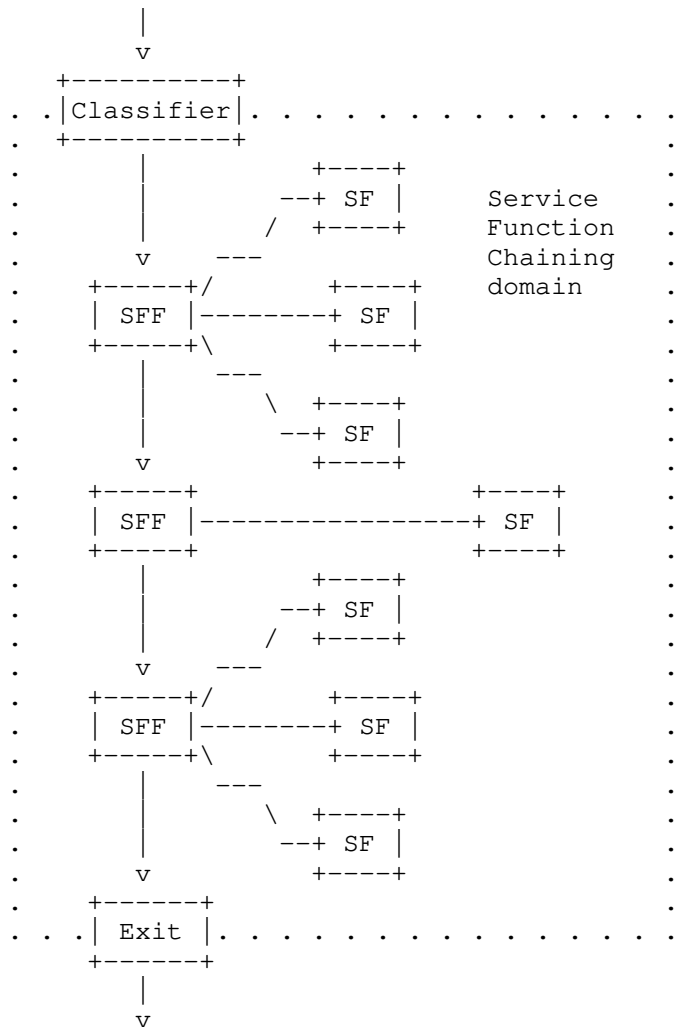


Figure 1. Example SFC Path Forwarding Nodes

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

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

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

1.2 ECN Background

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

1.3 Tunnel Congestion Feedback Background

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

(1) Traffic throttling (policing), where the downstream traffic flowing out of the ingress node is limited to reduce or eliminate congestion.

(2) Upstream congestion feedback, where the ingress node sends messages upstream to or towards the ultimate traffic source, a function that can throttle traffic generation/transmission.

(3) Traffic re-direction, where the ingress node configures the NSH so that some future traffic avoids congested paths.

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

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

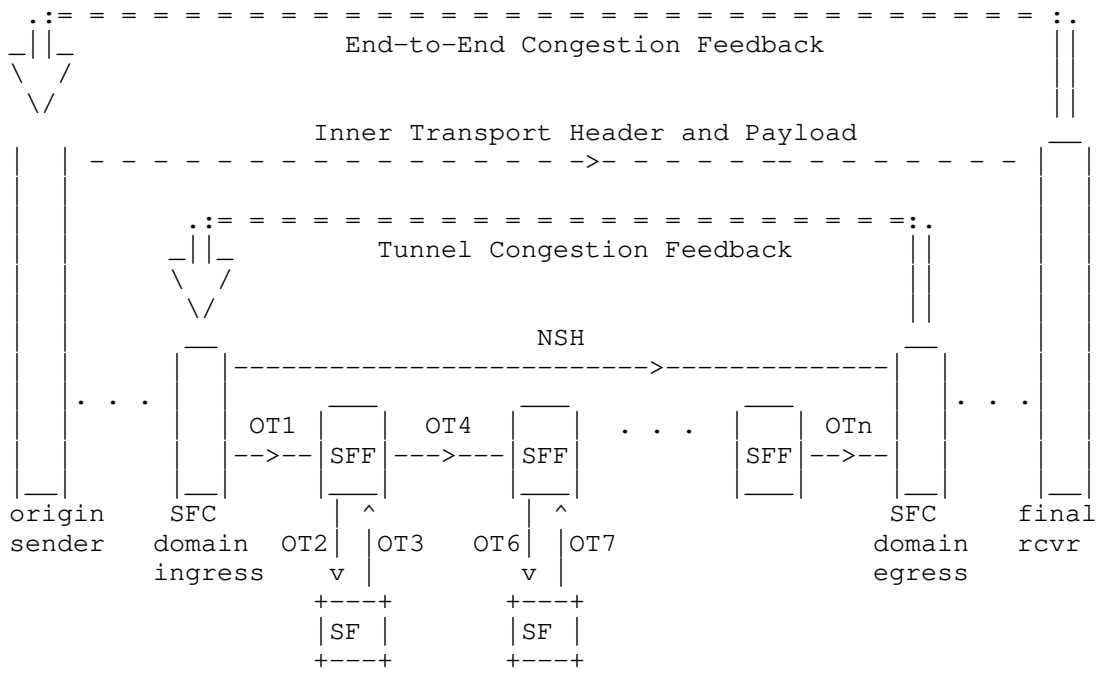


Figure 2: Congestion Feedback across an SFC Domain

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

1.4 Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this

document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Acronyms:

AQM - Active Queue Management [RFC7567]

CE - Congestion Experienced [RFC3168]

downstream - The direction from ingress to egress

ECN - Explicit Congestion Notification [RFC3168]

ECT - ECN Capable Transport [RFC3168]

IPFIX - IP Flow Information Export [RFC7011]

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

NSH - Network Service Header [RFC8300]

SF - Service Function [RFC7665]

SFC - Service Function Chaining [RFC7665]

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

TLV - Type Length Value

upstream - The direction from egress to ingress

2. The NSH ECN Field

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

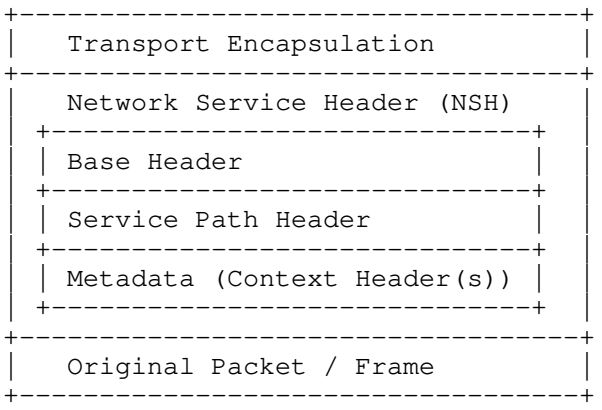


Figure 3. Data Encapsulation with the NSH

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

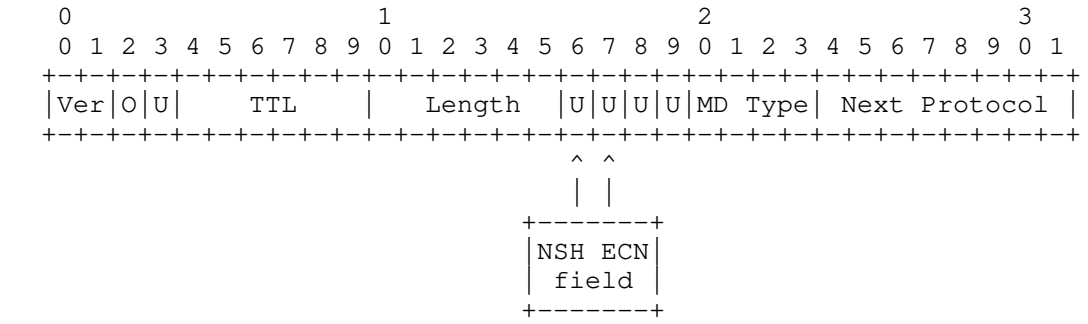


Figure 4: NSH Base Header

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

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

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

Table 1. ECN Field Code Points

3. ECN Support in the NSH

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

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

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

ECN Propagation:

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

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

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

ECN Marking:

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

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

3.1 At The Ingress

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

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

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

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

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

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

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

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

3.2 At Transit Nodes

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

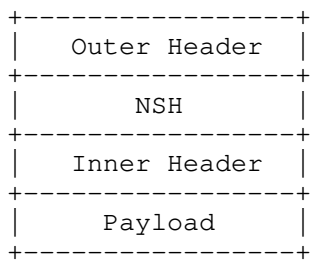


Figure 5. Packet in Transit

3.2.1 At NSH Transit Nodes

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

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

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

3.2.2 At an SF/Proxy

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

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

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

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

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

3.2.3 At Other Forwarding Nodes

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

3.3 At Exit/Egress

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

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

7. Acknowledgements

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

Joel Halpern, Xinpeng Wei

Normative References

[RFC2119] - Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3168] - Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC5129] - Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <https://www.rfc-editor.org/info/rfc5129>.

[RFC6040] - Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <http://www.rfc-editor.org/info/rfc6040>.

[RFC7011] - Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

[RFC7567] - Baker, F., Ed., and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

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

[RFC8300] - Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

[TunnelCongFeedback] - Wei, X., Zhu, L., and L. Deng, "Tunnel Congestion Feedback", draft-ietf-tsvwg-tunnel-congestion-feedback, work in progress.

Informative References

[RFC4301] - Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4960] - Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

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

[RFC8311] - Black, D., "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation", RFC 8311, DOI 10.17487/RFC8311, January 2018, <https://www.rfc-editor.org/info/rfc8311>.

[ecnL4S] - De Schepper, K., and B. Briscoe, "Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay (L4S)", draft-ietf-tsvwg-ecn-l4s-id, work in progress.

Authors' Addresses

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

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

Bob Briscoe
Independent
UK

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

Andrew G. Malis
Huawei Technologies

Email: agmalis@gmail.com

INTERNET-DRAFT
Intended status: Proposed Standard

Donald Eastlake
Huawei
Bob Briscoe
Independent
Andrew Malis
Huawei

Expires: August 4, 2019
February 5, 2019

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

Abstract

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

Status of This Memo

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

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html. The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Table of Contents

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 is 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.

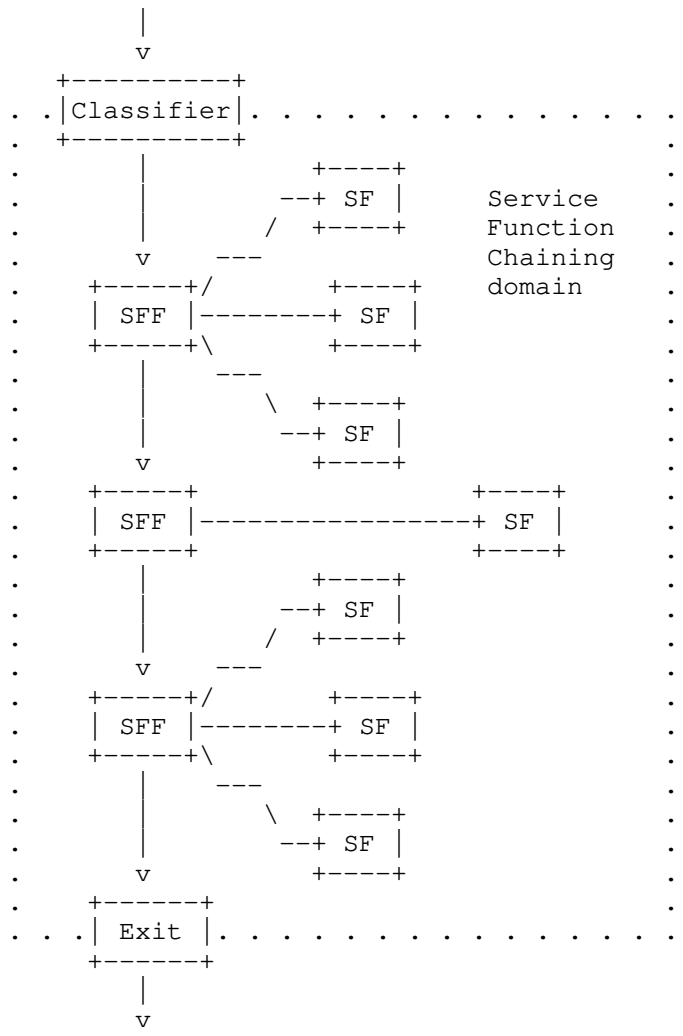


Figure 1. Example SFC Path Forwarding Nodes

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.

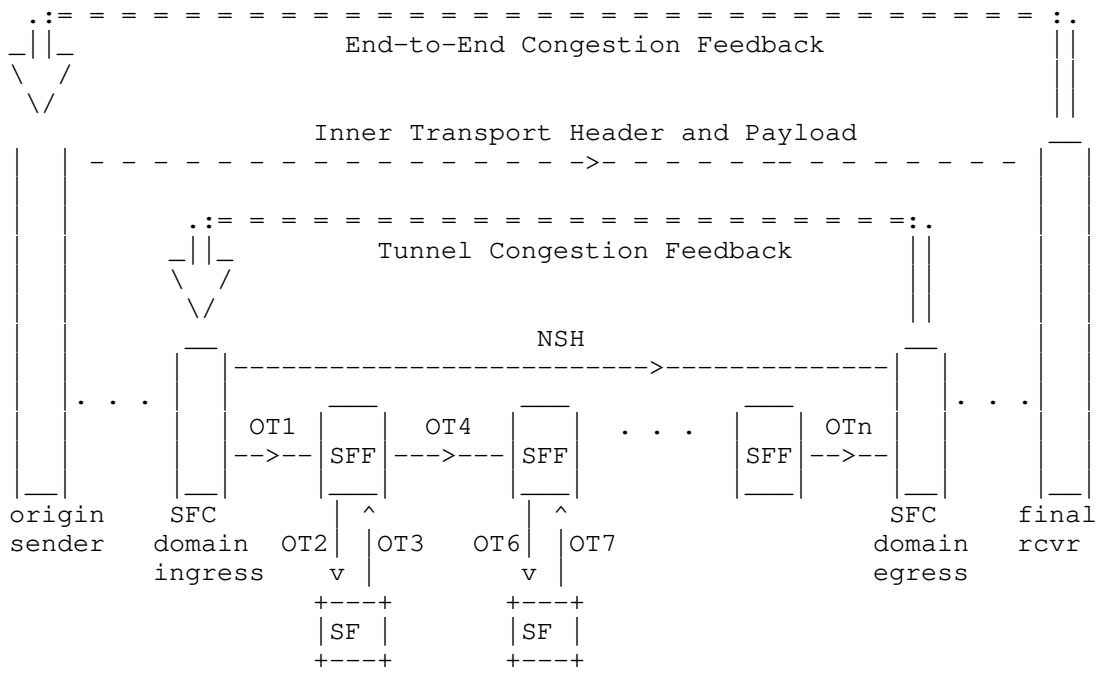


Figure 2: Congestion Feedback across an SFC Domain

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.

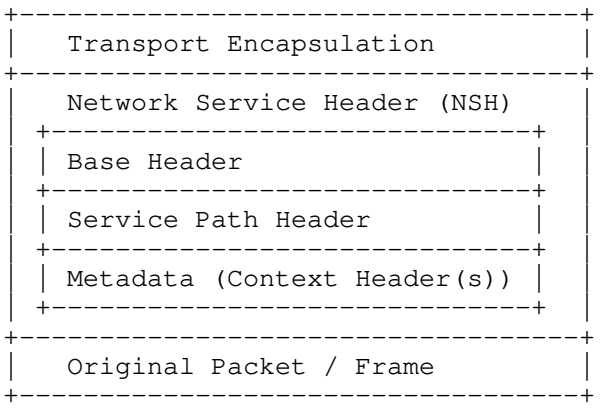


Figure 3. Data Encapsulation with the NSH

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.

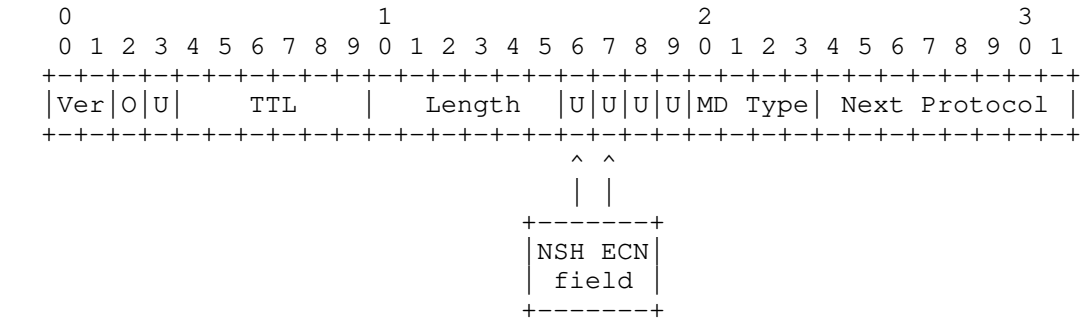


Figure 4: NSH Base Header

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 variatns 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.

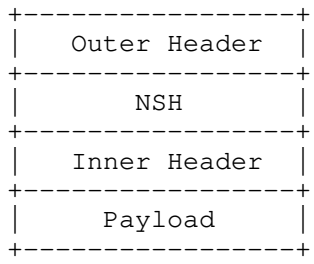


Figure 5. Packet in Transit

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

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

F. Brockners
S. Bhandari
V. Govindan
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JMPC
T. Mizrahi
Marvell
D. Mozes

P. Lapukhov
Facebook
R. Chang
Barefoot Networks
May 31, 2018

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

Abstract

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

This Internet-Draft will expire on December 2, 2018.

Table of Contents

1. Introduction

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

2. Conventions

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

Abbreviations used in this document:

IOAM: In-situ Operations, Administration, and Maintenance

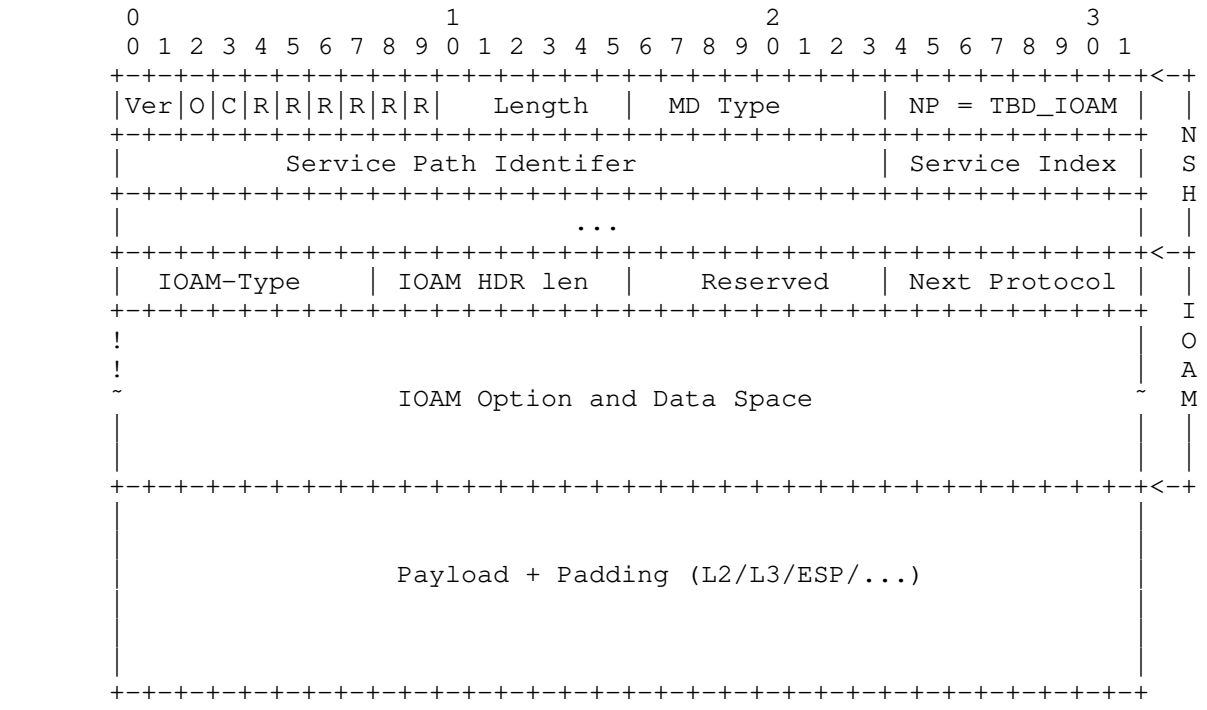
NSH: Network Service Header

OAM: Operations, Administration, and Maintenance

TLV: Type, Length, Value

3. IOAM data fields encapsulation in NSH

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



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.

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 x 4) octets.

2. Encapsulation of IOAM data fields using the "Next Protocol" field. Each IOAM data field option (trace, proof-of-transit, and edge-to-edge) would be specified by its own "next protocol".

3. Encapsulation of IOAM data fields using the "Next Protocol" field. A single NSH protocol type code point would be allocated for IOAM. A "sub-type" field would then specify what IOAM options type (trace, proof-of-transit, edge-to-edge) is carried.

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

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

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

5. IANA Considerations

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

Next Protocol	Description	Reference
x	TBD_IOAM	This document

6. Security Considerations

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

7. Acknowledgements

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

8. References

8.1. Normative References

[I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., daniel.bernier@bell.ca, d., and J. Lemon, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-02 (work in progress), March 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <https://www.rfc-editor.org/info/rfc2784>.

[RFC3232] Reynolds, J., Ed., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, DOI 10.17487/RFC3232, January 2002, <https://www.rfc-editor.org/info/rfc3232>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

8.2. Informative References

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

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

David Mozes

Email: mosesster@gmail.com

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

Email: petr@fb.com

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

SFC
Internet-Draft
Intended status: Standards Track
Expires: December 18, 2020

F. Brockners, Ed.
S. Bhandari, Ed.
Cisco

June 16, 2020

Network Service Header (NSH) Encapsulation for In-situ OAM (IOAM) Data
draft-ietf-sfc-ioam-nsh-04

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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on December 18, 2020.

Table of Contents

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

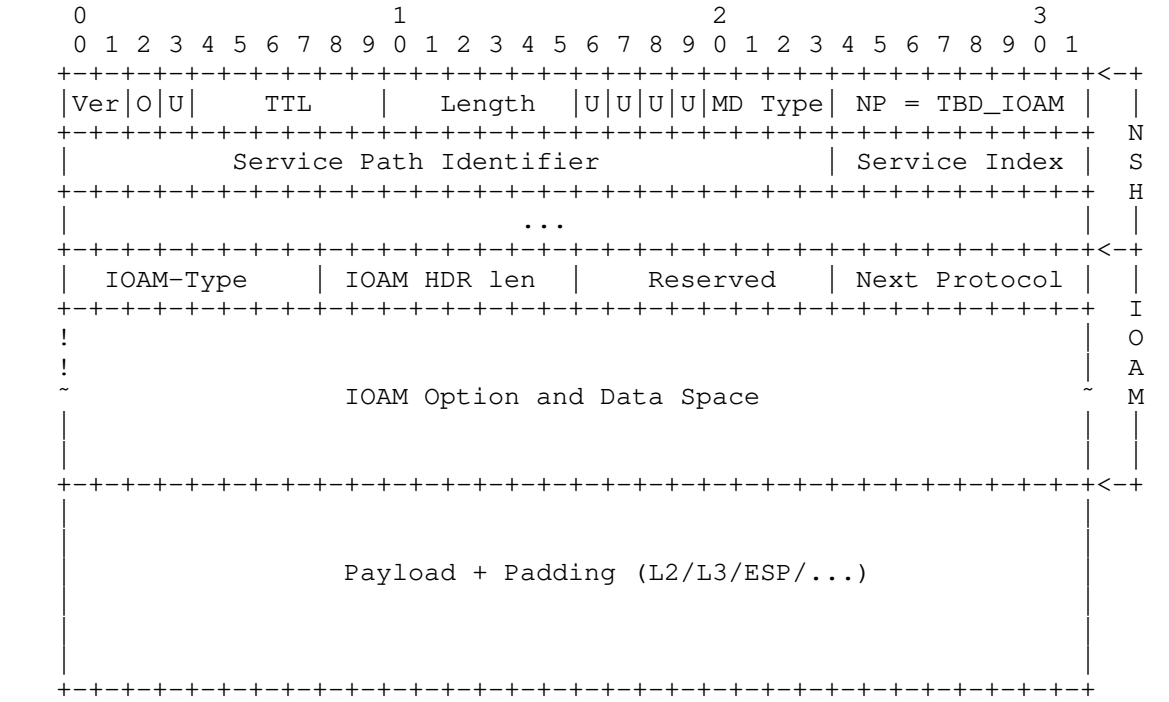
OAM: Operations, Administration, and Maintenance

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. The IOAM-Data-Fields MUST follow the definitions in [I-D.ietf-ippm-ioam-data]. If "proof-of-transit" is used in conjunction with NSH, the implementation of proof of transit MUST follow [I-D.ietf-sfc-proof-of-transit]. 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 the IOAM Option-Type Registry (see 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. The semantics of this field are identical to the Next Protocol field in [RFC8300].

IOAM Option and Data Space: IOAM-Option-Type and IOAM-Data-Field as specified by the IOAM-Type field are present (see Section 4 of [I-D.ietf-ippm-ioam-data]).

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.

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 discusses several approaches for encapsulating IOAM-Data-Fields in 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 x 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.

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. For additional IOAM related security considerations, see Section 8 in [I-D.ietf-ippm-ioam-data]. For proof of transit related security considerations, see Section 7 in [I-D.ietf-sfc-proof-of-transit].

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. 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

9. References

9.1. 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., remy@barefootnetworks.com, r., daniel.bernier@bell.ca, d., and J. Lemon, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-09 (work in progress), March 2020.

[I-D.ietf-sfc-proof-of-transit]
Brockners, F., Bhandari, S., Mizrahi, T., Dara, S., and S. Youell, "Proof of Transit", draft-ietf-sfc-proof-of-transit-05 (work in progress), May 2020.

[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

[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 (editor)
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 4, 2019

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

T. Mizrahi
Marvell
A. Aguado
Universidad Politecnica de Madrid
D. Lopez
Telefonica I+D

October 1, 2018

Proof of Transit
draft-ietf-sfc-proof-of-transit-01

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.

This Internet-Draft will expire on April 4, 2019.

Table of Contents

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 and exact set of pre-determined nodes.

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

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

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

2. Conventions

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

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

SFC: Service Function Chain

SR: Segment Routing

3. Proof of Transit

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

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

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

3.1. Basic Idea

The method relies on adding POT data to all packets that traverse a path. The added POT data allows a verifying node (egress node) to check whether a packet traversed the identified set of nodes on a path correctly or not. Security mechanisms are natively built into the generation of the POT data to protect against misuse (i.e. configuration mistakes, malicious administrators playing tricks with

routing, capturing, spoofing and replaying packets). The mechanism for POT leverages "Shamir's Secret Sharing" scheme [SSS].

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

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

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

3.2. Solution Approach

Solution approach: The overall algorithm uses two polynomials: POLY-1 and POLY-2. POLY-1 is secret and constant. Each node gets a point on POLY-1 at setup-time and keeps it secret. POLY-2 is public, random and per packet. Each node generates a point on POLY-2 each time a packet crosses it. Each node then calculates (point on POLY-1 + point on POLY-2) to get a (point on POLY-3) and passes it to verifier by adding it to each packet. The verifier constructs POLY-3 from the points given by all the nodes and cross checks whether 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".

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

3.2.1. Setup

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

3.2.2. In Transit

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

3.2.3. Verification

The verifier cross checks whether 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.

3.3.1. Basic Version

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

Choices: Prime = 53. POLY-1(x) = (3x^2 + 3x + 10) mod 53. The secret to be re-constructed is the constant coefficient of POLY-1,

i.e., SECRET=10. It is important to note that all operations are done over a finite field (i.e., modulo prime).

3.3.1.1. Secret Shares

The shares of the secret are the points on POLY-1 chosen for the 3 nodes. For example, let x0=2, x1=4, x2=5.

POLY-1(2) = 28 => (x0, y0) = (2, 28)

POLY-1(4) = 17 => (x1, y1) = (4, 17)

POLY-1(5) = 47 => (x2, y2) = (5, 47)

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

3.3.1.2. Lagrange Polynomials

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

l0(x) = (((x-x1) / (x0-x1)) * ((x-x2)/x0-x2))) mod 53 =
(((x-4) / (2-4)) * ((x-5)/2-5))) mod 53 =
(10/3 - 3x/2 + (1/6)x^2) mod 53

l1(x) = (((x-x0) / (x1-x0)) * ((x-x2)/x1-x2))) mod 53 =
(-5 + 7x/2 - (1/2)x^2) mod 53

l2(x) = (((x-x0) / (x2-x0)) * ((x-x1)/x2-x1))) mod 53 =
(8/3 - 2 + (1/3)x^2) mod 53

3.3.1.3. LPC Computation

Since x0=2, x1=4, x2=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(x0) = (10/3) mod 53 = 21

LPC(x1) = (-5) mod 53 = 48

LPC(x2) = (8/3) mod 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

POLY1(x) = l0(x) * y0 + l1(x) * y1 + l2(x) * y2

Subsequently, the SECRET, which is the constant coefficient of POLY1(x) can be computed as below

SECRET = (y0*LPC(l0)+y1*LPC(l1)+y2*LPC(l2)) mod 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 = mod (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. CML represents the cumulative value. It is the POT data in the packet that is updated at each hop with the node's respective (yi*LPC(i)), where i is their respective value.

3.3.1.5. Verification

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

3.3.2. Enhanced Version

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

3.3.2.1. Random Polynomial

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

3.3.2.2. Reconstruction

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

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) mod 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) mod 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) mod 53 = 39 + 16 = 55 mod 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) mod Prime, with Prime = 53 here

VERIFY = (RND-1 + RND-2) mod Prime = (10 + 45) mod 53 = 2

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

3.3.3. Final Version

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

3.4. Operational Aspects

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

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

1. The points (x, y) for each of the nodes on the public and private polynomials are picked such that the x component of the points match. This lends to the LPC values which are used to calculate the cumulative value CML to be constant. Note that the LPC are only depending on the x components. They can be computed at the controller and communicated to the nodes. Otherwise, one would need to distributed the x components to all the nodes.

2. A pre-evaluated portion of the public polynomial for each of the nodes is calculated and added to the POT-profile. Without this all the coefficients of the public polynomial had to be added to the POT profile and each node had to evaluate them. As stated before, the public portion is only the constant coefficient RND value, the pre-evaluated portion for each node should be kept secret as well.

3. To provide flexibility on the size of the cumulative and random numbers carried in the POT data a field to indicate this is shared and interpreted at the nodes.

3.5. Ordered POT (OPOT)

In certain scenarios preserving the order of the nodes traversed by the packet may be needed. Two alternatives, one based on "nested encryption", and another based on "symmetric masking between nodes" are described here for preserving the order

3.5.1. Nested Encryption

1. The controller provisions all the nodes with their respective secret keys.

2. The controller provisions the verifier with all the secret keys of the nodes.

3. For each packet, the ingress node generates a random number RND and encrypts it with its secret key to generate CML value

4. Each subsequent node on the path encrypts CML with their respective secret key and passes it along

5. The verifier is also provisioned with the expected sequence of nodes in order to verify the order

6. The verifier receives the CML, RND values, re-encrypts the RND with keys in the same order as expected sequence to verify.

With this nested encryption approach it is possible to retain the order in which the nodes are traversed, at the cost of:

1. Standard AES encryption would need 128 bits of RND, CML. This results in a 256 bits of additional overhead is added per packet

2. In hardware platforms that do not support native encryption capabilities like (AES-NI). This approach would have considerable impact on the computational latency

3.5.2. Symmetric Masking Between Nodes

1. The controller provisions all the nodes with (or asks them to agree on) a couple of secrets, that we will refer 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 = approx. 2*10^19	approx. 310,000 years
10 Gbps	64	2^64 = approx. 2*10^19	approx. 31,000 years
100 Gbps	64	2^64 = approx. 2*10^19	approx. 3,100 years
1 Gbps	32	2^32 = approx. 4*10^9	2,200 seconds
10 Gbps	32	2^32 = approx. 4*10^9	220 seconds
100 Gbps	32	2^32 = approx. 4*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 3.5.2), 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 those to the nodes. The sum of all parameters for a specific node is referred to as "POT-profile". This document does not define a specific protocol to be used between Controller and nodes. It only defines the procedures and the associated YANG data model.

5.1. Procedure

The Controller creates new POT-profiles at a constant rate and communicates the POT-profile to the nodes. The controller labels a POT-profile "even" or "odd" and the Controller cycles between "even" and "odd" labeled profiles. The rate at which the POT-profiles are

communicated to the nodes is configurable and is more frequent than the speed at which a POT-profile is "used up" (see table above). Once the POT-profile has been successfully communicated to all nodes (e.g., all NETCONF transactions completed, in case NETCONF is used as a protocol), the controller sends an "enable POT-profile" request to the ingress node.

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

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

5.2. YANG Model

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

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

  yang-version 1;

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

  prefix ietf-pot-profile;

  organization "IETF xxx Working Group";

  contact "";

  description "This module contains a collection of YANG
               definitions for proof of transit configuration
               parameters. The model is meant for proof of
               transit and is targeted for communicating the
```

```
                POT-profile between a controller and nodes
                participating in proof of transit.";

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

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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

6. IANA Considerations

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

7. Manageability Considerations

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

8. Security Considerations

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

8.1. Proof of Transit

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

- 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 3.5.2), the nodes need to keep two additional secrets: the upstream 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.

- 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, what 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 to bypass certain nodes in later packets. Such attacks could be avoided by carefully choosing POLY-2 as a (SEQ_NO + RND). For example, if 64 bits are being used for POLY-2 then first 16 bits could be a sequence number SEQ_NO and next 48 bits could be a random number.

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

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

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

8.4. Anti-Preplay

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

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

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

8.5. Anti-Tampering

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

8.6. Recycling

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

If symmetric masking is used for OPOT (Section 3.5.2), 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 3.5.2) is applied using symmetric masking, the Controller will be required to perform a a periodic refresh of the mask pairs. The use of OPOT SHOULD be configurable as part of the required level of assurance through the Controller management interface.

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

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

Email: sadara@cisco.com

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
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.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.

Table of Contents

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 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 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 (Share(POLY-1) + Share(POLY-2)) and CML is updated with this sum, specifically each node performs

CML = CML+(((Share(POLY-1)+ Share(POLY-2)) * LPC) mod 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) mod 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 x0=2, x1=4, x2=5.

POLY-1(2) = 28 => (x0, y0) = (2, 28)

POLY-1(4) = 17 => (x1, y1) = (4, 17)

POLY-1(5) = 47 => (x2, y2) = (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.

```
l0(x) = (((x-x1) / (x0-x1)) * ((x-x2)/x0-x2))) mod 53
      = (((x-4) / (2-4)) * ((x-5)/2-5))) mod 53
      = (10/3 - 3x/2 + (1/6)x^2) mod 53

l1(x) = (((x-x0) / (x1-x0)) * ((x-x2)/x1-x2))) mod 53
      = (-5 + 7x/2 - (1/2)x^2) mod 53

l2(x) = (((x-x0) / (x2-x0)) * ((x-x1)/x2-x1))) mod 53
      = (8/3 - 2 + (1/3)x^2) mod 53
```

3.3.1.3. LPC Computation

Since x0=2, x1=4, x2=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 mod(10/3, 53), mod(-5, 53), mod(8/3, 53).LPC are computed by the Controller and communicated to the individual nodes.

```
LPC(l0) = (10/3) mod 53 = 21

LPC(l1) = (-5) mod 53 = 48

LPC(l2) = (8/3) mod 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

```
POLY1(x) = l0(x) * y0 + l1(x) * y1 + l2(x) * y2
```

Subsequently, the SECRET, which is the constant coefficient of POLY1(x) can be computed as below

```
SECRET = (y0*LPC(l0)+y1*LPC(l1)+y2*LPC(l2)) mod 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 (yi*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 (RND + 7x + 10 x^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 Share(POLY-1). Each node calculates its respective Share(POLY-2) using the RND value retrieved from the packet. The CML reconstruction is enhanced as below. At every node, CML is updated as

```
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) mod 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) mod 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) mod 53 = 39 + 16 = 55 mod 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) mod Prime, with Prime = 53 here

VERIFY = (RND-1 + RND-2) mod Prime = (10 + 45) mod 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 = approx. 2*10^19	approx. 310,000 years
10 Gbps	64	2^64 = approx. 2*10^19	approx. 31,000 years
100 Gbps	64	2^64 = approx. 2*10^19	approx. 3,100 years
1 Gbps	32	2^32 = approx. 4*10^9	2,200 seconds
10 Gbps	32	2^32 = approx. 4*10^9	220 seconds
100 Gbps	32	2^32 = approx. 4*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,..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,..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+..+a_k*z^k, then the SECRET would be a_0.

* bitmask: Number of bits as mask used in controlling the size of the random value generation. 32-bits of mask is default. See Section 4 for details.

5.2.2. Tree Diagram

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

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

5.2.3. YANG Model

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

  prefix "pot";

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

  organization "IETF SFC Working Group";

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

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

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

         Redistribution and use in source and binary forms, with or
         without modification, is permitted pursuant to, and subject to
         the license terms contained in, the Simplified BSD License set
         forth in Section 4.c of the IETF Trust's Legal Provisions
         Relating to IETF Documents
         (https://trustee.ietf.org/license-info).
         This version of this YANG module is part of RFC XXXX
         (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
         itself for full legal notices.
         The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
         'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
         'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
         are to be interpreted as described in BCP 14 (RFC 2119)
         (RFC 8174) when, and only when, they appear in all
         capitals, as shown here.";

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

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

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

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

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

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

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

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

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

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

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

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

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

          uses opot-profile;

        }
      }

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

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

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

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

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

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

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

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

<CODE ENDS>

6. IANA Considerations

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

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

Registrant Contact: The IESG.

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

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

Name: ietf-pot-profile

Maintained by IANA: N

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

Prefix: pot

Reference: RFC XXXX

7. Security Considerations

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

7.1. Proof of Transit

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

* If there are k+1 nodes in the path, the polynomials (POLY-1, POLY-2) should be of degree k. Also k+1 points of POLY-1 are chosen and assigned to each node respectively. The verifier can re-construct the k degree polynomial (POLY-3) only when all the points are correctly retrieved.

* Precisely three values are kept secret by individual nodes. Share of SECRET (i.e. points on POLY-1), Share of POLY-2, LPC, P. Note that only constant coefficient, RND, of POLY-2 is public. x values and non-constant coefficient of POLY-2 are secret

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

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

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

7.2. Cryptanalysis

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

* Inter-Node: A passive attacker observing CML values across nodes (i.e., as the packets entering and leaving), cannot perform differential analysis to construct the points on POLY-1. This is because at each point there are four unknowns (i.e. Share(POLY-1), Share(Poly-2) LPC and prime number P) and three known values (i.e. RND, CML-before, CML-after). The application of symmetric masking for OPOT makes inter-node analysis less feasible.

* Inter-Packets: A passive attacker could observe CML values across packets (i.e., values of PKT-1 and subsequent PKT-2), in order to predict the secrets. Differential analysis across packets could be mitigated using a good PRNG for generating RND. Note that if constant coefficient is a sequence number than CML values become quite predictable and the scheme would be broken. If symmetric masking is used for OPOT, inter-packet analysis could be applied to guess mask values, which requires a proper refresh rate for masks, at least as high as the one used for LPCs.

7.3. Anti-Replay

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

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

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

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

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

7.4. Anti-Preplay

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

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

* Ingress node and Verifier are configured with common pre shared key

* Ingress node generates a Message Authentication Code (MAC) from packet fields using standard HMAC algorithm.

* The left most bits of the output are truncated to desired length to generate RND. It is recommended to use a minimum of 32 bits.

* The verifier regenerates the HMAC from the packet fields and compares with RND. To ensure the POT data is in fact that of the packet.

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

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

7.5. Tampering

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

7.6. Recycling

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

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

7.7. Redundant Nodes and Failover

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

7.8. Controller Operation

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

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

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

7.9. Verification Scope

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

7.9.1. Node Ordering

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

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

7.9.2. Stealth Nodes

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

7.10. POT Yang module

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

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

8. Acknowledgements

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

9. Contributors

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

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

John Leddy
Email: john@leddy.net

David Mozes
Email: mosesster@gmail.com

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

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

10. References

10.1. Normative References

[I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-data-10, 13 July 2020, <http://www.ietf.org/internet-drafts/draft-ietf-ippm-ioam-data-10.txt>.

[I-D.ietf-sfc-ioam-nsh]
Brockners, F. and S. Bhandari, "Network Service Header (NSH) Encapsulation for In-situ OAM (IOAM) Data", Work in Progress, Internet-Draft, draft-ietf-sfc-ioam-nsh-04, 16 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-sfc-ioam-nsh-04.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

[SSS] A., S., "How to share a secret", Communications of the ACM (22): 612-613, 1979.

10.2. Informative References

[I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", Work in Progress, Internet-Draft, draft-ietf-anima-autonomic-control-plane-18, 7 August 2018, <http://www.ietf.org/internet-drafts/draft-ietf-anima-autonomic-control-plane-18.txt>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Authors' Addresses

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

Email: fbrockne@cisco.com

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

IPPM Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2019

G. Mirsky
ZTE Corp.
W. Lingqiang
G. Zhui
ZTE Corporation

October 17, 2018

Hybrid Two-Step Performance Measurement Method
draft-mirsky-ippm-hybrid-two-step-02

Abstract

Development of, and advancements in, automation of network operations brought new requirements for measurement methodology. Among them is the ability to collect instant network state as the packet being processed by the networking elements along its path through the domain. This document introduces a new hybrid measurement method, referred to as hybrid two-step, as it separates the act of measuring and/or calculating the performance metric from the act of collecting and transporting network state.

Table of Contents

1. Introduction

Successful resolution of challenges of automated network operation, as part of, for example, overall service orchestration or data center operation, relies on a timely collection of accurate information that reflects the state of network elements on an unprecedented scale. Because performing the analysis and act upon the collected information requires considerable computing and storage resources, the network state information is unlikely to be processed by the network elements themselves but will be relayed into the data storage facilities, e.g., data lakes. The process of producing, collecting network state information also referred to in this document as network telemetry, and transporting it for post-processing should work equally well with data flows or injected in the network test packets. RFC 7799 [RFC7799] describes a combination of elements of passive and active measurement as a hybrid measurement.

Several technical methods have been proposed to enable collection of network state information instantaneous to the packet processing, among them [P4.INT] and [I-D.ietf-ippm-ioam-data].

This document introduces Hybrid Two-Step (HTS) as a new hybrid measurement method that separates measuring or calculating the performance metric from the collecting and transporting this information. The Hybrid Two-Step method extends the two-step mode of Residence Time Measurement (RTM) defined in [RFC8169] to on-path network state collection and transport.

2. Conventions used in this document

2.1. Terminology

RTM Residence Time Measurement

ECMP Equal Cost Multipath

MTU Maximum Transmission Unit

HTS Hybrid Two-Step

Network telemetry - the process of collecting and reporting of network state

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. Problem Overview

Performance measurements are meant to provide data that characterize conditions experienced by traffic flows in the network and possibly trigger operational changes (e.g., re-route of flows, or changes in resource allocations). Modifications to a network are determined based on the performance metric information available at the time that a change is to be made. The correctness of this determination is based on the quality of the collected metrics data. The quality of collected measurement data is defined by:

- the resolution and accuracy of each measurement;
- predictability of both the time at which each measurement is made and the timeliness of measurement collection data delivery for use.

Consider the case of delay measurement that relies on collecting time of packet arrival at the ingress interface and time of the packet transmission at the egress interface. The method includes recording a local clock value on receiving the first octet of an affected message at the device ingress, and again recording the clock value on transmitting the first byte of the same message at the device egress. In this ideal case, the difference between the two recorded clock times corresponds to the time that the message spent in traversing the device. In practice, the time that has been recorded can differ from the ideal case by any fixed amount and a correction can be applied to compute the same time difference taking into account the known fixed time associated with the actual measurement. In this way, the resulting time difference reflects any variable delay associated with queuing.

Depending on the implementation, it may be a challenge to compute the difference between message arrival and departure times and - on the fly - add the necessary residence time information to the same message. And that task may become even more challenging if the packet is encrypted. Implementations SHOULD NOT record a message departure time that may be significantly inaccurate in the same message, as the result of estimating the departure time that includes the variable time component (such as that associated with buffering and queuing of the message). A similar problem may cause a lower quality of, for example, information that characterizes utilization of the egress interface. If unable to obtain the data consistently, without variable delays for additional processing, information may not accurately reflect the state at the egress interface. To mitigate this problem [RFC8169] defined an RTM two-step mode.

Another challenge associated with methods that collect network state information into the actual data packet is the risk to exceed the Maximum Transmission Unit (MTU) size, especially if the packet traverses overlay domains or VPNs. Since the fragmentation is not available at the transport network, operators may have to reduce MTU size advertised to client layer or risk missing network state data for the part, most probably the latter part, of the path.

4. Theory of Operation

The HTS method consists of the two phases:

- performing a measurement or obtaining network state information, one or more than one type, on a node;
- collecting and transporting the measurement.

HTS uses HTS Trigger carried in a data packet or a specially constructed test packet. Nature of the HTS Trigger is transport network layer specific, and its description is outside the scope of this document. The packet that includes the HTS Trigger in this document also referred to as the trigger packet.

The HTS method uses the HTS Follow-up packet, in this document also referred to as the follow-up packet, to collect measurement and network state data from the nodes. The node that creates the HTS Trigger also generates the HTS Follow-up packet. The follow-up packet contains characteristic information, copied from the trigger packet, sufficient for participating HTS nodes to associate it with the original packet. The exact composition of the characteristic information is specific for each transport network, and its definition is outside the scope of this document. The follow-up packet also uses the same encapsulation as the data packet. If not payload but only network information used to load-balance flows in equal cost multipath (ECMP), use of the network encapsulation identical to the trigger packet should guarantee that the follow-up packet remains in-band, i.e., traverses the same set of network elements, with the original data packet with the HTS Trigger. Only one outstanding follow-up packet MUST be on the node for the given path. That means that if the node receives an HTS Trigger for the flow on which it still waits for the follow-up packet to the previous HTS Trigger, the node will originate the follow-up packet to transport the former set of the network state data and transmit it before it sends the follow-up packet with the latest collection of network state information.

4.1. Operation of the HTS Ingress Node

A node that originates the HTS Trigger is referred to as HTS ingress node. As stated, the ingress node originates the follow-up packet. The follow-up packet has the transport network encapsulation identical with the trigger packet followed by the HTS shim and one or more telemetry information elements encoded as Type-Length-Value {TLV}. Figure 1 displays the example of the follow-up packet format.

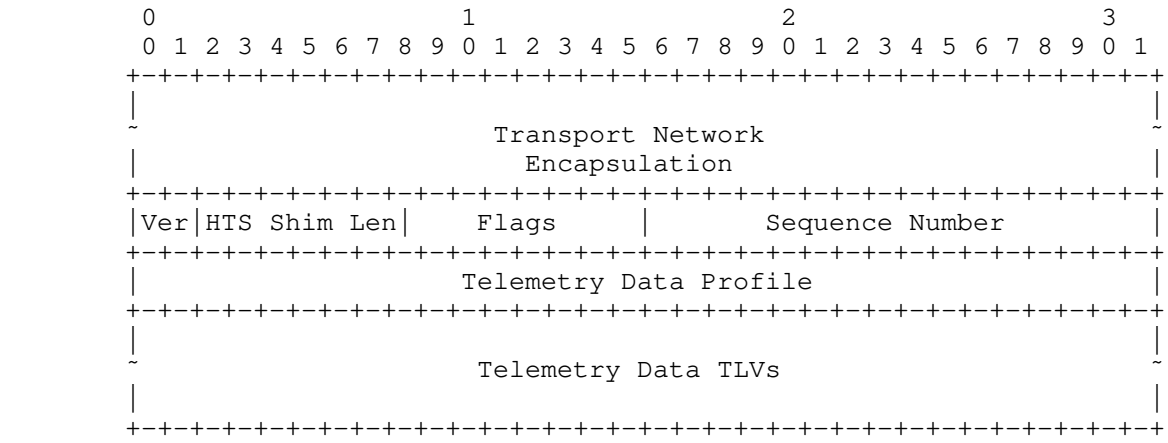


Figure 1: Follow-up Packet Format

Fields of the HTS shim are as follows:

Version (Ver) is the two-bits long field. It specifies the version of the HTS shim format. This document defines the format for the 0b00 value of the field.

HTS Shim Length is the six bits-long field. It defines the length of the HTS shim in bytes. The minimal value of the field is four bytes.

Flags is eight-bits long field. The format of the Flags field displayed in Figure 2.

Full (F) flag MUST be set to zero by the node originating the HTS follow-up packet and MUST be set to one by the node that does not add its telemetry data to avoid exceeding MTU size.

The node originating the follow-up packet MUST zero the Reserved field and ignore it on the receipt.

Sequence Number is 16 bits-long field. The value of the field reflects the number of the HTS follow-up packet in the sequence of the HTS follow-up packets originated in response to the same HTS trigger. The ingress node MUST set the value of the field to zero.

Telemetry Data Profile is the optional variable length field of bit-size flags. Each flag indicates requested type of telemetry data to be collected at the each HTS node. The increment of the field is four bytes with a minimum length of zero.

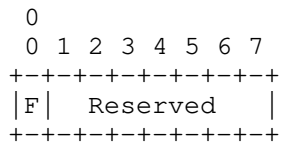


Figure 2: Flags Field Format

4.2. Operation of the HTS Transient Node

Upon receiving the trigger packet the HTS transient node MUST:

- copy the transport information;
- start the HTS Follow-up Timer for the obtained flow.

Upon receiving the follow-up packet the HTS transient node MUST:

- verify that the matching transport information exists and the Full flag is cleared, then stop the associated HTS Follow-up timer;
- collect telemetry data requested in the Telemetry Data Profile field or defined by the local HTS policy;
- if adding the collected telemetry would not exceed MTU, then append data into Telemetry Data TLVs field and transmit the follow-up packet;
- otherwise, set the value of the Full flag to one and transmit the received a follow-up packet;
- originate the new follow-up packet using the same transport information. The value of the Sequence Number field in the HTS shim MUST be set to the value of the field in the received follow-up packet incremented by one. Copy collected telemetry data and transmit the packet.

If the follow-up timer expires the transient node MUST:

- originate the follow-up packet using transport information associated with the expired timer;
- initialize the HTS shim by setting Version field to 0b00 and Sequence Number field to 0. Values of HTS Shim Length and Telemetry Data Profile fields MAY be set according to the local policy.

- copy telemetry information into Telemetry Data TLVs field and transmit the packet.

4.3. Operation of the HTS Egress Node

Upon receiving the trigger packet the HTS egress node MUST:

- copy the transport information;
- start the HTS Collection timer for the obtained flow.

When the egress node receives the follow-up packet for the known flow, i.e., the flow to which the Collection timer is running, the node MUST:

- copy telemetry information;
- restart the corresponding Collection timer.

When the Collection timer expires the egress relays the collected telemetry information for processing and analysis to a local or remote agent.

4.4. Considerations for HTS Timers

This specification defines two timers - HTS Follow-up and HTS Collection. Because for the particular flow there MUST be not more than one HTS Trigger, values of HTS timers bounded by the rate of the trigger generation for that flow.

5. IANA Considerations

TBD

6. Security Considerations

Nodes that practice HTS method are presumed to share a trust model that depends on the existence of a trusted relationship among nodes. This is necessary as these nodes are expected to correctly modify the specific content of the data in the follow-up packet, and the degree to which HTS measurement is useful for network operation depends on this ability. In practice, this means either confidentiality or integrity protection cannot cover those portions of messages that contain the network state data. Though there are methods that make it possible in theory to provide either or both such protections and still allow for intermediate nodes to make detectable yet authenticated modifications, such methods do not seem practical at

present, particularly for protocols that used to measure latency and/or jitter.

The ability to potentially authenticate and/or encrypt the network state data for scenarios both with and without the participation of intermediate nodes that participate in HTS measurement is left for further study.

While it is possible for a supposed compromised node to intercept and modify the network state information in the follow-up packet, this is an issue that exists for nodes in general - for all data that to be carried over the particular networking technology - and is therefore the basis for an additional presumed trust model associated with an existing network.

7. Acknowledgments

Authors express their gratitude and appreciation to Joel Halpern for the most helpful and insightful discussion on the applicability of HTS in a Service Function Chaining domain.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

8.2. Informative 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-03 (work in progress), June 2018.

[P4.INT] "In-band Network Telemetry (INT)", P4.org Specification, October 2017.

[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>.

[RFC8169] Mirsky, G., Ruffini, S., Gray, E., Drake, J., Bryant, S., and A. Vainshtein, "Residence Time Measurement in MPLS Networks", RFC 8169, DOI 10.17487/RFC8169, May 2017, <https://www.rfc-editor.org/info/rfc8169>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Wang Lingqiang
ZTE Corporation
No 19 ,East Huayuan Road
Beijing 100191
P.R.China

Phone: +86 10 82963945
Email: wang.lingqiang@zte.com.cn

Guo Zhui
ZTE Corporation
No 19 ,East Huayuan Road
Beijing 100191
P.R.China

Phone: +86 10 82963945
Email: guo.zhui@zte.com.cn

IPPM Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 7, 2021

G. Mirsky
ZTE Corp.
W. Lingqiang
G. Zhui
ZTE Corporation
H. Song
Futurewei Technologies

December 4, 2020

Hybrid Two-Step Performance Measurement Method
draft-mirsky-ippm-hybrid-two-step-07

Abstract

Development of, and advancements in, automation of network operations brought new requirements for measurement methodology. Among them is the ability to collect instant network state as the packet being processed by the networking elements along its path through the domain. This document introduces a new hybrid measurement method, referred to as hybrid two-step, as it separates the act of measuring and/or calculating the performance metric from the act of collecting and transporting network state.

Table of Contents

1. Introduction

Successful resolution of challenges of automated network operation, as part of, for example, overall service orchestration or data center operation, relies on a timely collection of accurate information that reflects the state of network elements on an unprecedented scale. Because performing the analysis and act upon the collected information requires considerable computing and storage resources, the network state information is unlikely to be processed by the network elements themselves but will be relayed into the data storage facilities, e.g., data lakes. The process of producing, collecting network state information also referred to in this document as network telemetry, and transporting it for post-processing should work equally well with data flows or injected in the network test

packets. RFC 7799 [RFC7799] describes a combination of elements of passive and active measurement as a hybrid measurement.

Several technical methods have been proposed to enable the collection of network state information instantaneous to the packet processing, among them [P4.INT] and [I-D.ietf-ippm-ioam-data]. The instantaneous, i.e., in the data packet itself, collection of telemetry information simplifies the process of attribution of telemetry information to the particular monitored flow. On the other hand, this collection method impacts the data packets, potentially changing their treatment by the networking nodes. Also, the amount of information the instantaneous method collects might be incomplete because of the limited space it can be allotted. Other proposals defined methods to collect telemetry information in a separate packet from each node traversed by the monitored data flow. Examples of this approach to collecting telemetry information are [I-D.ietf-ippm-ioam-direct-export] and [I-D.song-ippm-postcard-based-telemetry]. These methods allow data collection from any arbitrary path and avoid directly impacting data packets. On the other hand, the correlation of data and the monitored flow requires that each packet with telemetry information also includes characteristic information about the monitored flow.

This document introduces Hybrid Two-Step (HTS) as a new method of telemetry collection that improvers accuracy of a measurement by separating the act of measuring or calculating the performance metric from the collecting and transporting this information while minimizing the overhead of the generated load in a network. HTS method extends the two-step mode of Residence Time Measurement (RTM) defined in [RFC8169] to on-path network state collection and transport. HTS allows the collection of telemetry information from any arbitrary path, does not change data packets of the monitored flow and makes the process of attribution of telemetry to the data flow simple.

2. Conventions used in this document

2.1. Acronyms

RTM Residence Time Measurement

ECMP Equal Cost Multipath

MTU Maximum Transmission Unit

HTS Hybrid Two-Step

HMAC Hashed Message Authentication Code

Network telemetry - the process of collecting and reporting of network state

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. Problem Overview

Performance measurements are meant to provide data that characterize conditions experienced by traffic flows in the network and possibly trigger operational changes (e.g., re-route of flows, or changes in resource allocations). Modifications to a network are determined based on the performance metric information available when a change is to be made. The correctness of this determination is based on the quality of the collected metrics data. The quality of collected measurement data is defined by:

- the resolution and accuracy of each measurement;
- predictability of both the time at which each measurement is made and the timeliness of measurement collection data delivery for use.

Consider the case of delay measurement that relies on collecting time of packet arrival at the ingress interface and time of the packet transmission at the egress interface. The method includes recording a local clock value on receiving the first octet of an affected message at the device ingress, and again recording the clock value on transmitting the first byte of the same message at the device egress. In this ideal case, the difference between the two recorded clock times corresponds to the time that the message spent in traversing the device. In practice, the time recorded can differ from the ideal case by any fixed amount. A correction can be applied to compute the same time difference taking into account the known fixed time associated with the actual measurement. In this way, the resulting time difference reflects any variable delay associated with queuing.

Depending on the implementation, it may be a challenge to compute the difference between message arrival and departure times and - on the fly - add the necessary residence time information to the same message. And that task may become even more challenging if the packet is encrypted. Recording the departure of a packet time in the same packet may be decremental to the accuracy of the measurement

because the departure time includes the variable time component (such as that associated with buffering and queuing of the packet). A similar problem may lower the quality of, for example, information that characterizes utilization of the egress interface. If unable to obtain the data consistently, without variable delays for additional processing, information may not accurately reflect the egress interface state. To mitigate this problem [RFC8169] defined an RTM two-step mode.

Another challenge associated with methods that collect network state information into the actual data packet is the risk to exceed the Maximum Transmission Unit (MTU) size, especially if the packet traverses overlay domains or VPNs. Since the fragmentation is not available at the transport network, operators may have to reduce MTU size advertised to the client layer or risk missing network state data for the part, most probably the latter part, of the path.

4. Theory of Operation

The HTS method consists of two phases:

- performing a measurement or obtaining network state information, one or more than one type, on a node;
- collecting and transporting the measurement.

HTS uses HTS Trigger carried in a data packet or a specially constructed test packet. For example, an HTS Trigger could be a packet that has IOAM Option-Type set to the "IOAM Hybrid Two-Step Option-Type" value (TBA1) allocated by IANA (see Section 6.1). The HTS Trigger also includes IOAM Namespace-ID and IOAM-Trace-Type information [I-D.ietf-ippm-ioam-data]. A packet in the flow to which the Alternate-Marking method [RFC8321] is applied can be used as an HTS Trigger. The nature of the HTS Trigger is a transport network layer-specific, and its description is outside the scope of this document. The packet that includes the HTS Trigger in this document is also referred to as the trigger packet.

The HTS method uses the HTS Follow-up packet, referred to as the follow-up packet, to collect measurement and network state data from the nodes. The node that creates the HTS Trigger also generates the HTS Follow-up packet. The follow-up packet contains characteristic information, copied from the trigger packet, sufficient for participating HTS nodes to associate it with the original packet. The exact composition of the characteristic information is specific for each transport network, and its definition is outside the scope of this document. The follow-up packet also uses the same encapsulation as the data packet. If not payload but only network

information used to load-balance flows in equal cost multipath (ECMP), use of the network encapsulation identical to the trigger packet should guarantee that the follow-up packet remains in-band, i.e., traverses the same set of network elements, with the original data packet with the HTS Trigger. Only one outstanding follow-up packet MUST be on the node for the given path. That means that if the node receives an HTS Trigger for the flow on which it still waits for the follow-up packet to the previous HTS Trigger, the node will originate the follow-up packet to transport the former set of the network state data and transmit it before it sends the follow-up packet with the latest collection of network state information.

4.1. Operation of the HTS Ingress Node

A node that originates the HTS Trigger is referred to as the HTS ingress node. As stated, the ingress node originates the follow-up packet. The follow-up packet has the transport network encapsulation identical with the trigger packet followed by the HTS shim and one or more telemetry information elements encoded as Type-Length-Value {TLV}. Figure 1 displays an example of the follow-up packet format.

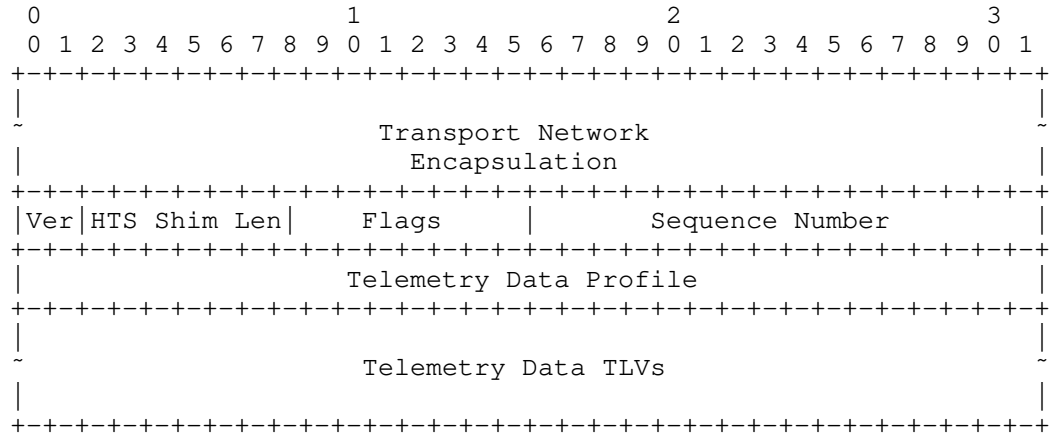


Figure 1: Follow-up Packet Format

Fields of the HTS shim are as follows:

Version (Ver) is the two-bits long field. It specifies the version of the HTS shim format. This document defines the format for the 0b00 value of the field.

HTS Shim Length is the six bits-long field. It defines the length of the HTS shim in bytes. The minimal value of the field is four bytes.

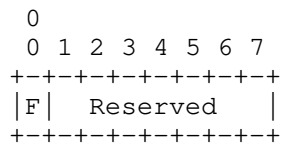


Figure 2: Flags Field Format

Flags is eight-bits long. The format of the Flags field displayed in Figure 2.

Full (F) flag MUST be set to zero by the node originating the HTS follow-up packet and MUST be set to one by the node that does not add its telemetry data to avoid exceeding MTU size.

The node originating the follow-up packet MUST zero the Reserved field and ignore it on the receipt.

Sequence Number is 16 bits-long field. The zero-based value of the field reflects the place of the HTS follow-up packet in the sequence of the HTS follow-up packets that originated in response to the same HTS trigger. The ingress node MUST set the value of the field to zero.

Telemetry Data Profile is the optional variable-length field of bit-size flags. Each flag indicates the requested type of telemetry data to be collected at each HTS node. The increment of the field is four bytes with a minimum length of zero. For example, IOAM-Trace-Type information defined in [I-D.ietf-ippm-ioam-data] can be used in the Telemetry Data Profile field.

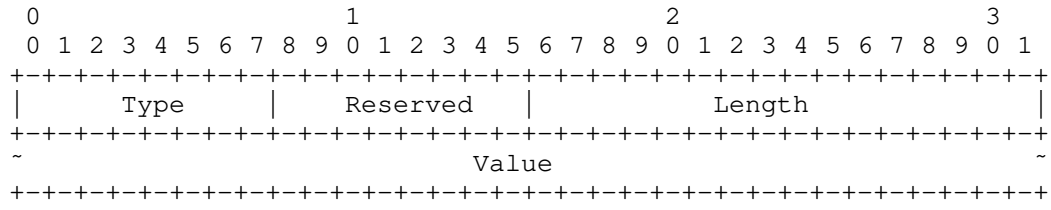


Figure 3: Telemetry Data TLV Format

Telemetry Data TLV is a variable-length field. Multiple TLVs MAY be placed in an HTS packet. Additional TLVs may be enclosed within a given TLV, subject to the semantics of the (outer) TLV in

question. Figure 3 presents the format of a Telemetry Data TLV, where fields are defined as the following:

Type - a one-octet-long field that characterizes the interpretation of the Value field.

Reserved - one-octet-long field.

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

Value - a variable-length field. The value of the Type field determines its interpretation and encoding. IOAM data fields, defined in [I-D.ietf-ippm-ioam-data], MAY be carried in the Value field.

All multibyte fields defined in this specification are in network byte order.

4.2. Operation of the HTS Intermediate Node

Upon receiving the trigger packet, the HTS intermediate node MUST:

- copy the transport information;
- start the HTS Follow-up Timer for the obtained flow.

Upon receiving the follow-up packet, the HTS intermediate node MUST:

- verify that the matching transport information exists and the Full flag is cleared, then stop the associated HTS Follow-up timer;
- collect telemetry data requested in the Telemetry Data Profile field or defined by the local HTS policy;
- if adding the collected telemetry would not exceed MTU, then append data as a new Telemetry Data TLV and transmit the follow-up packet;
- otherwise, set the value of the Full flag to one and transmit the received a follow-up packet;
- originate the new follow-up packet using the same transport information. The value of the Sequence Number field in the HTS shim MUST be set to the value of the field in the received follow-up packet incremented by one;

- copy collected telemetry data into the first Telemetry Data TLV's Value field and then transmit the packet.

If the HTS Follow-up Timer expires, the intermediate node MUST:

- originate the follow-up packet using transport information associated with the expired timer;
- initialize the HTS shim by setting Version field to 0b00 and Sequence Number field to 0. Values of HTS Shim Length and Telemetry Data Profile fields MAY be set according to the local policy.
- copy telemetry information into Telemetry Data TLV's Value field and transmit the packet.

If the intermediate node receives a "late" follow-up packet, i.e., a packet to which the node has no associated HTS Follow-up timer, the node MUST forward the "late" packet.

4.3. Operation of the HTS Egress Node

Upon receiving the trigger packet, the HTS egress node MUST:

- copy the transport information;
- start the HTS Collection timer for the obtained flow.

When the egress node receives the follow-up packet for the known flow, i.e., the flow to which the Collection timer is running, the node for each of Telemetry Data TLVs MUST:

- if HTS is used in the authenticated mode, verify the authentication of the Telemetry Data TLV using the Authentication sub-TLV (see Section 5);
- copy telemetry information from the Value field;
- restart the corresponding Collection timer.

When the Collection timer expires, the egress relays the collected telemetry information for processing and analysis to a local or remote agent.

4.4. Considerations for HTS Timers

This specification defines two timers - HTS Follow-up and HTS Collection. For the particular flow, there MUST be no more than one HTS Trigger, values of HTS timers bounded by the rate of the trigger generation for that flow.

4.5. Deploying HTS in a Multicast Network

Previous sections discussed the operation of HTS in a unicast network. Multicast services are important, and the ability to collect telemetry information is invaluable in delivering a high quality of experience. While the replication of data packets is necessary, replication of HTS follow-up packets is not. Replication of multicast data packets down a multicast tree may be set based on multicast routing information or explicit information included in the special header, as, for example, in Bit-Indexed Explicit Replication [RFC8296]. A replicating node processes the HTS packet as defined below:

- the first transmitted multicast packet MUST be followed by the received corresponding HTS packet as described in Section 4.2;
- each consecutively transmitted copy of the original multicast packet MUST be followed by the new HTS packet originated by the replicating node that acts as an intermediate HTS node when the HTS Follow-up timer expired.

As a result, there are no duplicate copies of Telemetry Data TLV for the same pair of ingress and egress interfaces. At the same time, all ingress/egress pairs traversed by the given multicast packet reflected in their respective Telemetry Data TLV. Consequently, a centralized controller would reconstruct and analyze the state of the particular multicast distribution tree based on HTS packets collected from egress nodes.

5. Authentication in HTS

Telemetry information may be used to drive network operation, closing the control loop for self-driving, self-healing networks. Thus it is critical to provide a mechanism to protect the telemetry information collected using the HTS method. This document defines an optional authentication of a Telemetry Data TLV that protects the collected information's integrity.

The format of the Authentication sub-TLV is displayed in Figure 4.

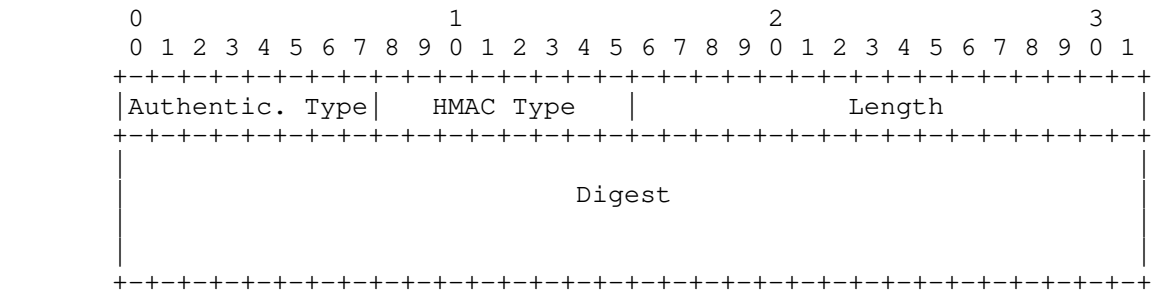


Figure 4: HMAC sub-TLV

where fields are defined as follows:

- Authentication Type - is a one-octet-long field, value TBA2 allocated by IANA Section 6.2.

- Length - two-octet-long field, set equal to the length of the Digest field in octets.

- HMAC Type - is a one-octet-long field that identifies the type of the HMAC and the length of the digest and the length of the digest according to the HTS HMAC Type sub-registry (see Section 6.4).

- Digest - is a variable-length field that carries HMAC digest of the text that includes the encompassing TLV.

This specification defines the use of HMAC-SHA-256 truncated to 128 bits ([RFC4868]) in HTS. Future specifications may define the use in HTS of more advanced cryptographic algorithms or the use of digest of a different length. HMAC is calculated as defined in [RFC2104] over text as the concatenation of the Sequence Number field of the follow-up packet (see Figure 1) and the preceding data collected in the Telemetry Data TLV. The digest then MUST be truncated to 128 bits and written into the Digest field. Distribution and management of shared keys are outside the scope of this document. In the HTS authenticated mode, the Authentication sub-TLV MUST be present in each Telemetry Data TLV. HMAC MUST be verified before using any data in the included Telemetry Data TLV. If HMAC verification fails, the system MUST stop processing corresponding Telemetry Data TLV and notify an operator. Specification of the notification mechanism is outside the scope of this document.

6. IANA Considerations

6.1. IOAM Option-Type for HTS

The IOAM Option-Type registry is requested in [I-D.ietf-ippm-ioam-data]. IANA is requested to allocate a new code point as listed in Table 1.

Value	Description	Reference
TBA1	IOAM Hybrid Two-Step Option-Type	This document

Table 1: IOAM Option-Type for HTS

6.2. HTS TLV Registry

IANA is requested to create the HTS TLV Type registry. All code points in the range 1 through 175 in this registry shall be allocated according to the "IETF Review" procedure specified in [RFC8126]. Code points in the range 176 through 239 in this registry shall be allocated according to the "First Come First Served" procedure specified in [RFC8126]. The remaining code points are allocated according to Table 2:

Value	Description	Reference
0	Reserved	This document
1- 175	Unassigned	This document
176 - 239	Unassigned	This document
240 - 251	Experimental	This document
252 - 254	Private Use	This document
255	Reserved	This document

Table 2: HTS TLV Type Registry

6.3. HTS Sub-TLV Type Sub-registry

IANA is requested to create the HTS sub-TLV Type sub-registry as part of the HTS TLV Type registry. All code points in the range 1 through 175 in this registry shall be allocated according to the "IETF Review" procedure specified in [RFC8126]. Code points in the range 176 through 239 in this registry shall be allocated according to the "First Come First Served" procedure specified in [RFC8126]. The remaining code points are allocated according to Table 3:

Value	Description	Reference
0	Reserved	This document
1- 175	Unassigned	This document
176 - 239	Unassigned	This document
240 - 251	Experimental	This document
252 - 254	Private Use	This document
255	Reserved	This document

Table 3: HTS Sub-TLV Type Sub-registry

This document defines the following new values in the IETF Review range of the HTS sub-TLV Type sub-registry:

Value	Description	TLV Used	Reference
TBA2	HMAC	Any	This document

Table 4: HTS sub-TLV Types

6.4. HMAC Type Sub-registry

IANA is requested to create the HMAC Type sub-registry as part of the HTS TLV Type registry. All code points in the range 1 through 127 in this registry shall be allocated according to the "IETF Review" procedure specified in [RFC8126]. Code points in the range 128 through 239 in this registry shall be allocated according to the "First Come First Served" procedure specified in [RFC8126]. The remaining code points are allocated according to Table 5:

Value	Description	Reference
0	Reserved	This document
1- 127	Unassigned	This document
128 - 239	Unassigned	This document
240 - 249	Experimental	This document
250 - 254	Private Use	This document
255	Reserved	This document

Table 5: HMAC Type Sub-registry

This document defines the following new values in the HMAC Type sub-registry:

Value	Description	Reference
1	HMAC-SHA-256 16 octets long	This document

Table 6: HMAC Types

7. Security Considerations

Nodes that practice the HTS method are presumed to share a trust model that depends on the existence of a trusted relationship among nodes. This is necessary as these nodes are expected to correctly modify the specific content of the data in the follow-up packet, and the degree to which HTS measurement is useful for network operation depends on this ability. In practice, this means either confidentiality or integrity protection cannot cover those portions of messages that contain the network state data. Though there are methods that make it possible in theory to provide either or both such protections and still allow for intermediate nodes to make detectable yet authenticated modifications, such methods do not seem practical at present, particularly for protocols that used to measure latency and/or jitter.

This document defines the use of authentication (Section 5) to protect the integrity of the telemetry information collected using the HTS method. Privacy protection can be achieved by, for example, sharing the IPsec tunnel with a data flow that generates information that is collected using HTS.

While it is possible for a supposed compromised node to intercept and modify the network state information in the follow-up packet; this is an issue that exists for nodes in general - for all data that to be carried over the particular networking technology - and is therefore the basis for an additional presumed trust model associated with an existing network.

8. Acknowledgments

Authors express their gratitude and appreciation to Joel Halpern for the most helpful and insightful discussion on the applicability of HTS in a Service Function Chaining domain.

9. References

9.1. Normative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

9.2. Informative References

[I-D.ietf-ippm-ioam-data] Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-11 (work in progress), November 2020.

[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", draft-ietf-ippm-ioam-direct-export-02 (work in progress), November 2020.

[I-D.song-ippm-postcard-based-telemetry] Song, H., Zhou, T., Li, Z., Mirsky, G., Shin, J., and K. Lee, "Postcard-based On-Path Flow Data Telemetry using Packet Marking", draft-song-ippm-postcard-based-telemetry-08 (work in progress), October 2020.

[P4.INT] "In-band Network Telemetry (INT)", P4.org Specification, October 2017.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <https://www.rfc-editor.org/info/rfc4868>.

[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>.

[RFC8169] Mirsky, G., Ruffini, S., Gray, E., Drake, J., Bryant, S., and A. Vainshtein, "Residence Time Measurement in MPLS Networks", RFC 8169, DOI 10.17487/RFC8169, May 2017, <https://www.rfc-editor.org/info/rfc8169>.

[RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <https://www.rfc-editor.org/info/rfc8296>.

[RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <https://www.rfc-editor.org/info/rfc8321>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Wang Lingqiang
ZTE Corporation
No 19 ,East Huayuan Road
Beijing 100191
P.R.China

Phone: +86 10 82963945
Email: wang.lingqiang@zte.com.cn

Guo Zhui
ZTE Corporation
No 19 ,East Huayuan Road
Beijing 100191
P.R.China

Phone: +86 10 82963945
Email: guo.zhui@zte.com.cn

Haoyu Song
Futurewei Technologies
2330 Central Expressway
Santa Clara
USA

Email: hsong@futurewei.com

SFC Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 13, 2019

G. Mirsky
ZTE Corp.
G. Fioccola
Huawei Technologies
T. Mizrahi
Huawei Network.IO Innovation Lab

October 10, 2018

Performance Measurement (PM) with Alternate Marking Method in Service Function Chaining (SFC) Domain
draft-mirsky-sfc-pmamm-06

Abstract

This document describes how the alternate marking method be used as the passive performance measurement method in a Service Function Chaining (SFC) domain.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 13, 2019.

Copyright Notice

Table of Contents

1. Introduction

[RFC7665] introduced architecture of a Service Function Chain (SFC) in the network and defined its components as classifier, Service Function Forwarder (SFF), and Service Function (SF). [RFC8321] describes the passive performance measurement method, which can be used to measure packet loss, latency, and jitter on live traffic. Because this method is based on marking consecutive batches of packets the method often referred to as Alternate Marking Method (AMM).

This document defines how the alternate marking method can be used to measure packet loss and delay metrics of a service flow over e2e or any segment of the SFC.

2. Conventions used in this document

2.1. Terminology

MM: Marking Method

OAM: Operations, Administration and Maintenance

SFC: Service Function Chain

SF: Service Function

SFF: Service Function Forwarder

SFP: Service Function Path

NSH: Network Service Header

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. Mark Field in NSH Base Header

[RFC8300] defines the format of the Network Service Header (NSH).

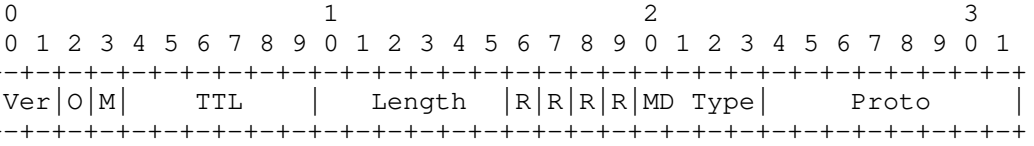


Figure 1: NSH Base format

This document defines the one-bit long field, referred to as Mark field (M in Figure 1, as part of NSH Base and designated for the alternate marking performance measurement method [RFC8321]. The Mark field MUST NOT be used in defining forwarding and/or quality of service treatment of an SFC packet. The Mark field MUST be used only for the performance measurement of data traffic in the SFC layer. Because the setting of the field to any value does not affect forwarding and/or quality of service treatment of a packet, the alternate marking method in SFC layer can be viewed as a real example of passive performance measurement method.

4. Theory of Operation

The marking method can be successfully used in the SFC. Without limiting any generality consider SFC presented in Figure 2. Any combination of markings, Loss and/or Delay, can be applied to a service flow by any component of the SFC at either ingress or egress

point to perform node, link, segment or end-to-end measurement to detect performance degradation defect and localize it efficiently.

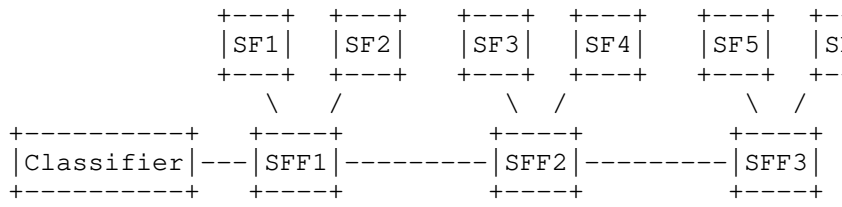


Figure 2: SFC network

Using the marking method a component of the SFC creates distinct sub-flows in the particular service traffic over SFC. Each sub-flow consists of consecutive blocks that are unambiguously recognizable by a monitoring point at any component of the SFC and can be measured to calculate packet loss and/or packet delay metrics.

4.1. Single Mark Enabled Measurement

As explained in the [RFC8321], marking can be applied to delineate blocks of packets based either on the equal number of packets in a block or based on the same time interval. The latter method offers better control as it allows better account for capabilities of downstream nodes to report statistics related to batches of packets and, at the same time, time resolution that affects defect detection interval.

The Loss flag is used to create alternate flows to measure the packet loss by switching the value of the Loss flag every N-th packet or at specified time intervals. Delay metrics MAY be calculated with the alternate flow using any of the following methods:

- First/Last Packet Delay calculation: whenever the marking, i.e., the value of Loss flag, changes a component of the SFC can store the timestamp of the first/last packet of the block. The timestamp can be compared with the timestamp of the packet that arrived in the same order through a monitoring point at a downstream component of the SFC to compute packet delay. Because timestamps collected based on order of arrival, this method is sensitive to packet loss and re-ordering of packets

- Average Packet Delay calculation: an average delay is calculated by considering the average arrival time of the packets within a single block. A component of the SFC may collect timestamps for

each packet received within a single block. Average of the timestamp is the sum of all the timestamps divided by the total number of packets received. Then the difference between averages calculated at two monitoring points is the average packet delay on that segment. This method is robust to out of order packets and also to packet loss (only a small error is introduced). This method only provides a single metric for the duration of the block, and it doesn't give the minimum and maximum delay values. Highly optimized implementation of the method can reduce the duration of the block and thus overcome the limitation.

4.2. Double Mark Enabled Measurement

Double Mark method allows measurement of minimum and maximum delays for the monitored flow, but it requires more nodal and network resources. If the Double Mark method used, then the Loss flag MUST be used to create the alternate flow, i.e., mark larger batches of packets. The Delay flag MUST be used to denote single packets to measure delay jitter.

The first marking (Loss flag alternation) is needed for packet loss and also for average delay measurement. The second marking (Delay flag is put to one) creates a new set of marked packets that are fully identified over the SFC, so that a component can store the timestamps of these packets; these timestamps can be compared with the timestamps of the same packets on another element of the SFC to compute packet delay values for each packet. The number of measurements can be easily increased by changing the frequency of the second marking. But the rate of the second marking must be not too high to avoid out of order issues. This method supports the calculation of not only the average delay but also the minimum and maximum delay values and, in broader terms, to know more about the statistic distribution of delay values.

4.3. Multiplexed Mark Enabled Measurement

There is also a scheme that provides the benefits of Double Mark method, but uses only one bit like Single Mark. This methodology is described in [I-D.mizrahi-ippm-compact-alternate-marking]. The concept is that in the middle of each block of packets with a certain value of the L flag, a single packet has the L flag inverted. So, by examining the stream, the packets with the inverted bit can be easily identified and employed for delay measurement. This Alternate Marking variation is advantageous because it requires only one bit from each packet, and such bits are always in short supply.

4.4. Residence Time Measurement with the Alternate Marking Method

Residence time is the variable part of the propagation delay that a packet experiences while traversing a network, e.g., SFC. Residence Time over an SFC is the sum of the nodal residence times, i.e., periods that the packet spent in each of SFFs that compose the SFC. The nodal residence time in SFC itself is the sum of sub-nodal residence times that the packet spent in each of SFs that are part of the given SFC and are mapped to the SFF. The residence time and deviation of the residence time metrics may include any combination of minimum, maximum, values over measurement period, as well as mean, median, percentile. These metrics may be used to evaluate the performance of the SFC and its elements before and during its operation.

Use of the specially marked packets simplifies residence time measurement and correlation of the measured metrics over the SFC end-to-end. For example, the alternate marking method may be used as described in Section 4.2 to identify packets in the data flow to be used to measure the residence time. The nodal and sub-nodal residence time metrics can be locally calculated and then collected using either in-band or out-band OAM mechanisms.

5. IANA Considerations

5.1. Mark Field in NSH Base Header

This document requests IANA to allocate the one-bit field from NSH Base Header Bits [RFC8300] as the Mark field of NSH as the following:

Bit Position	Description	Reference
TBA	Mark field	This document

Table 1: Mark field of SFC NSH

6. Security Considerations

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

7. Acknowledgment

TBD

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

8.2. Informative References

[I-D.mizrahi-ippm-compact-alternate-marking] Mizrahi, T., Arad, C., Fioccola, G., Cociglio, M., Chen, M., Zheng, L., and G. Mirsky, "Compact Alternate Marking Methods for Passive and Hybrid Performance Monitoring", draft-mizrahi-ippm-compact-alternate-marking-03 (work in progress), October 2018.

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

[RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <https://www.rfc-editor.org/info/rfc8321>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Giuseppe Fioccola
Huawei Technologies

Email: giuseppe.fioccola@huawei.com

Tal Mizrahi
Huawei Network.IO Innovation Lab
Israel

Email: tal.mizrahi.phd@gmail.com

SFC Working Group
Internet-Draft
Intended status: Experimental
Expires: December 31, 2020

G. Mirsky
ZTE Corp.
G. Fioccola
Huawei Technologies
T. Mizrahi
Huawei Network.IO Innovation Lab
June 29, 2020

Performance Measurement (PM) with Alternate Marking Method in Service Function Chaining (SFC) Domain
draft-mirsky-sfc-pmamm-10

Abstract

This document describes how the alternate marking method can be used as the efficient performance measurement method taking advantage of the actual data flows in a Service Function Chaining (SFC) domain.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 31, 2020.

Copyright Notice

Table of Contents

1. Introduction

[RFC7665] introduced the architecture of a Service Function Chain (SFC) in the network and defined its components. These include Classifier, Service Function Forwarder (SFF), Service Function (SF), and Service Function proxy. [I-D.ietf-sfc-oam-framework] provides a reference framework for Operations, Administration and Maintenance (OAM) for SFC. [RFC8321] describes the hybrid performance measurement method, which can be used to measure packet loss, latency, and jitter on live traffic. Because this method is based on marking consecutive batches of packets the method often referred to as Alternate Marking Method (AMM).

This document defines how packet loss and delay metrics of a service flow over end-to-end (Session-Reflector) or any segment of the SFC can be measured using AMM. This document is aligned with the SFC OAM Performance Measurement requirements defined in [I-D.ietf-sfc-oam-framework]. It states that any SFC-aware network device must have the ability to perform loss and delay measurements over the service function chain as a unit, i.e., E2E, or to a specific segment of service function through the SFC. Besides,, AMM can be used in combination with [I-D.ietf-sfc-ioam-nsh] and complement it to achieve the SFC performance measurement objective.

2. Conventions used in this document

2.1. Acronyms

AMM: Alternate Marking Method

OAM: Operations, Administration and Maintenance

SFC: Service Function Chain

SF: Service Function

SFF: Service Function Forwarder

SFP: Service Function Path

NSH: Network Service Header

E2E end-to-end

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. Mark Field in NSH Base Header

[RFC8300] defines the format of the Network Service Header (NSH).

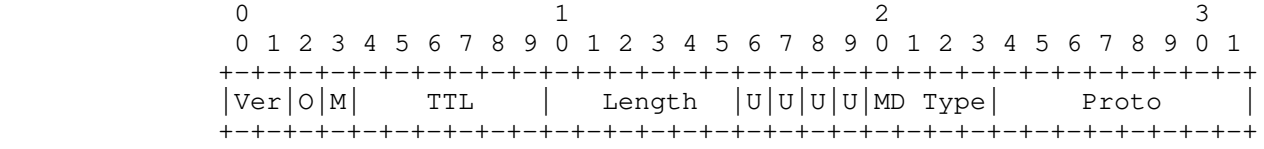


Figure 1: NSH Base format

This document defines the one-bit long field, referred to as Mark field (M in Figure 1, as part of NSH Base and designated for the alternate marking performance measurement method [RFC8321]. The Mark field MUST be set to 0 at initialization of NSH and ignored on the receipt when the method is not in use. The Mark field MUST NOT be used in defining forwarding and/or quality of service treatment of an SFC packet. The Mark field MUST be used only for the performance measurement of data traffic in the SFC layer. Though the setting of

the field to any value likely not affect forwarding and/or quality of service treatment of a packet, the alternate marking method in SFC layer is characterized as an example of a hybrid performance measurement method according to [RFC7799].

4. Theory of Operation

The marking method can be successfully used in the SFC. Without limiting any generality consider SFC presented in Figure 2. Any combination of markings, Loss and/or Delay, can be applied to a service flow by any component of the SFC at either ingress or egress point to perform node, link, segment or E2E measurement to detect performance degradation defect and localize it efficiently.

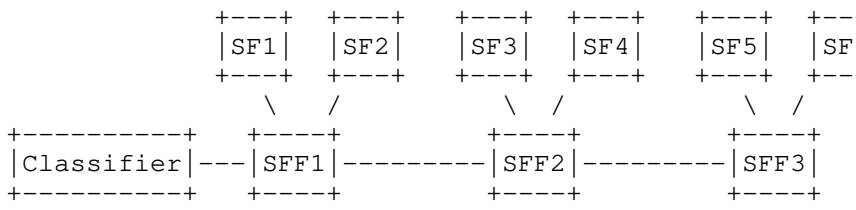


Figure 2: SFC network

Using the marking method, a component of the SFC creates distinct sub-flows in the particular service traffic over SFC. Each sub-flow consists of consecutive blocks that are unambiguously recognizable by a monitoring point at any component of the SFC and can be measured to calculate packet loss and/or packet delay metrics.

4.1. Single Mark Enabled Measurement

As explained in the [RFC8321], marking can be applied to delineate blocks of packets based either on the equal number of packets in a block or based on the same time interval. The latter method offers better control as it allows a better account for capabilities of downstream nodes to report statistics related to batches of packets and, at the same time, time resolution that affects defect detection interval.

The Mark flag is used to create distinctive flows to measure the packet loss by switching the value of the Mark flag every N-th packet or at specified time intervals. Delay metrics MAY be calculated with the alternate flow using any of the following methods:

- First/Last Packet Delay calculation: whenever the marking, i.e., the value of Mark flag changes a component of the SFC can store the timestamp of the first/last packet of the block. The timestamp can be compared with the timestamp of the packet that arrived in the same order through a monitoring point at a downstream component of the SFC to compute packet delay. Because timestamps collected based on the order of arrival, this method is sensitive to packet loss and re-ordering of packets

- Average Packet Delay calculation: an average delay is calculated by considering the average arrival time of the packets within a single block. A component of the SFC may collect timestamps for each packet received within a single block. Average of the timestamp is the sum of all the timestamps divided by the total number of packets received. Then the difference between averages calculated at two monitoring points is the average packet delay on that segment. This method is robust to out of order packets and also to packet loss (only a small error is introduced). This method only provides a single metric for the duration of the block, and it doesn't give the minimum and maximum delay values. Highly optimized implementation of the method can reduce the duration of the block and thus overcome the limitation.

4.2. Multiplexed Mark Enabled Measurement

There is also a scheme that method allows measurement of minimum and maximum delays for the monitored flow using a single marking flag. This methodology is described in [I-D.mizrahi-ippm-compact-alternate-marking]. The concept is that in the middle of each block of packets with a certain value of the M flag, a single packet has the M flag inverted. So, by examining the stream, the packets with the inverted bit can be easily identified and employed for delay measurement. This variation of AMM is advantageous because it requires only one bit from each packet, and such bits are always in short supply.

4.3. Residence Time Measurement with the Alternate Marking Method

Residence time is the variable part of the propagation delay that a packet experiences while traversing a network, e.g., SFC. Residence Time over an SFC is the sum of the nodal residence times, i.e., periods that the packet spent in each of SFFs that compose the SFC. The nodal residence time in SFC itself is the sum of sub-nodal residence times that the packet spent in each of SFs that are part of the given SFC and are mapped to the SFF. The residence time and deviation of the residence time metrics may include any combination of minimum, maximum, values over measurement period, as well as mean, median, percentile. These metrics may be used to evaluate the

performance of the SFC and its elements before and during its operation.

Use of the specially marked packets simplifies residence time measurement and correlation of the measured metrics over the E2E SFC. For example, AMM may be used as described in Section 4.2 to identify packets in the data flow to be used to measure the residence time. The nodal and sub-nodal residence time metrics can be locally calculated and then collected using either in-band or out-band OAM mechanisms.

5. IANA Considerations

5.1. Mark Field in NSH Base Header

This document requests IANA to allocate the one-bit field from NSH Base Header Bits [RFC8300] as the Mark field of NSH as the following:

Bit Position	Description	Reference
TBA	Mark field	This document

Table 1: Mark field of SFC NSH

6. Security Considerations

This document defines the use of AMM in an SFC domain and thus all security considerations specific to SFC discussed in [RFC7665] and [RFC8300] are applicable. By introducing AMM into SFC environment it inherits all seurity considerations discussed in [RFC8321]. A new Mark flag is defined in this specification to be used by AMM. Processing of AMM does require additional computational resources and creates certain amount of state information of per AMM flow perfromance metrics. An implementation MUST provide control over the number of concurrent AMM flows that a node process.

7. Acknowledgment

TBD

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

8.2. Informative References

[I-D.ietf-sfc-ioam-nsh] Brockners, F. and S. Bhandari, "Network Service Header (NSH) Encapsulation for In-situ OAM (IOAM) Data", draft-ietf-sfc-ioam-nsh-04 (work in progress), June 2020.

[I-D.ietf-sfc-oam-framework] Aldrin, S., Pignataro, C., Nainar, N., Krishnan, R., and A. Ghanwani, "Service Function Chaining (SFC) Operations, Administration and Maintenance (OAM) Framework", draft-ietf-sfc-oam-framework-15 (work in progress), May 2020.

[I-D.mizrahi-ippm-compact-alternate-marking] Mizrahi, T., Arad, C., Fioccola, G., Cociglio, M., Chen, M., Zheng, L., and G. Mirsky, "Compact Alternate Marking Methods for Passive and Hybrid Performance Monitoring", draft-mizrahi-ippm-compact-alternate-marking-05 (work in progress), July 2019.

[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>.

[RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <https://www.rfc-editor.org/info/rfc8321>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Giuseppe Fioccola
Huawei Technologies

Email: giuseppe.fioccola@huawei.com

Tal Mizrahi
Huawei Network.IO Innovation Lab
Israel

Email: tal.mizrahi.phd@gmail.com

SFC
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2019

B. Sarikaya
Denpel Informatique
M. Boucadair
Orange
D. von Hugo
Deutsche Telekom
October 16, 2018

Service Function Chaining: Subscriber and Policy Identification Variable-Length Network Service Header (NSH) Context Headers draft-sfc-serviceid-header-01

Abstract

This document discusses how to inform Service Functions about subscriber- and service-related information for the sake of policy enforcement and appropriate service function chaining operations.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 19, 2019.

Copyright Notice

Table of Contents

1. Introduction

This document discusses how to inform Service Functions (SFs) about subscriber- and service-related information when required for the sake of policy enforcement within a single administrative domain. Particularly, subscriber-related information may be required to enforce subscriber-specific SFC-based traffic forwarding policies. Nevertheless, the information carried in packets may not be sufficient to unambiguously identify a subscriber. This document fills this void by specifying a new Network Service Header (NSH) [RFC8300] context header to convey and disseminate such information.

Also, the enforcement of SFC-based differentiated traffic forwarding policies may be inferred by QoS considerations. Typically, QoS information may serve as an input to classification of the Service Function Path (SFP) for path computation, establishment, and selection. Furthermore, the dynamic structuring of service function chains and their subsequent enforcement may be conditioned by QoS requirements that will affect SF instance identification, location, and sequencing. Hence, the need to supply a policy identifier to upstream SFs to appropriately meet the service requirements.

SFs and SF Forwarders (SFFs) involved in a service chain have to contribute to the respective service policy (QoS, for example) requirements characterized by low transmission delay between each other, by exposing a high availability of resources to process function tasks, or by redundancy provided by stand-by machines for seamless execution continuation in case of failures. These requirements may be satisfied by means of control protocols, but in some contexts, (e.g., in networks where resources are very much constrained), carrying QoS-related information directly in packets

may improve the overall SFC operation instead of relying upon the potential complexity or adding overhead introduced by some SFC control plane features. This information is typically included as metadata in the NSH as the SFC encapsulation to provide the SFP identification.

The context information defined in this document can be applicable in the context of mobile networks (typically, in the 3GPP defined (S)Gi Interface) [I-D.ietf-sfc-use-case-mobility]. Because of the widespread use of private addressing in those networks, if SFs to be invoked are located after a NAT function (that can reside in the Packet Data Network (PDN) Gateway (PGW) or in a distinct node), the identification based on the internal IP address is not anymore possible once the NAT has been crossed. As such, means to allow passing the internal information may optimise packet traversal within an SFC-enabled mobile network domain. Furthermore, some SFs that are not enabled on the PGW may require a subscriber identifier to properly operate.

This document does not make any assumption about the structure of subscriber or policy identifiers; each such identifier is treated as an opaque value by the SFC operations and protocols. The semantics and validation of these identifiers are up to the control plane used for SFC. Expectations to SFC control plane protocols are laid down, e.g., in [RFC8459], but specifications of SFC control plane functionalities are also discussed in, for example, [I-D.ietf-bess-nsh-bgp-control-plane], [I-D.wu-pce-traffic-steering-sfc], or [I-D.maglione-sfc-nsh-radius].

The use cases considered in this document assume the NSH is used exclusively within a single administrative domain.

This document adheres to the architecture defined in [RFC7665]. This document assumes the reader is familiar with [RFC8300].

2. Conventions and Terminology

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

The reader should be familiar with the terms defined in [RFC7665].

3. Subscriber Identification NSH Variable-Length Context Header

Subscriber Identifier is defined as an optional variable-length NSH context header. Its structure is shown in Figure 1.

The subscriber identifier is used to convey an identifier already assigned by the service provider to uniquely identify a subscriber. This header conveys an opaque subscriber Identifier that can be used by the service functions to enforce per-subscriber policies.

The classifier and SFC-aware SFs MAY be instructed via a control interface to inject or strip a subscriber identifier context header. Also, the data to be injected in such header SHOULD be configured to nodes authorized to inject such headers. Failures to inject such headers SHOULD be logged locally while a notification alarm MAY be sent to a Control Element. The details of sending notification alarms (i.e., the parameters affecting the transmission of the notification alarms depend on the information in the context header such as frequency, thresholds, and content in the alarm (full header, header ID, timestamp), etc.) SHOULD be configurable by the control plane.

This document adheres to the recommendations in [RFC8300] for handling the context headers at both ingress and egress SFC boundary nodes. That is, to strip such context headers. Revealing any personal and subscriber-related information to third parties is avoided by design to prevent privacy breaches in terms of user tracking.

SFC-aware SFs and proxies MAY be instructed to strip a subscriber identifier context header from the packet or to pass the data to the next SF in the service chain after processing the content of the context headers. If no instruction is provided, the default behavior is to maintain such context headers so that the information can be passed to next SFC-aware hops.

SFC-aware SFs MAY be instructed via the control plane about the validation checks to run on the content of these context headers (e.g., accept only some lengths) and the behavior to adopt. For example, SFC-aware SFs may be instructed to ignore the context header, to remove the context header from the packet, etc. Nevertheless, this specification does not require nor preclude such additional validation checks. These validation checks are deployment-specific. If validation checks fail on a subscriber identifier context header, an SFC-aware SF MUST ignore that context header. The event SHOULD be logged locally while a notification alarm MAY be sent to a Control Element if the SFC-aware SF is instructed to do so.

Multiple subscriber Identifier context TLVs MAY be present in the NSH each carrying a distinct opaque value but all pointing to the same subscriber.

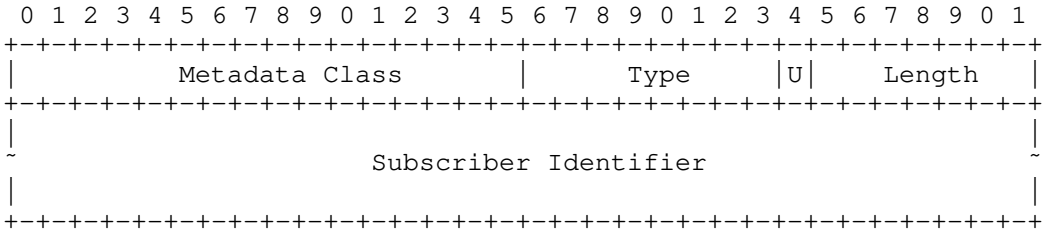


Figure 1: Subscriber Identifier Variable-Length Context Header

The description of the fields is as follows:

- Metadata Class: MUST be set to 0x0 [RFC8300].
- Type: TBD1 (See Section 5)
- Subscriber Identifier: Carries an opaque subscriber identifier.

4. Policy Identification NSH Variable-Length Context Headers

Dedicated service-specific performance identifier is defined to differentiate between services requiring specific treatment to exhibit a performance characterized by, e.g., ultra-low latency (ULL) or ultra-high reliability (UHR). These parameters are related to policy identifier, among others. They are contained in the policy identifier context header. The policy identifier thus allows for the enforcement of a per-service policy such as a service classification function to only consider specific SFs instances during service function path establishment. Details of this process are implementation-specific. For illustration purposes, the classifier may retrieve the details of usable SFs based upon the corresponding service identifier. Typical criteria for instantiating specific SFs include location, performance, or proximity considerations. For UHR services, the stand-by operation of back-up capacity or the deployment of multiple SF instances may be requested.

In other words, the classifier uses this kind of information to decide about the set of SFFs to invoke to honor the latency or reliability requirement (e.g., compute an Rendered Service Path (RSP), or insert a pointer to be shared with involved SFFs). Then, the policy identifier is inserted in the packet so that upstream SFC-aware nodes can make use of the information for proper distributed SFC path selection and SF instance selection.

Policy identifier is defined as optional variable length context header. Its structure is shown in Figure 2.

Similar control plane considerations as those discussed in Section 3 are to be followed.

Multiple policy identifier context headers MAY be present in the NSH; each carrying a distinct opaque value but all are pointing to policies that need to be enforced for a flow.

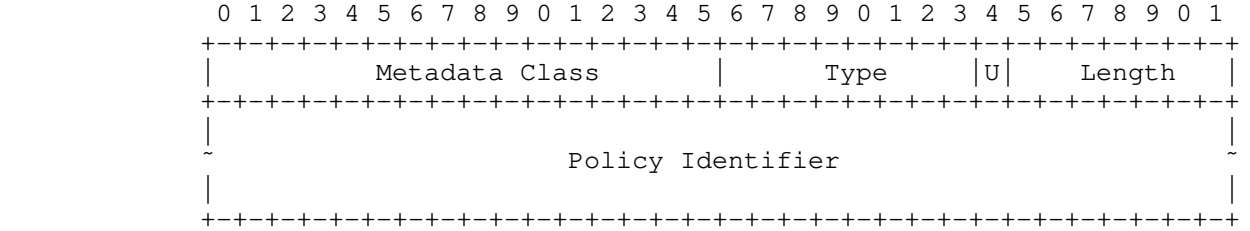


Figure 2: Policy Identifier Variable-Length Context Header

The description of the fields is as follows:

- Metadata Class: MUST be set to 0x0 [RFC8300].
- Type: TBD2 (See Section 5)
- Policy Identifier: Represents an opaque value pointing to specific policy to be enforced. The structure and semantic of this filed is deployment-specific.

5. IANA Considerations

This document requests IANA to assign the following types from the "NSH IETF- Assigned Optional Variable-Length Metadata Types" (0x0000 IETF Base NSH MD Class) registry available at: https://www.iana.org/assignments/nsh/nsh.xhtml#optional-variable-length-metadata-types.

Value	Description	Reference
TBD1	Subscriber Identifier	[ThisDocument]
TBD2	Policy Identifier	[ThisDocument]

6. Security Considerations

Data plane SFC-related security considerations, including privacy, are discussed in [RFC7665] and [RFC8300].

Nodes that are involved in an SFC-enabled domain are assumed to be trusted ([RFC8300]). Means to check that only authorized nodes are solicited when a packet is crossing an SFC-enabled domain.

7. Acknowledgements

Comments from Joel Halpern on a previous version and by Carlos Bernardos are appreciated. Contributions and review by Christian Jacquenet, Danny Lachos, Debashish Purkayastha, and Christian Esteve Rothenberg are thankfully acknowledged.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

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

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

8.2. Informative References

[I-D.ietf-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-04 (work in progress), July 2018.

[I-D.ietf-sfc-use-case-mobility]
Haeffner, W., Napper, J., Stiemerling, M., Lopez, D., and J. Uttaro, "Service Function Chaining Use Cases in Mobile Networks", draft-ietf-sfc-use-case-mobility-08 (work in progress), May 2018.

[I-D.maglione-sfc-nsh-radius]
Maglione, R., Trueba, G., and C. Pignataro, "RADIUS Attributes for NSH", draft-maglione-sfc-nsh-radius-01 (work in progress), October 2016.

[I-D.wu-pce-traffic-steering-sfc]
Wu, Q., Dhody, D., Boucadair, M., Jacquenet, C., and J. Tantsura, "PCEP Extensions for Service Function Chaining (SFC)", draft-wu-pce-traffic-steering-sfc-12 (work in progress), June 2017.

[RFC8459] Dolson, D., Homma, S., Lopez, D., and M. Boucadair, "Hierarchical Service Function Chaining (hSFC)", RFC 8459, DOI 10.17487/RFC8459, September 2018, <https://www.rfc-editor.org/info/rfc8459>.

Authors' Addresses

Behcet Sarikaya
Denpel Informatique

Email: sarikaya@ieee.org

Mohamed Boucadair
Orange
Rennes 3500
France

Email: mohamed.boucadair@orange.com

Dirk von Hugo
Deutsche Telekom
Deutsche-Telekom-Allee 7
D-64295 Darmstadt
Germany

Email: Dirk.von-Hugo@telekom.de

SFC
Internet-Draft
Intended status: Standards Track
Expires: May 18, 2019

M. Boucadair
Orange
D. von Hugo
Deutsche Telekom
B. Sarikaya
Denpel Informatique

November 14, 2018

Service Function Chaining: Subscriber and Policy Identification Variable-Length Network Service Header (NSH) Context Headers draft-sfc-serviceid-header-02

Abstract

This document discusses how to inform Service Functions about subscriber- and service-related information for the sake of policy enforcement and appropriate service function chaining operations.

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 May 18, 2019.

Table of Contents

1. Introduction

This document discusses how to inform Service Functions (SFs) about subscriber- and service-related information when required for the sake of policy enforcement within a single administrative domain. Particularly, subscriber-related information may be required to enforce subscriber-specific SFC-based traffic policies. Nevertheless, the information carried in packets may not be sufficient to unambiguously identify a subscriber. This document fills this void by specifying a new Network Service Header (NSH) [RFC8300] context header to convey and disseminate such information.

Also, the enforcement of SFC-based differentiated traffic policies may be inferred by QoS considerations. Typically, QoS information may serve as an input to classification of the Service Function Path (SFP) for path computation, establishment, and selection. Furthermore, the dynamic structuring of service function chains and their subsequent enforcement may be conditioned by QoS requirements that will affect SF instance identification, location, and sequencing. Hence, the need to supply a policy identifier to upstream SFs to appropriately meet the service requirements.

SFs and SF Forwarders (SFFs) involved in a service chain have to contribute to the respective service policy (QoS, for example) requirements characterized by low transmission delay between each other, by exposing a high availability of resources to process function tasks, or by redundancy provided by stand-by machines for seamless execution continuation in case of failures. These requirements may be satisfied by means of control protocols, but in some contexts, (e.g., in networks where resources are very much constrained), carrying QoS-related information directly in packets

may improve the overall SFC operation instead of relying upon the potential complexity or adding overhead introduced by some SFC control plane features. This information is typically included as metadata in the NSH as the SFC encapsulation to provide the SFP identification.

The context information defined in this document can be applicable in the context of mobile networks (typically, in the 3GPP defined (S)Gi Interface) [I-D.ietf-sfc-use-case-mobility]. Because of the widespread use of private addressing in those networks, if SFs to be invoked are located after a NAT function (that can reside in the Packet Data Network (PDN) Gateway (PGW) or in a distinct node), the identification based on the internal IP address is not anymore possible once the NAT has been crossed. As such, means to allow passing the internal information may optimise packet traversal within an SFC-enabled mobile network domain. Furthermore, some SFs that are not enabled on the PGW may require a subscriber identifier to properly operate.

This document does not make any assumption about the structure of subscriber or policy identifiers; each such identifier is treated as an opaque value by the SFC operations and protocols. The semantics and validation of these identifiers are up to the control plane used for SFC. Expectations to SFC control plane protocols are laid down, e.g., in [RFC8459], but specifications of SFC control plane functionalities are also discussed in, for example, [I-D.ietf-bess-nsh-bgp-control-plane], [I-D.wu-pce-traffic-steering-sfc], or [I-D.maglione-sfc-nsh-radius].

The use cases considered in this document assume the NSH is used exclusively within a single administrative domain.

This document adheres to the architecture defined in [RFC7665]. This document assumes the reader is familiar with [RFC8300].

2. Conventions and Terminology

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

The reader should be familiar with the terms defined in [RFC7665].

3. Subscriber Identification NSH Variable-Length Context Header

Subscriber Identifier is defined as an optional variable-length NSH context header. Its structure is shown in Figure 1.

The subscriber identifier is used to convey an identifier already assigned by the service provider to uniquely identify a subscriber. This header conveys an opaque subscriber Identifier that can be used by the service functions to enforce per-subscriber policies.

The classifier and SFC-aware SFs MAY be instructed via a control interface to inject or strip a subscriber identifier context header. Also, the data to be injected in such header SHOULD be configured to nodes authorized to inject such headers. Typically, a node can be instructed to insert such data following a type/set scheme (e.g., node X should inject subscriber ID type Y). Other schemes may be envisaged.

Failures to inject such headers SHOULD be logged locally while a notification alarm MAY be sent to a Control Element. The details of sending notification alarms (i.e., the parameters affecting the transmission of the notification alarms depend on the information in the context header such as frequency, thresholds, and content in the alarm (full header, header ID, timestamp), etc.) SHOULD be configurable by the control plane.

This document adheres to the recommendations in [RFC8300] for handling the context headers at both ingress and egress SFC boundary nodes. That is, to strip such context headers. Revealing any personal and subscriber-related information to third parties is avoided by design to prevent privacy breaches in terms of user tracking.

SFC-aware SFs and proxies MAY be instructed to strip a subscriber identifier context header from the packet or to pass the data to the next SF in the service chain after processing the content of the context headers. If no instruction is provided, the default behavior is to maintain such context headers so that the information can be passed to next SFC-aware hops.

SFC-aware SFs MAY be instructed via the control plane about the validation checks to run on the content of these context headers (e.g., accept only some lengths) and the behavior to adopt. For example, SFC-aware SFs may be instructed to ignore the context header, to remove the context header from the packet, etc. Nevertheless, this specification does not require nor preclude such additional validation checks. These validation checks are deployment-specific. If validation checks fail on a subscriber

identifier context header, an SFC-aware SF MUST ignore that context header. The event SHOULD be logged locally while a notification alarm MAY be sent to a Control Element if the SFC-aware SF is instructed to do so.

Multiple subscriber Identifier context TLVs MAY be present in the NSH each carrying a distinct opaque value but all pointing to the same subscriber. When multiple subscriber Identifier context TLVs are present and an SF is instructed to strip the subscriber Identifier context header, that SF has to remove all subscriber Identifier context TLVs.

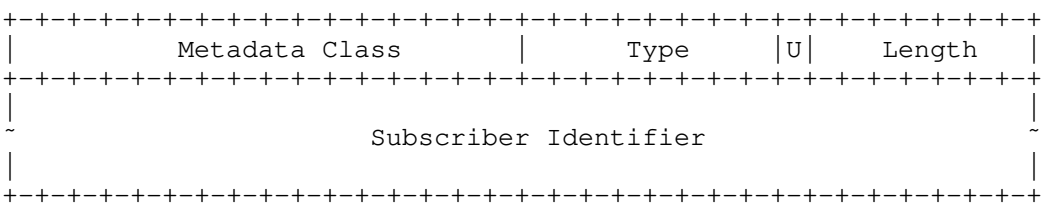


Figure 1: Subscriber Identifier Variable-Length Context Header

The description of the fields is as follows:

- Metadata Class: MUST be set to 0x0 [RFC8300].
- Type: TBD1 (See Section 5)
- Subscriber Identifier: Carries an opaque subscriber identifier.

4. Policy Identification NSH Variable-Length Context Headers

Dedicated service-specific performance identifier is defined to differentiate between services requiring specific treatment to exhibit a performance characterized by, e.g., ultra-low latency (ULL) or ultra-high reliability (UHR). Other policies can be considered when instantiating a service function chain within an SFC-enabled domain. They are contained in the Policy Identifier context header.

The policy identifier is inserted in an NSH packet so that upstream SFC-aware nodes can make use of the information for proper distributed SFC path selection, SF instance selection, or policy selection at SFs.

Thus, the policy identifier allows for the distributed enforcement of a per-service policy such as a service function path to only include specific SFs instances. Details of this process are implementation-

specific. For illustration purposes, an SFF may retrieve the details of usable SFs based upon the corresponding policy identifier. Typical criteria for instantiating specific SFs include location, performance, or proximity considerations. For the particular case of UHR services, the stand-by operation of back-up capacity or the deployment of multiple SF instances may be requested.

Policy identifier is defined as optional variable length context header. Its structure is shown in Figure 2.

Similar control plane considerations as those discussed in Section 3 are to be followed.

Multiple policy identifier context headers MAY be present in the NSH; each carrying a distinct opaque value but all are pointing to policies that need to be enforced for a flow. It is up to the control plane to ensure that these policies are not conflicting. When such conflict is detected by an SFC-aware node, the default behavior of the node is to discard the packet and send a notification alarm to a Control Element.

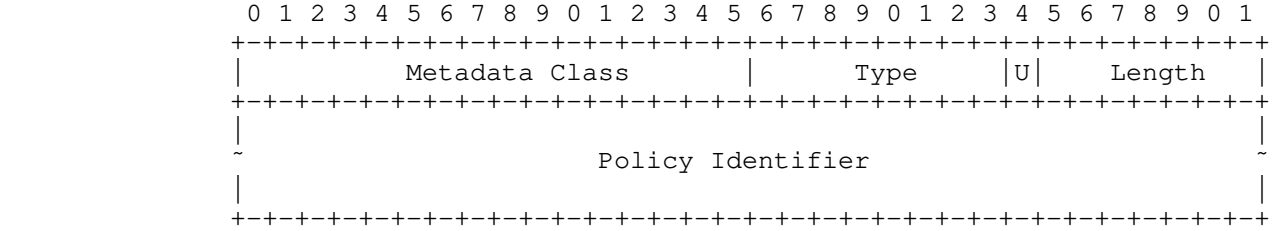


Figure 2: Policy Identifier Variable-Length Context Header

The description of the fields is as follows:

- Metadata Class: MUST be set to 0x0 [RFC8300].
- Type: TBD2 (See Section 5)
- Policy Identifier: Represents an opaque value pointing to specific policy to be enforced. The structure and semantic of this filed is deployment-specific.

5. IANA Considerations

This document requests IANA to assign the following types from the "NSH IETF- Assigned Optional Variable-Length Metadata Types" (0x0000 IETF Base NSH MD Class) registry available at:

https://www.iana.org/assignments/nsh/nsh.xhtml#optional-variable-length-metadata-types.

Value	Description	Reference
TBD1	Subscriber Identifier	[ThisDocument]
TBD2	Policy Identifier	[ThisDocument]

6. Security Considerations

Data plane SFC-related security considerations, including privacy, are discussed in [RFC7665] and [RFC8300].

Nodes that are involved in an SFC-enabled domain are assumed to be trusted ([RFC8300]). Means to check that only authorized nodes are solicited when a packet is crossing an SFC-enabled domain.

An SF maintaining logs for operational reasons MUST NOT log the content of subscriber identifier context header received in NSH packets if the SF does not use the content of that header for its operation.

7. Acknowledgements

Comments from Joel Halpern on a previous version and by Carlos Bernardos are appreciated. Contributions and review by Christian Jacquenet, Danny Lachos, Debashish Purkayastha, Christian Esteve Rothenberg, and Kyle Larose are thankfully acknowledged.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

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

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

8.2. Informative References

[I-D.ietf-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-04 (work in progress), July 2018.

[I-D.ietf-sfc-use-case-mobility] Napper, J., Stiemerling, M., Lopez, D., and J. Uttaro, "Service Function Chaining Use Cases in Mobile Networks", draft-ietf-sfc-use-case-mobility-08 (work in progress), May 2018.

[I-D.maglione-sfc-nsh-radius] Maglione, R., Trueba, G., and C. Pignataro, "RADIUS Attributes for NSH", draft-maglione-sfc-nsh-radius-01 (work in progress), October 2016.

[I-D.wu-pce-traffic-steering-sfc] Wu, Q., Dhody, D., Boucadair, M., Jacquenet, C., and J. Tantsura, "PCEP Extensions for Service Function Chaining (SFC)", draft-wu-pce-traffic-steering-sfc-12 (work in progress), June 2017.

[RFC8459] Dolson, D., Homma, S., Lopez, D., and M. Boucadair, "Hierarchical Service Function Chaining (hSFC)", RFC 8459, DOI 10.17487/RFC8459, September 2018, <https://www.rfc-editor.org/info/rfc8459>.

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 3500
France

Email: mohamed.boucadair@orange.com

Dirk von Hugo
Deutsche Telekom
Deutsche-Telekom-Allee 7
D-64295 Darmstadt
Germany

Email: Dirk.von-Hugo@telekom.de

Behcet Sarikaya
Denpel Informatique

Email: sarikaya@ieee.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 20, 2019

D. Trossen
D. Purkayastha
A. Rahman
InterDigital Communications, LLC
October 17, 2018

Name-Based Service Function Forwarder (nSFF) component within SFC framework
draft-trossen-sfc-name-based-sff-01

Abstract

Many stringent requirements are imposed on today's network, such as low latency, high availability and reliability in order to support several use cases such as IoT, Gaming, Content distribution, Robotics etc. Adoption of cloud and fog technology at the edge of the network allows operator to deploy a single "Service Function" to multiple "Execution locations". The decision to steer traffic to a specific location may change frequently based on load, proximity etc. Under the current SFC framework, steering traffic dynamically to the different execution end points require a specific 're-chaining', i.e., a change in the service function path reflecting the different IP endpoints to be used for the new execution points. In order to address this, we discuss separating the logical Service Function Path from the specific execution end points. This can be done by identifying the Service Functions using a name rather than a routable IP endpoint (or Layer 2 address). This draft describes the necessary extensions, additional functions and protocol details in SFF to handle name based relationships.

Table of Contents

1. Introduction

The requirements on today's networks are very diverse, enabling multiple use cases such as IoT, Content Distribution, Gaming and Network functions such as Cloud RAN and 5G control planes based on a service-based architecture . These services are deployed, provisioned and managed using Cloud based techniques as seen in the IT world.

Virtualization of compute and storage resources is at the heart of providing (often web) services to end users with the ability to quickly provisioning such virtualized service endpoints through, e.g., container based techniques. This creates a dynamicity with the capability to dynamically compose new services from available services as well as move a service instance in response to user mobility or resource availability where desirable. When moving from a pure 'distant cloud' model to one of localized micro data centers with regional, metro or even street level, often called 'edge' data centers, such virtualized service instances can be instantiated in topologically different locations with the overall 'distant' data center now being transformed into a network of distributed ones.

The Service Function Chaining (SFC) framework [RFC7665] allows network operators as well as service providers to compose new services by chaining individual "Service Functions". Such chains are expressed through explicit relationships of functional components (the service functions), realized through their direct Layer 2 (e.g., MAC address) or Layer 3 (e.g., IP address) relationship as defined through next hop information that is being defined by the network operator, see Section 3 for more background on SFC.

In a dynamic service environment of distributed data centers as the one outlined above, with the ability to create and recreate service endpoints frequently, the SFC framework requires to reconfigure the existing chain through information based on the new relationships, causing overhead in a number of components, specifically the orchestrator that initiates the initial service function chain and any possible reconfiguration.

This document describes how such changes can be handled without involving the initiation of new and reconfigured SFCs by lifting the chaining relationship from Layer 2 and 3 information to that of service function 'names', such as names for instance being expressed as URIs. In order to transparently support such named relationships, we propose to embed the necessary functionality directly into the Service Function Forwarder (SFF). With that, the SFF described in this document allows for keeping an existing SFC intact, as described by its service function path (SFP), while enabling the selection of an appropriate service function endpoint(s) during the traversal of packets through the SFC.

2. Example use case: 5G control plane services

We exemplify the need for chaining service functions at the level of a service name through a use case stemming from the current 3GPP Rel 16 work on Service Based Architecture (SBA) [_3GPP_SBA], [_3GPP_SBA_ENHANCEMENT]. In this work, mobile network control planes

are proposed to be realized by replacing the traditional network function interfaces with a fully service-based one. HTTP was chosen as the application layer protocol for exchanging suitable service requests [_3GPP_SBA]. With this in mind, the exchange between, say the session management function (SMF) and the authentication management function (AMF) in a 5G control plane is being described as a set of web service like requests which are in turn embedded into HTTP requests. Hence, interactions in a 5G control plane can be modelled based on service function chains where the relationship is between the specific (IP-based) service function endpoints that implement the necessary service endpoints in the SMF and AMF. The service functions are exposed through URIs with work ongoing to define the used naming conventions for such URIs.

This move from a network function model (in pre-Rel 15 systems of 3GPP) to a service-based model is motivated through the proliferation of data center operations for mobile network control plane services. In other words, typical IT-based methods to service provisioning, in particular that of virtualization of entire compute resources, are envisioned to being used in future operations of mobile networks. Hence, operators of such future mobile networks desire to virtualize service function endpoints and direct (control plane) traffic to the most appropriate current service instance in the most appropriate (local) data centre, such data centre envisioned as being interconnected through a software-defined wide area network (SD-WAN). 'Appropriate' here can be defined by topological or geographical proximity of the service initiator to the service function endpoint. Alternatively, network or service instance compute load can be used to direct a request to a more appropriate (in this case less loaded) instance to reduce possible latency of the overall request. Such data center centric operation is extended with the trend towards regionalization of load through a 'regional office' approach, where micro data centers provide virtualizable resources that can be used in the service execution, creating a larger degree of freedom when choosing the 'most appropriate' service endpoint for a particular incoming service request.

While the move to a service-based model aligns well with the framework of SFC, choosing the most appropriate service instance at runtime requires so-called 're-chaining' of the SFC since the relationships in said SFC are defined through Layer 2 or 3 identifiers, which in turn are likely to be different if the chosen service instances reside in different parts of the network (e.g., in a regional data center).

Hence, when a traffic flow is forwarded over a service chain expressed as an SFC-compliant Service Function Path (SFP), packets in the traffic flow are processed by the various service function

instances, with each service function instance applying a service function prior to forwarding the packets to the next network node. It is a Service layer concept and can possibly work over any Virtual network layer and an Underlay network, possibly IP and any Layer 2 technology. At the service layer, Service Functions are identified using a path identifier and an index. Eventually this index is translated to an IP address (or MAC address) of the host where the service function is running. Because of this, any change of service function instance is likely to require a change of the path information since either IP address (in the case of changing the execution from one data centre to another) or MAC address will change due to the newly selected service function instance.

Returning to our 5G Control plane example, a user's connection request to access an application server in the internet may start with signaling in the Control Plane to setup user plane bearers. The connection request may flow through service functions over a service chain in the Control plane, as deployed by network operator. Typical SFs in a 5G control plane may include "RAN termination / processing", "Slice Selection Function", "AMF" and "SMF". The Classifier, as described in SFC architecture, may reside in the user terminal or at the eNB. These service functions can be configured to be part of a Service Function Chain. We can also say that some of the configuration of the Service Function Path may change at the execution time. E.g. SMF may be relocated as user moves and a new SMF may be included in the Service Function Path based on user location. The following diagram in Figure 3 shows the example Service Function Chain described here.

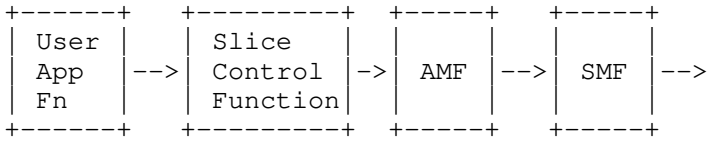


Figure 1: Mapping SFC onto Service Function Execution Points along a Service Function Path

3. Background

[RFC7665] describes an architecture for the specification, creation, and ongoing maintenance of Service Function Chains (SFCs). It includes architectural concepts, principles, and components used in the construction of composite services through deployment of SFCs.

In the following, we outline the parts of this SFC architecture relevant for our proposed extension, followed by the challenges with this current framework in the light of our example use case.

3.1. Relevant part of SFC architecture

SFC Architecture [RFC7665], describes architectural components such as Service Function (SF), Classifier, and Service Function Forwarder (SFF). It describes the Service Function Path (SFP) as the logical path of an SFC. Forwarding traffic along such SFP is the responsibility of the SFF. For this, the SFFs in a network maintain the requisite SFP forwarding information. Such SFP forwarding information is associated with a service path identifier (SPI) that is used to uniquely identify an SFP. The service forwarding state is represented by the Service Index (SI) and enables an SFF to identify which SFs of a given SFP should be applied, and in what order. The SFF also has information that allows it to forward packets to the next SFF after applying local service functions.

The operational steps to forward traffic are then as follows: Traffic arrives at an SFF from the network. The SFF determines the appropriate SF the traffic should be forwarded to via information contained in the SFC encapsulation. After SF processing, the traffic is returned to the SFF, and, if needed, is forwarded to another SF associated with that SFF. If there is another non-local (i.e., different SFF) hop in the SFP, the SFF further encapsulates the traffic in the appropriate network transport protocol and delivers it to the network for delivery to the next SFF along the path. Related to this forwarding responsibility, an SFF should be able to interact with metadata.

3.2. Challenges with current framework

As outlined in previous section, the Service Function Path defines an ordered sequence of specific Service Functions instances being used for the interaction between initiator and service functions along the SFP. These service functions are addressed by IP addresses and defined as next hop information in the network locator maps of traversing SFF nodes.

As outlined in our use case, however, the service provider may want to provision SFC nodes based on dynamically spun up service function instances so that these (now virtualized) service functions can be reached in the SFC domain using the SFC underlay layer.

Following the original model of SFC, any change in a specific execution points for a specific Service Function along the SFP will require a change of the SFP information (since the new service

function execution points likely carries different IP or L2 address information) and possibly even the Next Hop information in SFFs along the SFP. In case the availability of new service function instances is rather dynamic (e.g., through the use of container-based virtualization techniques), the current model and realization of SFC could lead to reducing the flexibility of service providers and increasing the management complexity incurred by the frequent changes of (service) forwarding information in the respective SFF nodes. This is because any change of the SFP (and possibly next hop info) will need to go through suitable management cycles.

To address these challenges through a suitable solution, we identify the following requirements:

- Relations between Service Execution Points MUST be abstracted so that, from an SFP point of view, the PATH never changes.

- Deriving the Service Execution Points from the abstract SFP SHOULD be fast and incur minimum delay.

- Identification of the Service Execution Points SHOULD not use a combination of Layer 2 or Layer 3 mechanism.

The next section outlines a solution to address the issue, allowing for keeping SFC information (represented in its SFP) intact while addressing the desired flexibility of the service provider.

4. Name based operation in SFF

4.1. General Idea

The general idea is two-pronged. Firstly, we elevate the definition of a Service Function Path onto the level of 'name-based interactions' rather than limiting SFPs to Layer 3 or Layer 2 information only. Secondly, we extend the operations of the SFF to allow for forwarding decisions that take into account such name-based interaction while remaining backward compatible to the current SFC architecture [RFC7665]. In the following sections, we outline these two components of our solution.

4.2. Name-Based Service Function Path (nSFP)

In the existing SFC framework [RFC7665], as outlined in Section 3, the SFP information is representing path information based on Layer 2 or 3 information, i.e., MAC or IP addresses , causing the afore mentioned frequent adaptations in cases of execution point changes. Instead, we introduce the notion of a 'name-based service function path (nSFP)'

In today's networking terms, any identifier can be treated as a name but we will illustrate the realization of a "Name based SFP" through extended SFF operations (see Section 5) based on URIs as names and HTTP as the protocol of exchanging information . Here, URIs are being used to name for a Service Function along the nSFP. It is to be noted that the Name based SFP approach is not restricted to HTTP (as the protocol) and URIs (as next hop identifier within the SFP). Other identifiers such as an IP address itself can also be used and are interpreted as a 'name' in the nSFP . With this, our notion of the nSFP goes beyond the initial proposals made in [I-D.purkayastha-sfc-service-indirection], which limited the notion of a 'name' to a URL (uniform resource locator), commonly used in the addressing of HTTP requests. In other words, IP addresses as well as fully qualified domain names forming complex URIs (uniform resource identifiers), such as www.foo.com/service_name1, are all captured by the notion of 'name' in this draft.

Generally, nSFPs are defined as an ordered sequence of the "name" of Service Functions (SF) and a typical name-based Service Function Path may look like: 192.168.x.x -> www.foo.com -> www.foo2.com/service1 -> www.foo2.com/service2

Our use case in Section 2 can then be represented as an ordered named sequence. An example for a session initiation that involves an authentication procedure, this could look like 192.168.x.x -> smf.3gpp.org/session_initiate -> amf.3gpp.org/auth -> smf.3gpp.org/ session_complete -> 192.168.x.x [Note that this example is only a conceptual one, since the exact nature of any future SBA-based exchange of 5G control plane functions is yet to be defined by standardization bodies such as 3GPP]

In accordance with our use case in Section 2, any of these named services can potentially be realized through more than one replicated SF instances. This leads to make dynamic decision on where to send packets along the SAME service function path information, being provided during the execution of the SFC. Through elevating the SFP onto the notion of name-based interactions, the SFP will remain the same even if those specific execution points change for a specific service interaction.

The following diagram describes this name-based SFP concept and the resulting mapping of those named interactions onto (possibly) replicated instances.

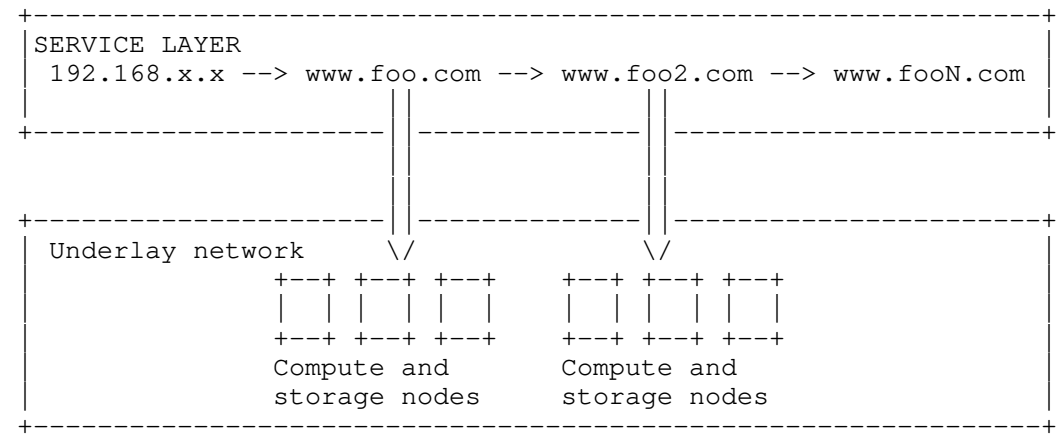


Figure 2: Mapping SFC onto Service Function Execution Points along a Service Function Path based on Virtualized Service Function Instance

4.3. Name Based Network Locator Map (nNLM)

In order to forward a name-based SFC, we need to extend the network locator map as originally defined in [RFC8300] with the ability to consider name relations based on URIs as well as high-level transport protocols such as HTTP for means of SFC packet forwarding. Another example for SFC packet forwarding could be that of CoAP.

The extended Network Locator Map or name-based Network Locator Map (nNLM) is shown in Figure 3 as an example for www.foo.com being part of the nSFP. Such extended nNLM is stored at each SFF throughout the SFC domain with suitable information populated to the nNLM during the configuration phase

SPI	SI	Next Hop(s)	Transport Encapsulation
10	255	192.0.2.1	VXLAN-gpe
10	254	198.51.100.10	GRE
10	253	www.foo.com	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Figure 3: Name-based Network Locator Map

Alternatively, the extended network locator map may be defined with implicit name information rather than explicit URIs as in Figure 3. In the example of Figure 4 below, the next hop is represented as a generic HTTP service without a specific URI being identified in the extended network locator map. In this scenario, the SFF forwards the packet based on parsing the HTTP request in order to identify the host name or URI. It retrieves the URI and may apply policy information to determine the destination host/service.

SPI	SI	Next Hop(s)	Transport Encapsulation
10	255	192.0.2.1	VXLAN-gpe
10	254	198.51.100.10	GRE
10	253	HTTP Service	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Figure 4: Name-based Network Locator Map with Implicit Name information

4.4. Name-based Service Function Forwarder (nSFF)

While [I-D.purkayastha-sfc-service-indirection] outlined the realization of forwarding packets in URL-based interaction through HTTP via a specific function (called Service Request Routing in [I-D.purkayastha-sfc-service-indirection]), it is desirable to extend the SFF of the SFC underlay in order to handle nSFPs transparently and without the need to insert a special (SRR) service function into the nSFP. Such extended name-based SFF would then be responsible for forwarding a packet in the SFC domain as per the definition of the (extended) nSFP.

In our exemplary realization for an extended SFF, the solution described in this document uses HTTP as the protocol of forwarding SFC packets to the next (name-based) hop in the nSFP. The URI in the HTTP transaction are the names in our nSFP information, which will be used for name based forwarding.

Following our reasoning so far, HTTP requests (and more specifically the plain text encoded requests above) are the equivalent of Packets that enter the SFC domain. In the existing SFC framework, typically an IP payload is assumed to be a packet entering the SFC domain. This packet is forwarded to destination nodes using the L2 encapsulation. Any layer 2 network can be used as an underlay network. This notion is now extended to packets being possibly entire higher layer application, such as HTTP requests. The handling of any intermediate layers such as TCP, IP is left to the realization

of the (extended) SFF operations towards the next (named) hop. For this, we will first outline the general lifecycle of an SFC packet in the following subsection, followed by two examples for determining next hop information in Section 5.2.3, finalized by a layered view on the realization of the nSFF in Section 5.2.4

4.5. High Level Architecture

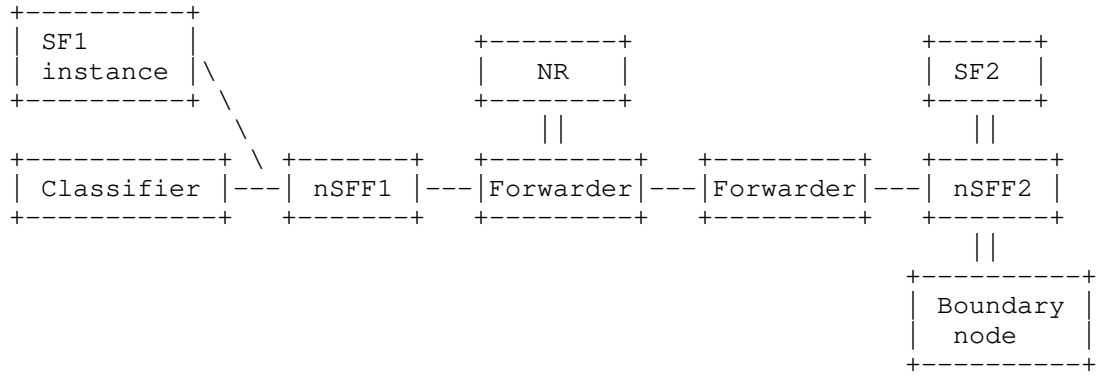


Figure 5: High-level architecture

The high-level architecture for name based operation shown in Figure 5 is very similar to the SFC architecture, as described in [RFC7665]. Two new functions are introduced, as shown in the above diagram, namely the name-based Service Function Forwarder (nSFF) and the Name Resolver (NR). The former is an extension of the existing SFF and is capable of processing SFC packets based on name-based network locator map (nNLM) information, determining the next SF, where the packet should be forwarded and the required transport encapsulation. Like standard SFF operation, it adds transport encapsulation to the SFC packet and forwards it. The Name Resolver is a new functional component, capable of identifying the execution end points, where a "named SF" is running, triggered by suitable resolution requests sent by the nSFF.

The other functional components such as Classifier, SF are same as described in SFC architecture [RFC7665], while the Forwarder shown in the above diagram are Layer 2 switches.

4.6. Operational Steps

In the proposed solution, the operations are realized by the name-based SFF, called nSFF. We utilize the high-level architecture in Figure 5 to describe the traversal between two service function instances of an nSFP-based transactions in an example chain of : 192.168.x.x -> SF1 (www.foo.com) -> SF2 (www.foo2.com) -> SF3 -> ...

According to the SFC lifecycle [RFC7665] and based on our example chain above, the traffic originates from a Classifier or another SFF on the left. The traffic is processed by the incoming nSFF1 (on the left side) through the following steps. The traffic exits at nSFF2.

- Step 1: At nSFF1 the following nNLM is assumed

SPI	SI	Next Hop(s)	Transport Encapsulation
10	255	192.0.2.1	VXLAN-gpe
10	254	198.51.100.10	GRE
10	253	www.foo.com	HTTP
10	252	www.foo2.com	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Figure 6: nNLM at nSFF1

- Step 2: nSFF1 removes the previous transport encapsulation (TE) for any traffic originating from another SFF or classifier (traffic from an SF instance does not carry any TE and is therefore directly processed at the nSFF).

- Step 3: nSFF1 then processes the NSH information to identify the next SF at the nSFP level by mapping the NSH information to the appropriate entry in its nNLM (see Figure 6) based on the provided SPI/SI information in the NSH (see Section 3) in order to determine the name-based identifier of the next hop SF. With such nNLM in mind, the nSFF searches the map for SPI = 10 and SI = 253.

It identifies the next hop as = www.foo.com and HTTP as the protocol to be used. Given the next hop resides locally , the SFC packet is forwarded to the SF1 instance of www.foo.com. Note that the next hop could also be identified from the provided HTTP request, if the next hop information was identified as a generic HTTP service, as defined in Section 5.3.

- Step 4: The SF1 instance then processes the received SFC packet according to its service semantics and modifies the NSH by setting SPI = 10, SI = 252 for forwarding the packet along the SFP. It then forwards the SFC packet to its local nSFF, i.e., nSFF1.

- Step 5: nSSF1 processes the NSH of the SFC packet again, now with the NSH modified (SPI = 10, SI = 252) by the SF1 instance. It retrieves the next hop information from its nNLM in Figure 6, to be www.foo2.com. Due to this SF not being locally available , the nSFF consults any locally available information regarding the routing/forwarding information towards a suitable nSFF that can serve this next hop

- Step 6: If such information exists, the Packet (plus the NSH information) is marked to be sent towards the nSFF serving the next hop based on such information in step 8.

- Step 7: If such information does not exist, nSFF1 consults the Name Resolver (NR) to determine the suitable routing/forwarding information towards the identified nSFF serving the next hop of the SFP. For future SFC packets towards this next hop, such resolved information may be locally cached , avoiding to contact the Name Resolver for every SFC packet forwarding. The packet is now marked to be sent via the network in step 8.

- Step 8: Utilizing the forwarding information determined in steps 6 or 7, nSFF1 adds the suitable transport encapsulation (TE) for the SFC packet before forwarding via the forwarders in the network towards the next nSFF22.

- Step 9: When the Packet (+NSH+TE) arrives at the outgoing nSFF2, i.e., the nSFF serving the identified next hop of the SFP, removes the TE and processes the NSH to identify the next hop information. At nSFF2 the nNLM in Figure 7 is assumed. Based on this nNLM and NSH information where SPI = 10 and SI = 252, nSFF2 identifies the next SF as www.foo2.com.

SPI	SI	Next Hop(s)	Transport Encapsulation
10	252	www.foo2.com	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Figure 7: nNLM at SFF2

- Step 10: If the next hop is locally registered at the nSFF, it forwards the packet (+NSH) to the service function instance, using suitable IP/MAC methods for doing so.

- Step 11: Otherwise, the outgoing nSFF adds a new TE information to the packet and forwards the packet (+NSH+TE) to the next SFF or boundary node, as shown in Figure 7.

5. nSFF Forwarding Operations

This section outlines the realization of various nSFF forwarding operations in Section 4.6. Although the operations in Section 4 utilize the notion of name-based transactions in general, we exemplify the operations here in Section 5 specifically for HTTP-based transactions to ground our description into a specific protocol for such name-based transaction. We will refer to the various steps in each of the following sub-sections.

5.1. nSFF Protocol Layers

Figure 8 shows the protocol layers, based on the high-level architecture in Figure 5.

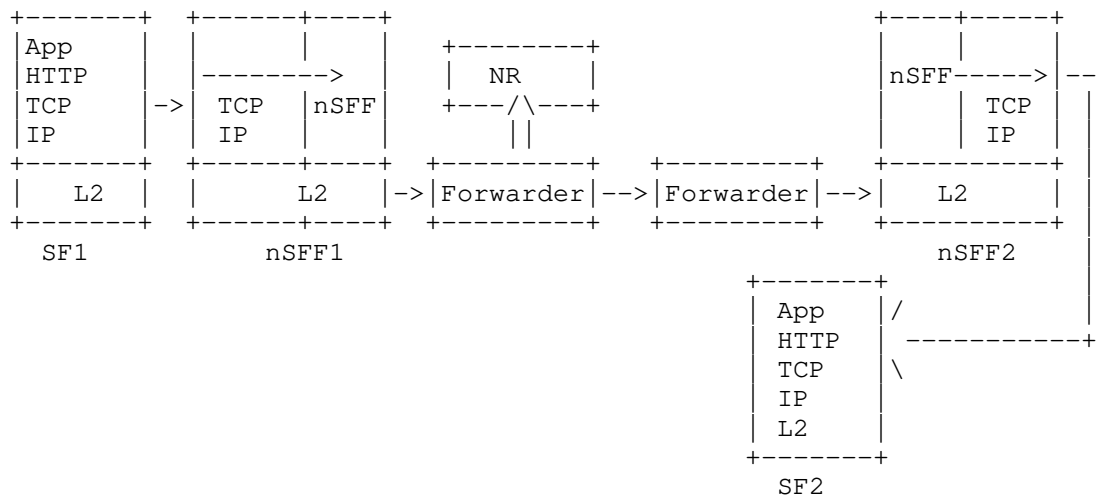


Figure 8: Protocol layers

The nSFF component here is shown as implementing a full incoming/ outgoing TCP/IP protocol stack towards the local service functions, while implementing the nSFF-NR and nSFF-nSFF protocols based on the descriptions in Section 5.2.3. For the exchange of HTTP-based service function transactions, the nSFF terminates incoming TCP connections from as well as outgoing TCP connections to local SFs.

5.2. nSFF Operations

In this section, we present three key aspects of operations for the realization of the steps in Section 4.6, namely (i) the registration of local SFs (for step 3 in Section 4.6), (ii) the forwarding of SFC packets to and from local SFs (for step 3 and 4 as well as 10 in Section 4.6), (iii) the forwarding to a remote SF (for steps 5, 6., and 7 in Section 4.6) and to the NR as well as (iv) for the lookup of a suitable remote SF (for step 7 in Section 4.6). We also cover aspects of maintaining local lookup information for reducing lookup latency and others issues.

5.2.1. Forwarding between nSFFs and nSFF-NR

Forwarding between the distributed nSFFs as well as between nSFF and NR is realized over the operator network via a path-based approach. A path-based approach utilizes path information provided by the source of the packet for forwarding said packet in the network. This is similar to segment routing albeit differing in the type of information provided for such source-based forwarding, as described

in this section. In this approach, the forwarding information to a remote nSFF or the NR is defined as a 'path identifier' (pathID) of a defined length where said length indicate the overall pathID length as the 2 to the power of 'length', i.e., maximum 2^16 bits as path information. The payload of the packet is defined by the various operations outlined in the following sub-sections, resulting in an overall packet being transmitted. With this, the generic forwarding format (GFF) for transport over the operator network is defined in Figure 9 with the length field defining the length of the pathID provided.

Figure 9: Generic Forwarding Format(GFF)

For the pathID information, solutions such as those in [Reed2016] can be used. Here, the IPv6 source and destination addresses are used to realize a so-called path-based forwarding from the incoming to the outgoing nSFF or the NR. The forwarders in Figure 8 are realized via SDN (software-defined networking) switches, implementing an AND/CMP operation based on arbitrary wildcard matching over the IPv6 source and destination addresses, as outlined in [Reed2016]. Note that in the case of using IPv6 address information for path-based forwarding, the step of removing the transport encapsulation at the outgoing nSFF in Figure 8 is realized by utilizing the provided (existing) IP header (which was used for the purpose of the path-based forwarding in [Reed2016]) for the purpose of next hop forwarding, such as that of IP-based routing. As described in step 8 of the extended nSFF operations, this forwarding information is used as traffic encapsulation. With the forwarding information utilizing existing IPv6 information, IP headers are utilized as TE in this case. The next hop nSFF (see Figure 8) will restore the IP header of the packet with the relevant IP information used to forward the SFC packet to SF2 or it will create a suitable TE (Transport Encapsulation) information to forward the information to another nSFF or boundary node. Forwarding operations at the intermediary forwarders, i.e., SDN switches, examine the pathID information through a flow matching rule in which a specific switch-local output port is represented through the specific assigned bit position in the pathID. Upon a positive match in said rule, the packet is forwarded on said output port.

Alternatively, the solution in [I-D.purkayastha-bier-multicast-http] suggests using a so-called BIER (Binary Indexed Explicit Replication) underlay. Here, the nSFF would be realized at the ingress to the BIER underlay, injecting the SFC packet (plus the NSH) header with BIER-based traffic encapsulation into the BIER underlay with each of the forwarders in Figure 8 being realized as a so-called Bit-Forwarding Router (BFR) [RFC8279].

5.2.1.1. Transport Protocol Considerations

Given that the proposed solution operates at the 'named transaction' level, particularly for HTTP transactions, forwarding between nSFFs and/or NR SHOULD be implemented via a transport protocol between nSFFs and/or NR in order to provide reliability, segmentation of large GFF packets, and flow control. The details of this protocol will be outlined at a later stage, with the GFF in Figure 9 being the basic forwarding format for this.

5.2.2. SF Registration

As outlined in step 3 and 10 of Section 4.6, the nSFF needs to determine if the SF derived from the nNLM is locally reachable or whether the packet needs forwarding to a remote SFF. For this, a registration mechanism is provided for such local SF with the local nSFF. Two mechanisms can be used for this:

1. SF-initiated: We assume that the SF registers its FQDN to the local nSFF. As local mechanisms, we foresee that either a REST-based interface over the link-local link or configuration of the SR (through configuration files or management consoles) can be utilized. Such local registration event leads to the nSFF to register the given FQDN with the NR in combination with a system-unique nSFF identifier that is being used for path computation purposes in the NR. For the registration, the packet format in Figure 10 is used (inserted as the payload in the GFF of Figure 9 with the pathID towards the NR), with a hash over the FQDN being conveyed in the registration and the R/D bit set to 0 (for registration). We assume that the pathID towards the NR is known to the nSFF through configuration means.

Figure 10: Registration packet format

The NR maintains an internal table that associates the hash(FQDN), the nSFF_id information as well as the pathID information being used for communication between nSFF and NR. The nSFF locally maintains a mapping of registered FQDNs to IP addresses, for the latter using link-local private IP addresses.

2. Orchestration-based: in this mechanism, we assume that SFC to be orchestrated and the chain being provided through an orchestration template with FQDN information associated to a compute/storage resource that is being deployed by the orchestrator. We also assume knowledge at the orchestrator of the resource topology. Based on this, the orchestrator can now use the same REST-based protocol defined in option 1 to instruct the NR to register the given FQDN, as provided in the template, at the nSFF it has identified as being the locally servicing nSFF, provided as the system-unique nSFF identifier.

5.2.3. Local SF Forwarding

There are two cases of local SF forwarding, namely the SF sending an SFC packet to the local nSFF (incoming requests) or the nSFF sending a packet to the SF (outgoing requests) as part of steps 3 and 10 in Section 4.6. In the following, we outline the operation for HTTP as an example named transaction.

As shown in Figure 8, incoming HTTP requests from SFs are extracted by terminating the incoming TCP connection at their local nSFFs at the TCP level. The nSFF MUST maintain a mapping of open TCP sockets to HTTP requests for HTTP response association.

For outgoing HTTP requests, the nSFF utilizes the maintained mapping of locally registered FQDNs to link-local IP addresses (see Section 5.2.2 option 1). Hence, upon receiving an SFC packet from a remote nSFF (in step 9 of Section 4.6), the nSFF determines the local existence of the SF through the registration mechanisms in Section 5.2.2. If said SF does exist locally, the HTTP (+NSH) packet, after stripping the TE, is sent to the local SF as step 10 in Section 4.6 via a TCP-level connection. Outgoing nSFF SHOULD keep TCP connections open to local SFs for improving SFC packet delivery in subsequent transactions.

5.2.4. Remote SF Forwarding

In steps 5, 6, 7, and 8 of Section 4.6, an SFC packet is forwarded to a remote nSFF based on the nNLM information for the next hop of the nSFP. Section 5.2.4.1 handles the case of suitable forwarding information to the remote nSFF not existing, therefore consulting the NR to obtain suitable information, while Section 5.2.4.2 describes

the maintenance of forwarding information at the local nSFF, while Section 5.2.4.3 describes the update of stale forwarding information. Note that the forwarding described in Section 5.2.1 is used for the actual forwarding to the various nSFF and SR components. Ultimately, Section 5.2.4.4 describes the forwarding to the remote nSFF via the forwarder network

5.2.4.1. Remote SF Discovery

The nSFF communicates with the NR for two purposes, namely the registration and discovery of FQDNs. The packet format for the former was shown in Figure 10 in Section 5.2.2, while Figure 11 outlines the packet format for the discovery request. A path to a specific FQDN is requested by sending a hash of the FQDN to the NR together with its nSFF_id, receiving as a response a pathID with a length identifier. The NR should maintain a table of discovery requests that map discovered (hash of) FQDN to the nSFF_id that requested it and the pathID that is being calculated as a result of the discovery request.

The discovery request for an FQDN that has not previously being served at the nSFF (or for an FQDN whose pathID information has been flushed as a result of the update operations in Section 5.2.4.3), results in an initial latency incurred by this discovery through the NR, while any SFC packet sent over the same SFP in a subsequent transaction will utilize the nSFF local mapping table. Such initial latency can be avoided by pre-populating the FQDN-pathID mapping proactively as part of the overall orchestration procedure, e.g., alongside the distribution of the nNLM information to the nSFF.

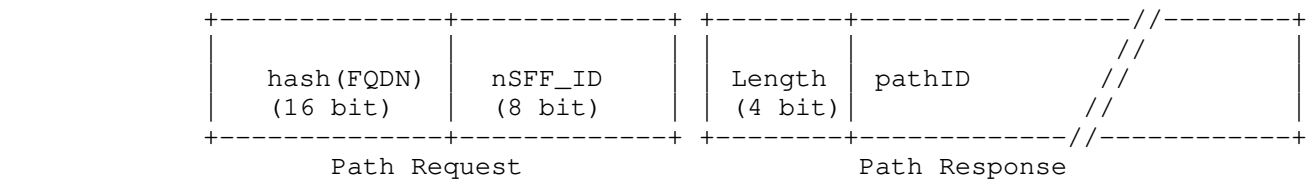


Figure 11: Discovery packet format

5.2.4.2. Maintaining Forwarding Information at Local nSFF

Each nSFF MUST maintain an internal table that maps the (hash of the) FQDN information to a suitable pathID information. As outlined in step 7 of Section 4.6, if a suitable entry does not exist for a given FQDN, the pathID information is requested with the operations in Section 5.2.4.1 and the suitable entry is locally created upon

receiving a reply with the forwarding operation being executed as described in Section 5.2.1.

If such entry does exist (i.e., step 6 of Section 4.6) the pathID is locally retrieved and used for the forwarding operation in Section 5.2.1.

5.2.4.3. Updating Forwarding Information at nSFF

The forwarding information maintained at each nSFF (see Section 5.2.4.2) might need to be updated for three reasons:

- An existing SF is no longer reachable: In this case, the nSFF with which the SF is locally registered, deregisters the SF explicitly at the NR by sending the packet in Figure 10 with the hashed FQDN and the R/D bit set to 1 (for deregister).

- Another SF instance has become reachable in the network (and therefore might provide a better alternative to the existing SF): in this case, the NR has received another packet with format defined in Figure 11 but a different nSFF_id value.

- Links along paths might no longer be reachable: the NR might use suitable southbound interface to transport networks to detect link failures, which it associates to the appropriate pathID bit position

For this purpose, the packet format in Figure 12 is sent from the NR to all affected nSFFs, using the generic format in Figure 9. The pathID to the affected nSFFs is computed as the binary OR over all pathIDs to those nSFF_ids affected where the pathID information to the affected nSFF_id values is determined from the NR-local table maintained in the registration/deregistration operation of Section 5.2.2.

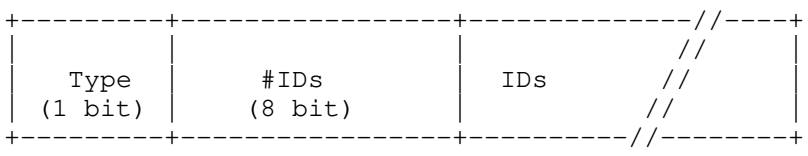


Figure 12: Path update format

In case 1 and 2, the Type bit is set to 1 (type nSFF_id) and the affected nSFFs are determined by those nSFFs that have previously sent SF discovery requests, utilizing the optional table mapping

previously registered FQDNs to nSFF_id values. If no table mapping the (hash of) FQDN to nSFF_id is maintained, the update is sent to all nSFFs. Upon receiving the path update at the affected nSFF, all appropriate nSFF-local mapping entries to pathIDs for the hash(FQDN) identifiers provided will be removed, leading to a new NR discovery request at the next remote nSFF forwarding to the appropriate FQDN.

In case 3, the Type bit is set to 0 (type linkID) and the affected nSFFs are determined by those nSFFs whose discovery requests have previously resulted in pathIDs which include the affected link, utilizing the optional table mapping previously registered FQDNs to pathID values (see Section 5.2.4.1). Upon receiving the path update, the affected nSFF will check its internal table that maps FQDNs to pathIDs to determine those pathIDs affected by the link problems. For this, the pathID entries of said table are checked against the linkID values provided in the ID entry of the path update through a binary AND/CMP operation to check the inclusion of the link in the pathIDs to the FQDNs. If any pathID is affected, the FQDN-pathID entry is removed, leading to a new NR discovery request at the next remote nSFF forwarding to the appropriate FQDN.

5.2.4.4. Forwarding to remote nSFF

Once step 5, 6, and 7 in Section 4.6 are being executed, step 8 finally sends the SFC packet to the remote nSFF, utilizing the pathID returned in the discovery request (Section 5.2.4.1) or retrieved from the local pathID mapping table. The SFC packet is placed in the payload of the generic forwarding format in Figure 14 together with the pathID and the nSFF eventually executes the forwarding operations in Section 5.2.1.

6. IANA Considerations

This document requests no IANA actions.

7. Security Considerations

The operations in Section 4 and 5 consider the forwarding of SFC packets between named SFs based on HTTP. The support for HTTPS is foreseen to ensure suitable encryption capability of such exchanges. Future updates to this draft will outline the support for such HTTPS-based SFC exchanges.

8. Informative References

[_3GPP_SBA]
3GPP, "Technical Realization of Service Based Architecture", 3GPP TS 29.500 0.4.0, January 2018, <http://www.3gpp.org/ftp/Specs/html-info/29500.htm>.

[_3GPP_SBA_ENHANCEMENT]
3GPP, "New SID for Enhancements to the Service-Based 5G System Architecture", 3GPP S2-182904 , February 2018, <http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_126_Montreal/Docs/S2-182904.zip>.

[I-D.purkayastha-bier-multicast-http]
Purkayastha, D., Rahman, A., and D. Trossen, "Multicast HTTP using BIER", draft-purkayastha-bier-multicast-http-00 (work in progress), March 2018.

[I-D.purkayastha-sfc-service-indirection]
Purkayastha, D., Rahman, A., Trossen, D., Despotovic, Z., and R. Khalili, "Alternative Handling of Dynamic Chaining and Service Indirection", draft-purkayastha-sfc-service-indirection-02 (work in progress), March 2018.

[Reed2016]
Reed, M., Al-Naday, M., Thomas, N., Trossen, D., and S. Spirou, "Stateless multicast switching in software defined networks", ICC 2016, 2016.

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

[RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <https://www.rfc-editor.org/info/rfc8279>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

Authors' Addresses

Dirk Trossen
InterDigital Communications, LLC
64 Great Eastern Street, 1st Floor
London EC2A 3QR
United Kingdom

Email: Dirk.Trossen@InterDigital.com
URI: http://www.InterDigital.com/

Debashish Purkayastha
InterDigital Communications, LLC
Conshohocken
USA

Email: Debashish.Purkayastha@InterDigital.com

Akbar Rahman
InterDigital Communications, LLC
Montreal
Canada

Email: Akbar.Rahman@InterDigital.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 27, 2019

D. Trossen
InterDigital Europe, Ltd
D. Purkayastha
A. Rahman
InterDigital Communications, LLC

May 26, 2019

Name-Based Service Function Forwarder (nSFF) component within SFC framework
draft-trossen-sfc-name-based-sff-07

Abstract

Adoption of cloud and fog technology allows operators to deploy a single "Service Function" to multiple "Execution locations". The decision to steer traffic to a specific location may change frequently based on load, proximity etc. Under the current SFC framework, steering traffic dynamically to the different execution end points require a specific 're-chaining', i.e., a change in the service function path reflecting the different IP endpoints to be used for the new execution points. This procedure may be complex and take time. In order to simplify re-chaining and reduce the time to complete the procedure, we discuss separating the logical Service Function Path from the specific execution end points. This can be done by identifying the Service Functions using a name rather than a routable IP endpoint (or Layer 2 address). This document describes the necessary extensions, additional functions and protocol details in SFF (Service Function Forwarder) to handle name based relationships.

This document presents InterDigital's approach to name-based service function chaining. It does not represent IETF consensus and is presented here so that the SFC community may benefit from considering this mechanism and the possibility of its use in the edge data centers.

Copyright Notice

Table of Contents

1. Introduction

The requirements on today's networks are very diverse, enabling multiple use cases such as IoT, Content Distribution, Gaming and Network functions such as Cloud RAN and 5G control planes based on a service-based architecture. These services are deployed, provisioned and managed using Cloud based techniques as seen in the IT world. Virtualization of compute and storage resources is at the heart of providing (often web) services to end users with the ability to quickly provisioning such virtualized service endpoints through, e.g., container based techniques. This creates a dynamicity with the capability to dynamically compose new services from available services as well as move a service instance in response to user mobility or resource availability where desirable. When moving from a pure 'distant cloud' model to one of localized micro data centers with regional, metro or even street level, often called 'edge' data centers, such virtualized service instances can be instantiated in topologically different locations with the overall 'distant' data center now being transformed into a network of distributed ones. The reaction of content providers, like Facebook, Google, NetFlix and others, are not just relying on deploying content server at the ingress of the customer network. Instead the trend is towards deploying multiple POPs within the customer network, those POPs being connected through proprietary mechanisms [Schlinker2017] to push content.

The Service Function Chaining (SFC) framework [RFC7665] allows network operators as well as service providers to compose new services by chaining individual "Service Functions (SFs)". Such chains are expressed through explicit relationships of functional components (the service functions), realized through their direct Layer 2 (e.g., MAC address) or Layer 3 (e.g., IP address) relationship as defined through next hop information that is being defined by the network operator, see Section 4 for more background on SFC.

In a dynamic service environment of distributed data centers as the one outlined above, with the ability to create and recreate service endpoints frequently, the SFC framework requires to reconfigure the existing chain through information based on the new relationships, causing overhead in a number of components, specifically the orchestrator that initiates the initial service function chain and any possible reconfiguration.

This document describes how such changes can be handled without involving the initiation of new and reconfigured SFCs by lifting the chaining relationship from Layer 2 and 3 information to that of service function 'names', such as names for instance being expressed as URIs. In order to transparently support such named relationships, we propose to embed the necessary functionality directly into the Service Function Forwarder (SFF), as described in [RFC7665]). With that, the SFF described in this document allows for keeping an existing SFC intact, as described by its service function path (SFP), while enabling the selection of an appropriate service function endpoint(s) during the traversal of packets through the SFC. This document is an Independent Submission to the RFC Editor. It is not an output of the IETF SFC WG.

2. Terminology

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

3. Example use case: 5G control plane services

We exemplify the need for chaining service functions at the level of a service name through a use case stemming from the current 3GPP Rel 16 work on Service Based Architecture (SBA) [_3GPP_SBA], [_3GPP_SBA_ENHANCEMENT]. In this work, mobile network control planes are proposed to be realized by replacing the traditional network function interfaces with a fully service-based one. HTTP was chosen as the application layer protocol for exchanging suitable service requests [_3GPP_SBA]. With this in mind, the exchange between, say the 3GPP (Rel. 15) defined Session Management Function (SMF) and the Access and Mobility management Function (AMF) in a 5G control plane is being described as a set of web service like requests which are in turn embedded into HTTP requests. Hence, interactions in a 5G control plane can be modelled based on service function chains where the relationship is between the specific (IP-based) service function endpoints that implement the necessary service endpoints in the SMF and AMF. The service functions are exposed through URIs with work ongoing to define the used naming conventions for such URIs.

This move from a network function model (in pre-Rel 15 systems of 3GPP) to a service-based model is motivated through the proliferation of data center operations for mobile network control plane services. In other words, typical IT-based methods to service provisioning, in particular that of virtualization of entire compute resources, are envisioned to being used in future operations of mobile networks.

Hence, operators of such future mobile networks desire to virtualize service function endpoints and direct (control plane) traffic to the most appropriate current service instance in the most appropriate (local) data centre, such data centre envisioned as being interconnected through a software-defined wide area network (SD-WAN). 'Appropriate' here can be defined by topological or geographical proximity of the service initiator to the service function endpoint. Alternatively, network or service instance compute load can be used to direct a request to a more appropriate (in this case less loaded) instance to reduce possible latency of the overall request. Such data center centric operation is extended with the trend towards regionalization of load through a 'regional office' approach, where micro data centers provide virtualizable resources that can be used in the service execution, creating a larger degree of freedom when choosing the 'most appropriate' service endpoint for a particular incoming service request.

While the move to a service-based model aligns well with the framework of SFC, choosing the most appropriate service instance at runtime requires so-called 're-chaining' of the SFC since the relationships in said SFC are defined through Layer 2 or 3 identifiers, which in turn are likely to be different if the chosen service instances reside in different parts of the network (e.g., in a regional data center).

Hence, when a traffic flow is forwarded over a service chain expressed as an SFC-compliant Service Function Path (SFP), packets in the traffic flow are processed by the various service function instances, with each service function instance applying a service function prior to forwarding the packets to the next network node. It is a Service layer concept and can possibly work over any Virtual network layer and an Underlay network, possibly IP or any Layer 2 technology. At the service layer, Service Functions are identified using a path identifier and an index. Eventually this index is translated to an IP address (or MAC address) of the host where the service function is running. Because of this, any change of service function instance is likely to require a change of the path information since either IP address (in the case of changing the execution from one data centre to another) or MAC address will change due to the newly selected service function instance.

Returning to our 5G Control plane example, a user's connection request to access an application server in the internet may start with signaling in the Control Plane to setup user plane bearers. The connection request may flow through service functions over a service chain in the Control plane, as deployed by network operator. Typical SFs in a 5G control plane may include "RAN termination / processing", "Slice Selection Function", "AMF" and "SMF". A Network Slice is a

complete logical network including Radio Access Network (RAN) and Core Network (CN). Distinct RAN and Core Network Slices may exist. A device may access multiple Network Slices simultaneously through a single RAN. The device may provide Network Slice Selection Assistance Information (NSSAI) parameters to the network to help it select a RAN and a Core network part of a slice instance. Part of the control plane, the Common Control Network Function (CCNF), the Network Slice Selection Function (NSSF) is in charge of selecting core Network Slice instances. The Classifier, as described in SFC architecture, may reside in the user terminal or at the eNB. These service functions can be configured to be part of a Service Function Chain. We can also say that some of the configurations of the Service Function Path may change at the execution time. For example, the SMF may be relocated as user moves and a new SMF may be included in the Service Function Path based on user location. The following diagram in Figure 1 shows the example Service Function Chain described here.

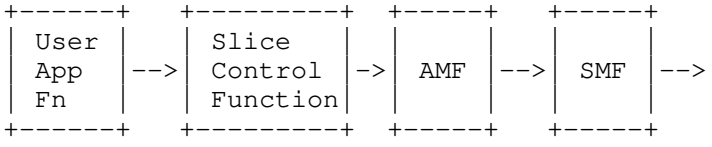


Figure 1: Mapping SFC onto Service Function Execution Points along a Service Function Path

4. Background

[RFC7665] describes an architecture for the specification, creation and ongoing maintenance of Service Function Chains (SFCs). It includes architectural concepts, principles, and components used in the construction of composite services through deployment of SFCs. In the following, we outline the parts of this SFC architecture relevant for our proposed extension, followed by the challenges with this current framework in the light of our example use case.

4.1. Relevant part of SFC architecture

SFC Architecture, as defined in [RFC7665], describes architectural components such as Service Function (SF), Classifier, and Service Function Forwarder (SFF). It describes the Service Function Path (SFP) as the logical path of an SFC. Forwarding traffic along such SFP is the responsibility of the SFF. For this, the SFFs in a

network maintain the requisite SFP forwarding information. Such SFP forwarding information is associated with a service path identifier (SPI) that is used to uniquely identify an SFP. The service forwarding state is represented by the Service Index (SI) and enables an SFF to identify which SFs of a given SFP should be applied, and in what order. The SFF also has information that allows it to forward packets to the next SFF after applying local service functions.

The operational steps to forward traffic are then as follows: Traffic arrives at an SFF from the network. The SFF determines the appropriate SF the traffic should be forwarded to via information contained in the SFC encapsulation. After SF processing, the traffic is returned to the SFF, and, if needed, is forwarded to another SF associated with that SFF. If there is another non-local hop (i.e., to an SF with a different SFF) in the SFP, the SFF further encapsulates the traffic in the appropriate network transport protocol and delivers it to the network for delivery to the next SFF along the path. Related to this forwarding responsibility, an SFF should be able to interact with metadata.

4.2. Challenges with current framework

As outlined in previous section, the Service Function Path defines an ordered sequence of specific Service Functions instances being used for the interaction between initiator and service functions along the SFP. These service functions are addressed by IP (or any L2/MAC) addresses and defined as next hop information in the network locator maps of traversing SFF nodes.

As outlined in our use case, however, the service provider may want to provision SFC nodes based on dynamically spun up service function instances so that these (now virtualized) service functions can be reached in the SFC domain using the SFC underlay layer.

Following the original model of SFC, any change in a specific execution point for a specific Service Function along the SFP will require a change of the SFP information (since the new service function execution point likely carries different IP or L2 address information) and possibly even the Next Hop information in SFFs along the SFP. In case the availability of new service function instances is rather dynamic (e.g., through the use of container-based virtualization techniques), the current model and realization of SFC could lead to reducing the flexibility of service providers and increasing the management complexity incurred by the frequent changes of (service) forwarding information in the respective SFF nodes. This is because any change of the SFP (and possibly next hop info) will need to go through suitable management cycles.

To address these challenges through a suitable solution, we identify the following requirements:

- Relations between Service Execution Points MUST be abstracted so that, from an SFP point of view, the Logical Path never changes.
- Deriving the Service Execution Points from the abstract SFP SHOULD be fast and incur minimum delay.
- Identification of the Service Execution Points SHOULD not use a combination of Layer 2 or Layer 3 mechanisms.

The next section outlines a solution to address the issue, allowing for keeping SFC information (represented in its SFP) intact while addressing the desired flexibility of the service provider.

5. Name based operation in SFF

5.1. General Idea

The general idea is two-pronged. Firstly, we elevate the definition of a Service Function Path onto the level of 'name-based interactions' rather than limiting SFPs to Layer 3 or Layer 2 information only. Secondly, we extend the operations of the SFF to allow for forwarding decisions that take into account such name-based interaction while remaining backward compatible to the current SFC architecture, as defined in [RFC7665]. In the following sections, we outline these two components of our solution.

If the next hop information in the Network Locator Map (NLM) is described using L2/L3 identifier, the name-based SFF (nSFF) may operate as described for [traditional] SFF, as defined in [RFC7665]. On the other hand, if the next hop information in the NLM is described as a name, then the nSFF operates as described in the following sections.

In the following sections, we outline the two components of our solution.

5.2. Name-Based Service Function Path (nSFP)

The existing SFC framework is defined in [RFC7665]. Section 4 outlines that the SFP information is representing path information based on Layer 2 or 3 information, i.e., MAC or IP addresses, causing the aforementioned frequent adaptations in cases of execution point changes. Instead, we introduce the notion of a "name-based service function path (nSFP)".

In today's networking terms, any identifier can be treated as a name but we will illustrate the realization of a "Name based SFP" through extended SFF operations (see Section 6) based on URIs as names and HTTP as the protocol of exchanging information. Here, URIs are being used to name for a Service Function along the nSFP. It is to be noted that the Name based SFP approach is not restricted to HTTP (as the protocol) and URIs (as next hop identifier within the SFP). Other identifiers such as an IP address itself can also be used and are interpreted as a 'name' in the nSFP. IP addresses as well as fully qualified domain names forming complex URIs (uniform resource identifiers), such as www.example.com/service_name1, are all captured by the notion of 'name' in this document.

Generally, nSFPs are defined as an ordered sequence of the "name" of Service Functions (SF) and a typical name-based Service Function Path may look like: 192.0.x.x -> www.example.com -> www.example2.com/ service1 -> www.example2.com/service2.

Our use case in Section 3 can then be represented as an ordered named sequence. An example for a session initiation that involves an authentication procedure, this could look like 192.0.x.x -> smf.example.org/session_initiate -> amf.example.org/auth -> smf.example.org/session_complete -> 192.0.x.x. [Note that this example is only a conceptual one, since the exact nature of any future SBA-based exchange of 5G control plane functions is yet to be defined by standardization bodies such as 3GPP].

In accordance with our use case in Section 3, any of these named services can potentially be realized through more than one replicated SF instances. This leads to make dynamic decision on where to send packets along the SAME service function path information, being provided during the execution of the SFC. Through elevating the SFP onto the notion of name-based interactions, the SFP will remain the same even if those specific execution points change for a specific service interaction.

The following diagram in Figure 2, describes this name-based SFP concept and the resulting mapping of those named interactions onto (possibly) replicated instances.

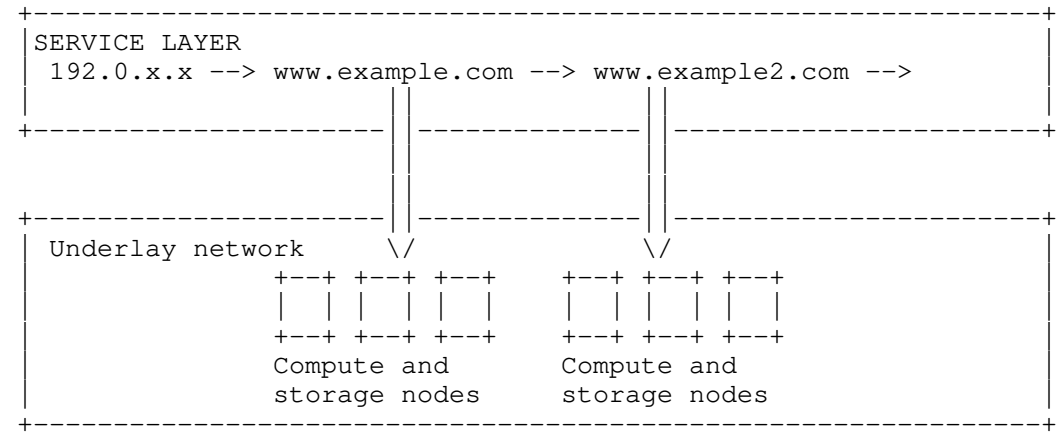


Figure 2: Mapping SFC onto Service Function Execution Points along a Service Function Path based on Virtualized Service Function Instance

5.3. Name Based Network Locator Map (nNLM)

In order to forward a packet within a name-based SFP, we need to extend the network locator map as defined in [RFC8300] with the ability to consider name relations based on URIs as well as high-level transport protocols such as HTTP for means of SFC packet forwarding. Another example for SFC packet forwarding could be that of CoAP.

The extended Network Locator Map or name-based Network Locator Map (nNLM) is shown in Figure 3 as an example for www.example.com being part of the nSFP. Such extended nNLM is stored at each SFF throughout the SFC domain with suitable information populated to the nNLM during the configuration phase.

SPI	SI	Next Hop(s)	Transport Encapsulation (TE)
10	255	192.0.2.1	VXLAN-gpe
10	254	198.51.100.10	GRE
10	253	www.example.com	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Figure 3: Name-based Network Locator Map

Alternatively, the extended network locator map may be defined with implicit name information rather than explicit URIs as in Figure 3. In the example of Figure 4 below, the next hop is represented as a generic HTTP service without a specific URI being identified in the extended network locator map. In this scenario, the SFF forwards the packet based on parsing the HTTP request in order to identify the host name or URI. It retrieves the URI and may apply policy information to determine the destination host/service.

SPI	SI	Next Hop(s)	Transport Encapsulation (TE)
10	255	192.0.2.1	VXLAN-gpe
10	254	198.51.100.10	GRE
10	253	HTTP Service	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Figure 4: Name-based Network Locator Map with Implicit Name information

5.4. Name-based Service Function Forwarder (nSFF)

It is desirable to extend the SFF of the SFC underlay to handle nSFPs transparently and without the need to insert any service function into the nSFP. Such extended name-based SFF would then be responsible for forwarding a packet in the SFC domain as per the definition of the (extended) nSFP.

In our exemplary realization for an extended SFF, the solution described in this document uses HTTP as the protocol of forwarding SFC packets to the next (name-based) hop in the nSFP. The URI in the HTTP transaction are the names in our nSFP information, which will be used for name based forwarding.

Following our reasoning so far, HTTP requests (and more specifically the plain text encoded requests above) are the equivalent of Packets that enter the SFC domain. In the existing SFC framework, typically an IP payload is assumed to be a packet entering the SFC domain. This packet is forwarded to destination nodes using the L2 encapsulation. Any layer 2 network can be used as an underlay network. This notion is now extended to packets being possibly part of a entire higher layer application, such as HTTP requests. The handling of any intermediate layers such as TCP, IP is left to the realization of the (extended) SFF operations towards the next (named) hop. For this, we will first outline the general lifecycle of an SFC packet in the following subsection, followed by two examples for

determining next hop information in Section 6.2.3, finalized by a layered view on the realization of the nSFF in Section 6.2.4.

5.5. High Level Architecture

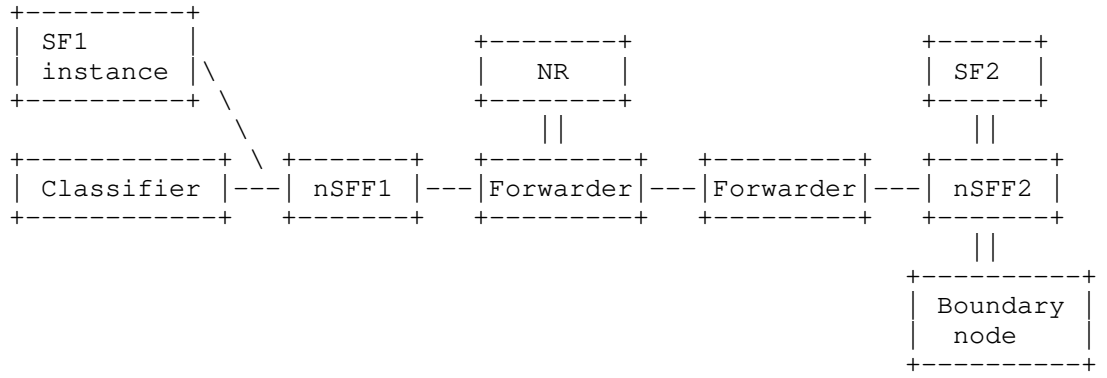


Figure 5: High-level architecture

The high-level architecture for name based operation shown in Figure 5 is very similar to the SFC architecture, as described in [RFC7665]. Two new functions are introduced, as shown in the above diagram, namely the name-based Service Function Forwarder (nSFF) and the Name Resolver (NR).

nSFF (name-based Service Function Forwarder) is an extension of the existing SFF and is capable of processing SFC packets based on name-based network locator map (nNLM) information, determining the next SF, where the packet should be forwarded and the required transport encapsulation. Like standard SFF operation, it adds transport encapsulation to the SFC packet and forwards it.

The Name Resolver is a new functional component, capable of identifying the execution end points, where a "named SF" is running, triggered by suitable resolution requests sent by the nSFF. Though this is similar to DNS function, but it is not same. It does not use DNS protocols or data records. A new procedure to determine the suitable routing/forwarding information towards the Nsff (name-based SFF) serving the next hop of the SFP (Service Function Path) is used. The details is described later.

The other functional components such as Classifier, SF are same as described in SFC architecture, as defined in [RFC7665], while the Forwarders shown in the above diagram are traditional Layer 2 switches.

5.6. Operational Steps

In the proposed solution, the operations are realized by the name-based SFF, called nSFF. We utilize the high-level architecture in Figure 5 to describe the traversal between two service function instances of an nSFP-based transactions in an example chain of : 192.0.x.x -> SF1 (www.example.com) -> SF2 (www.example2.com) -> SF3 -> ... Service Function 3 (SF3)is assumed to be a classical Service Function, hence existing SFC mechanisms can be used to reach it and will not be considered in this example.

According to the SFC lifecycle, as defined in [RFC7665], based on our example chain above, the traffic originates from a Classifier or another SFF on the left. The traffic is processed by the incoming nSFF1 (on the left side) through the following steps. The traffic exits at nSFF2.

- Step 1: At nSFF1 the following nNLM is assumed

SPI	SI	Next Hop(s)	Transport Encapsulation(TE)
10	255	192.0.2.1	VXLAN-gpe
10	254	198.51.100.10	GRE
10	253	www.example.com	HTTP
10	252	www.example2.com	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Figure 6: nNLM at nSFF1

- Step 2: nSFF1 removes the previous transport encapsulation (TE) for any traffic originating from another SFF or classifier

(traffic from an SF instance does not carry any TE and is therefore directly processed at the nSFF).

- Step 3: nSFF1 then processes the Network Service Header (NSH) information, as defined in [RFC8300], to identify the next SF at the nSFP level by mapping the NSH information to the appropriate entry in its nNLM (see Figure 6) based on the provided SPI/SI information in the NSH (see Section 4) in order to determine the name-based identifier of the next hop SF. With such nNLM in mind, the nSFF searches the map for SPI = 10 and SI = 253. It identifies the next hop as = www.example.com and HTTP as the protocol to be used. Given the next hop resides locally, the SFC packet is forwarded to the SF1 instance of www.example.com. Note that the next hop could also be identified from the provided HTTP request, if the next hop information was identified as a generic HTTP service, as defined in Section 5.3.

- Step 4: The SF1 instance then processes the received SFC packet according to its service semantics and modifies the NSH by setting SPI = 10, SI = 252 for forwarding the packet along the SFP. It then forwards the SFC packet to its local nSFF, i.e., nSFF1.

- Step 5: nSSF1 processes the NSH of the SFC packet again, now with the NSH modified (SPI = 10, SI = 252) by the SF1 instance. It retrieves the next hop information from its nNLM in Figure 6, to be www.example2.com. Due to this SF not being locally available, the nSFF consults any locally available information regarding routing/forwarding towards a suitable nSFF that can serve this next hop.

- Step 6: If such information exists, the Packet (plus the NSH information) is marked to be sent towards the nSFF serving the next hop based on such information in step 8.

- Step 7: If such information does not exist, nSFF1 consults the Name Resolver (NR) to determine the suitable routing/forwarding information towards the identified nSFF serving the next hop of the SFP. For future SFC packets towards this next hop, such resolved information may be locally cached, avoiding to contact the Name Resolver for every SFC packet forwarding. The packet is now marked to be sent via the network in step 8.

- Step 8: Utilizing the forwarding information determined in steps 6 or 7, nSFF1 adds the suitable transport encapsulation (TE) for the SFC packet before forwarding via the forwarders in the network towards the next nSFF22.

- Step 9: When the Packet (+NSH+TE) arrives at the outgoing nSFF2, i.e., the nSFF serving the identified next hop of the SFP, removes the TE and processes the NSH to identify the next hop information. At nSFF2 the nNLM in Figure 7 is assumed. Based on this nNLM and NSH information where SPI = 10 and SI = 252, nSFF2 identifies the next SF as www.example2.com.

SPI	SI	Next Hop(s)	Transport Encapsulation (TE)
10	252	www.example2.com	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Figure 7: nNLM at SFF2

- Step 10: If the next hop is locally registered at the nSFF, it forwards the packet (+NSH) to the service function instance, using suitable IP/MAC methods for doing so.

- Step 11: Otherwise, the outgoing nSFF adds a new TE information to the packet and forwards the packet (+NSH+TE) to the next SFF or boundary node, as shown in Figure 7.

6. nSFF Forwarding Operations

This section outlines the realization of various nSFF forwarding operations in Section 5.6. Although the operations in Section 5 utilize the notion of name-based transactions in general, we exemplify the operations here in Section 5 specifically for HTTP-based transactions to ground our description into a specific protocol for such name-based transaction. We will refer to the various steps in each of the following sub-sections.

6.1. nSFF Protocol Layers

Figure 8 shows the protocol layers, based on the high-level architecture in Figure 5.

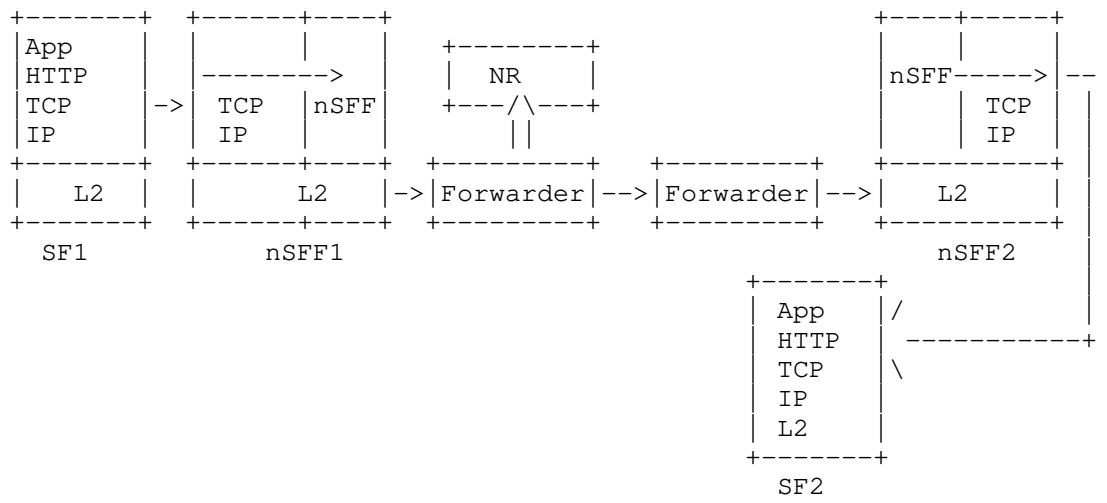


Figure 8: Protocol layers

The nSFF component here is shown as implementing a full incoming/ outgoing TCP/IP protocol stack towards the local service functions, while implementing the nSFF-NR and nSFF-nSFF protocols based on the descriptions in Section 6.2.3.

For the exchange of HTTP-based service function transactions, the nSFF terminates incoming TCP connections from as well as outgoing TCP connections to local SFs, e.g., the TCP connection from SF1 terminates at nSFF1, and nSFF1 may store the connection information, such as socket information. It also maintains the mapping information for the HTTP request such as originating SF, destination SF and socket ID. nSFF1 may implement sending keep-alive messages over the socket to maintain the connection to SF1. Upon arrival of an HTTP request from SF1, nSFF1 extracts the HTTP Request and forwards it towards the next node, as outlined in Section 6.2. Any returning response is mapped onto the suitable open socket (for the original request) and send towards SF1.

At the outgoing nSFF2, the destination SF2/Host is identified from the HTTP request message. If no TCP connection exists to the SF2, a new TCP connection is opened towards the destination SF2 and the HTTP request is sent over said TCP connection. The nSFF2 may also save the TCP connection information (such as socket information) and maintain the mapping of the socket information to the destination SF2. When an HTTP response is received from SF2 over the TCP connection, nSFF2 extracts the HTTP response, which is forwarded to

the next node. nSFF2 may maintain the TCP connection through keep-alive messages.

6.2. nSFF Operations

In this section, we present three key aspects of operations for the realization of the steps in Section 5.6, namely (i) the registration of local SFs (for step 3 in Section 5.6), (ii) the forwarding of SFC packets to and from local SFs (for step 3 and 4 as well as 10 in Section 5.6), (iii) the forwarding to a remote SF (for steps 5, 6 and 7 in Section 5.6) and to the NR as well as (iv) for the lookup of a suitable remote SF (for step 7 in Section 5.6). We also cover aspects of maintaining local lookup information for reducing lookup latency and others issues.

6.2.1. Forwarding between nSFFs and nSFF-NR

Forwarding between the distributed nSFFs as well as between nSFF and NR is realized over the operator network via a path-based approach. A path-based approach utilizes path information provided by the source of the packet for forwarding said packet in the network. This is similar to segment routing albeit differing in the type of information provided for such source-based forwarding, as described in this section. In this approach, the forwarding information to a remote nSFF or the NR is defined as a 'path identifier' (pathID) of a defined length where said "Length" field indicates the full pathID length. The payload of the packet is defined by the various operations outlined in the following sub-sections, resulting in an overall packet being transmitted. With this, the generic forwarding format (GFF) for transport over the operator network is defined in Figure 9 with the length field defining the length of the pathID provided.

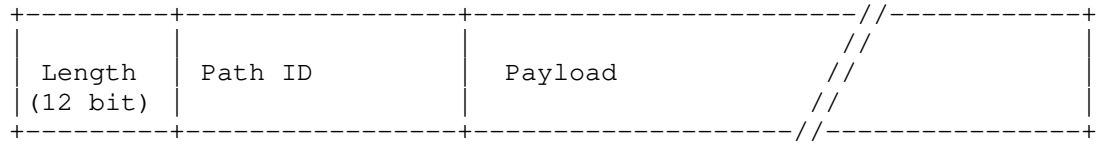


Figure 9: Generic Forwarding Format(GFF)

- Length (12 bits): Defines the length of the pathID, i.e., up to 4096 bits

- Path ID (): Variable length field, Bit field derived from IPv6 source and destination address

For the pathID information, solutions such as those in [Reed2016] can be used. Here, the IPv6 source and destination addresses are used to realize a so-called path-based forwarding from the incoming to the outgoing nSFF or the NR. The forwarders in Figure 8 are realized via SDN (software-defined networking) switches, implementing an AND/CMP operation based on arbitrary wildcard matching over the IPv6 source and destination addresses, as outlined in [Reed2016]. Note that in the case of using IPv6 address information for path-based forwarding, the step of removing the transport encapsulation at the outgoing nSFF in Figure 8 is realized by utilizing the provided (existing) IP header (which was used for the purpose of the path-based forwarding in [Reed2016]) for the purpose of next hop forwarding, such as that of IP-based routing. As described in step 8 of the extended nSFF operations, this forwarding information is used as traffic encapsulation. With the forwarding information utilizing existing IPv6 information, IP headers are utilized as TE in this case. The next hop nSFF (see Figure 8) will restore the IP header of the packet with the relevant IP information used to forward the SFC packet to SF2 or it will create a suitable TE (Transport Encapsulation) information to forward the information to another nSFF or boundary node. Forwarding operations at the intermediary forwarders, i.e., SDN switches, examine the pathID information through a flow matching rule in which a specific switch-local output port is represented through the specific assigned bit position in the pathID. Upon a positive match in said rule, the packet is forwarded on said output port.

Alternatively, the solution in [I-D.ietf-bier-multicast-http-response] suggests using a so-called BIER (Binary Indexed Explicit Replication) underlay. Here, the nSFF would be realized at the ingress to the BIER underlay, injecting the SFC packet (plus the NSH) header with BIER-based traffic encapsulation into the BIER underlay with each of the forwarders in Figure 8 being realized as a so-called Bit-Forwarding Router (BFR) [RFC8279].

6.2.1.1. Transport Protocol Considerations

Given that the proposed solution operates at the 'named transaction' level, particularly for HTTP transactions, forwarding between nSFFs and/or NR SHOULD be implemented via a transport protocol between nSFFs and/or NR in order to provide reliability, segmentation of large GFF packets, and flow control, with the GFF in Figure 9 being the basic forwarding format for this.

Note that the nSFFs act as TCP proxies at ingress and egress, thus terminating incoming and initiating outgoing HTTP sessions to SFs.

Figure 10 shows the packet format being used for the transmission of data, being adapted from the TCP header. Segmentation of large transactions into single transport protocol packets is realized through maintaining a 'Sequence number'. A 'Checksum' is calculated over a single data packet with the ones-complement TCP checksum calculation being used. The 'Window Size' field indicates the current maximum number of transport packets that are allowed in-flight by the egress nSFF. A data packet is sent without 'Data' field to indicate the end of (e.g., HTTP) transaction.

Note that in order to support future named transactions based on other application protocols, such as CoAP, future versions of the transport protocol MAY introduce a 'Type' field that indicates the type of application protocol being used between SF and nSFF with 'Type' 0x01 proposed for HTTP. This is being left for future study.

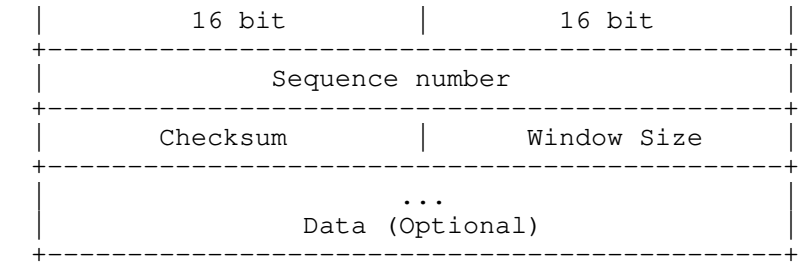


Figure 10: Transport protocol data packet format

Given the path-based forwarding being used between nSFFs, the transport protocol between nSFFs utilizes negative acknowledgements from the egress nSFF towards the ingress nSFF. The transport protocol NACK packet carries the number of NACKs as well as the specific sequence numbers being indicated as lost in the 'NACK number' field(s), as shown in Figure 11.

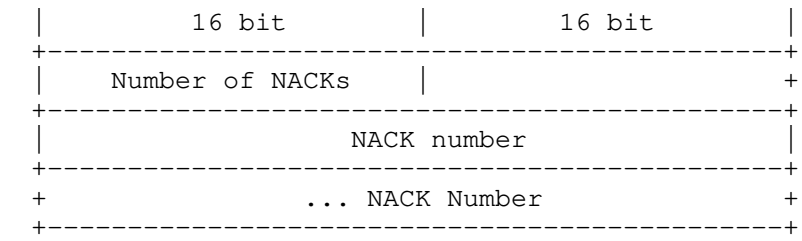


Figure 11: Transport protocol NACK packet format

If the indicated number of NACKs in a received NACK packet in non-zero, the ingress nSFF will retransmit all sequence numbers signalled in the packet, while decreasing its congestion window size for future transmissions.

If the indicated number of NACKs in a received NACK packet in zero, it will indicate the current congestion window as being successfully (and completely) being transmitted, increasing the congestion window size if smaller than the advertised 'Window Size' in Figure 10.

The maintenance of the congestion window is subject to realization at the ingress nSFF and left for further study in nSFF realizations.

6.2.2. SF Registration

As outlined in step 3 and 10 of Section 5.6, the nSFF needs to determine if the SF derived from the nNLM is locally reachable or whether the packet needs forwarding to a remote SFF. For this, a registration mechanism is provided for such local SF with the local nSFF. Two mechanisms can be used for this:

1. SF-initiated: We assume that the SF registers its FQDN to the local nSFF. As local mechanisms, we foresee that either a REST-based interface over the link-local link or configuration of the nSFF (through configuration files or management consoles) can be utilized. Such local registration event leads to the nSFF to register the given FQDN with the NR in combination with a system-unique nSFF identifier that is being used for path computation purposes in the NR. For the registration, the packet format in Figure 12 is used (inserted as the payload in the GFF of Figure 9 with the pathID towards the NR).

Figure 12: Registration packet format

- o R/D: 1 bit length (0 for Register, 1 for De-register)
- o Hash(FQDN): 16 bit length for a hash over the FQDN of the SF
- o nSFF_ID: 8 bit for a system-unique identifier for the SFF related to the SF.

We assume that the pathID towards the NR is known to the nSFF through configuration means.

The NR maintains an internal table that associates the hash(FQDN), the nSFF_id information as well as the pathID information being used for communication between nSFF and NR. The nSFF locally maintains a mapping of registered FQDNs to IP addresses, for the latter using link-local private IP addresses.

2. Orchestration-based: in this mechanism, we assume that SFC to be orchestrated and the chain being provided through an orchestration template with FQDN information associated to a compute/storage resource that is being deployed by the orchestrator. We also assume knowledge at the orchestrator of the resource topology. Based on this, the orchestrator can now use the same REST-based protocol defined in option 1 to instruct the NR to register the given FQDN, as provided in the template, at the nSFF it has identified as being the locally servicing nSFF, provided as the system-unique nSFF identifier.

6.2.3. Local SF Forwarding

There are two cases of local SF forwarding, namely the SF sending an SFC packet to the local nSFF (incoming requests) or the nSFF sending a packet to the SF (outgoing requests) as part of steps 3 and 10 in Section 5.6. In the following, we outline the operation for HTTP as an example named transaction.

As shown in Figure 8, incoming HTTP requests from SFs are extracted by terminating the incoming TCP connection at their local nSFFs at the TCP level. The nSFF MUST maintain a mapping of open TCP sockets

to HTTP requests (utilizing the URI of the request) for HTTP response association.

For outgoing HTTP requests, the nSFF utilizes the maintained mapping of locally registered FQDNs to link-local IP addresses (see Section 6.2.2 option 1). Hence, upon receiving an SFC packet from a remote nSFF (in step 9 of Section 5.6), the nSFF determines the local existence of the SF through the registration mechanisms in Section 6.2.2. If said SF does exist locally, the HTTP (+NSH) packet, after stripping the TE, is sent to the local SF as step 10 in Section 5.6 via a TCP-level connection. Outgoing nSFF SHOULD keep TCP connections open to local SFs for improving SFC packet delivery in subsequent transactions.

6.2.4. Handling of HTTP responses

When executing step 3 and 10 in Section 5.6, the SFC packet will be delivered to the locally registered next hop. As part of the HTTP protocol, responses to the HTTP request will need to be delivered on the return path to the originating nSFF (i.e., the previous hop). For this, the nSFF maintains a list of link-local connection information, e.g., sockets to the local SF and the pathID on which the request was received. Once receiving the response, nSFF consults the table to determine the pathID of the original request, forming a suitable GFF-based packet to be returned to the previous nSFF.

When receiving the HTTP response at the previous nSFF, the nSFF consults the table of (locally) open sockets to determine the suitable local SF connection, mapping the received HTTP response URI to the stored request URI. Utilizing the found socket, the HTTP response is forwarded to the locally registered SF.

6.2.5. Remote SF Forwarding

In steps 5, 6, 7, and 8 of Section 5.6, an SFC packet is forwarded to a remote nSFF based on the nNLM information for the next hop of the nSFP. Section 6.2.5.1 handles the case of suitable forwarding information to the remote nSFF not existing, therefore consulting the NR to obtain suitable information, while Section 6.2.5.2 describes the maintenance of forwarding information at the local nSFF, while Section 6.2.5.3 describes the update of stale forwarding information. Note that the forwarding described in Section 6.2.1 is used for the actual forwarding to the various nSFF components. Ultimately, Section 6.2.5.4 describes the forwarding to the remote nSFF via the forwarder network.

6.2.5.1. Remote SF Discovery

The nSFF communicates with the NR for two purposes, namely the registration and discovery of FQDNs. The packet format for the former was shown in Figure 10 in Section 6.2.2, while Figure 13 outlines the packet format for the discovery request.

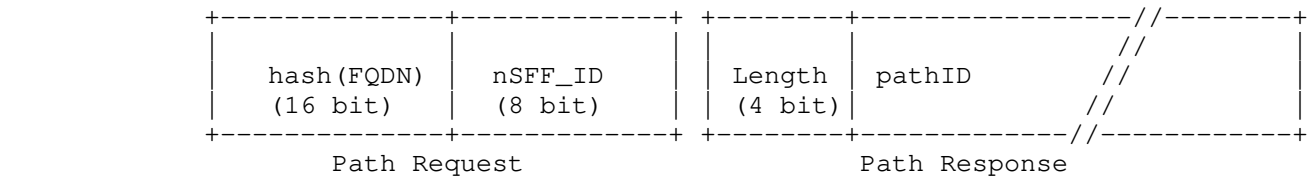


Figure 13: Discovery packet format

For Path Request:

- Hash(FQDN): 16 bit length for a hash over the FQDN of the SF
- nSFF_ID: 8 bit for a system-unique identifier for the SFF related to the SF

For Path Response:

- Length (4 bits): Defines the length of the pathID
- Path ID (): Variable length field, Bit field derived from IPv6 source and destination address

A path to a specific FQDN is requested by sending a hash of the FQDN to the NR together with its nSFF_id, receiving as a response a pathID with a length identifier. The NR SHOULD maintain a table of discovery requests that map discovered (hash of) FQDN to the nSFF_id that requested it and the pathID that is being calculated as a result of the discovery request.

The discovery request for an FQDN that has not previously been served at the nSFF (or for an FQDN whose pathID information has been flushed as a result of the update operations in Section 6.2.5.3), results in an initial latency incurred by this discovery through the NR, while any SFC packet sent over the same SFP in a subsequent transaction will utilize the nSFF local mapping table. Such initial latency can be avoided by pre-populating the FQDN-pathID mapping proactively as part of the overall orchestration procedure, e.g., alongside the distribution of the nNLM information to the nSFF.

6.2.5.2. Maintaining Forwarding Information at Local nSFF

Each nSFF MUST maintain an internal table that maps the (hash of the) FQDN information to a suitable pathID information. As outlined in step 7 of Section 5.6, if a suitable entry does not exist for a given FQDN, the pathID information is requested with the operations in Section 6.2.5.1 and the suitable entry is locally created upon receiving a reply with the forwarding operation being executed as described in Section 6.2.1.

If such entry does exist (i.e., step 6 of Section 5.6) the pathID is locally retrieved and used for the forwarding operation in Section 6.2.1.

6.2.5.3. Updating Forwarding Information at nSFF

The forwarding information maintained at each nSFF (see Section 6.2.5.2) might need to be updated for three reasons:

- An existing SF is no longer reachable: In this case, the nSFF with which the SF is locally registered, de-registers the SF explicitly at the NR by sending the packet in Figure 10 with the hashed FQDN and the R/D bit set to 1 (for de-register).

- Another SF instance has become reachable in the network (and therefore might provide a better alternative to the existing SF): in this case, the NR has received another packet with format defined in Figure 11 but a different nSFF_id value.

- Links along paths might no longer be reachable: the NR might use suitable southbound interface to transport networks to detect link failures, which it associates to the appropriate pathID bit position.

For this purpose, the packet format in Figure 14 is sent from the NR to all affected nSFFs, using the generic format in Figure 9.

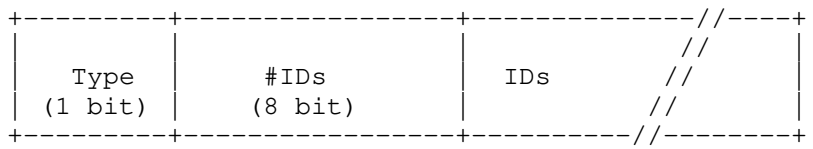


Figure 14: Path update format

- Type: 1 bit length (0 for Nsff ID, 1 for Link ID)

- #IDs: 8 bit length for number of IDs in the list

- IDs: List of IDs (Nsff ID or Link ID)

The pathID to the affected nSFFs is computed as the binary OR over all pathIDs to those nSFF_ids affected where the pathID information to the affected nSFF_id values is determined from the NR-local table maintained in the registration/deregistration operation of Section 6.2.2.

The pathID may include the type of information being updated (e.g., node identifiers of leaf nodes or link identifiers for removed links). The node identifier itself may be a special identifier to signal "ALL NODES" as being affected. The node identifier may signal changes to the network that are substantial (e.g., parallel link failures). The node identifier may trigger (e.g., recommend) purging of the entire path table (e.g., rather than the selective removal of a few nodes only).

It will include the information according to the type. The included information may also be related to the type and length information for the number of identifiers being provided.

In case 1 and 2, the Type bit is set to 1 (type nSFF_id) and the affected nSFFs are determined by those nSFFs that have previously sent SF discovery requests, utilizing the optional table mapping previously registered FQDNs to nSFF_id values. If no table mapping the (hash of) FQDN to nSFF_id is maintained, the update is sent to all nSFFs. Upon receiving the path update at the affected nSFF, all appropriate nSFF-local mapping entries to pathIDs for the hash(FQDN) identifiers provided will be removed, leading to a new NR discovery request at the next remote nSFF forwarding to the appropriate FQDN.

In case 3, the Type bit is set to 0 (type linkID) and the affected nSFFs are determined by those nSFFs whose discovery requests have previously resulted in pathIDs which include the affected link, utilizing the optional table mapping previously registered FQDNs to pathID values (see Section 6.2.5.1). Upon receiving the node identifier information in the path update, the affected nSFF will check its internal table that maps FQDNs to pathIDs to determine those pathIDs affected by the link problems and remove path information that includes the received node identifier(s). For this, the pathID entries of said table are checked against the linkID values provided in the ID entry of the path update through a binary AND/CMP operation to check the inclusion of the link in the pathIDs to the FQDNs. If any pathID is affected, the FQDN-pathID entry is removed, leading to a new NR discovery request at the next remote nSFF forwarding to the appropriate FQDN.

6.2.5.4. Forwarding to remote nSFF

Once step 5, 6, and 7 in Section 5.6 are being executed, step 8 finally sends the SFC packet to the remote nSFF, utilizing the pathID returned in the discovery request (Section 6.2.5.1) or retrieved from the local pathID mapping table. The SFC packet is placed in the payload of the generic forwarding format in Figure 9 together with the pathID and the nSFF eventually executes the forwarding operations in Section 6.2.1.

7. IANA Considerations

This document requests no IANA actions.

8. Security Considerations

The operations in Sections 5 and 6 describes the forwarding of SFC packets between named SFs based on URIs exchanged in HTTP messages. For security considerations, TLS is sufficient between originating node and Nsff, Nsff to Nsff, Nsff to destination. TLS handshake allows to determine the FQDN, which in turn is enough for the service routing decision. Supporting TLS also allows the possibility of HTTPS based transactions.

9. Acknowledgement

The authors would like to thank Dirk von Hugo and Andrew Malis for their reviews and valuable comments. We would also like to thank Joel Halpern, the chair of the SFC WG, and Adrian Farrel for guiding us through the IETF Independent Submission Editor (ISE) path.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

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

[RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <https://www.rfc-editor.org/info/rfc8279>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

10.2. Informative References

[_3GPP_SBA]
3GPP, "Technical Realization of Service Based Architecture", 3GPP TS 29.500 0.4.0, January 2018, <http://www.3gpp.org/ftp/Specs/html-info/29500.htm>.

[_3GPP_SBA_ENHANCEMENT]
3GPP, "New SID for Enhancements to the Service-Based 5G System Architecture", 3GPP S2-182904 , February 2018, <http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_126_Montreal/Docs/S2-182904.zip>.

[I-D.ietf-bier-multicast-http-response]
Purkayastha, D., Rahman, A., Trossen, D., and T. Eckert, "Applicability of BIER Multicast Overlay for Adaptive Streaming Services", draft-ietf-bier-multicast-http-response-00 (work in progress), February 2019.

[Reed2016]
Reed, M., Al-Naday, M., Thomas, N., Trossen, D., and S. Spirou, "Stateless multicast switching in software defined networks", ICC 2016, 2016, <https://arxiv.org/pdf/1511.06069.pdf>.

[Schlinker2017]
Schlinker, B., Kim, H., Cui, T., Katz-Bassett, E., Madhyastha, Harsha., Cunha, I., Quinn, J., Hassan, S., Lapukhov, P., and H. Zeng, "Engineering Egress with Edge Fabric, Steering Oceans of Content to the World", ACM SIGCOMM 2017, 2017, <https://research.fb.com/wp-content/uploads/2017/08/sigcomm17-final177-2billion.pdf>.

Authors' Addresses

Dirk Trossen
InterDigital Europe, Ltd
64 Great Eastern Street, 1st Floor
London EC2A 3QR
United Kingdom

Email: Dirk.Trossen@InterDigital.com

Debashish Purkayastha
InterDigital Communications, LLC
1001 E Hector St
Conshohocken
USA

Email: Debashish.Purkayastha@InterDigital.com

Akbar Rahman
InterDigital Communications, LLC
1000 Sherbrooke Street West
Montreal
Canada

Email: Akbar.Rahman@InterDigital.com

SFC WG
Internet-Draft
Updates: 8300 (if approved)
Intended status: Standards Track
Expires: April 10, 2019

G. Mirsky
ZTE Corp.
W. Meng
ZTE Corporation
B. Khasnabish
ZTE TX, Inc.
C. Wang
October 7, 2018

Active OAM for Service Function Chains in Networks
draft-wang-sfc-multi-layer-oam-12

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 April 10, 2019.

Copyright Notice

Table of Contents

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

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.

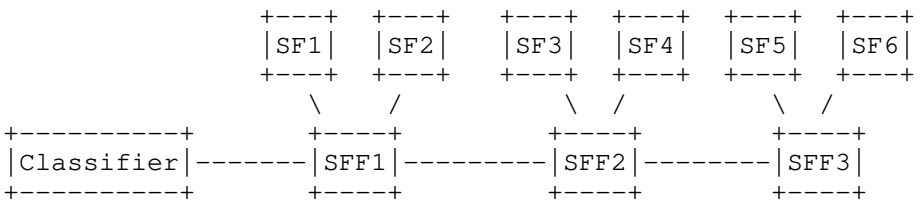


Figure 1: SFC reference model

In the example presented in Figure 1, the service SFP1 may be realized through two RSPs, RSP1(SF1--SF3--SF5) and RSP2(SF2--SF4--SF5). 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 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 - TLVs contain OAM command or data, and the type of payload determined by the Next Protocol field;

- 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 bit is clear - no OAM in TLV and the payload determined by the value of 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.

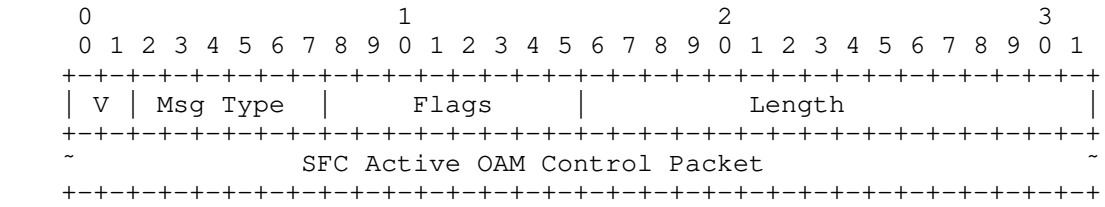


Figure 2: SFC Active OAM Header

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 BFD.

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.

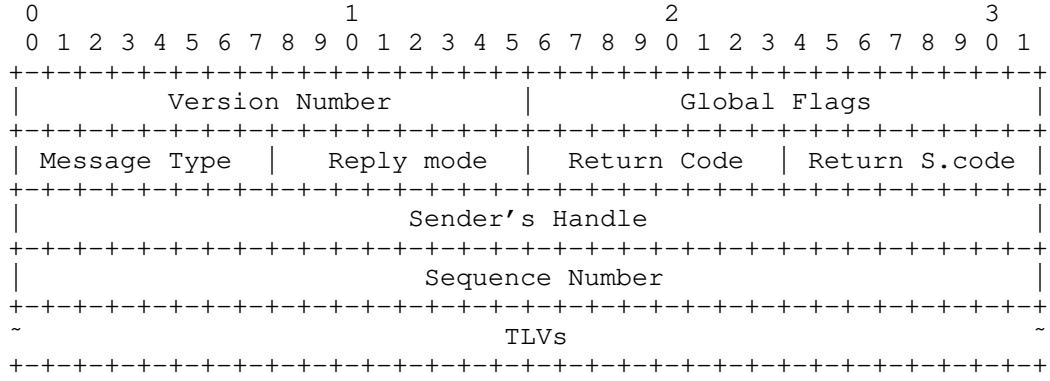


Figure 3: SFC Echo Request/Reply format

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 filed 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.

5.1. 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:

- 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.

- Reply via an IPv4/IPv6 UDP Packet (TBA6) value likely will be the most used.

- Reply via Application Level Control Channel (TBA7) value if the SFP may have bi-directional paths.

- Reply via Specified Path (TBA7) 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.

5.2. SFC Echo Request Reception

5.3. 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.

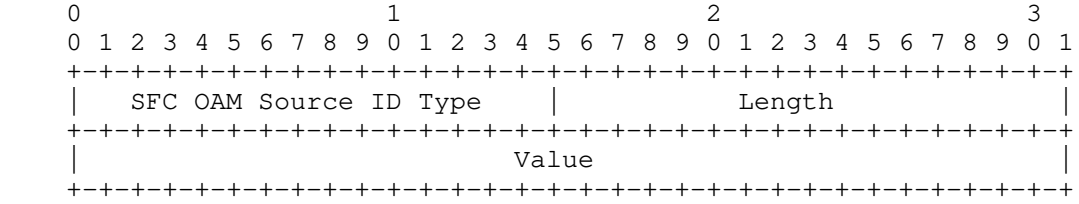


Figure 4: SFC Source TLV

where

SFC OAM Source Id Type is two octets in length and has the value of TBA9 Section 8.6.

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 8.7.

5.4. Overlay Echo Reply Reception

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.

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 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 6:

Value	Description	Reference
0	Reserved	This document
1- 32759	Unassigned	IETF Review
32760 - 65279	Unassigned	First Come First Served
65280 - 65519	Experimental	This document
65520 - 65534	Private Use	This document
65535	Reserved	This document

Table 6: 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 7: SFC OAM Source IP Address Type

8.7. 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 5.3	This document

Table 8: 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>.

[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
ZTE TX, Inc.
55 Madison Avenue, Suite 160
Morristown, New Jersey 07960
USA

Email: bhumip.khasnabish@ztetx.com

Cui Wang

Email: lindawangjoy@gmail.com