Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2019.
1.  Introduction

Service Function Chain (SFC) is a chain with a series of ordered Service Functions (SFs). Service Function Path (SFP) is a path of a SFC. SFC is described in detail in the SFC architecture document [RFC7665]. The SFs in the SFC are ordered and only when one SF processes traffic then it can be processed by the next SF. Changes in the order may cause errors. Sometimes, an SF uses the metadata...
from its upstream SF process. That’s why it’s very important for the operator to make sure that the order of traversing the SFs is exactly as defined by the control plane or the orchestrator. This document refers to the correspondence between the state of the control plane and the SFP itself as the SFP consistency.

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

2. Conventions used in this document

2.1. Terminology

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

SFF: Service Function Forwarder

SF: Service Function

OAM: Operation, Administration and Maintenance

SFP: Service Function Path

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

2.2. Requirements Language

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

3. Consistency OAM: Theory of Operation

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

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

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

3.1. COAM packet

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

- TBA1 - COAM Request
- TBA2 - COAM Reply

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

The COAM packet is displayed in Figure 1.

```
+---------------+---------------+---------------+---------------+
| Message Type  |   Reply mode  |  Return Code  | Return S.code |
+---------------+---------------+---------------+---------------+
|               |               |               |               |
+---------------+---------------+---------------+---------------+
| Sender’s Handle |
+---------------+---------------+---------------+---------------+
| Sequence Number |
+---------------+---------------+---------------+---------------+
| Type       | Length        |
+---------------+---------------+---------------+---------------+
| - Value      |
+---------------+---------------+---------------+---------------+
```

Figure 1: COAM Packet Header

3.2. SFF Information Record TLV

For COAM Request, the SFF MUST include the Information of SFs into the SF Information Record TLV in the COAM Reply message. Every SFF send back one COAM Reply Message with all the SFs that are attaching to the SFF along the SFP indicated by the COAM Request.
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     SFF Record TLV Type           |          Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Service Path Identifier(SPI)            |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                   SF Information  Sub-TLV                     |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Figure 2: SFF Information Record TLV

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

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

3.3. SF Information Sub-TLV

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

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

0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     SF sub-TLV Type           |          Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Service Index  |          SF Type              |   SF ID Type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          SF Identifiers                       |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Figure 3: Service Function information sub-TLV

SF sub-TLV Type: Two octets long field. It indicates that the TLV is a SF TLV which contains the information of one SF.
Length: Two octets long field. The value of the field is the length of the data following the Length field counted in octets.

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

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

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

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

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

3.4. SF Information Sub-TLV Construction

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

3.4.1. Multiple SFs as hops of SFP

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

An example of the COAM procedure for this case is shown in Figure 4. The Service Function Path(SFI=x) is SFI1->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 SFF. The SF information Sub-TLV in Figure 3 contains information for each SF(SF1, SF2, SF3 and SF4).
### 3.4.2. Multiple SFs for load balance

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

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

![Figure 4](image4.png)

**Figure 4: Example 1 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA1</td>
<td>SFP Consistency Echo Request</td>
<td>This document</td>
</tr>
<tr>
<td>TBA2</td>
<td>SFP Consistency Echo Reply</td>
<td>This document</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA3</td>
<td>SFF Information Record Type</td>
<td>This document</td>
</tr>
</tbody>
</table>

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:
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:

```
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>This document</td>
</tr>
<tr>
<td>TBA6</td>
<td>IPv4</td>
<td>This document</td>
</tr>
<tr>
<td>TBA7</td>
<td>IPv6</td>
<td>This document</td>
</tr>
<tr>
<td>TBA8</td>
<td>MAC</td>
<td>This document</td>
</tr>
<tr>
<td>TBA8+1-191</td>
<td>Unassigned</td>
<td>IETF Review</td>
</tr>
<tr>
<td>192-251</td>
<td>Unassigned</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>252-254</td>
<td>Unassigned</td>
<td>Private Use</td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td>This document</td>
</tr>
</tbody>
</table>
```

Table 4: SF Identifier Type

6. Acknowledgements

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

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

Thanks to Dirk von Hugo for his useful comments.

7. References

7.1. NormativeReferences
[I-D.ietf-bess-nsh-bgp-control-plane]

[I-D.ietf-sfc-multi-layer-oam]

[I-D.ietf-sfc-nsh-tlv]


7.2. Informational References


Authors' Addresses
Abstract

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

Service Function Chain (SFC) is a chain with a series of ordered Service Functions (SFs). Service Function Path (SFP) is a path of a SFC. SFC is described in detail in the SFC architecture document [RFC7665]. The SFs in the SFC are ordered and only when one SF processes traffic then it can be processed by the next SF. Changes in the order may cause errors. Sometimes, an SF uses the metadata
from its upstream SF process. That’s why it’s very important for the operator to make sure that the order of traversing the SFs is exactly as defined by the control plane or the orchestrator. This document refers to the correspondence between the state of the control plane and the SFP itself as the SFP consistency.

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

2. Conventions used in this document

2.1. Terminology

SFC(Service Function Chain): An ordered set of some abstract SFs.
SFF: Service Function Forwarder
SF: Service Function
OAM: Operation, Administration and Maintenance
SFP: Service Function Path
COAM(Consistency OAM): OAM that can be used to check the consistency of the Service Function Path.

2.2. Requirements Language

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

3. Consistency OAM: Theory of Operation

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

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

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

3.1. COAM packet

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

- TBA1 - COAM Request
- TBA2 - COAM Reply

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

The COAM packet is displayed in Figure 1.

```plaintext
+-----------------+-----------------+-----------------+-----------------+
<table>
<thead>
<tr>
<th>Message Type</th>
<th>Reply mode</th>
<th>Return Code</th>
<th>Return S.code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender's Handle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Type</td>
<td>Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 1: COAM Packet Header

3.2. SFF Information Record TLV

For COAM Request, the SFF MUST include the Information of SFs into the SF Information Record TLV in the COAM Reply message. Every SFF send back one COAM Reply Message with all the SFs that are attaching to the SFF along the SFP indicated by the COAM Request.
Service Path Identifier (SPI): The identifier of SFP to which all the SFs in this TLV belong.

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

### 3.3. SF Information Sub-TLV

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

After the COAM traversed the SFP, all the information of the SFs on the SFP are collected from the TLVs with COAM Reply.
Length: Two octets long field. The value of the field is the length of the data following the Length field counted in octets.

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

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

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

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

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

3.4. SF Information Sub-TLV Construction

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

3.4.1. Multiple SFs as hops of SFP

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

An example of the COAM procedure for this case is shown in Figure 4. The Service Function Path(SFI=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 SFF. The SF information Sub-TLV in Figure 3 contains information for each SF(SF1, SF2, SF3 and SF4).
3.4.2. Multiple SFs for load balance

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

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

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA1</td>
<td>SFP Consistency Echo Request</td>
<td>This document</td>
</tr>
<tr>
<td>TBA2</td>
<td>SFP Consistency Echo Reply</td>
<td>This document</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA3</td>
<td>SFF Information Record Type</td>
<td>This document</td>
</tr>
</tbody>
</table>

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:
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>This document</td>
</tr>
<tr>
<td>TBA6</td>
<td>IPv4</td>
<td>This document</td>
</tr>
<tr>
<td>TBA7</td>
<td>IPv6</td>
<td>This document</td>
</tr>
<tr>
<td>TBA8</td>
<td>MAC</td>
<td>This document</td>
</tr>
<tr>
<td>TBA8+1-191</td>
<td>Unassigned</td>
<td>IETF Review</td>
</tr>
<tr>
<td>192-251</td>
<td>Unassigned</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>252-254</td>
<td>Unassigned</td>
<td>Private Use</td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 4: SF Identifier Type

6. Acknowledgements

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

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

Thanks to Dirk von Hugo for his useful comments.

7. References

7.1. Normative References
7.2. Informational References

[I-D.ietf-bess-nsh-bgp-control-plane]

[I-D.ietf-sfc-multi-layer-oam]

[I-D.ietf-sfc-nsh-tlv]


7.2. Informational References


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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2019.

Copyright Notice

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

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

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

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

2.1. Terminology

SF - Service Function

SFF - Service Function Forwarder

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

SFP - Service Function Path

SPI - Service Path Index

OAM - Operation, Administration, and Maintenance

2.2. Requirements Language

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

3. Extension

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

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

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

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

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     SFC Reply Path Type       |          Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Reply Service Function Path                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: SFC Reply TLV Format

where:

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

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

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reply Service Function Path Identifier     | Service Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Reply Service Function Path Field Format

where:

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

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

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

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

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

5.1. Bi-directional SFC Case

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

6. Security Considerations

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

In addition, the SFC Return Path extension, defined in this document, can be used for potential "proxying" attacks. For example, an echo request initiator may specify a return path that has a destination different from that of the initiator. But usually, such attacks will
not happen in an SFC domain where the initiators and receivers belong to the same domain, as specified in [RFC7665]. Even if the attack occurs, in order to prevent using the SFC Return Path extension for proxying any possible attacks, the return path SFP SHOULD have a path to reach the sender of the echo request, identified in SFC Source TLV [I-D.ietf-sfc-multi-layer-oam]. The receiver MAY drop the echo request when it cannot determine whether the return path SFP has the route to the initiator. That means, when sending echo request, the sender SHOULD choose a proper source address according to specified return path SFP to help the receiver to make the decision.

7. IANA Considerations

7.1. SFC Return Path Type

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA1</td>
<td>SFC Reply Path Type</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 1: SFC Return Path Type

7.2. New Return Codes

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA2</td>
<td>Reply Path TLV is missing</td>
<td>This document</td>
</tr>
<tr>
<td>TBA3</td>
<td>Reply SFP was not found</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 2: SFC Echo Reply Return Codes

8. References

8.1. Normative References

[I-D.ietf-sfc-multi-layer-oam]
Internet-Draft     Controlled Return Path for SFC OAM         March 2019


8.2. Informative References


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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2019.

Copyright Notice

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

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

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

2. **Design tree for SFC YANG data model**

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

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

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

    import ietf-yang-types {
        prefix "yang";
    }
}
<CODEEND>

The YANG module defines a generic configuration model for SFC.
enum sf-appfw {
  value 7 ;
}
enum sf-adc {
  value 8 ;
}
enum sf-woc {
  value 9 ;
}
enum sf-mon {
  value 10 ;
}
enum sf-sgw {
  value 11 ;
}
enum sf-pgw {
  value 12 ;
}
enum sf-hss {
  value 13 ;
}
enum sf-mme {
  value 14 ;
}
enum sf-pcrf {
  value 15 ;
}
enum sf-pcef {
  value 16 ;
}
enum sf-tdf {
  value 17 ;
}
enum sf-tssf {
  value 18 ;
}
enum sf-tds {
  value 19 ;
}
enum sf-pep {
  value 20 ;
}
enum sf-ims {
  value 21 ;
}
enum sf-li {
  value 22 ;
}
enum sf-proxy {
    value 23;
}

description "The nexthop node type.";
}

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

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

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

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

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

TBD.

5. IANA Considerations

TBD.

6. References

6.1. Normative References


6.2. Information References

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

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

Authors’ Addresses

Ting Ao
ZTE Corporation
No.889, BiBo Road
Shanghai  201203
China
Phone: +86 21 68897642
Email: ao.ting@zte.com.cn

Ran Chen
ZTE Corporation
Email: ran.chen@zte.com.cn

Wei Wei
ZTE Corporation
Email: wei.wei@zte.com.cn
Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on December 5, 2019.

Copyright Notice

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

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

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

2. Design tree for SFC YANG data model

module: ietf-sfc

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

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

```xml
<CODEBEGIN> file "ietf-sfc@2019-03-10.yang"
module ietf-sfc {
    namespace "urn:ietf:params:xml:ns:yang:ietf-sfc";
    prefix "sfc";
    import ietf-inet-types {
        prefix "inet";
    }
    import ietf-yang-types {
        prefix "yang";
    }
}
</CODEBEGIN>
```
organization "IETF SFC Working Group";

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

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

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

revision 2019-03-07{
    description
        "Initial revision."
    reference "RFC XXXX: YANG Data Model for SFC Protocol."
}

/*Typedefs*/
typedef sfp-nexthop-type {
    type enumeration {
        enum sff {
            value 1 ;
        }
        enum sf-firewall {
            value 2 ;
        }
        enum sf-dpi {
            value 3 ;
        }
        enum sf-ids {
            value 4 ;
        }
        enum sf-edgefw {
            value 5 ;
        }
        enum sf-segfw {
            value 6 ;
        }
    }
}
enum sf-appfw {
  value 7;
}
enum sf-adc {
  value 8;
}
enum sf-woc {
  value 9;
}
enum sf-mon {
  value 10;
}
enum sf-sgw {
  value 11;
}
enum sf-pgw {
  value 12;
}
enum sf-hss {
  value 13;
}
enum sf-mme {
  value 14;
}
enum sf-pcrf {
  value 15;
}
enum sf-pceef {
  value 16;
}
enum sf-tdf {
  value 17;
}
enum sf-tssf {
  value 18;
}
enum sf-tds {
  value 19;
}
enum sf-pep {
  value 20;
}
enum sf-ims {
  value 21;
}
enum sf-li {
  value 22;
}
enum sf-proxy {
    value 23;
}

description "The nexthop node type."
}

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

type inet:ipv4-address;
description "Remote IPv4 address."
}
description "The configuration for SFP nexthop which encapsulation type is ethernet&ipv4."
}
case ipv6-nexthop {
  leaf nh-node-type {
    type sfp-nexthop-type;
description "Nexthop node type.";
  }
  leaf remote-ipv6 {
    type inet:ipv6-address;
description "Remote IPv6 address.";
  }

description "The configuration for SFP nexthop which encapsulation type is ethernet&ipv6."
}
case mac-nexthops {
  leaf nh-node-type {
    type sfp-nexthop-type;
description "Nexthop node type.";
  }
  leaf remote-mac {
    type yang:mac-address;
description "MAC address.";
  }

description "The configuration for SFP nexthop which specifies the MAC address."
}
case vxlan-gpe-nexthop {
  leaf nh-node-type {
    type sfp-nexthop-type;
description "Nexthop node type.";
  }
  leaf remote-ip {
    type inet:ip-address;
description "Remote IP address.";
  }
  leaf source-ip {
    type inet:ipv4-address;
description "The source IP address.";
  }
  leaf destination-ip {
    type inet:ipv4-address;
description "The destination address.";
  }
  leaf vni {

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

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

<CODE ENDS>

4. Security Considerations
TBD.

5. IANA Considerations
TBD.

6. References

6.1. Normative References

           Chaining (SFC) Architecture", RFC 7665,
           DOI 10.17487/RFC7665, October 2015,

           "Network Service Header (NSH)", RFC 8300,
           DOI 10.17487/RFC8300, January 2018,

6.2. Information References

[I-D.ietf-bess-nsh-bgp-control-plane]
  Farrel, A., Drake, J., Rosen, E., Urtaro, J., and L.
  Jalil, "BGP Control Plane for NSH SFC", draft-ietf-bess-nsh-bgp-control-plane-11
  (work in progress), May 2019.
[I-D.ietf-sfc-dc-use-cases]

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

Authors’ Addresses

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

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

Ran Chen
ZTE Corporation

Email: ran.chen@zte.com.cn

Wei Wei
ZTE Corporation

Email: wei.wei@zte.com.cn
Network Service Header (NSH) Encapsulation for In-situ OAM (IOAM) Data

draft-ietf-sfc-ioam-nsh-01

Abstract

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

Status of This Memo

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

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

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

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

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

Abbreviations used in this document:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOAM</td>
<td>In-situ Operations, Administration, and Maintenance</td>
</tr>
<tr>
<td>NSH</td>
<td>Network Service Header</td>
</tr>
<tr>
<td>OAM</td>
<td>Operations, Administration, and Maintenance</td>
</tr>
<tr>
<td>SFC</td>
<td>Service Function Chaining</td>
</tr>
<tr>
<td>TLV</td>
<td>Type, Length, Value</td>
</tr>
</tbody>
</table>

3. IOAM data fields encapsulation in NSH

The NSH is defined in [RFC8300]. IOAM data fields are carried in NSH using a next protocol header which follows the NSH MD context headers. An IOAM header is added containing the different IOAM data fields defined in [I-D.ietf-ippm-ioam-data]. In an administrative domain where IOAM is used, insertion of the IOAM header in NSH is enabled at the NSH tunnel endpoints, which also serve as IOAM encapsulating/decapsulating nodes by means of configuration.
The NSH header and fields are defined in [RFC8300]. The "NSH Next Protocol" value (referred to as "NP" in the diagram above) is TBD_IOAM.

The IOAM related fields in NSH are defined as follows:

IOAM-Type: 8-bit field defining the IOAM Option type, as defined in Section 7.2 of [I-D.ietf-ippm-ioam-data].

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

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

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The semantics of this field are identical to the Next Protocol field in [RFC8300].
IOAM Option and Data Space: IOAM option header and data is present as specified by the IOAM-Type field, and is defined in Section 4 of [I-D.ietf-ippm-ioam-data].

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

4. Considerations

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

4.1. Discussion of the encapsulation approach

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

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

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

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

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

1. Encapsulation of IOAM data fields as "NSH MD Type 2" (see [RFC8300], Section 2.5). Each IOAM data field option (trace, proof-of-transit, and edge-to-edge) would be specified by a type, with the different IOAM data fields being TLVs within this the particular option type. NSH MD Type 2 offers support for variable length meta-data. The length field is 6-bits, resulting in a maximum of 256 (2^6 x 4) octets.

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

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

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

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

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

5. IANA Considerations

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


8.2. Informative References


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
Network Service Header (NSH) Encapsulation for In-situ OAM (IOAM) Data

draft-ietf-sfc-ioam-nsh-02

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 14, 2020.

Copyright Notice

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

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

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

2. Conventions

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

Abbreviations used in this document:

IOAM: In-situ Operations, Administration, and Maintenance

NSH: Network Service Header

OAM: Operations, Administration, and Maintenance

SFC: Service Function Chaining
3. IOAM data fields encapsulation in NSH

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

The NSH header and fields are defined in [RFC8300]. The "NSH Next Protocol" value (referred to as "NP" in the diagram above) is TBD_IOAM.

The IOAM related fields in NSH are defined as follows:

**IOAM-Type**: 8-bit field defining the IOAM Option type, as defined in Section 7.2 of [I-D.ietf-ippm-ioam-data].
IOAM HDR Len: 8 bit Length field contains the length of the IOAM header in 4-octet units.

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

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

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

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

4. Considerations

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

4.1. Discussion of the encapsulation approach

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

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

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

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

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

1. Encapsulation of IOAM data fields as "NSH MD Type 2" (see [RFC8300], Section 2.5). Each IOAM data field option (trace, proof-of-transit, and edge-to-edge) would be specified by a type, with the different IOAM data fields being TLVs within this the particular option type. NSH MD Type 2 offers support for variable length meta-data. The length field is 6-bits, resulting in a maximum of 256 \((2^6 \times 4)\) octets.

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

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

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

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

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

5. IANA Considerations

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

<table>
<thead>
<tr>
<th>Next Protocol</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>TBD_IOAM</td>
<td>This document</td>
</tr>
</tbody>
</table>

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. Contributors

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

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

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

[I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H.,
Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov,
P., Chang, R., daniel.bernier@bell.ca, d., and J. Lemon,
"Data Fields for In-situ OAM", draft-ietf-ippm-ioam-
data-06 (work in progress), July 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,

"Network Service Header (NSH)", RFC 8300,
DOI 10.17487/RFC8300, January 2018,

9.2. Informative References


Chaining (SFC) Architecture", RFC 7665,
DOI 10.17487/RFC7665, October 2015,

Authors' Addresses

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

Email: fbrockne@cisco.com
Active OAM for Service Function Chains in Networks
draft-ietf-sfc-multi-layer-oam-02

Abstract

A set of requirements for active Operation, Administration and
Maintenance (OAM) of Service Function Chains (SFCs) in networks is
presented. Based on these requirements an encapsulation of active
OAM message in SFC and a mechanism to detect and localize defects
described. Also, this document updates RFC 8300 in the definition of
O (OAM) bit in the Network Service Header (NSH) and defines how the
active OAM message identified in SFC NSH.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 9, 2019.

Copyright Notice

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

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

[RFC7665] defines components necessary to implement Service Function Chain (SFC). These include a classifier which performs the classification of incoming packets. A Service Function Forwarder (SFF) is responsible for forwarding traffic to one or more connected Service Functions (SFs) according to the information carried in the SFC encapsulation. SFF also handles traffic coming back from the SF and transports the data packets to the next SFF. And the SFF serves as termination element of the Service Function Path (SFP). SF is responsible for the specific treatment of received packets.
Resulting from that SFC is constructed by a number of these components, there are different views from different levels of the SFC. One is the SFC, entirely abstract entity, which defines an ordered set of SFFs that must be applied to packets selected as a result of classification. But SFC doesn’t specify the exact mapping between SFFs and SFs. Thus there exists another semi-abstract entity referred to as SFP. SFP is the instantiation of the SFC in the network and provides a level of indirection between the entirely abstract SFC and a fully specified ordered list of SFFs and SFs identities that the packet will visit when it traverses the SFC. The latter entity is being referred to as Rendered Service Path (RSP). The main difference between SFP and RSP is that in the former the authority to select the SFF/SF has been delegated to the network.

This document defines how active Operation, Administration and Maintenance (OAM), per [RFC7799] definition of active OAM, identified in Network Service Header (NSH) SFC, lists requirements to improve the troubleshooting efficiency, and defines SFC Echo request and Echo reply that enables on-demand Continuity Check, Connectivity Verification among other operations over SFC in networks. Also, this document updates Section 2.2 of [RFC8300] in part of the definition of 0 bit in the (NSH).

2. Conventions

2.1. Requirements Language

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

2.2. Terminology

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

e2e: End-to-End

FM: Fault Management

NSH: Network Service Header

OAM: Operations, Administration, and Maintenance

PRNG: Pseudorandom number generator
3. Requirements for Active OAM in SFC Network

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

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

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

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

The egress, SFF3 in the example in Figure 1, is the entity that detects the failure of the SFC. It must be able to signal the new defect state to the ingress SFF1. Hence the following requirement:
REQ#3: SFC OAM MUST support Remote Defect Indication (RDI) notification by the egress to the ingress.

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

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

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

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

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

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

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

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

4. Active OAM Identification in SFC NSH

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

O bit: Setting this bit indicates an OAM packet.

This document updates the definition of O bit as follows:

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

Active SFC OAM defined as a combination of OAM commands and/or data included in a message that immediately follows the NSH. To identify the active OAM message the value on the Next Protocol field MUST be
set to Active SFC OAM (TBA1) according to Section 8.1. The rules of interpreting the values of O bit and the Next Protocol field are as follows:

- **O** bit set, and the Next Protocol value is not one of identifying active or hybrid OAM protocol (per [RFC7799] definitions), e.g., defined in this specification Active SFC OAM - a Fixed-Length Context Header or Variable-Length Context Header(s) contain OAM command or data. and the type of payload determined by the Next Protocol field;

- **O** bit set, and the Next Protocol value is one of identifying active or hybrid OAM protocol - the payload that immediately follows SFC NSH contains OAM command or data;

- **O** bit is clear – no OAM in a Fixed-Length Context Header or Variable-Length Context Header(s) and the payload determined by the value of the Next Protocol field;

- **O** bit is clear and the Next Protocol value is one of identifying active or hybrid OAM protocol MUST be identified and reported as the erroneous combination. An implementation MAY have control to enable processing of the OAM payload.

From the above-listed rules follows the recommendation to avoid combination of OAM in a Fixed-Length Context Header or Variable-Length Context Header(s) and in the payload immediately following the SFC NSH because there is no unambiguous way to identify such combination using the O bit and the Next Protocol field.

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

```
+---------------------------------------------+-
| V | Message Type | Flags | Length |
+---------------------------------------------+---
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

**Figure 2: SFC Active OAM Header**
V - two bits long field indicates the current version of the SFC active OAM header. The current value is 0.

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

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

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

5. Echo Request/Echo Reply for SFC in Networks

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

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Version Number        |         Global Flags          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Message Type  |   Reply mode  |  Return Code  | Return S.code |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sender’s Handle                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Sequence Number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                              TLVs                             ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: SFC Echo Request/Reply format

The interpretation of the fields is as follows:

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

The Global Flags is a bit vector field.
The Message Type field reflects the type of the packet. Value TBA3 identifies echo request and TBA4 - echo reply

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

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

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

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

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

5.1. Return Codes

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Return Code</td>
</tr>
<tr>
<td>1</td>
<td>Malformed echo request received</td>
</tr>
<tr>
<td>2</td>
<td>One or more of the TLVs was not understood</td>
</tr>
</tbody>
</table>
5.2. SFC Echo Request Transmission

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

Value of the Reply Mode field MAY be set to:

- **Do Not Reply (TBA5)** if one-way monitoring is desired. If the echo request is used to measure synthetic packet loss; the receiver may report loss measurement results to a remote node.

- **Reply via an IPv4/IPv6 UDP Packet (TBA6)** value likely will be the most used.

- **Reply via Application Level Control Channel (TBA7)** value if the SFP may have bi-directional paths.

- **Reply via Specified Path (TBA8)** value to enforce the use of the particular return path specified in the included TLV to verify bi-directional continuity and also increase the robustness of the monitoring by selecting a more stable path.

5.3. SFC Echo Request Reception

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

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

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

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

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SFC OAM Source ID Type    |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Value                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: SFC Source TLV

where

- SFC OAM Source Id Type is two octets in length and has the value of TBA9 Section 8.7.
- Length is two octets long field, and the value equals the length of the Value field in octets.
- Value field contains the IP address of the sender of the SFC OAM control message, IPv4 or IPv6.

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

5.5. SFC Echo Reply Reception

An SFF SHOULD NOT accept SFC echo reply unless the received passes the following checks:
o the received SFC echo reply is well-formed;

o it has outstanding SFC echo request sent from the UDP port that matches destination UDP port number of the received packet;

o if the matching to the echo request found, the value of Sender’s Handle in the echo request sent is equal to the value of Sender’s Handle in the echo reply received;

o if all checks passed, the SFF checks if the Sequence Number in the echo request sent matches to the Sequence Number in the echo reply received.

6. Security Considerations

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

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

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

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

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

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

8. IANA Considerations

8.1. SFC Active OAM Protocol

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA1</td>
<td>SFC Active OAM</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 1: SFC Active OAM Protocol

8.2. SFC Active OAM Message Type

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>IETF Consensus</td>
</tr>
<tr>
<td>1 - 32767</td>
<td>Reserved</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>32768 - 65530</td>
<td>Reserved</td>
<td>Private Use</td>
</tr>
<tr>
<td>65531 - 65534</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>65535</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: SFC Active OAM Message Type

IANA is requested to assign new type from the SFC Active OAM Message Type registry as follows:
### Table 3: SFC Echo Request/Echo Reply Type

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA2</td>
<td>SFC Echo Request/Echo Reply</td>
<td>This document</td>
</tr>
</tbody>
</table>

#### 8.3. SFC Echo Request/Echo Reply Parameters

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

#### 8.4. SFC Echo Request/Echo Reply Message Types

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>TBA3</td>
<td>SFC Echo Request</td>
<td>This document</td>
</tr>
<tr>
<td>TBA4</td>
<td>SFC Echo Reply</td>
<td>This document</td>
</tr>
<tr>
<td>TBA4+1-191</td>
<td>Unassigned</td>
<td>IETF Review</td>
</tr>
<tr>
<td>192-251</td>
<td>Unassigned</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>252-254</td>
<td>Unassigned</td>
<td>Private Use</td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

#### Table 4: SFC Echo Request/Echo Reply Message Types

#### 8.5. SFC Echo Reply Modes

IANA is requested to create in the SFC Echo Request/Echo Reply Parameters registry the new sub-registry Reply Modes All code points in the range 1 through 191 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC8126] and assign values as follows:
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>TBA5</td>
<td>Do Not Reply</td>
<td>This document</td>
</tr>
<tr>
<td>TBA6</td>
<td>Reply via an IPv4/IPv6 UDP Packet</td>
<td>This document</td>
</tr>
<tr>
<td>TBA7</td>
<td>Reply via Application Level Control Channel</td>
<td>This document</td>
</tr>
<tr>
<td>TBA8</td>
<td>Reply via Specified Path</td>
<td>This document</td>
</tr>
<tr>
<td>TBA8+1-191</td>
<td>Unassigned</td>
<td>IETF Review</td>
</tr>
<tr>
<td>192-251</td>
<td>Unassigned</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>252-254</td>
<td>Unassigned</td>
<td>Private Use</td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: SFC Echo Reply Modes

8.6. SFC Echo Return Codes

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-191</td>
<td>Unassigned</td>
<td>IETF Review</td>
</tr>
<tr>
<td>192-251</td>
<td>Unassigned</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>252-254</td>
<td>Unassigned</td>
<td>Private Use</td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: SFC Echo Return Codes

Return Codes defined in this document are the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Return Code</td>
</tr>
<tr>
<td>1</td>
<td>Malformed echo request received</td>
</tr>
<tr>
<td>2</td>
<td>One or more of the TLVs was not understood</td>
</tr>
</tbody>
</table>

8.7. SFC TLV Type

IANA is requested to create SFC OAM TLV Type registry. All code points in the range 1 through 32759 in this registry shall be allocated according to the "IETF Review" procedure as specified in
[RFC8126]. Code points in the range 32760 through 65279 in this registry shall be allocated according to the "First Come First Served" procedure as specified in [RFC8126]. Remaining code points are allocated according to the Table 7:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>This document</td>
</tr>
<tr>
<td>1-32767</td>
<td>Mandatory TLV, unassigned</td>
<td>IETF Review</td>
</tr>
<tr>
<td>32768-65279</td>
<td>Optional TLV, unassigned</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>65280-65519</td>
<td>Experimental</td>
<td>This document</td>
</tr>
<tr>
<td>65520-65534</td>
<td>Private Use</td>
<td>This document</td>
</tr>
<tr>
<td>65535</td>
<td>Reserved</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 7: SFC TLV Type Registry

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA9</td>
<td>Source IP Address</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 8: SFC OAM Source IP Address Type

8.8. SFC OAM UDP Port

IANA is requested to allocate UDP port number according to

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Port Number</th>
<th>Transport Protocol</th>
<th>Description</th>
<th>Semantics Definition</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFC OAM</td>
<td>TBA10</td>
<td>UDP</td>
<td>SFC OAM</td>
<td>Section 5.4</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 9: SFC OAM Port
9. References

9.1. Normative References


9.2. Informative References


Authors’ Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

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

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

Bhumip Khasnabish
Individual contributor

Email: vumipi@gmail.com

Cui Wang

Email: lindawangjoy@gmail.com
Abstract

A set of requirements for active Operation, Administration and Maintenance (OAM) of Service Function Chains (SFCs) in networks is presented. Based on these requirements an encapsulation of active OAM message in SFC and a mechanism to detect and localize defects described. Also, this document updates RFC 8300 in the definition of O (OAM) bit in the Network Service Header (NSH) and defines how the active OAM message identified in SFC NSH.

Status of This Memo

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

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

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

This Internet-Draft will expire on November 9, 2019.

Copyright Notice

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

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

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

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

The main difference between SFP and RSP is that in the former the authority to select the SFF/SF has been delegated to the network.

This document defines how active Operation, Administration and Maintenance (OAM), per [RFC7799] definition of active OAM, identified in Network Service Header (NSH) SFC, lists requirements to improve the troubleshooting efficiency, and defines SFC Echo request and Echo reply that enables on-demand Continuity Check, Connectivity Verification among other operations over SFC in networks. Also, this document updates Section 2.2 of [RFC8300] in part of the definition of 0 bit in the (NSH).

2. Conventions

2.1. Requirements Language

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

2.2. Terminology

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

e2e: End-to-End

FM: Fault Management

NSH: Network Service Header

OAM: Operations, Administration, and Maintenance
3. Requirements for Active OAM in SFC Network

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

```
+---+  +---+   +---+  +---+  +---+  +---+
|SF1|  |SF2|   |SF3|  |SF4|  |SF5|  |SF6|
+---+  +---+   +---+  +---+  +---+  +---+
 \ /          \  /         \  /         \  /
+----------+       +----+         +----+        +----+
|Classifier|-------|SFF1|---------|SFF2|--------|SFF3|
+----------+       +----+         +----+        +----+
```

Figure 1: SFC reference model

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

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

REQ#2: SFC OAM MUST support pro-active monitoring of any element in the SFC availability.
The egress, SFF3 in the example in Figure 1, is the entity that detects the failure of the SFC. It must be able to signal the new defect state to the ingress SFF1. Hence the following requirement:

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

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

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

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

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

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

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

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

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

4. Active OAM Identification in SFC NSH

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

O bit: Setting this bit indicates an OAM packet.

This document updates the definition of O bit as follows:

O bit: Setting this bit indicates an OAM command and/or data in the NSH Context Header or packet payload.
Active SFC OAM defined as a combination of OAM commands and/or data included in a message that immediately follows the NSH. To identify the active OAM message the value on the Next Protocol field MUST be set to Active SFC OAM (TBA1) according to Section 8.1. The rules of interpreting the values of O bit and the Next Protocol field are as follows:

- O bit set, and the Next Protocol value is not one of identifying active or hybrid OAM protocol (per [RFC7799] definitions), e.g., defined in this specification Active SFC OAM - a Fixed-Length Context Header or Variable-Length Context Header(s) contain OAM command or data. and the type of payload determined by the Next Protocol field;

- O bit set, and the Next Protocol value is one of identifying active or hybrid OAM protocol - the payload that immediately follows SFC NSH contains OAM command or data;

- O bit is clear - no OAM in a Fixed-Length Context Header or Variable-Length Context Header(s) and the payload determined by the value of the Next Protocol field;

- O bit is clear and the Next Protocol value is one of identifying active or hybrid OAM protocol MUST be identified and reported as the erroneous combination. An implementation MAY have control to enable processing of the OAM payload.

From the above-listed rules follows the recommendation to avoid combination of OAM in a Fixed-Length Context Header or Variable-Length Context Header(s) and in the payload immediately following the SFC NSH because there is no unambiguous way to identify such combination using the O bit and the Next Protocol field.

Several active OAM protocols will be needed to address all the requirements listed in Section 3. Destination UDP port number may identify protocols if IP/UDP encapsulation used. But extra IP/UDP headers, especially in the case of IPv6, add noticeable overhead. This document defines Active OAM Header Figure 2 to demultiplex active OAM protocols on an SFC.
V - two bits long field indicates the current version of the SFC active OAM header. The current value is 0.

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

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

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

5. Echo Request/Echo Reply for SFC in Networks

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

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

The Global Flags is a bit vector field.

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Return Code</td>
</tr>
<tr>
<td>1</td>
<td>Malformed echo request received</td>
</tr>
<tr>
<td>2</td>
<td>One or more of the TLVs was not understood</td>
</tr>
</tbody>
</table>

5.2. SFC Echo Request Transmission

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

Value of the Reply Mode field MAY be set to:

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

5.3. SFC Echo Request Reception

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

Firstly, the SFF that has received an SFC echo request verifies the general sanity of the received packet. If the packet is not well-formed, the receiver SFF SHOULD send an SFC echo reply with the
Return Code set to "Malformed echo request received" and the Subcode set to zero. If there are any TLVs not marked as "Ignore" (i.e., if the TLV type is less than 32768, see Section 3) that SFF does not understand, the SFF MUST send an SFC echo reply with the Return Code set to 2 ("One or more TLVs was not understood") and set the Subcode to zero. In the latter case, the SFF MAY include an Errored TLVs TLV (Section 5.3.1) that as sub-TLVs contains only the misunderstood TLVs. The header field's Sender's Handle, Sequence Number are not examined but are included in the SFC echo reply message.

5.3.1. Errored TLVs TLV

If the Return Code for the echo reply is determined as 2 ("One or more TLVs was not understood"), then the Errored TLVs TLV MAY be included in an echo reply. The use of this TLV allows to inform the sender of an echo request of mandatory TLVs either not supported by an implementation or parsed and found to be in error.

```
0                   1                   2                   3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Errored TLVs Type        |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Value                             |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Errored TLVs TLV

where

The Errored TLVs Type MUST be set to TBA11 Section 8.7.

The Value field contains the mandatory TLVs, encoded as sub-TLVs, that were not understood or failed to be parsed correctly.

5.4. SFC Echo Reply Transmission

The Reply Mode field directs whether and how the echo reply message should be sent. The sender of the echo request MAY use TLVs to request that the corresponding echo reply is transmitted over the specified path. Value TBA3 is referred to as "Do not reply" mode and suppresses transmission of echo reply packet. The default value
Responder to the SFC echo request sends the echo reply over IP network if the Reply mode is Reply via an IPv4/IPv6 UDP Packet. Because SFC NSH does not identify the ingress of the SFP the echo request, the source ID MUST be included in the message and used as the IP destination address for IP/UDP encapsulation of the SFC echo reply. The sender of the SFC echo request MUST include SFC Source TLV Figure 5.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|   SFC OAM Source ID Type    |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                           Value                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 5: SFC Source TLV

where

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

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

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

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

5.5. SFC Echo Reply Reception

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

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

o if all checks passed, the SFF checks if the Sequence Number in the echo request sent matches to the Sequence Number in the echo reply received.

6. Security Considerations

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

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

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

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

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

7. Acknowledgments

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

8.1.  SFC Active OAM Protocol

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

+-----------------+---------------+---------------+
| Value | Description | Reference     |
|-------+----------------+---------------+
| TBA1  | SFC Active OAM | This document |
+-----------------+---------------+---------------+

Table 1: SFC Active OAM Protocol

8.2.  SFC Active OAM Message Type

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

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

Table 2: SFC Active OAM Message Type

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

+-----------------+---------------+---------------+
| Value | Description        | Reference     |
|-------+-----------------+---------------+
| TBA2  | SFC Echo Request/Echo Reply | This document |
+-----------------+---------------+---------------+

Table 3: SFC Echo Request/Echo Reply Type
8.3. SFC Echo Request/Echo Reply Parameters

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

8.4. SFC Echo Request/Echo Reply Message Types

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>TBA3</td>
<td>SFC Echo Request</td>
<td>This document</td>
</tr>
<tr>
<td>TBA4</td>
<td>SFC Echo Reply</td>
<td>This document</td>
</tr>
<tr>
<td>TBA4+1-191</td>
<td>Unassigned</td>
<td>IETF Review</td>
</tr>
<tr>
<td>192-251</td>
<td>Unassigned</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>252-254</td>
<td>Unassigned</td>
<td>Private Use</td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: SFC Echo Request/Echo Reply Message Types

8.5. SFC Echo Reply Modes

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-191</td>
<td>Unassigned</td>
<td>IETF Review</td>
</tr>
<tr>
<td>192-251</td>
<td>Unassigned</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>252-254</td>
<td>Unassigned</td>
<td>Private Use</td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: SFC Echo Return Codes

Return Codes defined in this document are the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Return Code</td>
</tr>
<tr>
<td>1</td>
<td>Malformed echo request received</td>
</tr>
<tr>
<td>2</td>
<td>One or more of the TLVs was not understood</td>
</tr>
</tbody>
</table>

### 8.7. SFC TLV Type

IANA is requested to create SFC OAM TLV Type registry. All code points in the range 1 through 32759 in this registry shall be allocated according to the "IETF Review" procedure as specified in
Code points in the range 32760 through 65279 in this registry shall be allocated according to the "First Come First Served" procedure as specified in [RFC8126]. Remaining code points are allocated according to the Table 7:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>This document</td>
</tr>
<tr>
<td>1 - 32767</td>
<td>Mandatory TLV,</td>
<td>IETF Review</td>
</tr>
<tr>
<td></td>
<td>unassigned</td>
<td></td>
</tr>
<tr>
<td>32768 - 65279</td>
<td>Optional TLV,</td>
<td>First Come First Served</td>
</tr>
<tr>
<td></td>
<td>unassigned</td>
<td></td>
</tr>
<tr>
<td>65280 - 65519</td>
<td>Experimental</td>
<td>This document</td>
</tr>
<tr>
<td>65520 - 65534</td>
<td>Private Use</td>
<td>This document</td>
</tr>
<tr>
<td>65535</td>
<td>Reserved</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 7: SFC TLV Type Registry

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA9</td>
<td>Source IP Address</td>
<td>This document</td>
</tr>
<tr>
<td>TBA11</td>
<td>Errored TLVs</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 8: SFC OAM Source IP Address Type

8.8. SFC OAM UDP Port

IANA is requested to allocate UDP port number according to

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Port Number</th>
<th>Transport Protocol</th>
<th>Description</th>
<th>Semantics Definition</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFC OAM</td>
<td>TBA10 UDP</td>
<td>SFC OAM</td>
<td>Section 5.4</td>
<td>This document</td>
<td></td>
</tr>
</tbody>
</table>

Table 9: SFC OAM Port
9. References

9.1. Normative References


9.2. Informative References


Authors’ Addresses

Greg Mirsky
ZTE Corp.
Email: gregimirsky@gmail.com

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

Bhumip Khasnabish
Individual contributor
Email: vumipl@gmail.com

Cui Wang
Individual contributor
Email: lindawangjoy@gmail.com
Explicit Congestion Notification (ECN) and Congestion Feedback
Using the Network Service Header (NSH)
<draft-ietf-sfc-nsh-ecn-support-00.txt>

Abstract

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

Status of This Memo

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

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

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

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

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

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

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

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

1.1 NSH Background

The Service Function Chaining (SFC [RFC7665]) architecture calls for the encapsulation of traffic within a service function chaining domain with a Network Service Header (NSH [RFC8300]) added by the "Classifier" (ingress node) on entry to the domain and the NSH being removed on exit from the domain at the egress node. The NSH is used to control the path of a packet in an SFC domain. The NSH is a natural place, in a domain where traffic is NSH encapsulated, to note congestion, avoiding possible confusion due, for example, to changes in the outer transport header in different parts of the domain.
Figure 1 shows an SFC domain for the purpose of illustrating the use of NSH. Traffic passes through a sequence of Service Function Forwarders (SFFs) each of which sends the traffic to one or more Service Functions (SFs). Each SF performs some operation on the traffic, for example firewall or Network Address Translation (NAT), and then returns it to the SFF from which it was received.

Logically, during the transit of each SFF, the outer transport header that got the packet to the SFF is stripped, the SFF decides on the next forwarding step, either adding a transport header or, if the SFF is the exit/egress, removing the NSH header. The transport headers
added may be different in different regions of the SFC domain. For example, IP could be used for some SFF-to-SFF communication and MPLS used for other such communication.

1.2 ECN Background

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

1.3 Tunnel Congestion Feedback Background

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

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

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

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

(3) Traffic re-direction, where the ingress node configures the NSH of some future traffic so that it avoids congested paths. Great care must be taken to avoid (a) significant re-ordering of traffic in flows that it is desirable to keep in order and (b) oscillation/instability in traffic paths due to alternate congestion of previously idle paths and the idling of previously congested paths. For example, it is preferable to classify...
traffic into flows of a sufficiently coarse granularity that the
flows are long lived and use a stable path per flow sending only
newly appearing flows on apparently uncongested paths.

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

Figure 2: Congestion Feedback across an SFC Domain

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

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

Acronyms:

AQM - Active Queue Management [RFC7567]

CE - Congestion Experienced [RFC3168]

downstream - The direction from ingress to egress

ECN - Explicit Congestion Notification [RFC3168]

ECT - ECN Capable Transport [RFC3168]

IPFIX - IP Flow Information Export [RFC7011]

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

NSH - Network Service Header [RFC8300]

SF - Service Function [RFC7665]

SFC - Service Function Chaining [RFC7665]

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

TLV - Type Length Value

upstream - The direction from egress to ingress
2. The NSH ECN Field

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

+-----------------------------------+
| Transport Encapsulation           |
|                                  |
| Network Service Header (NSH)      |
| +------------------------------+  |
| | Base Header                   |  |
| +------------------------------+  |
| | Service Path Header           |  |
| +------------------------------+  |
| | Metadata (Context Header(s))   |  |
| +------------------------------+  |
| +------------------------------+  |
|                                  |
| Original Packet / Frame          |
+-----------------------------------+

Figure 3. Data Encapsulation with the NSH

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

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Ver</td>
<td>O</td>
<td>U</td>
<td>TTL</td>
</tr>
</tbody>
</table>
| +-----------------------------------+
| ^ ^                                |
| +------+
| NSH ECN field                      |
| +------+

Figure 4: NSH Base Header

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

Table 1 shows the meaning of the code points in the NSH ECN field. These have the same meaning as the ECN field code points in the IPv4 or IPv6 header as defined in [RFC3168].
<table>
<thead>
<tr>
<th>Binary</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Not-ECT</td>
<td>Not ECN-Capable Transport</td>
</tr>
<tr>
<td>01</td>
<td>ECT(1)</td>
<td>ECN-Capable Transport</td>
</tr>
<tr>
<td>10</td>
<td>ECT(0)</td>
<td>ECN-Capable Transport</td>
</tr>
<tr>
<td>11</td>
<td>CE</td>
<td>Congestion Experienced</td>
</tr>
</tbody>
</table>

Table 1. ECN Field Code Points
3. ECN Support in the NSH

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

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

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

ECN Propagation:

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

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

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

ECN Marking:

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Incoming Header (also equal to departing Inner Header)</th>
<th>Departing NSH and Outer Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not-ECT</td>
<td>ECT(0)</td>
</tr>
<tr>
<td>ECT(0)</td>
<td>ECT(0)</td>
</tr>
<tr>
<td>ECT(1)</td>
<td>ECT(1)</td>
</tr>
<tr>
<td>CE</td>
<td>CE</td>
</tr>
</tbody>
</table>

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

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

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

```
+-----------------+
|   Outer Header  |
+-----------------+
|       NSH       |
+-----------------+
|   Inner Header  |
+-----------------+
|     Payload     |
+-----------------+
```

Figure 5. Packet in Transit

3.2.1 At NSH Transit Nodes

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

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

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

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

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

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

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

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

3.2.3 At Other Forwarding Nodes

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

3.3 At Exit/Egress

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

<table>
<thead>
<tr>
<th>Arriving Inner Header</th>
<th>Arriving Outer Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not-ECT</td>
<td>Not-ECT</td>
</tr>
<tr>
<td>ECT(0)</td>
<td>ECT(0)</td>
</tr>
<tr>
<td>ECT(1)</td>
<td>ECT(1)</td>
</tr>
<tr>
<td>CE</td>
<td>CE</td>
</tr>
</tbody>
</table>

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

### 3.4 Conservation of Packets

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>tbd(16-17)</td>
<td>NSH ECN</td>
<td>[this document]</td>
</tr>
</tbody>
</table>
6. Security Considerations

For general NSH security considerations, see [RFC8300].

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

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

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

7. Acknowledgements

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

Joel Halpern, Tal Mizrahi, Xinpeng Wei
Normative References


Informative References


Authors’ Addresses

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

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

Bob Briscoe
Independent
UK

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

Andrew G. Malis
Huawei Technologies

Email: agmalis@gmail.com
Copyright and IPR Provisions

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License. The definitive version of an IETF Document is that published by, or under the auspices of, the IETF. Versions of IETF Documents that are published by third parties, including those that are translated into other languages, should not be considered to be definitive versions of IETF Documents. The definitive version of these Legal Provisions is that published by, or under the auspices of, the IETF. Versions of these Legal Provisions that are published by third parties, including those that are translated into other languages, should not be considered to be definitive versions of these Legal Provisions. For the avoidance of doubt, each Contributor to the IETF Standards Process licenses each Contribution that he or she makes as part of the IETF Standards Process to the IETF Trust pursuant to the provisions of RFC 5378. No language to the contrary, or terms, conditions or rights that differ from or are inconsistent with the rights and licenses granted under RFC 5378, shall have any effect and shall be null and void, whether published or posted by such Contributor, or included with or in such Contribution.
Explicit Congestion Notification (ECN) and Congestion Feedback
Using the Network Service Header (NSH)
<draft-ietf-sfc-nsh-ecn-support-01.txt>

Abstract

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

Status of This Memo

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

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

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

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

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

1. Introduction ............................................3
   1.1 NSH Background .........................................3
   1.2 ECN Background .........................................5
   1.3 Tunnel Congestion Feedback Background ..................5
   1.4 Conventions Used in This Document ......................7

2. The NSH ECN Field .......................................8

3. ECN Support in the NSH .................................10
   3.1 At The Ingress ........................................11
   3.2 At Transit Nodes .................................12
   3.2.1 At NSH Transit Nodes .................................12
   3.2.2 At an SF/Proxy ................................13
   3.2.3 At Other Forwarding Nodes ...........................13
   3.3 At Exit/Egress ........................................13
   3.4 Conservation of Packets ................................14

4. Tunnel Congestion Feedback Support .....................15

5. IANA Considerations ....................................16

6. Security Considerations ................................17

7. Acknowledgements .......................................17

Normative References ......................................18
Informative References ....................................19

Authors’ Addresses ........................................20
1. Introduction

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

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

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

1.1 NSH Background

The Service Function Chaining (SFC [RFC7665]) architecture calls for the encapsulation of traffic within a service function chaining domain with a Network Service Header (NSH [RFC8300]) added by the "Classifier" (ingress node) on entry to the domain and the NSH being removed on exit from the domain at the egress node. The NSH is used to control the path of a packet in an SFC domain. The NSH is a natural place, in a domain where traffic is NSH encapsulated, to note congestion, avoiding possible confusion due, for example, to changes in the outer transport header in different parts of the domain.
Figure 1 shows an SFC domain for the purpose of illustrating the use of NSH. Traffic passes through a sequence of Service Function Forwarders (SFFs) each of which sends the traffic to one or more Service Functions (SFs). Each SF performs some operation on the traffic, for example firewall or Network Address Translation (NAT), and then returns it to the SFF from which it was received.

Logically, during the transit of each SFF, the outer transport header that got the packet to the SFF is stripped, the SFF decides on the next forwarding step, either adding a transport header or, if the SFF is the exit/egress, removing the NSH header. The transport headers
added may be different in different regions of the SFC domain. For example, IP could be used for some SFF-to-SFF communication and MPLS used for other such communication.

1.2 ECN Background

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

1.3 Tunnel Congestion Feedback Background

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

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

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

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

(3) Traffic re-direction, where the ingress node configures the NSH of some future traffic so that it avoids congested paths. Great care must be taken to avoid (a) significant re-ordering of traffic in flows that it is desirable to keep in order and (b) oscillation/instability in traffic paths due to alternate congestion of previously idle paths and the idling of previously congested paths. For example, it is preferable to classify
traffic into flows of a sufficiently coarse granularity that the flows are long lived and use a stable path per flow, then send only newly appearing flows on apparently uncongested paths.

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

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

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

Acronyms:

AQM - Active Queue Management [RFC7567]
CE - Congestion Experienced [RFC3168]
downstream - The direction from ingress to egress
ECN - Explicit Congestion Notification [RFC3168]
ECT - ECN Capable Transport [RFC3168]
IPFIX - IP Flow Information Export [RFC7011]
Not-ECT - Not ECN-Capable Transport [RFC3168]
NSH - Network Service Header [RFC8300]
SF - Service Function [RFC7665]
SFC - Service Function Chaining [RFC7665]
SFF - Service Function Forwarder [RFC7665] - A type of node that forwards based on the NSH.
TLV - Type Length Value
upstream - The direction from egress to ingress
2. The NSH ECN Field

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

Figures 3. Data Encapsulation with the NSH

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

Figures 4. NSH Base Header

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

Table 1 shows the meaning of the code points in the NSH ECN field. These have the same meaning as the ECN field code points in the IPv4 or IPv6 header as defined in [RFC3168].
<table>
<thead>
<tr>
<th>Binary</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Not-ECT</td>
<td>Not ECN-Capable Transport</td>
</tr>
<tr>
<td>01</td>
<td>ECT(1)</td>
<td>ECN-Capable Transport</td>
</tr>
<tr>
<td>10</td>
<td>ECT(0)</td>
<td>ECN-Capable Transport</td>
</tr>
<tr>
<td>11</td>
<td>CE</td>
<td>Congestion Experienced</td>
</tr>
</tbody>
</table>

Table 1. ECN Field Code Points
3. ECN Support in the NSH

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

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

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

ECN Propagation:

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

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

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

ECN Marking:

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Incoming Header (also equal to departing Inner Header)</th>
<th>Departing NSH and Outer Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not-ECT</td>
<td>ECT(0)</td>
</tr>
<tr>
<td>ECT(0)</td>
<td>ECT(0)</td>
</tr>
<tr>
<td>ECT(1)</td>
<td>ECT(1)</td>
</tr>
<tr>
<td>CE</td>
<td>CE</td>
</tr>
</tbody>
</table>

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

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

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

```
+-----------------+
|   Outer Header  |
+-----------------+
|       NSH       |
+-----------------+
|   Inner Header  |
+-----------------+
|     Payload     |
+-----------------+
```

Figure 5. Packet in Transit

3.2.1 At NSH Transit Nodes

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

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

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

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

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

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

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

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

3.2.3 At Other Forwarding Nodes

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

3.3 At Exit/Egress

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

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

3.4 Conservation of Packets

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>tbd(16-17)</td>
<td>NSH ECN</td>
<td>[this document]</td>
</tr>
</tbody>
</table>
6. Security Considerations

For general NSH security considerations, see [RFC8300].

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

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

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

7. Acknowledgements

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

   Joel Halpern, Tal Mizrahi, Xinpeng Wei
Normative References


Informative References


Authors’ Addresses

Donald E. Eastlake, 3rd
Futurewei Technologies
1424 Pro Shop Court
Davenport, FL 33896 USA
Tel: +1-508-333-2270
Email: d3e3e3@gmail.com

Bob Briscoe
Independent
UK
Email: ietf@bobbriscoe.net
URI: http://bobbriscoe.net/

Andrew G. Malis
Futurewei Technologies
Email: agmalis@gmail.com
Copyright and IPR Provisions

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

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

This Internet-Draft will expire on September 12, 2019.

Copyright Notice

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

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

Table of Contents

1. Introduction .................................................. 3
2. Conventions .................................................. 4
3. Proof of Transit .............................................. 5
   3.1. Basic Idea ................................................ 6
   3.2. Solution Approach ........................................ 7
      3.2.1. Setup ............................................... 7
      3.2.2. In Transit ......................................... 7
      3.2.3. Verification ....................................... 8
   3.3. Illustrative Example ...................................... 8
      3.3.1. Baseline ........................................... 8
         3.3.1.1. Secret Shares ................................. 8
         3.3.1.2. Lagrange Polynomials ......................... 9
         3.3.1.3. LPC Computation .............................. 9
         3.3.1.4. Reconstruction ............................... 9
         3.3.1.5. Verification .................................. 10
      3.3.2. Complete Solution .................................. 10
         3.3.2.1. Random Polynomial ............................ 10
         3.3.2.2. Reconstruction .............................. 10
         3.3.2.3. Verification .................................. 11
      3.3.3. Solution Deployment Considerations .................. 11
   3.4. Operational Aspects ...................................... 12
   3.5. Ordered POT (OPOT) ...................................... 12
4. Sizing the Data for Proof of Transit .......................... 13
5. Node Configuration ........................................... 14
   5.1. Procedure ............................................... 15
   5.2. YANG Model .............................................. 15
1. Introduction

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

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

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

2. Conventions

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

Abbreviations used in this document:

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

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

LPC: Lagrange Polynomial Constants

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

NSH: Network Service Header
POT: Proof of Transit

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

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

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

SFC: Service Function Chain

SSSS: Shamir’s Secret Sharing Scheme

SR: Segment Routing

3. Proof of Transit

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

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

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

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

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

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

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

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

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

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

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

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

3.2.1. Setup

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

3.2.2. In Transit

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

\[
\text{CML} = \text{CML} + \left(\left(\text{Share}(\text{POLY-1}) + \text{Share}(\text{POLY-2})\right) \times \text{LPC}\right) \mod \text{Prime},
\]

where "LPC" is the Lagrange Polynomial Constant and "Prime" is the prime number which defines the finite field arithmetic that all operations are done over. Please also refer to Section 3.3.2 below for further details how the operations are performed.

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

### 3.2.3. Verification

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

### 3.3. Illustrative Example

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

#### 3.3.1. Baseline

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

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

#### 3.3.1.1. Secret Shares

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

\[
\text{POLY-1}(2) = 28 \Rightarrow (x0, y0) = (2, 28)
\]

\[
\text{POLY-1}(4) = 17 \Rightarrow (x1, y1) = (4, 17)
\]
POLY-1(5) = 47 => (x2, y2) = (5, 47)

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

### 3.3.1.2. Lagrange Polynomials

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

\[
l_0(x) = \left(\frac{(x-x_1)}{(x_0-x_1)} \cdot \frac{(x-x_2)}{(x_0-x_2)}\right) \mod 53 = \left(\frac{(x-4)}{(2-4)} \cdot \frac{(x-5)}{(2-5)}\right) \mod 53 = (10/3 - 3x/2 + (1/6)x^2) \mod 53
\]

\[
l_1(x) = \left(\frac{(x-x_0)}{(x_1-x_0)} \cdot \frac{(x-x_2)}{(x_1-x_2)}\right) \mod 53 = (-5 + 7x/2 - (1/2)x^2) \mod 53
\]

\[
l_2(x) = \left(\frac{(x-x_0)}{(x_2-x_0)} \cdot \frac{(x-x_1)}{(x_2-x_1)}\right) \mod 53 = (8/3 - 2 + (1/3)x^2) \mod 53
\]

### 3.3.1.3. LPC Computation

Since x0=2, x1=4, x2=5 are chosen points. Given that computations are done over a finite arithmetic field ("modulo a prime number"), the Lagrange basis polynomial constants are computed modulo 53. The Lagrange Polynomial Constant (LPC) would be 10/3, -5, 8/3. LPC are computed by the Controller and communicated to the individual nodes.

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

LPC(11) = (-5) mod 53 = 48

LPC(12) = (8/3) mod 53 = 38

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

### 3.3.1.4. Reconstruction

Reconstruction of the polynomial is well-defined as

\[
POLY1(x) = l_0(x) \cdot y_0 + l_1(x) \cdot y_1 + l_2(x) \cdot y_2
\]

Subsequently, the SECRET, which is the constant coefficient of POLY1(x) can be computed as below.
SECRET = (y0*LPC(l0)+y1*LPC(l1)+y2*LPC(l2)) mod 53

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

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

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

3.3.1.5. Verification

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

3.3.2. Complete Solution

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

3.3.2.1. Random Polynomial

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

3.3.2.2. Reconstruction

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

\[ CML = CML + \left((\text{Share}(\text{POLY-1}) + \text{Share}(\text{POLY-2})) \times \text{LPC}\right) \mod \text{Prime} \]
Let us observe the packet level transformations in detail. For the example packet here, let the value RND be 45. Thus POLY-2 would be \((45 + 7x + 10x^2)\).

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

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

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

\[
CML = 0 + \left( (28 + 46) \times 21 \right) \mod 53 = 17
\]

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

\[
CML = 17 + \left( (17 + 21) \times 48 \right) \mod 53 = 17 + 22 = 39
\]

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

\[
CML = 39 + \left( (47 + 12) \times 38 \right) \mod 53 = 39 + 16 = 55 \mod 53 = 2
\]

The verification using CML is discussed in next section.

3.3.2.3. Verification

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

\[
VERIFY = (SECRET + RND) \mod Prime, \text{ with } Prime = 53 \text{ here}
\]

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

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

3.3.3. Solution Deployment Considerations

The "complete solution" described above in Section 3.3.2 could still be prone to replay or preplay attacks. An attacker could e.g. reuse the POT metadata for bypassing the verification. These threats can be mitigated by appropriate parameterization of the algorithm. Please refer to Section 8 for details.
3.4. Operational Aspects

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

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

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

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

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

3.5. Ordered POT (OPOT)

POT as discussed in this document so far only verifies that a defined set of nodes have been traversed by a packet. The order in which nodes where traversed is not verified. "Ordered Proof of Transit (OPOT)" addresses the need of deployments, that require to verify the order in which nodes were traversed. OPOT extends the POT scheme with symmetric masking between the nodes.
1. For each path the controller provisions all the nodes with (or asks them to agree on) two secrets per node, that we will refer to as masks, one for the connection from the upstream node(s), another for the connection to the downstream node(s). For obvious reasons, the ingress and egress (verifier) nodes only receive one, for downstream and upstream, respectively.

2. Any two contiguous nodes in the OPOT stream share the mask for the connection between them, in the shape of symmetric keys. Masks can be refreshed as per-policy, defined at each hop or globally by the controller.

3. Each mask has the same size in bits as the length assigned to CML plus RND, as described in the above sections.

4. Whenever a packet is received at an intermediate node, the CML+RND sequence is deciphered (by XORing, though other ciphering schemas MAY be possible) with the upstream mask before applying the procedures described in Section 3.3.2.

5. Once the new values of CML+RND are produced, they are ciphered (by XORing, though other ciphering schemas MAY be possible) with the downstream mask before transmitting the packet to the next node downstream.

6. The ingress node only applies step 5 above, while the verifier only applies step 4 before running the verification procedure.

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

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

4. Sizing the Data for Proof of Transit

Proof of transit requires transport of two data fields in every packet that should be verified:
1. RND: Random number (the constant coefficient of public polynomial)

2. CML: Cumulative

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

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

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

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

5. Node Configuration

A POT system consists of a number of nodes that participate in POT and a Controller, which serves as a control and configuration entity. The Controller is to create the required parameters (polynomials, prime number, etc.) and communicate the associated values (i.e. prime number, secret-share, LPC, etc.) to the nodes. The sum of all
parameters for a specific node is referred to as "POT-profile". For details see the YANG model in Section 5.2. This document does not define a specific protocol to be used between Controller and nodes. It only defines the procedures and the associated YANG data model.

5.1. Procedure

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

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

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

5.2. YANG Model

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

<CODE BEGINS> file "ietf-pot-profile@2016-06-15.yang"
module ietf-pot-profile {
  yang-version 1;
}


prefix ietf-pot-profile;

organization "IETF xxx Working Group";

contact "";

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

<CODE ENDS>

6. IANA Considerations

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

7. Manageability Considerations

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

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

8.1. Proof of Transit

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

- 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 reconstruct the k degree polynomial (POLY-3) only when all the points are correctly retrieved.

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

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

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

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

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

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

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

8.3. Anti-Replay

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

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

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

For all practical purposes in the rest of the document RND means SEQ_NO + RND to keep it simple.
The solution discussed in this memo does not currently mitigate replay attacks. An anti-replay mechanism may be included in future versions of the solution.

8.4. Anti-Preplay

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

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

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

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

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

8.5. Tampering

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

8.6. Recycling

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

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

8.7. Redundant Nodes and Failover

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

8.8. Controller Operation

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

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

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

8.9. Verification Scope

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

8.9.1. Node Ordering

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

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

8.9.2. Stealth Nodes

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

9. Acknowledgements

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

10. References

10.1. Normative References

[I-D.ietf-ippm-ioam-data]
10.2. Informative References

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

Authors' Addresses

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany
Email: fbrockne@cisco.com

Shwetha Bhandari
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India
Email: shwethab@cisco.com

Sashank Dara
Seconize
BANGALORE, Bangalore, KARNATAKA
INDIA
Email: sashank@seconize.co
Carlos Pignataro
Cisco Systems, Inc.
7200-11 Kit Creek Road
Research Triangle Park, NC 27709
United States

Email: cpignata@cisco.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

David Mozes

Email: mosesster@gmail.com

Tal Mizrahi
Huawei Network.IO Innovation Lab
Israel

Email: tal.mizrahi.phd@gmail.com

Alejandro Aguado
Universidad Politecnica de Madrid
Campus Montegancedo, Boadilla del Monte
Madrid 28660
Spain

Phone: +34 910 673 086
Email: a.aguadom@fi.upm.es
Diego R. Lopez  
Telefonica I+D  
Editor Jose Manuel Lara, 9 (1-B)  
Seville  41013  
Spain  

Phone: +34 913 129 041  
Email: diego.r.lopez@telefonica.com
Proof of Transit
draft-ietf-sfc-proof-of-transit-03

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 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 March 14, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents
(https://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction .......................... 3
2. 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 ................... 8
3.3. Illustrative Example ............... 8
   3.3.1. Baseline ....................... 8
      3.3.1.1. Secret Shares ............... 8
      3.3.1.2. Lagrange Polynomials .......... 9
      3.3.1.3. LPC Computation .............. 9
      3.3.1.4. Reconstruction .............. 9
      3.3.1.5. Verification ................ 10
   3.3.2. Complete Solution .............. 10
      3.3.2.1. Random Polynomial ........... 10
      3.3.2.2. Reconstruction .............. 10
      3.3.2.3. Verification ............... 11
   3.3.3. Solution Deployment Considerations ...... 11
3.4. Operational Aspects ............... 12
3.5. Ordered POT (OPOT) ............... 12
4. Sizing the Data for Proof of Transit .. 13
5. Node Configuration ................... 14
   5.1. Procedure ....................... 15
   5.2. YANG Model for POT .............. 15
      5.2.1. Main Parameters ............... 16
      5.2.2. Tree Diagram ................. 16
      5.2.3. YANG Model ................. 17
6. IANA Considerations .................. 20
7. Security Considerations .............. 20
   7.1. Proof of Transit .................. 21
   7.2. Cryptanalysis .................... 21
   7.3. Anti-Replay ...................... 22
   7.4. Anti-Preplay ..................... 23
   7.5. Tampering ....................... 23
1. Introduction

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

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

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

2. Conventions

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

Abbreviations used in this document:

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

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

LPC: Lagrange Polynomial Constants

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

NSH: Network Service Header

POT: Proof of Transit

POT-Profile: Proof of Transit Profile that has the necessary data for nodes to participate in proof of transit
RND: Random Bits generated per packet. Packet fields that do not change during the traversal are given as input to HMAC-256 algorithm. A minimum of 32 bits (left most) need to be used from the output if RND is used to verify the packet integrity. This is a standard recommendation by NIST.

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

SFC: Service Function Chain

SSSS: Shamir’s Secret Sharing Scheme

SR: Segment Routing

3. Proof of Transit

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

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

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

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

3.1. Basic Idea

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

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

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

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

3.2. Solution Approach

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

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

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

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

### 3.2.1. Setup

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

### 3.2.2. In Transit

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

\[
CML = CML + (((\text{Share}(POLY-1) + \text{Share}(POLY-2)) \times \text{LPC}) \mod \text{Prime}, \text{with} \ \ "\text{LPC}" \ \ \text{being the Lagrange Polynomial Constant and} \ "\text{Prime}" \ \ \text{being the prime number which defines the finite field arithmetic that all}\]

operations are done over. Please also refer to Section 3.3.2 below for further details how the operations are performed.

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

3.2.3. Verification

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

3.3. Illustrative Example

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

3.3.1. Baseline

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

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

3.3.1.1. Secret Shares

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

POLY-1(2) = 28 => (x0, y0) = (2, 28)
POLY-1(4) = 17 => (x1, y1) = (4, 17)
POLY-1(5) = 47 => (x2, y2) = (5, 47)

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

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

\[ l_0(x) = (((x-x_1) / (x_0-x_1)) * ((x-x_2)/x_0-x_2)) \mod 53 \]
\[ = (((x-4) / (2-4)) * ((x-5)/2-5))) \mod 53 \]
\[ = (10/3 - 3x/2 + (1/6)x^2) \mod 53 \]

\[ l_1(x) = (((x-x_0) / (x_1-x_0)) * ((x-x_2)/x_1-x_2)) \mod 53 \]
\[ = (-5 + 7x/2 - (1/2)x^2) \mod 53 \]

\[ l_2(x) = (((x-x_0) / (x_2-x_0)) * ((x-x_1)/x_2-x_1)) \mod 53 \]
\[ = (8/3 - 2 + (1/3)x^2) \mod 53 \]

3.3.1.3. LPC Computation

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

\[ \text{LPC}(l_0) = (10/3) \mod 53 = 21 \]
\[ \text{LPC}(l_1) = (-5) \mod 53 = 48 \]
\[ \text{LPC}(l_2) = (8/3) \mod 53 = 38 \]

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

3.3.1.4. Reconstruction

Reconstruction of the polynomial is well-defined as

\[ \text{POLY1}(x) = l_0(x) \cdot y_0 + l_1(x) \cdot y_1 + l_2(x) \cdot y_2 \]

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

\[ \text{SECRET} = (y_0 \cdot \text{LPC}(l_0) + y_1 \cdot \text{LPC}(l_1) + y_2 \cdot \text{LPC}(l_2)) \mod 53 \]
The secret can be easily reconstructed using the y-values and the LPC:

\[
\text{SECRET} = (y_0 \times \text{LPC}(l_0) + y_1 \times \text{LPC}(l_1) + y_2 \times \text{LPC}(l_2)) \mod 53 \\
= (28 \times 21 + 17 \times 48 + 47 \times 38) \mod 53 \\
= 3190 \mod 53 \\
= 10
\]

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

### 3.3.1.5. Verification

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

### 3.3.2. Complete Solution

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

#### 3.3.2.1. Random Polynomial

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

#### 3.3.2.2. Reconstruction

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

\[
\text{CML} = \text{CML} + ((\text{Share(POLY-1)} + \text{Share(POLY-2)}) \times \text{LPC}) \mod \text{Prime}
\]
Let us observe the packet level transformations in detail. For the example packet here, let the value RND be 45. Thus POLY-2 would be 

\[(45 + 7x + 10x^2)\].

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

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

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

\[CML = 0 + ((28 + 46)* 21) \mod 53 = 17\]

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

\[CML = 17 + ((17 + 21)*48) \mod 53 = 17 + 22 = 39\]

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

\[CML = 39 + ((47 + 12)*38) \mod 53 = 39 + 16 = 55 \mod 53 = 2\]

The verification using CML is discussed in next section.

3.3.2.3. Verification

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

\[\text{VERIFY} = (\text{SECRET} + \text{RND}) \mod \text{Prime}, \text{ with Prime } = 53\text{ here}\]

\[\text{VERIFY} = (\text{RND}-1 + \text{RND}-2) \mod \text{Prime} = (10 + 45) \mod 53 = 2\]

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

3.3.3. Solution Deployment Considerations

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

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

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

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

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

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

3.5. Ordered POT (OPOT)

POT as discussed in this document so far only verifies that a defined set of nodes have been traversed by a packet. The order in which nodes were traversed is not verified. "Ordered Proof of Transit (OPOT)" addresses the need of deployments, that require to verify the order in which nodes were traversed. OPOT extends the POT scheme with symmetric masking between the nodes.
1. For each path the controller provisions all the nodes with (or asks them to agree on) two secrets per node, that we will refer to as masks, one for the connection from the upstream node(s), another for the connection to the downstream node(s). For obvious reasons, the ingress and egress (verifier) nodes only receive one, for downstream and upstream, respectively.

2. Any two contiguous nodes in the OPOT stream share the mask for the connection between them, in the shape of symmetric keys. Masks can be refreshed as per-policy, defined at each hop or globally by the controller.

3. Each mask has the same size in bits as the length assigned to CML plus RND, as described in the above sections.

4. Whenever a packet is received at an intermediate node, the CML+RND sequence is deciphered (by XORing, though other ciphering schemas MAY be possible) with the upstream mask before applying the procedures described in Section 3.3.2.

5. Once the new values of CML+RND are produced, they are ciphered (by XORing, though other ciphering schemas MAY be possible) with the downstream mask before transmitting the packet to the next node downstream.

6. The ingress node only applies step 5 above, while the verifier only applies step 4 before running the verification procedure.

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

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

4. Sizing the Data for Proof of Transit

Proof of transit requires transport of two data fields in every packet that should be verified:
1. RND: Random number (the constant coefficient of public polynomial)

2. CML: Cumulative

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

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

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

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

5. Node Configuration

A POT system consists of a number of nodes that participate in POT and a Controller, which serves as a control and configuration entity. The Controller is to create the required parameters (polynomials, prime number, etc.) and communicate the associated values (i.e. prime number, secret-share, LPC, etc.) to the nodes. The sum of all
parameters for a specific node is referred to as "POT-Profile". For
details see the YANG model in Section 5.2. This document does not
define a specific protocol to be used between Controller and nodes.
It only defines the procedures and the associated YANG data model.

5.1. Procedure

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

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

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

5.2. YANG Model for POT

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

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

- **pot-profile-index**: Section 5.1 details that two POT-Profiles are used. Only one of the POT-Profiles is active at a given point in time, allowing the Controller to refresh the non-active one for future use. pot-profile-index defines which of the POT-Profiles (the "even" or "odd" POT-Profile) is currently active. pot-profile-index will be set in the first hop of the path or chain. Other nodes will not use this field.

- **prime-number**: Prime number used for module math computation.

- **secret-share**: Share of the secret of polynomial-1 used in computation for the node. If POLY-1 is defined by points \((x1_i, y1_i)\) with \(i=0,..k\), then for node \(i\), the secret-share will be \(y1_i\).

- **public-polynomial**: Public polynomial value for the node. If POLY-2 is defined by points \((x2_i, y2_i)\) with \(i=0,..k\), then for node \(i\), the secret-share will be \(y2_i\).

- **lpc**: Lagrange Polynomial Coefficient for the node, i.e. for node \(i\), this would be \(LPC(l_i)\), with \(l_i\) being the \(i\)-th Lagrange Basis Polynomial.

- **validator?**: True if the node is a verifier node.

- **validator-key?**: The validator-key represents the SECRET as described in the sections above. The SECRET is the constant coefficient of POLY-1(z). If \(POLY-1(z) = a_0 + a_1z + a_2z^2 + ... + a_kz^k\), then the SECRET would be \(a_0\).

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

5.2.2. Tree Diagram

This section shows a simplified graphical representation of the YANG data model for POT. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).

Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

Ellipsis ("...") stands for contents of subtrees that are not shown.

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

5.2.3. YANG Model

```yaml
file "ietf-pot-profile@2016-06-15.yang"
module ietf-pot-profile { 
    yang-version 1;
    prefix ietf-pot-profile;
    organization "IETF SFC Working Group";
    contact "WG Web: <https://tools.ietf.org/wg/sfc/>
    WG List: <mailto:sfc@ietf.org>";
    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 for the node. If POLY-1

is defined by points \((x_{1,i}, y_{1,i})\) with 
i=0,..k, then for node \(i\), the secret-share 
will be \(y_{1,i}\)."
}

leaf public-polynomial {
  type uint64;
  mandatory true;
  description
  "Public polynomial value for the node.
  If POLY-2 is defined by points \((x_{2,i}, y_{2,i})\)
  with i=0,..k, then for node \(i\),
  the secret-share will be \(y_{2,i}\)."
}

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

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

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

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

This document does not require any actions from IANA.

7. Security Considerations

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

7.1. Proof of Transit

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

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

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

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

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

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

7.2. Cryptanalysis

A passive attacker could try to harvest the POT data (i.e., CML, RND values) in order to determine the configured secrets. Subsequently two types of differential analysis for guessing the secrets could be done.
o Inter-Node: A passive attacker observing CML values across nodes (i.e., as the packets entering and leaving), cannot perform differential analysis to construct the points on POLY-1. This is because at each point there are four unknowns (i.e., Share(POLY-1), Share(Poly-2) LPC and prime number P) and three known values (i.e., RND, CML-before, CML-after). The application of symmetric masking for OPOT makes inter-node analysis less feasible.

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

7.3. Anti-Replay

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

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

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

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

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

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

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

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

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

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

7.5. Tampering

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

7.6. Recycling

The solution approach is flexible for recycling long term secrets like POLY-1. All the nodes could be periodically updated with shares of new SECRET as best practice. The table above could be consulted for refresh cycles (see Section 4).
If symmetric masking is used for OPOT (Section 3.5), mask values must be periodically updated as well, at least as frequently as the other secrets are.

7.7. Redundant Nodes and Failover

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

7.8. Controller Operation

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

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

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

7.9. Verification Scope

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

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

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

7.9.2. Stealth Nodes

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

8. Acknowledgements

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

9. Contributors

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

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

John Leddy  
Email: john@leddy.net
10. References

10.1. Normative References


10.2. Informative References

[I-D.ietf-anima-autonomic-control-plane]

Authors’ Addresses

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

Shwetha Bhandari (editor)
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India
Email: shwethab@cisco.com

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

Sashank Dara
Seconize
BANGALORE, Bangalore, KARNATAKA
INDIA
Email: sashank@seconize.co
Stephen Youell
JP Morgan Chase
25 Bank Street
London   E14 5JP
United Kingdom

Email: stephen.youell@jpmorgan.com