

SFC WG
Internet-Draft
Intended status: Standards Track
Expires: August 31, 2018

T. Ao
ZTE Corporation
G. Mirsky
ZTE Corp.
Z. Chen
China Telecom
February 27, 2018

SFC OAM for path consistency
draft-ao-sfc-oam-path-consistency-02

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.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 31, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Conventions used in this document	3
2.1. Terminology	3
2.2. Requirements Language	3
3. Consistency OAM: Theory of Operation	3
3.1. COAM packet	4
3.2. SF information Sub-TLV	4
3.3. SF information Sub-TLV construction	5
4. Security Considerations	6
5. IANA Considerations	6
5.1. COAM Message Types	6
5.2. SFF Information Record TLV Type	7
5.3. SF Information Sub-TLV Type	7
5.4. SF Identifier Types	7
6. Acknowledgements	8
7. References	8
7.1. Normative References	8
7.2. Informational References	9
Authors' Addresses	9

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:

- o TBA1 - COAM Request
- o 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.

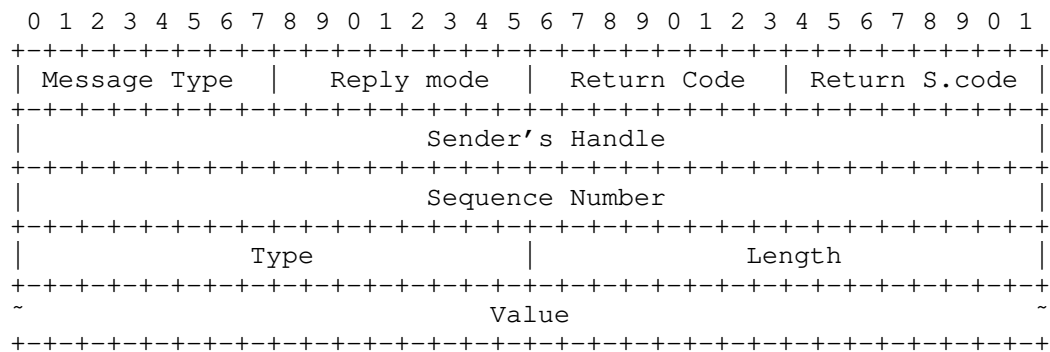


Figure 1: COAM Packet Header

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.

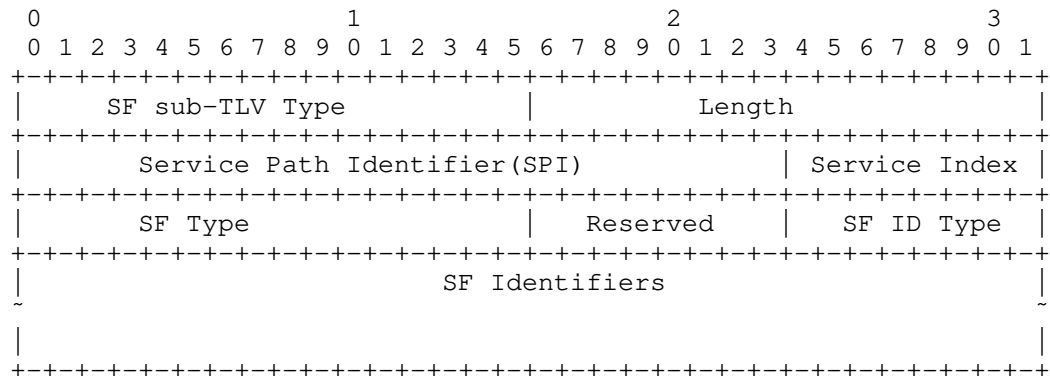


Figure 2: Service Function information sub-TLV

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.

4. Security Considerations

Security considerations discussed in [I-D.ietf-sfc-nsh] 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-02 (work in progress), October 2017.
- [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, "Multi-Layer Active OAM for Service Function Chains in Networks", draft-wang-sfc-multi-layer-oam-10 (work in progress), September 2017.
- [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>>.

- [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>>.

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: 1 October 2021

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

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 1 October 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Conventions used in this document	3
2.1. Acronyms	3
2.2. Requirements Language	3
3. Consistency OAM: Theory of Operation	3
3.1. COAM packet	4
3.2. SFF Information Record TLV	5
3.3. SF Information Sub-TLV	6
3.4. SF Information Sub-TLV Construction	7
3.4.1. Multiple SFs as hops of SFP	7
3.4.2. Multiple SFs for load balance	8
4. Security Considerations	8
5. IANA Considerations	8
5.1. COAM Message Types	9
5.2. SFF Information Record TLV Type	9
5.3. SF Information Sub-TLV Type	9
5.4. SF Identifier Types	10
6. Acknowledgements	10
7. References	10
7.1. Normative References	10
7.2. Informational References	11
Authors' Addresses	12

1. Introduction

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

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

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

2. Conventions used in this document

2.1. Acronyms

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

SFF: Service Function Forwarder

SF: Service Function

OAM: Operation, Administration and Maintenance

SFP: Service Function Path

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

MAC: Message Authentication Code

2.2. Requirements Language

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

3. Consistency OAM: Theory of Operation

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

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

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

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

3.1. COAM packet

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

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

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

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

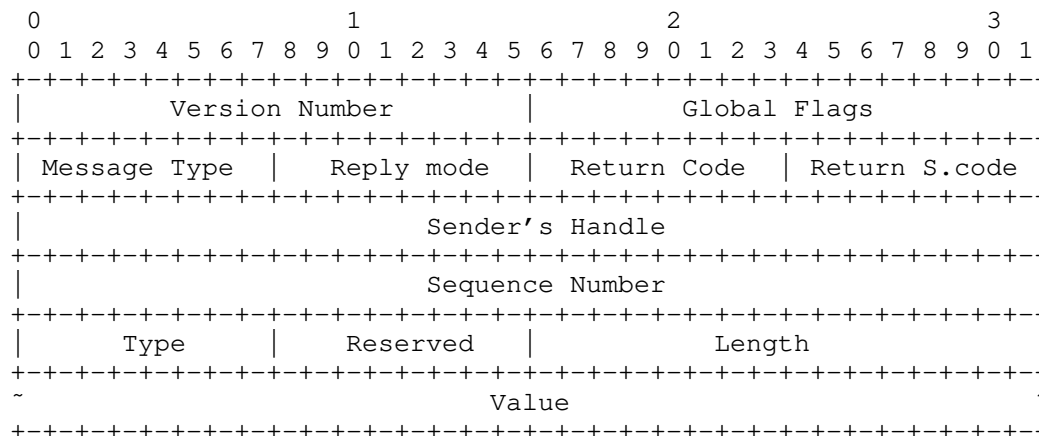


Figure 1: COAM Packet Header

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

3.2. SFF Information Record TLV

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

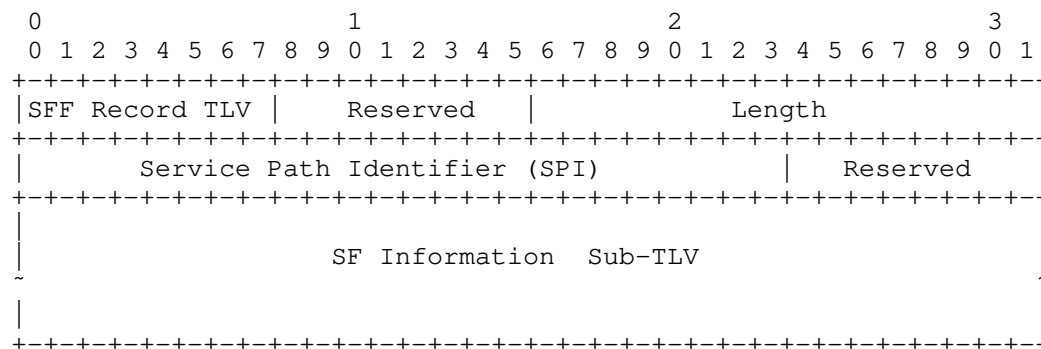


Figure 2: SFF Information Record TLV

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

Reserved - one-octet-long field.

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

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

3.3. SF Information Sub-TLV

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

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

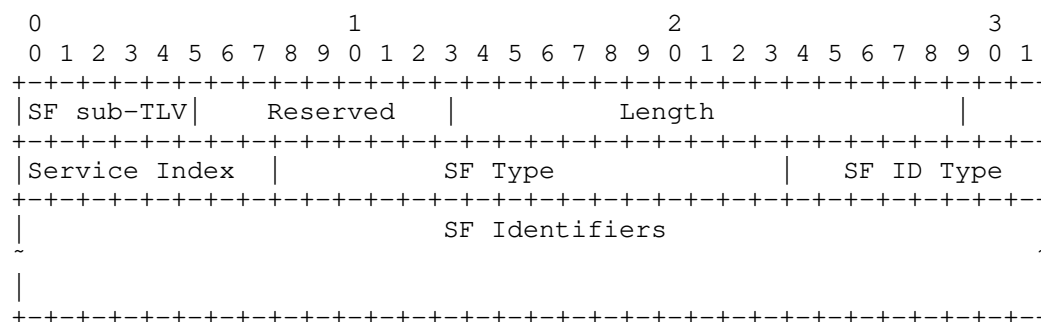


Figure 3: Service Function information sub-TLV

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

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

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

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

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

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

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

3.4. SF Information Sub-TLV Construction

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

3.4.1. Multiple SFs as hops of SFP

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

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

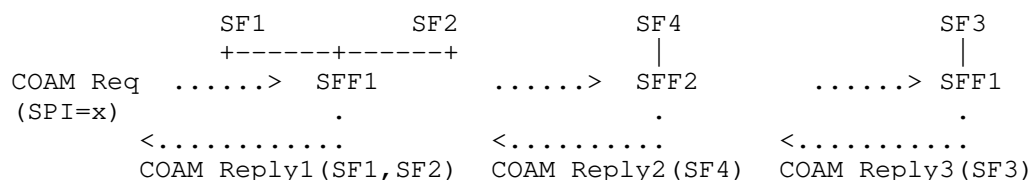


Figure 4: Example 1 for COAM Reply with multiple SFs

3.4.2. Multiple SFs for load balance

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

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

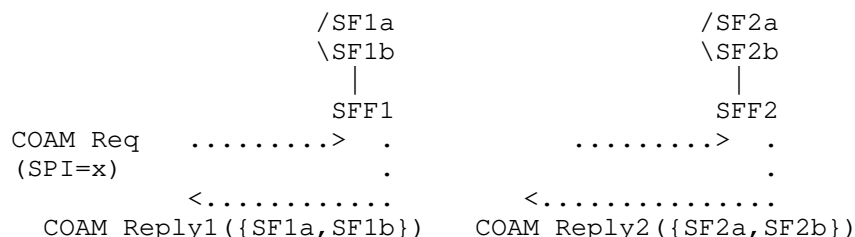


Figure 5: Example 2 for COAM Reply with multiple SFs

4. Security Considerations

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

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

5. IANA Considerations

5.1. COAM Message Types

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

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

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

5.2. SFF Information Record TLV Type

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

Value	Description	Reference
TBA3	SFF Information Record Type	This document

Table 2: SFF-Information Record

5.3. SF Information Sub-TLV Type

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

Value	Description	Reference
TBA4	SF Information	This document

Table 3: SF-Information Sub-TLV Type

5.4. SF Identifier Types

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

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

Table 4: SF Identifier Type

6. Acknowledgements

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

7. References

7.1. Normative References

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

- [I-D.ietf-sfc-nsh-integrity]
Boucadair, M., Reddy, T., and D. Wing, "Integrity Protection for the Network Service Header (NSH) and Encryption of Sensitive Context Headers", Work in Progress, Internet-Draft, draft-ietf-sfc-nsh-integrity-05, 23 March 2021, <<https://tools.ietf.org/html/draft-ietf-sfc-nsh-integrity-05>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <<https://www.rfc-editor.org/info/rfc8300>>.

7.2. Informational References

- [I-D.ietf-bess-nsh-bgp-control-plane]
Farrel, A., Drake, J., Rosen, E., Uttaro, J., and L. Jalil, "BGP Control Plane for the Network Service Header in Service Function Chaining", Work in Progress, Internet-Draft, draft-ietf-bess-nsh-bgp-control-plane-18, 21 August 2020, <<https://tools.ietf.org/html/draft-ietf-bess-nsh-bgp-control-plane-18>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC8924] Aldrin, S., Pignataro, C., Ed., Kumar, N., Ed., Krishnan, R., and A. Ghanwani, "Service Function Chaining (SFC) Operations, Administration, and Maintenance (OAM) Framework", RFC 8924, DOI 10.17487/RFC8924, October 2020, <<https://www.rfc-editor.org/info/rfc8924>>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

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

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

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

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

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

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

Email: kleung@cisco.com

Gyan Mishra
Verizon Inc.

Email: gyan.s.mishra@verizon.com

SFC WG
Internet-Draft
Intended status: Experimental
Expires: September 6, 2018

CJ. Bernardos
UC3M
A. Mourad
InterDigital
March 5, 2018

Service Function discovery in fog environments
draft-bernardos-sfc-discovery-00

Abstract

Service function chaining (SFC) allows the instantiation of an ordered set of service functions and subsequent "steering" of traffic through them. Service functions provide a specific treatment of received packets, therefore they need to be known so they can be used in a given service composition via SFC. This document discusses the need for service function discovery mechanisms and propose some solutions for sfc-aware nodes to discover available service functions in fog environments.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Problem statement	4
3.1. Discovery of SF in a multi-provider fog/edge environment	4
4. Network-based SF discovery	6
4.1. ICMPv6-based SF discovery	8
4.2. DHCPv6-based SF discovery	8
5. IANA Considerations	8
6. Security Considerations	8
7. Acknowledgments	8
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Authors' Addresses	9

1. Introduction

Virtualization of functions provides operators with tools to deploy new services much faster, as compared to the traditional use of monolithic and tightly integrated dedicated machinery. As a natural next step, mobile network operators need to re-think how to evolve their existing network infrastructures and how to deploy new ones to address the challenges posed by the increasing customers' demands, as well as by the huge competition among operators. All these changes are triggering the need for a modification in the way operators and infrastructure providers operate their networks, as they need to significantly reduce the costs incurred in deploying a new service and operating it. Some of the mechanisms that are being considered and already adopted by operators include: sharing of network infrastructure to reduce costs, virtualization of core servers running in data centers as a way of supporting their load-aware elastic dimensioning, and dynamic energy policies to reduce the monthly electricity bill. However, this has proved to be tough to put in practice, and not enough. Indeed, it is not easy to deploy new mechanisms in a running operational network due to the high dependency on proprietary (and sometime obscure) protocols and interfaces, which are complex to manage and often require configuring multiple devices in a decentralized way.

Service Functions are widely deployed and essential in many networks. These Service Functions provide a range of features such as security,

WAN acceleration, and server load balancing. Service Functions may be instantiated at different points in the network infrastructure such as data center, the WAN, the RAN, and even on mobile nodes.

Service functions (SFs), also referred to as VNFs, or just functions, are hosted on compute, storage and networking resources. The hosting environment of a function is called Service Function Provider or NFVI-PoP (using ETSI NFV terminology).

With the arrival of virtualization, the deployment model for service function is evolving to one where the traffic is steered through the functions wherever they are deployed (functions do not need to be deployed in the traffic path anymore). For a given service, the abstracted view of the required service functions and the order in which they are to be applied is called a Service Function Chain (SFC). An SFC is instantiated through selection of specific service function instances on specific network nodes to form a service graph: this is called a Service Function Path (SFP). The service functions may be applied at any layer within the network protocol stack (network layer, transport layer, application layer, etc.).

A mobile terminal can benefit from using service function chaining at the edge/fog to enhance existing applications or to enable new ones. In order to do so, discovery of available service functions is required. This document focuses on this aspect.

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 [RFC2119].

While [RFC2119] describes interpretations of these key words in terms of protocol specifications and implementations, they are used in this document to describe requirements for the SFC mechanisms to efficiently enable fog RAN.

The following terms used in this document are defined by the IETF in [RFC7665] and [I-D.ietf-bess-nsh-bgp-control-plane]:

Service Function (SF): a function that is responsible for specific treatment of received packets (e.g., firewall, load balancer).

Service Function Chain (SFC): for a given service, the abstracted view of the required service functions and the order in which they are to be applied. This is somehow equivalent to the Network Function Forwarding Graph (NF-FG) at ETSI.

Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF.

SFI: SF instance.

Service Function Path (SFP): the selection of specific service function instances on specific network nodes to form a service graph through which an SFC is instantiated.

A Service Function Type (SFT) that is the category of Service Function that is provided (such as "firewall").

3. Problem statement

[RFC7665] describes an architecture for the specification, creation, and ongoing maintenance of Service Function Chains (SFCs) in a network. It includes architectural concepts, principles, and components used in the construction of composite services through deployment of SFCs. In this architecture, a key element is the service function (SF), which is a function that is responsible for specific treatment of received packets (e.g., a firewall).

So far, how the SFs are discovered and composed has been out of the scope of discussions in IETF. There is however a need to define mechanisms that allow SF discovery in fog environments [I-D.bernardos-sfc-fog-ran]. Note that the mechanisms described in this document address fog environments. There are other mechanisms described, like [I-D.ietf-bess-nsh-bgp-control-plane], that cover generic SF discovery in more traditional environments. Some of the solutions described in the present document might be of applicable to other scenarios as well.

3.1. Discovery of SF in a multi-provider fog/edge environment

The need to provide networking, computing, and storage capabilities closer to the users has recently emerged, due to the demands from 5G applications of very low latency, leading to what is known today as the concept of intelligent edge. ETSI has been the first to address this need recently by developing the framework of mobile edge computing (MEC). Such an intelligent edge could not be envisaged without virtualization. Beyond applications, it raises a clear opportunity for networking functions to execute at the edge benefiting from inherent low latencies. Being in close proximity to the access, the edge becomes an attractive place for hosting different functions, saving bandwidth in their respective domains and

offering local breakout options where required. Whilst it is appreciated the particular challenge for the intelligent edge concept in dealing with mobile users, the edge virtualization substrate has been largely assumed to be fixed or stationary. Although little developed, the intelligent edge concept is being extended further to scenarios where for example the edge computing substrate is on the move, e.g., on-board a car or a train, or that it is distributed further down the edge, even integrating resources from different stakeholders, into what is known as the fog.

Service composition is a powerful tool which can provide significant benefits when applied in a softwarized network environment. While it is being explored in the core part of networks to compose services using DPIs (Deep Packet Inspections), firewalls, parental control, video accelerators, etc., its applicability to the RAN (Radio Access Network), and in particular to the edge and the fog, has not been explored yet.

Running functions (standalone functions or service function chains) at the edge of the network has clear advantages. For example, it enables offloading functions from the end-user terminal so that it can become more efficient in terms of cost and energy consumption.

A mobile terminal can benefit from using service function chaining at the edge/fog to enhance existing applications or to enable new ones. Some examples of such applications are: privacy enhancement by local anchoring, opportunistic local breakout, assisted encryption, video transcoding, personal firewalling, etc. The mobile terminal might look for function hosting opportunities at the edge for various reasons such as:

- o to increase battery life in critical situations by offloading energy demanding operations (e.g., video transcoding, augmented reality) to the edge/cloud;
- o to reduce communications latency (e.g., by using local breakout at the edge for selected applications demanding low latency);
- o to enable new functions (e.g., privacy improvements, personal firewalling) which demand additional intelligence/resources at the network;
- o to benefit from context information available at the edge (e.g., enrich networking decisions by executing functions at the edge using RAN information);

Several key challenges need to be addressed to enable controlled service function chaining for a mobile terminal, and one of them is

the discovery of the functions available for use at the Fog/Edge/Cloud.

4. Network-based SF discovery

In this section we describe several mechanisms for a mobile SFC-aware node to discover what SFs are available in the network. Different alternatives (protocol containers) are considered to enable the mobile node to obtain the following information per SF available:

- o Service Function Type, identifying the category of SF provided.
- o SFC-aware: Yes/No. Indicates if the SF is SFC-aware.
- o Route Distinguisher (RD): IP address indicating the location of the SF(I).
- o Pricing/costs details.
- o Migration capabilities of the SF: whether a given function can be moved to another provider (potentially including information about compatible providers topologically close).
- o Mobility of the device hosting the SF, with e.g. the following sub-options:
 - Level: no, low, high; or a corresponding scale (e.g., 1 to 10).
 - Current geographical area (e.g., GPS coordinates, post code).
 - Target moving area (e.g., GPS coordinates, post code).
- o Power source of the device hosting the SF, with e.g. the following sub-options:
 - Battery: Yes/No. If Yes, the following sub-options could be defined:
 - Capacity of the battery (e.g., mmWh).
 - Charge status (e.g., %).
 - Lifetime (e.g., minutes).

Figure 1 shows the generic mechanism for SF discovery, with network support. In this scenario, SFs (which might belong to different administrative domains) are previously registered at the network, which can then reply to requests sent from mobile nodes that have

just attached to the network. A request might optionally include the SFs of interest for the terminal, instead of a request for all known SFs.

The network might also send periodic advertisements in addition to responses to solicited requests. These responses/advertisements include the information about known SFs (or only about the ones queried by the terminal), which can then be used by the terminal to decide whether to use (some of) them in a certain SFC. How the mobile terminal then configures this SFC is not covered in this document.

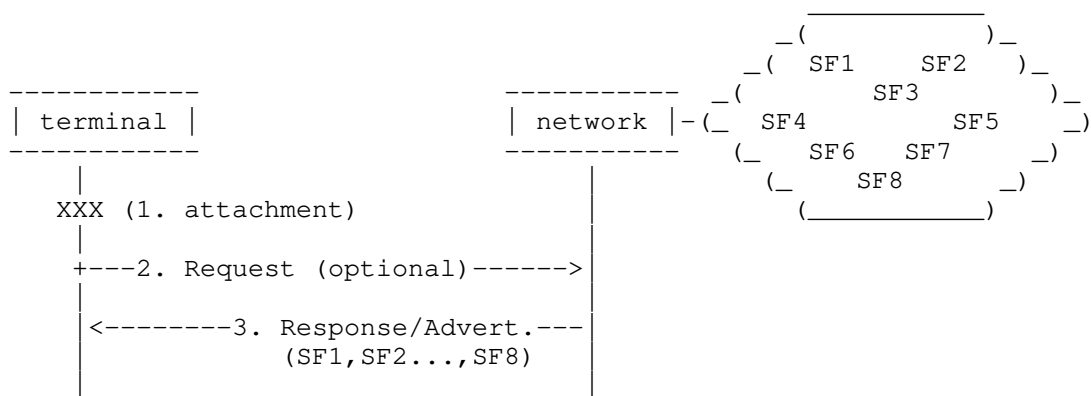


Figure 1: SF (network) discovery

In addition to the discovery of SFs at the infrastructure, mobile terminals can also host SF(I)s, and therefore they also need to be discovered. A similar approach can be followed, as shown in Figure 2.

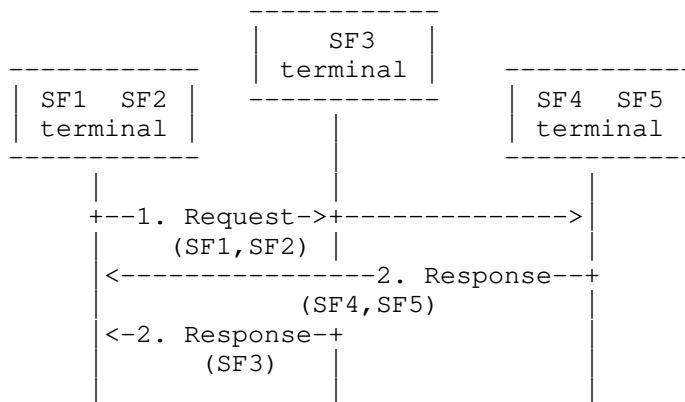


Figure 2: SF (mobiles) discovery

SFs might belong to different administrative domains. This might require the use of additional security and authentication mechanisms. Policies can be used (both in single and multi-domain scenarios) to adapt/limit the type and number of SFs that are advertised, depending on the relationship of the requester and the advertiser.

Next sections describe different protocol alternatives for this SF discovery in fog environments.

4.1. ICMPv6-based SF discovery

TBD.

4.2. DHCPv6-based SF discovery

TBD.

5. IANA Considerations

N/A.

6. Security Considerations

TBD.

7. Acknowledgments

The work in this draft will be further developed and explored under the framework of the H2020 5G-CORAL project (Grant 761586).

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>>.

8.2. Informative References

- [I-D.bernardos-sfc-fog-ran]
Bernardos, C., Rahman, A., and A. Mourad, "Service Function Chaining Use Cases in Fog RAN", draft-bernardos-sfc-fog-ran-02 (work in progress), June 2017.
- [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-02 (work in progress), October 2017.
- [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

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: <http://www.it.uc3m.es/cjbc/>

Alain Mourad
InterDigital Europe

Email: Alain.Mourad@InterDigital.com
URI: <http://www.InterDigital.com/>

SFC WG
Internet-Draft
Intended status: Experimental
Expires: April 24, 2022

CJ. Bernardos
UC3M
A. Mourad
InterDigital
October 21, 2021

Service Function discovery in fog environments
draft-bernardos-sfc-discovery-07

Abstract

Service function chaining (SFC) allows the instantiation of an ordered set of service functions and subsequent "steering" of traffic through them. Service functions provide a specific treatment of received packets, therefore they need to be known so they can be used in a given service composition via SFC. This document discusses the need for service function discovery mechanisms and propose some solutions for sfc-aware nodes to discover available service functions in fog environments.

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 24, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Problem statement	4
3.1. Discovery of SF in a multi-provider fog/edge environment	4
4. Network-based SF discovery	5
4.1. ICMPv6-based SF discovery	8
4.2. DHCPv6-based SF discovery	8
5. IANA Considerations	8
6. Security Considerations	8
7. Acknowledgments	8
8. Informative References	8
Authors' Addresses	8

1. Introduction

Virtualization of functions provides operators with tools to deploy new services much faster, as compared to the traditional use of monolithic and tightly integrated dedicated machinery. As a natural next step, mobile network operators need to re-think how to evolve their existing network infrastructures and how to deploy new ones to address the challenges posed by the increasing customers' demands, as well as by the huge competition among operators. All these changes are triggering the need for a modification in the way operators and infrastructure providers operate their networks, as they need to significantly reduce the costs incurred in deploying a new service and operating it. Some of the mechanisms that are being considered and already adopted by operators include: sharing of network infrastructure to reduce costs, virtualization of core servers running in data centers as a way of supporting their load-aware elastic dimensioning, and dynamic energy policies to reduce the monthly electricity bill. However, this has proved to be tough to put in practice, and not enough. Indeed, it is not easy to deploy new mechanisms in a running operational network due to the high dependency on proprietary (and sometime obscure) protocols and interfaces, which are complex to manage and often require configuring multiple devices in a decentralized way.

Service Functions are widely deployed and essential in many networks. These Service Functions provide a range of features such as security, WAN acceleration, and server load balancing. Service Functions may

be instantiated at different points in the network infrastructure such as data center, the WAN, the RAN, and even on mobile nodes.

Service functions (SFs), also referred to as VNFs, or just functions, are hosted on compute, storage and networking resources. The hosting environment of a function is called Service Function Provider or NFVI-PoP (using ETSI NFV terminology).

With the arrival of virtualization, the deployment model for service function is evolving to one where the traffic is steered through the functions wherever they are deployed (functions do not need to be deployed in the traffic path anymore). For a given service, the abstracted view of the required service functions and the order in which they are to be applied is called a Service Function Chain (SFC). An SFC is instantiated through selection of specific service function instances on specific network nodes to form a service graph: this is called a Service Function Path (SFP). The service functions may be applied at any layer within the network protocol stack (network layer, transport layer, application layer, etc.).

A mobile terminal can benefit from using service function chaining at the edge/fog to enhance existing applications or to enable new ones. In order to do so, discovery of available service functions is required. This document focuses on this aspect.

2. Terminology

The following terms used in this document are defined by the IETF in [RFC7665] and [RFC9015]:

Service Function (SF): a function that is responsible for specific treatment of received packets (e.g., firewall, load balancer).

Service Function Chain (SFC): for a given service, the abstracted view of the required service functions and the order in which they are to be applied. This is somehow equivalent to the Network Function Forwarding Graph (NF-FG) at ETSI.

Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF.

SFI: SF instance.

Service Function Path (SFP): the selection of specific service function instances on specific network nodes to form a service graph through which an SFC is instantiated.

A Service Function Type (SFT) that is the category of Service Function that is provided (such as "firewall").

3. Problem statement

[RFC7665] describes an architecture for the specification, creation, and ongoing maintenance of Service Function Chains (SFCs) in a network. It includes architectural concepts, principles, and components used in the construction of composite services through deployment of SFCs. In this architecture, a key element is the service function (SF), which is a function that is responsible for specific treatment of received packets (e.g., a firewall).

So far, how the SFs are discovered and composed has been out of the scope of discussions in IETF. There is however a need to define mechanisms that allow SF discovery in fog environments [I-D.bernardos-sfc-fog-ran]. Note that the mechanisms described in this document address fog environments. There are other mechanisms described, like [RFC9015], that cover generic SF discovery in more traditional environments. Some of the solutions described in the present document might be of applicable to other scenarios as well.

3.1. Discovery of SF in a multi-provider fog/edge environment

The need to provide networking, computing, and storage capabilities closer to the users has recently emerged, due to the demands from 5G applications of very low latency, leading to what is known today as the concept of intelligent edge. ETSI has been the first to address this need recently by developing the framework of mobile edge computing (MEC). Such an intelligent edge could not be envisaged without virtualization. Beyond applications, it raises a clear opportunity for networking functions to execute at the edge benefiting from inherent low latencies. Being in close proximity to the access, the edge becomes an attractive place for hosting different functions, saving bandwidth in their respective domains and offering local breakout options where required. Whilst it is appreciated the particular challenge for the intelligent edge concept in dealing with mobile users, the edge virtualization substrate has been largely assumed to be fixed or stationary. Although little developed, the intelligent edge concept is being extended further to scenarios where for example the edge computing substrate is on the move, e.g., on-board a car or a train, or that it is distributed further down the edge, even integrating resources from different stakeholders, into what is known as the fog.

Service composition is a powerful tool which can provide significant benefits when applied in a softwarized network environment. While it is being explored in the core part of networks to compose services using DPIs (Deep Packet Inspections), firewalls, parental control, video accelerators, etc., its applicability to the RAN (Radio Access Network), and in particular to the edge and the fog, has not been explored yet.

Running functions (standalone functions or service function chains) at the edge of the network has clear advantages. For example, it enables offloading functions from the end-user terminal so that it can become more efficient in terms of cost and energy consumption.

A mobile terminal can benefit from using service function chaining at the edge/fog to enhance existing applications or to enable new ones. Some examples of such applications are: privacy enhancement by local anchoring, opportunistic local breakout, assisted encryption, video transcoding, personal firewalling, etc. The mobile terminal might look for function hosting opportunities at the edge for various reasons such as:

- o to increase battery life in critical situations by offloading energy demanding operations (e.g., video transcoding, augmented reality) to the edge/cloud;
- o to reduce communications latency (e.g., by using local breakout at the edge for selected applications demanding low latency);
- o to enable new functions (e.g., privacy improvements, personal firewalling) which demand additional intelligence/resources at the network;
- o to benefit from context information available at the edge (e.g., enrich networking decisions by executing functions at the edge using RAN information);

Several key challenges need to be addressed to enable controlled service function chaining for a mobile terminal, and one of them is the discovery of the functions available for use at the Fog/Edge/Cloud.

4. Network-based SF discovery

In this section we describe several mechanisms for a mobile SFC-aware node to discover what SFs are available in the network. Different alternatives (protocol containers) are considered to enable the mobile node to obtain the following information per SF available:

- o Service Function Type, identifying the category of SF provided.
- o SFC-aware: Yes/No. Indicates if the SF is SFC-aware.
- o Route Distinguisher (RD): IP address indicating the location of the SF(I).
- o Pricing/costs details.
- o Migration capabilities of the SF: whether a given function can be moved to another provider (potentially including information about compatible providers topologically close).
- o Mobility of the device hosting the SF, with e.g. the following sub-options:
 - Level: no, low, high; or a corresponding scale (e.g., 1 to 10).
 - Current geographical area (e.g., GPS coordinates, post code).
 - Target moving area (e.g., GPS coordinates, post code).
- o Power source of the device hosting the SF, with e.g. the following sub-options:
 - Battery: Yes/No. If Yes, the following sub-options could be defined:
 - Capacity of the battery (e.g., mmWh).
 - Charge status (e.g., %).
 - Lifetime (e.g., minutes).

Figure 1 shows the generic mechanism for SF discovery, with network support. In this scenario, SFs (which might belong to different administrative domains) are previously registered at the network, which can then reply to requests sent from mobile nodes that have just attached to the network. A request might optionally include the SFs of interest for the terminal, instead of a request for all known SFs.

The network might also send periodic advertisements in addition to responses to solicited requests. These responses/advertisements include the information about known SFs (or only about the ones queried by the terminal), which can then be used by the terminal to decide whether to use (some of) them in a certain SFC. How the

mobile terminal then configures this SFC is not covered in this document.

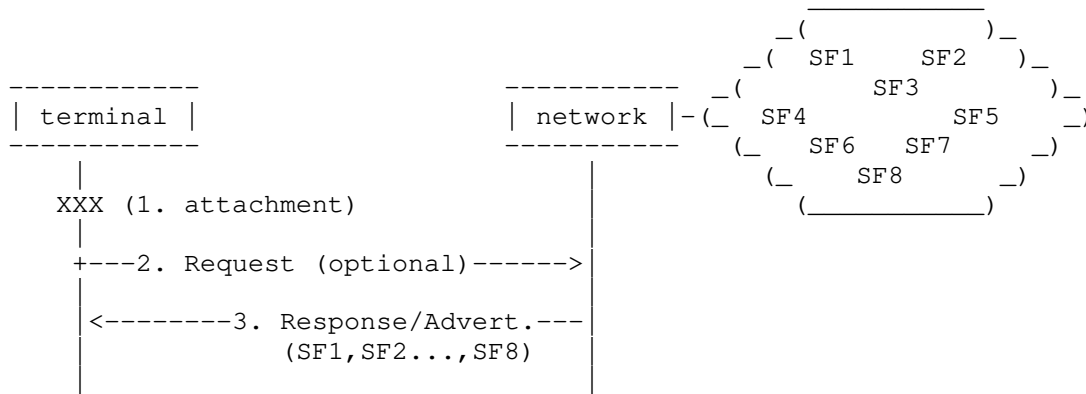


Figure 1: SF (network) discovery

In addition to the discovery of SFs at the infrastructure, mobile terminals can also host SF(I)s, and therefore they also need to be discovered. A similar approach can be followed, as shown in Figure 2.

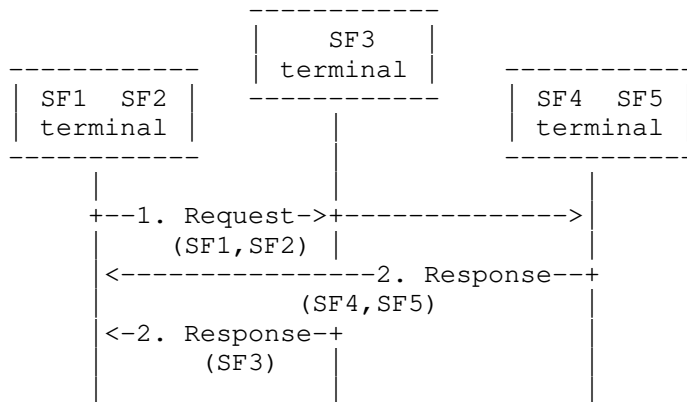


Figure 2: SF (mobiles) discovery

SFs might belong to different administrative domains. This might require the use of additional security and authentication mechanisms. Policies can be used (both in single and multi-domain scenarios) to adapt/limit the type and number of SFs that are advertised, depending on the relationship of the requester and the advertiser.

Next sections describe different protocol alternatives for this SF discovery in fog environments.

4.1. ICMPv6-based SF discovery

TBD.

4.2. DHCPv6-based SF discovery

TBD.

5. IANA Considerations

N/A.

6. Security Considerations

TBD.

7. Acknowledgments

The work in this draft has been explored under the framework of the H2020 5G-DIVE project (Grant 859881).

8. Informative References

- [I-D.bernardos-sfc-fog-ran]
Bernardos, C. J., Rahman, A., and A. Mourad, "Service Function Chaining Use Cases in Fog RAN", draft-bernardos-sfc-fog-ran-09 (work in progress), March 2021.
- [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>>.
- [RFC9015] Farrel, A., Drake, J., Rosen, E., Uttaro, J., and L. Jalil, "BGP Control Plane for the Network Service Header in Service Function Chaining", RFC 9015, DOI 10.17487/RFC9015, June 2021, <<https://www.rfc-editor.org/info/rfc9015>>.

Authors' Addresses

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: <http://www.it.uc3m.es/cjbc/>

Alain Mourad
InterDigital Europe

Email: Alain.Mourad@InterDigital.com
URI: <http://www.InterDigital.com/>

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: May 3, 2018

F. Brockners
S. Bhandari
S. Dara
C. Pignataro
Cisco
J. Leddy
Comcast
S. Youell
JMPC
D. Mozes
Mellanox Technologies Ltd.
T. Mizrahi
Marvell
October 30, 2017

Proof of Transit
draft-brockners-proof-of-transit-04

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

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

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

1. Introduction

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

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

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

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

2. Conventions

Abbreviations used in this document:

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

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

LPC: Lagrange Polynomial Constants

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

NSH: Network Service Header

POT: Proof of Transit

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

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

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

SFC: Service Function Chain

SR: Segment Routing

3. Proof of Transit

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

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

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

3.1. Basic Idea

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

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

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

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

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

3.2. Solution Approach

Solution approach: The overall algorithm uses two polynomials: POLY-1 and POLY-2. POLY-1 is secret and constant. Each node gets a point on POLY-1 at setup-time and keeps it secret. POLY-2 is public, random and per packet. Each node generates a point on POLY-2 each time a packet crosses it. Each node then calculates (point on POLY-1 + point on POLY-2) to get a (point on POLY-3) and passes it to verifier by adding it to each packet. The verifier constructs POLY-3 from the points given by all the nodes and cross checks whether $POLY-3 = POLY-1 + POLY-2$. 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 $(\text{Share}(\text{POLY-1}) + \text{Share}(\text{POLY-2}))$ and CML is updated with this sum. This step is performed by each node until the packet completes the path. The verifier also performs the step with its respective share.

3.2.3. Verification

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

3.3. Illustrative Example

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

3.3.1. Basic Version

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

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

3.3.1.1. Secret Shares

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

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

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

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

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

3.3.1.2. Lagrange Polynomials

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

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

$$\begin{aligned} 11(x) &= (((x-x_0) / (x_1-x_0)) * ((x-x_2)/(x_1-x_2))) \bmod 53 = \\ &(-5 + 7x/2 - (1/2)x^2) \bmod 53 \end{aligned}$$

$$\begin{aligned} 12(x) &= (((x-x_0) / (x_2-x_0)) * ((x-x_1)/(x_2-x_1))) \bmod 53 = \\ &(8/3 - 2 + (1/3)x^2) \bmod 53 \end{aligned}$$

3.3.1.3. LPC Computation

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

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

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

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

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

3.3.1.4. Reconstruction

Reconstruction of the polynomial is well-defined as

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

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

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

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

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

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

3.3.1.5. Verification

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

3.3.2. Enhanced Version

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

3.3.2.1. Random Polynomial

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

3.3.2.2. Reconstruction

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

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

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

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

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

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

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

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

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

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

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

The verification using CML is discussed in next section.

3.3.2.3. Verification

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

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

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

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

3.3.3. Final Version

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

3.4. Operational Aspects

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

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

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

3.5. Alternative Approach

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

3.5.1. Basic Idea

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

3.5.2. Pros

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

3.5.3. Cons

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

4. Sizing the Data for Proof of Transit

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

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

2. CML: Cumulative

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

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

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

5. Node Configuration

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

5.1. Procedure

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

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

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

5.2. YANG Model

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

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

  yang-version 1;

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

  prefix ietf-pot-profile;

  organization "IETF xxx Working Group";
```

```
contact "";

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

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

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

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

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

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

    leaf secret-share {
```

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

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

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

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

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

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

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

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

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

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

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

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

6. IANA Considerations

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

7. Manageability Considerations

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

8. Security Considerations

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

8.1. Proof of Transit

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

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

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

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

8.2. Cryptanalysis

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

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

8.3. Anti-Replay

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

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

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

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

8.4. Anti-Preplay

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

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

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

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

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

8.5. Anti-Tampering

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

8.6. Recycling

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

8.7. Redundant Nodes and Failover

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

8.8. Controller Operation

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

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

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

8.9. Verification Scope

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

8.9.1. Node Ordering

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

8.9.2. Stealth Nodes

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

9. Acknowledgements

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

10. References

10.1. Normative References

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

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

10.2. Informative References

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: November 8, 2018

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

T. Mizrahi
Marvell
May 7, 2018

Proof of Transit
draft-brockners-proof-of-transit-05

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on November 8, 2018.

Copyright Notice

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

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

Table of Contents

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

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

1. Introduction

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

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

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

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

2. Conventions

Abbreviations used in this document:

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

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

LPC: Lagrange Polynomial Constants

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

NSH: Network Service Header

POT: Proof of Transit

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

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

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

SFC: Service Function Chain

SR: Segment Routing

3. Proof of Transit

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

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

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

3.1. Basic Idea

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

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

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

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

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

3.2. Solution Approach

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

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

3.2.1. Setup

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

3.2.2. In Transit

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

3.2.3. Verification

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

3.3. Illustrative Example

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

3.3.1. Basic Version

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

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

3.3.1.1. Secret Shares

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

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

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

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

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

3.3.1.2. Lagrange Polynomials

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

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

$$\begin{aligned} 11(x) &= (((x-x_0) / (x_1-x_0)) * ((x-x_2)/(x_1-x_2))) \bmod 53 = \\ &(-5 + 7x/2 - (1/2)x^2) \bmod 53 \end{aligned}$$

$$\begin{aligned} 12(x) &= (((x-x_0) / (x_2-x_0)) * ((x-x_1)/(x_2-x_1))) \bmod 53 = \\ &(8/3 - 2 + (1/3)x^2) \bmod 53 \end{aligned}$$

3.3.1.3. LPC Computation

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

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

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

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

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

3.3.1.4. Reconstruction

Reconstruction of the polynomial is well-defined as

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

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

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

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

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

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

3.3.1.5. Verification

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

3.3.2. Enhanced Version

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

3.3.2.1. Random Polynomial

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

3.3.2.2. Reconstruction

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

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

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

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

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

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

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

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

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

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

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

The verification using CML is discussed in next section.

3.3.2.3. Verification

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

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

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

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

3.3.3. Final Version

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

3.4. Operational Aspects

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

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

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

3.5. Alternative Approach

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

3.5.1. Basic Idea

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

3.5.2. Pros

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

3.5.3. Cons

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

4. Sizing the Data for Proof of Transit

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

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

2. CML: Cumulative

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

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

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

5. Node Configuration

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

5.1. Procedure

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

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

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

5.2. YANG Model

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

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

    yang-version 1;

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

    prefix ietf-pot-profile;

    organization "IETF xxx Working Group";
```

```
contact "";

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

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

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

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

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

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

    leaf secret-share {
```

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

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

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

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

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

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

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

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

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

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

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

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

6. IANA Considerations

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

7. Manageability Considerations

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

8. Security Considerations

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

8.1. Proof of Transit

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

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

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

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

8.2. Cryptanalysis

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

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

8.3. Anti-Replay

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

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

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

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

8.4. Anti-Preplay

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

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

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

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

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

8.5. Anti-Tampering

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

8.6. Recycling

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

8.7. Redundant Nodes and Failover

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

8.8. Controller Operation

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

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

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

8.9. Verification Scope

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

8.9.1. Node Ordering

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

8.9.2. Stealth Nodes

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

9. Acknowledgements

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

10. References

10.1. Normative References

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

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

10.2. Informative References

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

sfc
Internet-Draft
Intended status: Standards Track
Expires: September 4, 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
March 3, 2018

NSH Encapsulation for In-situ OAM Data
draft-brockners-sfc-ioam-nsh-01

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 4, 2018.

Copyright Notice

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

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

Table of Contents

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

1. Introduction

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

2. Conventions

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

Abbreviations used in this document:

IOAM: In-situ Operations, Administration, and Maintenance

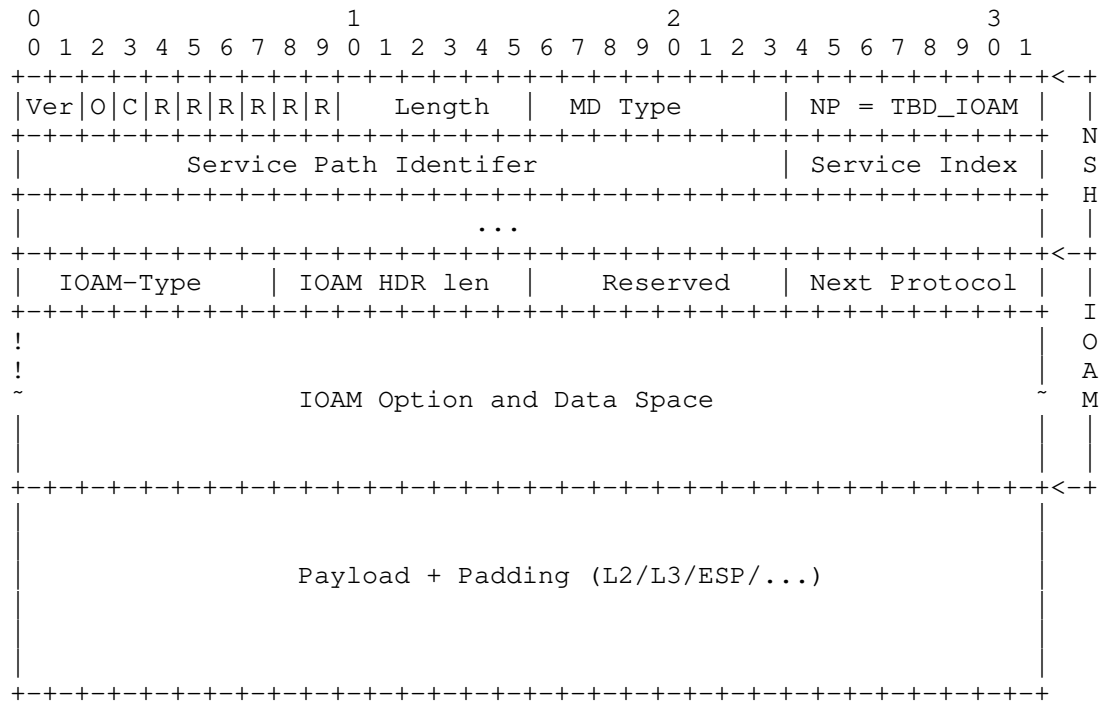
NSH: Network Service Header

OAM: Operations, Administration, and Maintenance

TLV: Type, Length, Value

3. IOAM data fields encapsulation in NSH

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



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 ($2^6 \times 4$) octets.
2. Encapsulation of IOAM data fields using the "Next Protocol" field. Each IOAM data field option (trace, proof-of-transit, and edge-to-edge) would be specified by its own "next protocol".
3. Encapsulation of IOAM data fields using the "Next Protocol" field. A single NSH protocol type code point would be allocated for IOAM. A "sub-type" field would then specify what IOAM options type (trace, proof-of-transit, edge-to-edge) is carried.

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

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

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

5. IANA Considerations

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

Next Protocol	Description	Reference
x	TBD_IOAM	This document

6. Security Considerations

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

7. Acknowledgements

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

8. References

8.1. Normative References

- [I-D.ietf-ippm-ioam-data] Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., and d. daniel.bernier@bell.ca, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-01 (work in progress), October 2017.
- [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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 2, 2018

D. Purkayastha
A. Rahman
D. Trossen
InterDigital Communications, LLC
Z. Despotovic
R. Khalili
Huawei
March 1, 2018

Alternative Handling of Dynamic Chaining and Service Indirection
draft-purkayastha-sfc-service-indirection-02

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. Networks need to be flexible and dynamic in terms of allocation of services and resources. Network Operators should be able to reconfigure the composition of a service and steer users towards new service end points as user move or resource availability changes. SFC allows network operators to easily create and reconfigure service function chains dynamically in response to changing network requirements. We discuss a use case where Service Function Chain can adapt or self-organize as demanded by the network condition without requiring SPI re-classification. This can be achieved, for example, by decoupling the service consumer and service endpoint by a new service function proposed in this draft. We describe few requirements for this service function to enable dynamic switching between consumer and end point.

Status of This Memo

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

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

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

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

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Use Case Description	3
2.1. Data Center	3
2.2. Third party cloud service provider	4
2.3. ETSI MEC USE CASE	5
2.4. 3GPP	6
2.5. Use Case Analysis	6
3. NSH and Re-classification	8
3.1. Dynamic service chain creation using NSH	9
4. Challenges with dynamic indirection	10
5. HTTP as a transport	12
6. Service Request Routing (SRR) Service Function	14
6.1. Overview	14
6.2. Details of SRR Function	16
7. Protocol Consideration	21
8. Next Steps	21
9. IANA Considerations	21
10. Security Considerations	22
11. Informative References	22
Authors' Addresses	23

1. Introduction

The requirements on today's networks are very diverse, enabling multiple use cases such as IoT, Content Distribution, Gaming, Network functions such as Cloud RAN. Every use case imposes certain requirements on the network. These requirements vary from one extreme to other and often they are in a divergent direction. Network operator and service providers are pushing many functions towards the edge of the network in order to be closer to the users.

This reduces latency and backhaul traffic, as user request can be processed locally.

It becomes more challenging when network congestion, user mobility as well as non-deterministic availability of compute and storage resources are considered. The impact is felt most in the edge of the network because as the users move, their point of attachment changes frequently, which results in (at least partially) relocating the service as well as the service endpoint. Furthermore, network functions are pushed more and more towards the edge, where network, compute and storage resources are constrained and availability is non-deterministic. Constrained network resources may lead into congestion in the network. Also, storage resources may need to be moved where the user concentration is more in case of content delivery applications.

We describe few use cases in the next section and derive the requirement for composing new services and service path in a dynamic edge network. We address this dynamicity by introducing a special Service Function, called SRR (service request routing). We describe the problems associated with today's network and Layer 3 based approach to handle dynamicity in the network. We then discuss how such new Service Function with certain capabilities can handle the dynamicity better than these conventional methods.

2. Use Case Description

2.1. Data Center

The data center use case draft [I-D.ietf-sfc-dc-use-cases] describes an East West traffic use case. This is the predominant traffic in data centers today. Server virtualization has led to the new paradigm where virtual machines can migrate from one server to another across the data center. This explosion in east-west traffic is leading to newer data center network fabric architectures that provide consistent latencies from one point in the fabric to another.

SFCs applied in an enterprise or service provider data center can be broadly categorized into two types:

- o Access SFCs
- o Application SFCs

Access SFCs are focused on servicing traffic entering and leaving the data center while Application SFCs are focused on servicing traffic destined to applications. Service providers deploy a single "Access SFC" and multiple "Application SFCs" for each tenant. Enterprise

data center operators on the other hand may not have a need for Access SFCs depending on the size and requirements of the enterprise.

In carrier networks, operators may deploy multiple data centers dispersed geographically. Each data center may host different types of service functions. For example, latency sensitive or high usage service functions are deployed in regional data centers while other latency tolerant, low usage service functions are deployed in global or central data centers. In such deployments, SFCs may span multiple data centers and enable operators to deploy services in a flexible and inexpensive way.

It is clear that within the data center as well as in inter data center scenarios, users are serviced by multiple SFs distributed inside as well as outside a location. In this scenario, it is clear that Service function chains should be able to reselect, redirect traffic very fast. The draft identifies that Static service chains do not allow for modifying the SFCs as they require the ability to add SNs or remove SNs to scale up and down the service capacity. Likewise the ability to dynamically pick one among the many SN instance is not available.

2.2. Third party cloud service provider

This use case is related to an emerging business model, where computational resources for edge cloud service are provided by alternative facility providers that are non-traditional network operators. This is due to the situation for many specific localized use cases, where network operators may not have necessary real estate available. They may even not be willing to spend on CAPEX and OPEX for said point-of-presence, because there is no clear path for sustainable cost recovery [UKNIC].

The industry is witnessing the emergence of real estate owners such as building asset or management companies, cell tower owners, railway companies or other facility owners willing to deploy edge cloud resources. The facility provider, e.g. cell tower owner or building management company, deploys edge computing resources throughout their installation in the country. They have their own operation and management software, which is capable of resource deployment, scale up or scale down resources, deploy edge applications from third party service providers. They are capable of offering service to more than one network operator at a specific location, thus acting as a "neutral host". The facility provider, which owns cloud resources and provides application services, is referred to as "Third party Edge Owner (TEO)".

There is more than one stakeholder in this ecosystem, E.g. Network Service Provider, Real estate owner, Cloud capability (compute and storage resource) provider, Application/service provider. An entity can assume more than one role. From network operators point of view there may be "Cloud provider" or "Cloud service provider" depending on the roles assumed by external entity.

"Cloud Providers" provide cloud resources (compute and storage) to network operators. Network operators rent those resources and manage MEC host by themselves. Network operator can set up application traffic rules, so that traffic can be processed, by that host.

"Cloud Service Providers" not only make resources available to network operators or service providers, but also provides management and hosting service. They can host edge applications on behalf of application service providers and sets up user plane traffic to be steered towards the edge application.

Cloud Service Providers, as well as many organizations that need to share and analyze a quickly growing amount of data, such as retailers, manufacturers, telcos, financial services firms, and many more, are turning to localized Micro Data Centers (MDC) installed on the factory floor, in the telco central office, the back of a retail outlet, etc. The solution applies to a broad base of applications that require low latency, high bandwidth, or both.

As Micro Data centers are deployed at the edge of the network, common deployment options are:

- o Micro Data Centers are deployed on L2 in the edge of the network
- o Instead of single internet Point Of Presence (POP) deployment, multiple internet POP deployment is desirable to localize data
- o Service is composed out of these multiple POP deployment of MDC, where data exchange and collaboration is expected among these MDCs
- o Due to mobility, changes in network condition (e.g. congestion, load), service composition may change frequently to support promised quality of experience

2.3. ETSI MEC USE CASE

Take the following video orchestration service example from ETSI MEC Requirements document [ETSI_MEC]. The proposed use case of edge video orchestration suggests a scenario where visual content can be produced and consumed at the same location close to consumers in a densely populated and clearly limited area. Such a case could be a

sports event or concert where a remarkable number of consumers are using their handheld devices to access user select tailored content. The overall video experience is combined from multiple sources, such as local recording devices, which may be fixed as well as mobile, and master video from central production server. The user is given an opportunity to select tailored views from a set of local video sources.

2.4. 3GPP

3GPP Rel. 15 introduces the notion of the service-based interface (SBI) as an alternative to the traditional call pattern invocation of network functions. This introduction targets the support for replication, e.g., driven by virtualized functions, as well as supporting alternative interactions, e.g., for different vertical market specific control planes, by making the discovery as well as composition of new interactions more flexible.

We believe that SFC is a suitable framework for the interconnection of such network functions through the new SBI. One of the aforementioned driving forces, namely the replication of functions aligns with our thinking in this draft in that indirections to new vertical instances need to be dynamic in reacting to the appearance of new virtual instances or to changes in policies for the selection of specific instances by specific calling entities.

2.5. Use Case Analysis

SFC allows network operators as well as service providers to compose new services by chaining individual service functions.

In a dynamic network environment, like the edge of a network, the capability to dynamically compose new services from available services as well as move a service instance is desirable. Dynamic composition and relocation of services may be attributed to:

- o Congestion in the network: Due to constrained network resources, increase in the network load may create congestion in the network, resulting in a congested Service Function Path. Service functions may detect congestion and reconfigure the Service Function Path to avoid it.
- o In response to latency: in a dynamic network environment and with the need for ultra-low latency communication, instantiation of new service function endpoints might be the only remedy to combat the increase of latency caused, e.g., by increased load on a previous endpoint or mobility of the user and therefore increasing the 'distance' to the service function endpoint. Keeping the service

function endpoint 'close' to the user allows for reducing latency, segregating communication in localized islands of service interaction.

- o In response to user mobility: In a dynamic network environment where service functions move frequently because of user movement, load balancing or resource modification, service function chains and the service end points need to be created and recreated frequently
- o Resource availability.: Availability of compute and storage resources varies with network load, number and type of applications running etc. In the edge of the network, due to sudden increase of users, compute load may increase. In this situation applications, running on the compute resources may be moved to another location where more resources are available.

In SFC, there is a notion of logical chaining of SFs and chaining of actual physical locations, known as Rendered Service Path (RSP). RSP provides a static binding of SFs to their physical location. In order to create a chain in dynamic fashion, late binding of SFs and physical location may be desired. SFC is capable of modifying the service chain to certain extent in response to network conditions, but not a complete solution has been described

In order to route the service requests to service end points in a dynamic manner, we identify the following desirable features in a service function chain:

- o Capability to trigger service chain reconfiguration based on network information such as congestion indication, mobility, degradation of user experience etc. Service Functions should be able to process such network information, identify which section of the chain needs to be reconfigured and take action
- o Fast switching from one service instance to another by not relying on the DNS for service location resolution. Instead of DNS, the function should be able to identify the path, which will allow to reach the service end point.
- o Direct path mobility, where the path between the requester and the responding service can be determined as being optimal (e.g., shortest path or direct path to a selected instance), is needed to avoid the use of anchor points and further reduce service-level latency

- o Indirect service requests at the network level, transparent to the requesting client and without the involvement of the DNS. End user is not aware of the decision made by the SF.
- o New methods for forwarding, such as path-based forwarding, direct path routing in mobility cases, path pinning for traffic steering and simplified service-specific peering towards the Internet.

3. NSH and Re-classification

[RFC7498] captures the problems associated with existing service deployments that are problematic. The problems are described below at a high level:

- o Network topology: Network service deployment is tightly coupled with network topology thus reducing the flexibility in service delivery. It adds complexity in deploying network service when certain traffic types may need some service and other traffic types do not need the same service.
- o Configuration complexity is the direct result of dependency on network topology.
- o Limited availability of services
- o Altering the order of a deployed chain is complex and cumbersome
- o Coupling of service functions to topology may require service functions to support many transport encapsulations or for a transport gateway function to be present.
- o In a dynamic environment like the Edge of a network service delivery, routing changes fast. It may be difficult to deliver service dynamically due to the risk and complexity of VLANs and/or routing modifications.

These factors provide motivation for a simplified and flexible service insertion model that addresses many of the current shortcomings and provides new, much needed functionality to enable service deployments in modern network environments. Service chaining accomplishes this by considering service functions as resources, with associated attributes, available for scheduled consumption. Selective traffic, subject to policy, may then be "steered" to the requisite service resources, along with any "extra" information referred to as metadata. This metadata is used for policy enforcement.

A basic form of service chaining may be realized using existing transport encapsulations. This method of chaining relies upon the tunneling of selected data between service functions. Although this form of service chaining achieves some level of abstraction from the underlying topology, it does not truly create a service plane. NSH [RFC8300] is a distinct identifiable plane that can be used across all transports to create a service chain and exchange metadata along the chain.

Fundamentally, however, the notion of "services" in SFC is tied into specific service function endpoints, which lie along a well-defined service function path (SFP) where the path is defined through lower layer transport encapsulations. If any such service function endpoint changes, the service chain needs to be adjusted; a procedure we outline in the following sub-section.

3.1. Dynamic service chain creation using NSH

We revisit the dynamic service chain creation capability of NSH. NSH defines a new service plane protocol [RFC8300]. A Network Service Header (NSH) contains service path information and optionally metadata that are added to a packet or frame and used to create a service plane. A control plane is required in order to exchange NSH values with participating nodes, and to provision the same nodes with requisite information such as service path ID to overlay mapping.

The Network Service Header has three parts, Base header, Service Path Header and Context Header. NSH Service Path Header is a 4-byte service path header follows the base header and defines two fields used to construct a service path:

- o Service path identifier (SPI)
- o Service index (SI)

The following figure depicts the service path header.

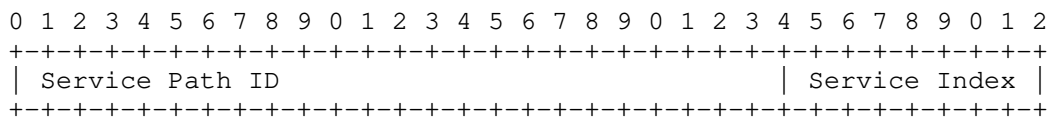


Figure 1: NSH Path Header

The service path identifier (SPI) is used to identify the service path that interconnects the needed service functions. It allows nodes to utilize the identifier to select the appropriate network

transport protocol and forwarding techniques. The service index (SI) identifies the location of a packet within a service path. As packets traverse a service path, the SI is decremented post-service.

SPI represents the service path and altering the path identifier results in a change of a service path. A change in SPI value is a result of re-classification. It means a node in the service path determined, based on policy, that the initial classification was incorrect or incomplete. If the updated classification results in the necessity of a new service path, the node updates the SPI and SI fields accordingly. The new identifier is then used to select the appropriate overlay topology. This allows service functions to alter the path of a packet without having to participate in the network topology and its associated control plane(s). The method to determine that an existing classification is incorrect and how to determine the new classification is not defined.

4. Challenges with dynamic indirection

The emerging trend in today's network is to deploy network functions, services and applications at the edge of the network to support latency requirements, computational offload, traffic optimization etc. As users are moving, application or services being used by users, may need to be moved closer to the user's new location. This implies another instance of the service function may need to be instantiated close to the user's new location. It may result in re-establishing service path from the newly instantiated service function to other service instances. It is also possible that the newly instantiated service function may be redirected to a new service end point (e.g. Application Server) for various reasons, such as incomplete content, proximity to data store, load balancing etc. In another scenario, a single instance of the service function may not handle all users due to latency or load constraints. A single service function may be instantiated more than once to balance user load. As the number of instances increase and along with mobility, the complexity of service routing increases. It is anticipated that there may be a constant action of function chaining, re-chaining occurring in the network.

The challenge of dynamic indirection may be better described by analyzing the working of CDNs, which dynamically (re-)direct user-initiated requests towards the most appropriate content instance. This task becomes more difficult if granularity of the instance placement increases. For instance, in case of a CDN being realized close to end users, specifically in edge of the network, the specific content instance might need to be selected dynamically. After initial selection, the instance may change during service execution.

In a conventional network, an instance of a service is found and selected using DNS. The subsequent service request is then routed through the network between the client and the service. If the user is doing a DNS lookup to access content served by a CDN then the DNS service will maintain a list of IP addresses that can be returned for a given domain name and will try to return an IP address of a node geographically close to the client. Should the service provider want to replace an instance of their service with another one at a different IP address (and potentially a different physical location for various reasons such as load balancing, reliability etc.) then the DNS tables must be updated, i.e., the service needs to be (re-)registered quickly. This is done by updating the local authoritative DNS server which then propagates the new mapping to DNS services across the world. DNS propagation can take up to 48 hours so fast and dynamic switching from one service instance to another is not possible in conventional networks; even in more localized scenarios, the propagation of DNS updates might still be insufficient. When relying on many surrogate service endpoints to exist in the edge network, there is a clear issue of certain resources not being available in one surrogate instance while existing in another so that changes in redirection might be desirable, while also changes in local load drive the need for such change in redirection. With the emergence of container-based virtualization platforms, service function endpoints can be established in a matter of seconds and we therefore believe that the 'reachability' of such said service instance, i.e., the possibility of route service requests to it from a client that was previously served elsewhere, must follow a similar timeline, i.e., a few seconds or even less.

The other issue in conventional network lies with mobility management procedure. These procedures use an anchor point, which terminates a session at the network edge. As user moves around, traffic is redirected from the anchor point to the new point of attachment. Relying on typical mobility management approaches found in IP networks, usually leads to inefficient 'triangular' routing of requests through this common 'anchor' point. This triangular routing increases the latency in reaching the new service function or service end points as users move.

Traffic steering is a common procedure in managed networks, particularly at the edge, due to desired subscriber-centric traffic policies (e.g., related to pricing structures), resource requirements (e.g., related to using particular paths in the network) or mobility (e.g., users moving in a cellular network). Today's methods for traffic steering include anchor-based mobility management as well as traffic classification, for instance, in packet gateways of cellular systems (using, e.g., deep packet inspection as well as port and

address classification). While the former leads to inefficient 'triangular' traffic forwarding, the latter often requires additional state in the forwarders to differentiate traffic from one user to another.

The analysis of CDN network shows that dynamic indirection is a necessary requirement, which needs to be supported by the networks. The goal for this indirection is to provide user applications lowest possible latency. But as discussed above, relying on today's technique does not help in guaranteeing same latency to user applications. On the other hand, there is a high possibility that latency may increase if we rely on Layer 3 based service redirection techniques.

SFC handles indirection through the use of SPI. A packet needs to be reclassified and the intermediate node changes the SPI. Following are the typical steps that happens in order to implement the indirection.

- o A packet arrives at a particular node
- o The node contacts the policy manager
- o Identifies the current classification is incorrect
- o Reclassifies the packet, i.e. change the SPI
- o Inserts the packet in the pipe, possibly towards the SFF

The indirection mechanism in SFC involves certain steps to process policy information and change the SPI in the packet header, making it suitable to handle dynamic indirection requirements. Our proposed SF in this document provides an additional method to handle dynamic indirection of service requests, not relying on the reclassification mechanism. Combining these two techniques may provide flexibility and improvement over single method.

5. HTTP as a transport

With the extensive use of "web technology", "distributed services" and availability of heterogeneous network, HTTP has effectively transitioned into the common transport for name-based E2E communication across the web. In the context of SFC and SF, HTTP requests and response are considered as the "Service Request (SR)". This use case describes how these SRs are directed towards correct SF in a fast and dynamic way. The routing and indirection of SRs are abstracted at HTTP level, instead of the traditional approach where routing decision for a service request is made at Layer 3.

If we abstract HTTP as a transport, HTTP requests, such as GET, PUT and POST can be routed based on the URI associated with the request, with the URI being simply the name of a resource or the invocation point for a service transaction. Based on the name of the resource requested, the appropriate HTTP request can be routed to the suitable service endpoint. If Service Functions (SF) could be identified using URI or name, HTTP requests to an SF would be routed or directed using name based routing. With that, the redirection to the most suitable service instance is purely done based on named services with HTTP being a specific (application layer) transport service.

The ongoing EU H2020 efforts like FLAME [H2020FLAME] are driven by city-scale many-POP deployments of compute infrastructure, all SDN-connected and OpenStack managed. Localized media use cases drive the need for name-based (HTTP as the main transport protocol here) service instances being chained with the relationship between specific virtual instances being controlled at the underlying routing/switching level.

The notion of 'HTTP as-a transport', utilizing URLs as addressing scheme, can be used to create SFP as shown in Fig 2., i.e., 192.168.x.x -> www.example.com -> 192.168.x.x -> www.example2.com -> 192.168.x.x -> ... -> www.exampleN.com. It is this 'name-based' relationship that we see possibly realized through specific replicated instances, where in turn the routing towards those specific instances is realized by the SRR.

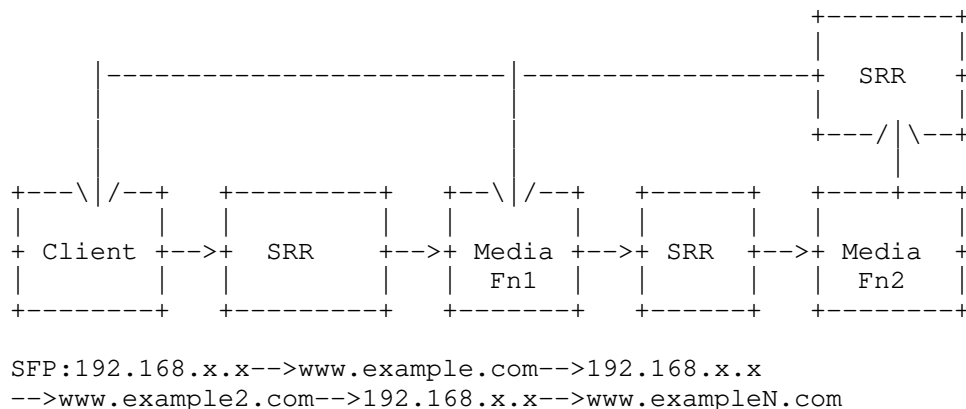


Figure 2: SFP with new HTTP-based Transport option

In a pure SFC architectural framework, Classifier function may interact with SRR to obtain an SE (Service Encapsulation). E.g. the

Classifier function may look into the network locator map in Fig 2 and determine the next SF is www.example.com. It provides this information to SRR to obtain the next hop information. SRR returns the SE for next hop, which can be a "bitfield" information that is being used in the overlay routing for this part of the SFP. The Classifier function uses this SE to route the incoming packet directly at the transport network level.

6. Service Request Routing (SRR) Service Function

6.1. Overview

The following diagram shows the application of the new proposed SRR service function in an example of media clients connecting to media servers. There may be more than one media functions to support CDN like architecture, Surrogate servers to handle mobility and load balancing.

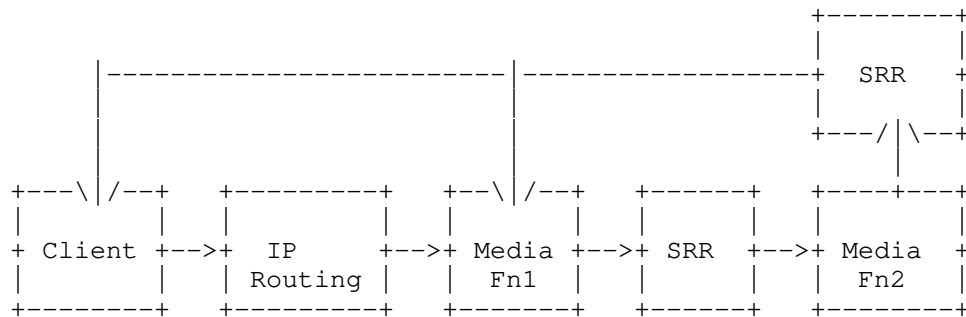


Figure 3: General SFC with SRR Flexible Chaining, initiated via IP Routed Client Connection

The clients are connected to media functions through frontend routed network, e.g., relying on standard IP routing, while media functions are chained via the new proposed service request routing (SRR) function. Alternatively, we also envision to utilize the SRR function directly between client SF and media function SF, as outlined in the figure below

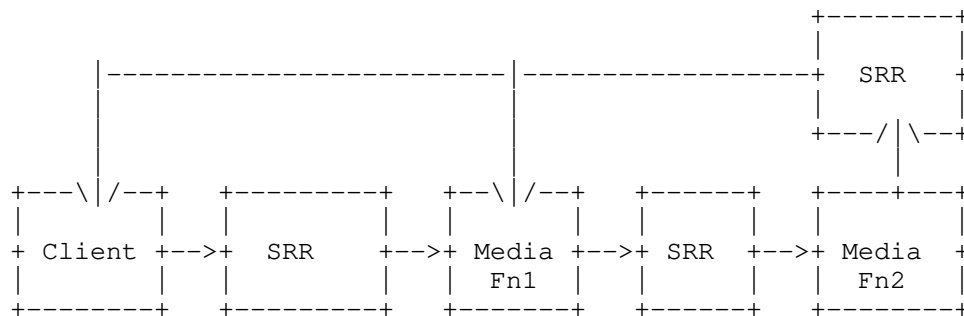


Figure 4: General SFC with SRR Flexible Chaining, initiated via SRR Chained Client

For our considerations, we assume that each SF is realized by at least one or more service function endpoints (SFEs). Hence, instead of looking at "chaining" as a concept that connects specific SFEs along a well-defined SFP, we propose to look at "chaining" at the level of "named" service functions rather than their specific endpoint instances. With this in mind, the SRR service function lifts the relationship between the connecting SFs to the level of "logical" service functions rather than their specific realizing endpoints. Instead of relying on dynamic re-chaining in case of any dynamically changing relationship between specific SFEs, the SRR provides the selection of suitable SFEs while maintaining the logical relationship between the SFs. In Section 6.3, we will present the necessary extensions to the SFP concept to support this higher abstraction of "chaining" via "named" logical SFs. The SRR introduces the flexibility in routing service requests from client to specific SFEs. In the edge network, where users are moving and service end points may also change, having flexibility to decide and steer service requests directly helps in guaranteeing the same latency to user applications. Clearly, that is achieved by reducing the switching time from SF to another. As service end point changes, the routing functions makes instantaneous decision to route the request to the appropriate media server.

The SRR introduces the flexibility in routing service requests from client to specific SFEs in response to conditions such as congestion in the network, user mobility etc. In the edge network, where users are moving and service end points may also change, having flexibility to decide and steer service requests directly helps in guaranteeing the same latency to user applications. The edge of the network maybe congested due to limited network resources. The SRR may be able to determine network congestion and quickly route service requests to other Service End point, which is not experiencing congestion. In

addition, application-layer control functions might utilize latency measurements to ensure that suitable service instances are being created during runtime of the scenario such as to ensure that service function endpoints are available 'nearby' (possibly) moving so as to keep a desired latency under a desired value.

Clearly, that is achieved by reducing the switching time from one SF endpoint to another. As the service end point changes, the routing functions makes instantaneous decision to route the request to the appropriate media server.

The possible improvements of using SRR within an SFC framework are listed below:

- o Fast (between 10 and 20ms) switching times from one service instance to another by not relying on the DNS for service discovery and directly routing service requests at the level of the transport network.
- o The capability to indirect service requests at the network level will help in reducing latency, when service end points change. E.g. when a service request is being sent to one surrogate instance but results in a HTTP 404 or 5xx error response, the original request is redirected to another alternative surrogate with minimal latency, i.e., right at the destination of said failed service request. Nesting these operations effectively leads to a net-level 'search' among all available surrogate instances until the search is exhausted (with a negative result) or the resource is found.
- o New methods for forwarding, such as path-based forwarding, will enable direct path routing in mobility cases, path pinning for traffic steering and simplified service-specific peering towards the Internet. Such capability would allow for localizing traffic, reduce latency and costs.

6.2. Details of SRR Function

Assuming such introduction of an HTTP-level transport notion, the SRR function can be decomposed further as shown in Fig 5.

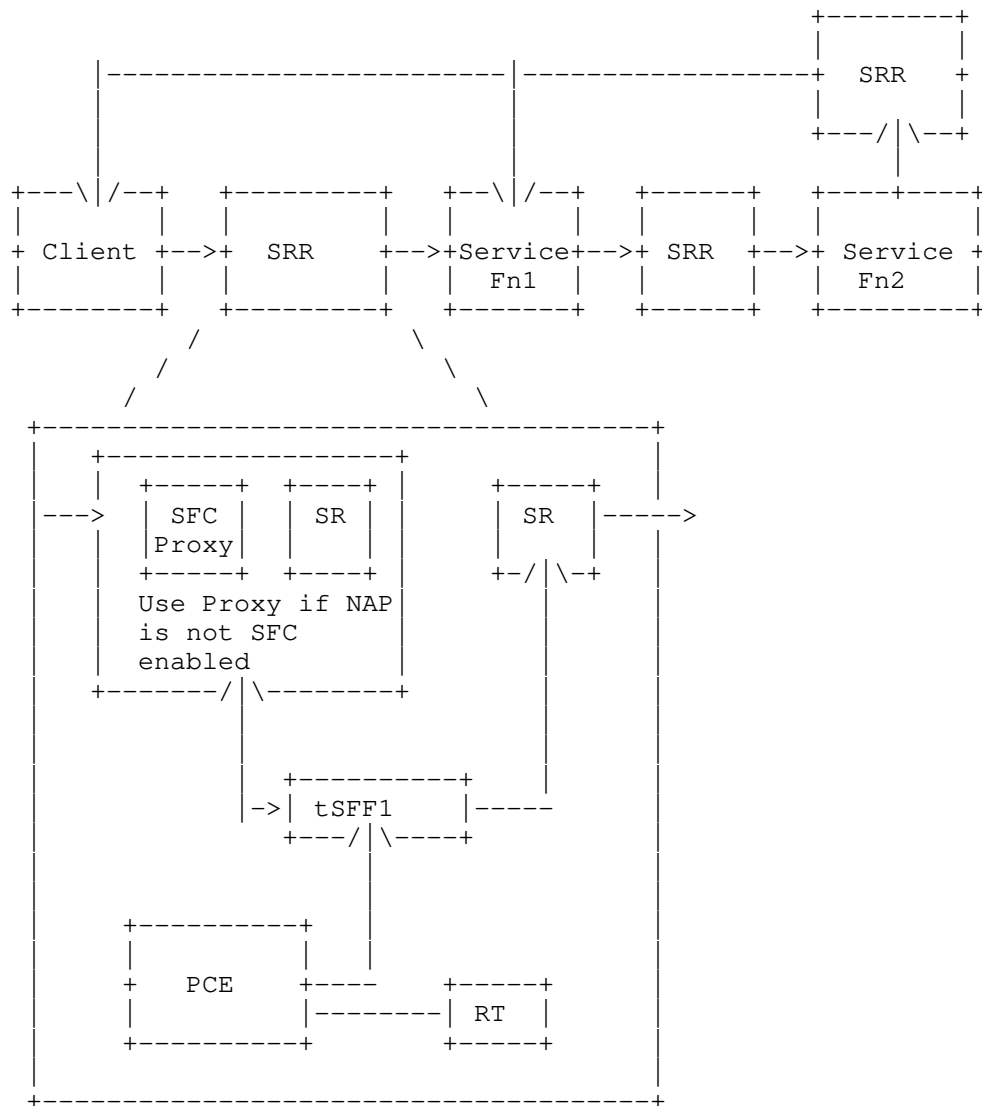


Figure 5: SRR decomposition

Another option for the two functions routing via the SRR could be entirely link-local, i.e., there's another simple tSFF2 between client and SRR as well as SF1 and SRR that is simply a link-local transport. The following figure describes this alternate option.

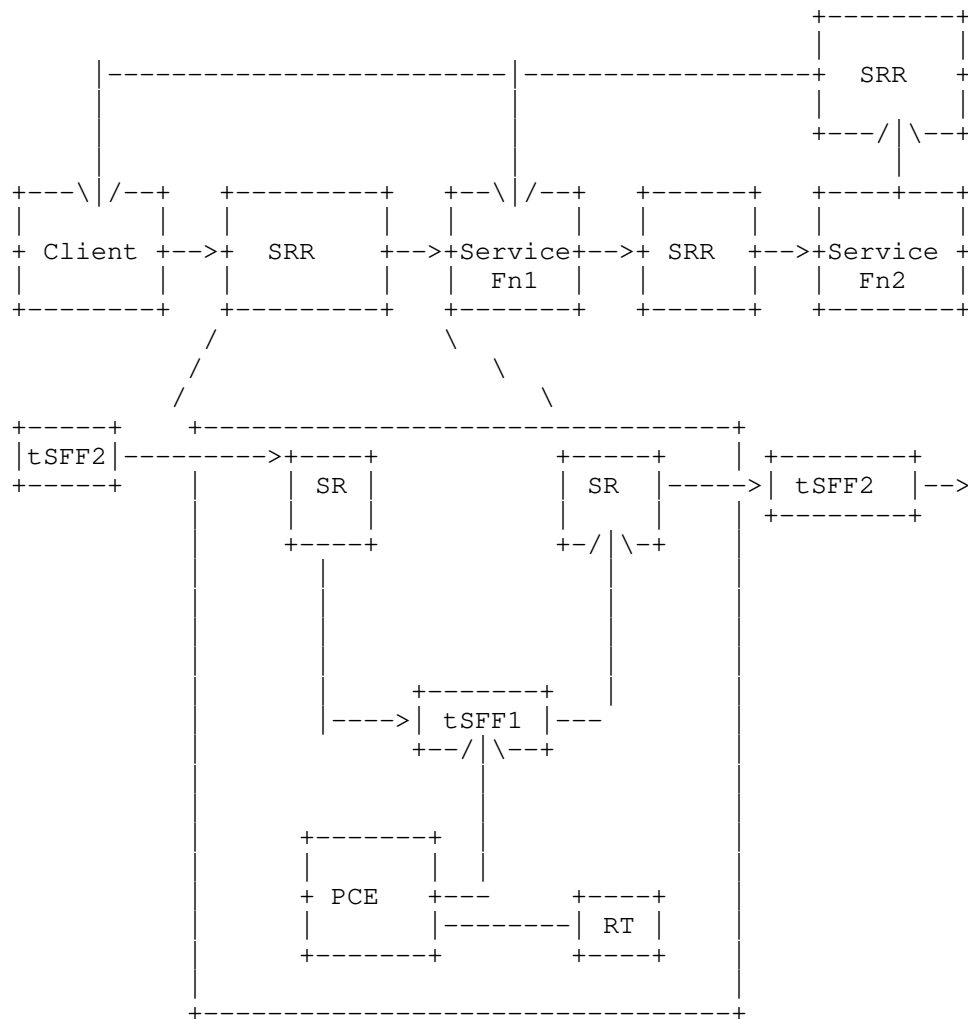


Figure 6: SRR decomposition using link-local client/function communication

The SRR function may be composed of the following functions:

- o Service Router(SR) at the ingress, terminates on the client side Layer 3 and above protocols, such as TCP

- o Service Router(SR) at the egress, terminates any transport protocol on the outgoing (server) side
- o PCE, Path Computation Element function is responsible for selecting the correct next SF, also possibly realizing path policy enforcement. The result of the selection is a path identifier which is delivered to the ingress SR upon initial path computation request (i.e., when sending a request to a specific URL on the SFP for the first time). The path identifier is utilized for any future request for a given URL-based SF. In case of another SF instance becoming available, indicated to the PCE through a registration procedure, the PCE will instruct all ingress SRs to invalidate path identifiers to the specific URL of the SF, resulting in an initial path computation request at the next SF request forwarding. Through this, the newly registered SF instance might be utilized if the policy-governed path computation will select said SF instance.
- o Reclassification Trigger Handler (RT) : Network measurement information, such as latency, packet loss or network congestion, etc. could be processed by the handler. This may trigger reconfiguration of the specific service function endpoint chain over which the SFC is being executed. The handler forwards the information about the chain reconfiguration to PCE.
- o Transport-derived SFF (tSFF1): the communication between ingress/egress SRs as well as SRs to PCE is realized via a transport-derived SFF. We outline here three possible tSFFs
 - * SDN-based: This option utilizes path-based forwarding through SDN-based wildcard matching fields, supported with OF1.2+[Reed2016]. It can be embedded into slicing approach of underlying transport infrastructure by leaving typical slicing fields available (e.g., VLAN tags). The forwarding utilizes the Ethernet frame format at Layer 2, representing the topological links of a specific forwarding path in the transport network as unique bits in a fixed size bit array. For the latter, the approach utilizes the IPv6 source and destination fields for storing the bit array information (in a simple version for this forwarding, this limits the topology to 256 links but extensions schemes are possible, which are left out of this document at this stage). As mentioned, the SDN forwarding decision action is a simple wildcard matching, supported with OF1.2+, with the wildcard representing the unique bit of a switch-specific output port. With that, the switch needs to consider as many forwarding rules as switch local output ports - see [Reed2016] for more information. Fig. xx illustrate this forwarding solution, including the ability

to create ad-hoc multicast relations by simply ORing individual bitarrays representing unicast paths.

- * Another approach is outlined in [I-D.ietf-bier-use-cases] where the SFF is suggested to be realized via a BIER overlay, in turn realized over a BIER-compliant underlay, such as MPLS. BIER utilizes a similar bit array approach for representing a forwarding path in the overlay network but unlike [Reed2016], the bit fields indicate the egress BIER-compliant router that the packet is supposed to reach.
- * As yet another alternative, the tSFF may utilize a flow aggregation approach, outlined in [Khalili2016], called edge switch classification (ESC). In this approach, a path from an ingress to egress SR is described as a so-called edge classification vector (ECV), which combines information on the aggregated flow (following [Khalili2016]) and the switch-local endpoint. The representation has similar bitarray characteristics as the previous two approaches
- o NOTE: with the ingress and egress SRs terminating SF Layer 3 connections and the utilization of bitarray-based tSFFs, the transmission of packets can effectively take place as an ad-hoc Layer multicast while the SFC itself is denoted as an n-times unicast SFC. As an example, consider the chaining of a set of n clients to a single video server. Each sub-SFC from an individual client to the video server will semantically result in a unicast response from the server back to the client (e.g., carrying the video chunk for a MPEG DASH-based video stream). When combining the sub-SFCs to the single SFC with n times unicast relations to the server, the SRR will deliver the responses from the server via one or more multicast responses to one or more clients. The size of the individual multicast groups will depend on the synchronicity of the client requests (and therefore on the synchronicity of the server responses). Note that the multicast relations here are ad-hoc created by ORing the bitarrays representing the specific clients to which the responses are meant to be sent. This is illustrated in the figure below. The HTTP multicast use case is being presented in the BIER use case draft [I-D.ietf-bier-use-cases] albeit without specific a SFC relation.

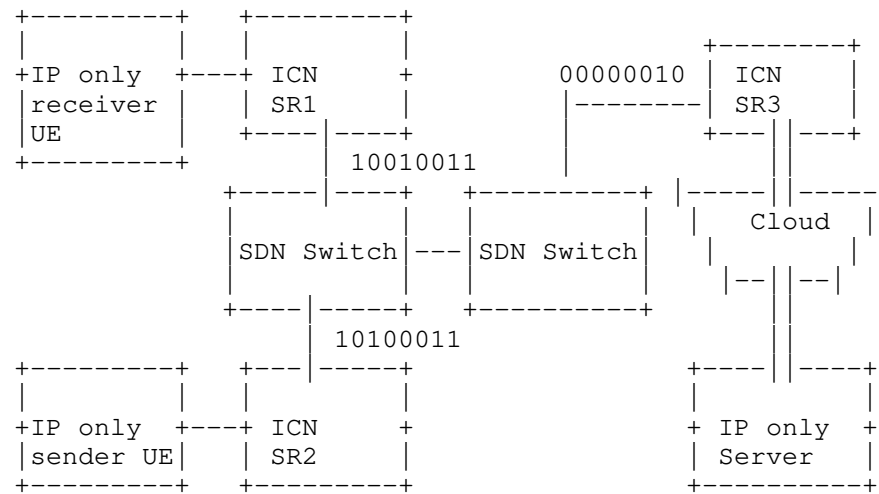


Figure 7: Illustration of Bitfield-based Forwarding using SDN

7. Protocol Consideration

For the operations outlined in the previous section, we foresee the following protocol changes are required:

- o SR-to-SR protocol for HTTP: HTTP based message exchange between client and server SRs
- o SR-PCE protocol: Used for path computation, obtaining routing information as well as provide path updates
- o Registration protocol: Used to register FQDN service endpoints

8. Next Steps

Feedback from the SFC WG on the validity of this solution and its scope within the SFC WG. If such alternative to the re-classification for service indirection is seen beneficial as well as fitting with the charter of the WG, the next steps would be to update the draft to outline potential protocol solutions required for the realization of such SRR SF.

9. IANA Considerations

This document requests no IANA actions.

10. Security Considerations

TBD.

11. Informative References

[ETSI_MEC]

ETSI, "Mobile Edge Computing (MEC), Technical Requirements", GS MEC 002 1.1.1, March 2016, <http://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf>.

[H2020FLAME]

EU, "EU H2020 FLAME PROJECT", , March 2016, <<https://www.ict-flame.eu/>>.

[I-D.ietf-bier-use-cases]

Kumar, N., Asati, R., Chen, M., Xu, X., Dolganow, A., Przygienda, T., Gulko, A., Robinson, D., Arya, V., and C. Bestler, "BIER Use Cases", draft-ietf-bier-use-cases-06 (work in progress), January 2018.

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

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

[Khalili2016]

Khalili, R., Poe, W., Despotovic, Z., and A. Hecker, "Reducing State of SDN Switches in Mobile Core Networks by Flow Rule Aggregation", ICCCN, August, 2016.

[Reed2016]

Reed, M., Al-Naday, M., Thomas, N., Trossen, D., and S. Spirou, "Reducing State of SDN Switches in Mobile Core Networks by Flow Rule Aggregation", ICC 2016, 2016.

[RFC7498]

Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <<https://www.rfc-editor.org/info/rfc7498>>.

[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>>.

[UKNIC] UK NIC, "5G Infrastructure Requirements in the UK", Final Report 3.0, December 2016,
<https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/577940/5G_Infrastructure_requirements_for_the_UK_-_LS_Telcom_report_for_the_NIC.pdf>.

Authors' Addresses

Debashish Purkayastha
InterDigital Communications, LLC
Conshohocken
USA

Email: Debashish.Purkayastha@InterDigital.com

Akbar Rahman
InterDigital Communications, LLC
Montreal
Canada

Email: Akbar.Rahman@InterDigital.com

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/>

Zoran Despotovic
Huawei

Email: Zoran.Despotovic@huawei.com

URI: <http://www.huawei.com/>

Ramin Khalili
Huawei

Email: Ramin.khalili@huawei.com

URI: <http://www.huawei.com/>