SFC
Internet-Draft
Intended status: Standards Track
Expires: April 7, 2016

R. Penno
C. Pignataro
C. Yen
E. Wang
K. Leung
Cisco Systems
October 05, 2015

Packet Generation in Service Function Chains
draft-penno-sfc-packet-02

Abstract

Service Functions (e.g., Firewall, NAT, Proxies and Intrusion Prevention Systems) generate packets in the reverse flow direction to the source of the current in-process packet/flow. In this document we discuss and propose how to support this required functionality within the SFC framework.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Service Functions (e.g., Firewall, NAT, Proxies and Intrusion Prevention Systems) generate packets in the reverse flow direction destined to the source of the current in-process packet/flow. This is a basic intrinsic functionality and therefore needs to be supported in a service function chaining deployment.

2. Problem Statement

The challenge of this functionality in service chain environments is that generated packets need to traverse in the reverse order the same Service Functions traversed by original packet that triggered the packet generation.

Although this might seem to be a straightforward problem, on further inspection there are a few interesting challenges that need to be solved. First and foremost a few requirements need to be met in order to allow a packet to make its way through back to its source through the service path:

- o A symmetric path ID needs to exist. Symmetric path is discussed in [SymmetricPaths]

- o The SF needs to be able encapsulate such error or proxy packets in a encapsulation transport such as VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe] + NSH header [I-D.ietf-sfc-nsh]

- o The SF needs to be able to determine, directly or indirectly, the symmetric path ID and associated next service-hop index or, alternatively, indicate reverse path for the service path ID in the original packet

3. Definitions and Acronyms

The reader should be familiar with the terms contained in [I-D.ietf-sfc-nsh] ,[I-D.ietf-sfc-architecture] and [I-D.ietf-nvo3-vxlan-gpe]

4. Assumptions

We make the following assumption throughout this document

1. An SF could be connected to more than one SFF directly. In other words, a SF can be multi-homed and each connection can use different encapsulations.

2. After forwarding a packet to an SF, the SFF always has connectivity to the next hop SFF to complete the path. This means the following scenario is not possible (SFF2 cannot complete the forward path which contains SFF3 and potentially SFs connected to SFF3

```
                .-.           .-.
               /   \         /   \
              ( SF1 )       ( SF2 )
               \   /         \   / \
                `+'           `+'   \
                 |             |     \
                 |             |      \
              +--+---+      +--+---+   \+------+
        ...---+ SFF1 +------+ SFF2 |    | SFF3 +---...
              +------+      +--+---+    +------+
                               |
                               |
                               +-----...
```

RSFP Forward -> SFF1 : SF1 : SFF1 : SFF2 : SF2 : SFF3 : ...

3. In the figure below, if SF2 is directly connected to SFF2A and SFF2B, there could be a case that SFF2A only has the forwarding rules for the forward path, and SFF2B only has the forwarding rules for the reverse path

```
                .-.             .-.             .-.
               /   \           /   \           /   \
              ( SF1 )         ( SF2 )         ( SF3 )
               \   /\          \   /\          \   /\
                `+'  \          `+'  \          `+'  \
                 |    \          |    \          |    \
                 |     |         |     |         |     |
             +---+---+ |     +-------+ |     +---+---+ |
       ...---+ SFF1A +-|-----+ SFF2A +-|-----+ SFF3A +-|---...
             +-------+ |     +-------+ |     +-------+ |
                       |               |               |
                   +---+---+       +---+---+       +---+---+
             ...---+ SFF1B +-------+ SFF2B +-------+ SFF3B +-----...
                   +-------+       +-------+       +-------+
```

```
       Symmetric Paths:

       RSFP Forward -> SFF1A : SF1 : SFF1A : SFF2A : SF2 :
                       SFF2A : SFF3A : SF3 : SFF3A ...
       RSFP Reverse <- SFF1B : SF1 : SFF1B : SFF2B : SF2 :
                       SFF2B : SFF3B : SF3 : SFF3B


       Asymmetric Paths (skipping SF2 on reverse):

       RSFP Forward -> SFF1A : SF1 : SFF1A : SFF2A : SF2 : SFF2A :
                       SFF3A : SF3 : SFF3A ...
       RSFP Reverse <- SFF1B : SF1 : SFF1B         :       SFF2B :
                       SFF3B : SF3 : SFF3B
```

4. Assumption #2 allows an SF to always bounce a packet back to the SFF that originally sent the packet. Due to #3, an SF has to determine which SFF to send the generated packet to. It cannot treat generated packet the same way as forwarded packet, as in #2.

These assumptions make sense for certain implementation. However, some implementations may not have the constraints in #3, which will simplify the SF logic in handling generated traffic. The 3 assumptions can be illustrated below. The SFF "A"s only have knowledge for the forward path, and SFF "B"s only have knowledge for the reverse path. When SF2 generates a packet in the reverse direction, SF2 must determine which SFF ('A' or 'B') should receive the packet.

5. Service Function Behavior

When a Service Function wants to send packets to the reverse direction back to the source it needs to know the symmetric service path ID (if it exists) and associated service index. This information is not available to Service Functions since they do not need to perform a next-hop service lookup. There are four recommended approaches to solve this problem and we assume different implementations might make different choices.

1. The SF can receive service path forwarding information in the same manner a SFF does.

2. The SF can send the packet in the forward direction but set appropriate bits in the NSH header requesting a SFF to send the packet back to the source

3. The classifier can encode all information the SF needs to send a reverse packet in the metadata header

4. The controller uses a deterministic algorithm when creating the associated symmetric path ID and service index.

We will discuss the ramifications of these approaches in the next sections.

5.1. SF receives Reverse Forwarding Information

This solution is easy to understand but brings a change on how traditionally service functions operate. It requires SFs to receive and process a subset of the information a SFF does. When a SF wants to send a packet to the source, the SF uses information conveyed via the control plane to impose the correct NSH header values.

Advantages:

- Changes are restricted to SF and controller, no changes to SFF

- Incremental deployment possible

- No protocol between SF and SFF, which avoids interoperability issues

- No performance penalty on SFF due to in or out-of-band protocol

Disadvantages:

o SFs need to process and understand Rendered Service Path messages from controller

This solution can be characterized by putting the burden on the SF, but that brings the advantage of being self-contained (as well as providing a mechanism for other features). Also, many SFs have policy or classification function which in fact makes them a classifier and SF combination in practice.

5.2. SF requests SFF cooperation

These solutions can be characterized by distributing the burden between SF and SFF. In this section we discuss two possible in-band solutions: using OAM header and using a reserved bit 'R' in the NSH header.

5.2.1. OAM Header

When the SF needs to send a packet in the reverse direction it will set the OAM bit in the NSH header and use an OAM protocol [I-D.penno-sfc-trace] to request that the SFF impose a new, reverse path NSH header. Post imposition, the SFF forwards the packet correctly.

SF Reverse Packet Request

```
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+\
 |Ver|1|C|R|R|R|R|R|R|   Length  |  MD-type=0x1  |  OAM Protocol | |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
 |          Service Path ID                      | Service Index | |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
 |                Mandatory Context Header                       | |S
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |F
 |                Mandatory Context Header                       | |C
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
 |                Mandatory Context Header                       | |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
 |                Mandatory Context Header                       | |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ <
 |Rev. Pkt Req   |         Original NSH headers (optional)       | |O
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |A
                                                                  |M
                                                                 /
```

(postamble)

Ver: 1

OAM Bit: 1

Length: 6

MD-Type: 1

Next Protocol: OAM Protocol

Rev. Pkt Req: 1 Reverse packet request

Advantages:

- o SF does not need to process and understand control plane path messages.
- o Clear division of labor between SF and SFF.
- o Extensible
- o Original NSH header could be carried inside OAM protocol which leaves metadata headers available for SF-SFF communication.

Disadvantages:

- o SFFs need to process and understand a new OAM message type
- o Possible interoperability issues between SF-SFF
- o SFF Performance penalty

5.2.2. Service Function Forwarder Behavior

In the case where the SF has all the information to send the packet back to the origin no changes are needed at the SFF. When an SF requests SFF cooperation the SFF MUST be able to process the OAM message used to signal reverse path forwarding.

- o Process/decode OAM message
- o Examine and act on any metadata present in the NSH header
- o Examine its forwarding tables and find the reverse path-id and index of the next service-hop

The reverse path can be found in the Rendered Service Path Yang model [RSPYang] that conveyed to the SFF when a path is constructed.

If a SFF does not understand the OAM message it just forwards the packet based on the original path-id and index. Since it is a special OAM packet, it tells other SFFs and SFs that they should process it differently. For example, a downstream intrusion detection SF might not associate flow state with this packet.

5.2.3. Reserved bit

In this solution the SF sets a reversed bit in the NSH that carries the same semantic as in the OAM solution discussed previously. This solution is simpler from a SF perspective but requires allocating one of the reserved bits. Another issue is that the metadata in the original packet might be overwritten by SFs or SFFs in the path.

When a SFF receives a NSH packet with the reversed bit set, it shall look up a preprogrammed table to map the Service Path ID and Index in the NSH packet into the reverse Service Path ID and Index. The SFF would then use the new reverse ID and Index pair to determine the SF/SFF which is in the reverse direction.

Advantages:

- No protocol header overhead

- Limited performance impact on SF

Disadvantages:

- Use of a reserved bit

- SFF Performance penalty

- Not extensible

5.3. Classifier Encodes Information

This solution allows the Service Function to send a reverse packet without interactions with the controller or SFF, therefore it is very attractive. Also, it does not need to have the OAM bit set or use a reserved bit. The penalty is that for a MD Type-1 packet a significant amount of information (48 bits) need to be encoded in the metadata section of the packet and this data can not be overwritten. Ideally this metadata would need to be added by the classifier.

The Rendered Service Path yang model [RSPYang] already provides all the necessary information that a classifier would need to add to the metadata header. An explanation of this method is better served with an examples.

5.3.1. Symmetric Service Paths

The figure below shows a simple SFC with symmetric service paths comprising three SFs.

(preamble)

```
.....................SFP2 Forward........................>

  Forward SI  253          252          251

+---+         .-.          .-.          .-.             +---+
|   |        /   \        /   \        /   \            |   |
| A +-------( SF1 )------( SF2 )------( SF3 )----------+ B |
|   |        \   /        \   /        \   /            |   |
+---+         `-'          `-'          `-'             +---+

  Reverse SI  253          254          255

 <....................SFP3 (Reverse of SFP2)........................


  SFP2 Forward ->  SF1 : SF2 : SF3
  SFP3 Reverse <-  SF1 : SF2 : SF3

  RSP2 Forward -> SF1 : SF2 : SF3
  RSP3 Reverse <- SF1 : SF2 : SF3
```

Figure 1: SFC example with symmetric path

Below we see the JSON objects of the two symmetric paths depicted above.


```
RENDERED_SERVICE_PATH_RESP_JSON = """
{
  "rendered-service-paths": {
    "rendered-service-path": [
      {
        "name": "SFC1-SFP1-Path-2-Reverse",
        "transport-type": "service-locator:vxlan-gpe",
        "parent-service-function-path": "SFC1-SFP1",
        "path-id": 3,
        "service-chain-name": "SFC1",
        "starting-index": 255,
        "rendered-service-path-hop": [
          {
            "hop-number": 0,
            "service-index": 255,
```

```
            "service-function-forwarder-locator": "eth0",
            "service-function-name": "SF3",
            "service-function-forwarder": "SFF3"
          },
          {
            "hop-number": 1,
            "service-index": 254,
            "service-function-forwarder-locator": "eth0",
            "service-function-name": "SF2",
            "service-function-forwarder": "SFF2"
          },
          {
            "hop-number": 2,
            "service-index": 253,
            "service-function-forwarder-locator": "eth0",
            "service-function-name": "SF1",
            "service-function-forwarder": "SFF1"
          }
        ],
        "symmetric-path-id": 2
      },
      {
        "name": "SFC1-SFP1-Path-2",
        "transport-type": "service-locator:vxlan-gpe",
        "parent-service-function-path": "SFC1-SFP1",
        "path-id": 2,
        "service-chain-name": "SFC1",
        "starting-index": 253,
        "rendered-service-path-hop": [
          {
            "hop-number": 0,
            "service-index": 253,
            "service-function-forwarder-locator": "eth0",
            "service-function-name": "SF1",
            "service-function-forwarder": "SFF1"
          },
          {
            "hop-number": 1,
            "service-index": 252,
            "service-function-forwarder-locator": "eth0",
            "service-function-name": "SF2",
            "service-function-forwarder": "SFF2"
          },
          {
            "hop-number": 2,
            "service-index": 251,
            "service-function-forwarder-locator": "eth0",
            "service-function-name": "SF3",
```

```
               "service-function-forwarder": "SFF3"
             }
           ],
           "symmetric-path-id": 3
         }
       ]
     }
   }"""
```


We will assume the classifier will encode the following information in the metadata:

- symmetric path-id = 2 (24 bits)

- symmetric starting index = 253 (8 bits)

- symmetric number of hops = 3 (8 bits)

- starting index = 255 (8 bits)

In the method below we will assume SF will generate a reverse packet after decrementing the index of the current packet. We will call that current index.

If SF1 wants to generate a reverse packet it can find the appropriate index by applying the following algorithm:

```
current_index = 252

remaining_hops = symmetric_number_hops - starting_index - current_index
remaining_hops = 3 - (255 - 252) = 0
reverse_service_index = symmetric_starting_index - remaining_hops - 1
reverse_service_index = next_service_hop_index = 253 - 0 - 1 = 252
The "-1"  is necessary for the service index to point to the next service_hop.
```

If SF2 wants to send reverse packet:

```
current index = 253

remaining_hops = 3 - (255 - 253) = 1
reverse_service_index = next_service_hop_index = 253 - 1 - 1 = 251
```

IF SF3 wants to send reverse packet:

```
current index = 254

remaining_hops = 3 - (255 - 254) = 2
reverse_service_index = next_service_hop_index = 253 - 2 - 1 = 250
```

The following tables summarize the service indexes as calculated by each SF in the forward and reverse paths respectively.

(preamble)

Fwd SI = forward Service Index
Cur SI = Current Service Index
Gen SI = Service Index for Generated packets

RSFP1 Forward -
Number of Hops: 3
Forward Starting Index: 253
Reverse Starting Index: 255

SF	SF1	SF2	SF3
Fwd SI	253	252	251
Cur SI	252	251	250
Gen SI	252	253	254

RSFP1 Reverse -
Number of Hops: 3
Reverse Starting Index: 255
Forward Starting Index: 253

SF	SF1	SF2	SF3
Rev SI	253	254	255
Cur SI	252	253	254
Gen SI	252	251	250

Figure 2: Service indexes generated by each SF in the symmetric forward and reverse paths

5.3.2. Analysis

Advantages:

o SF does not need to request SFF cooperation or contact controller

- No SFF performance impact

Disadvantages:

- Metadata overhead in case MD-Type 2 is used

- Relies on classifier or SFF to encode metadata information

- If classifier will encode information it needs to receive and process rendered service path information

- SFF needs to decrement NOP associated indexes

5.4. Algorithmic Reversed Path ID Generation

In these proposals no extra storage is required from the NSH and SFF does not need to know how to handle the reversed packet nor does it know about it. Reverse Path is programmed by Orchestrator and used by SF having the need to send upstream traffic.

5.4.1. Same Path-ID and Disjoint Index Spaces

Instead of defining a new Service Path ID, the same Service Path ID is used. The Orchestrator must define the reverse chain of service using a different range of Service Path Index. It is also assumed that the reverse packet must go through the same number of Services as its forward path. It is proposed that Service Path Index (SPI) 1..127 and 255..129 are the exact mirror of each other.

Here is an example: SF1, SF2, and SF3 are identified using Service Path Index (SPI) 8, 7 and 6 respectively.

Path 100 Index 8 - SF1

Path 100 Index 7 - SF2

Path 100 Index 6 - SF3

Path 100 Index 5 - Terminate

At the same time, Orchestrator programs SPI 248, 249 and 250 as SF1, SF2 and SF3. Orchestrator also programs SPI 247 as "terminate". Reverse-SPI = 256 - SPI.

Path 100 Index 247 - Terminate

Path 100 Index 248 (256 - 8) - SF1

Path 100 Index 249 (256 - 7) - SF2

Path 100 Index 250 (256 - 6) - SF3

If SF3 needs to send the packet in reverse direction, it calculates the new SPI as 256 - 6 (6 is the SPI of the packet) and obtained 250. It then subtract the SPI by 1 and send the packet back to SFF

Subsequently, SFF received the packet and sees the SPI 249. It then diverts the packet to SF2, etc. Eventually, the packet SPI will drop to 247 and the SFF will strip off the NSH and deliver the packet.

The same mechanism works even if SF1 later decided to send back another upstream packet. The packet can ping-pong between SF1 and SF3 using existing mechanism.

Advantages:

- No precious NSH area is consumed

- SF self-contained solution

- No SFF performance impact and no cooperation needed

- No Special Classification required

Disadvantages:

- SPI range is reduced and may become incompatible with existing topology

- Assumption that the reverse path Service Functions are the same as forward path, only in reverse

- Reverse paths need to use Service Index = 128 for loop detection instead of SI = 0.

In either case, the SF must have the knowledge through Orchestrator that the reverse path has been programmed and the method (SPI only or SPI + SPID bit) to use.

The symmetrization mechanism keep reverse path symmetric as described in section 6 can be applied in this method as well.

5.4.2. Flip Path-Id and Index High Order bits

An alternative to reducing Service Path Index range is to make use of a different Service Path ID, e.g. the most significant bit. The bit can be flipped when the SF needs to send packet in reverse. However, the negation of the SPI is still required, e.g. SPI 6 becomes SPI 134

This approach is fully compatible with the current NSH protocol standard and provides a fully deterministic way of determining reverse paths. It the recommended approach.

6. Asymmetric Service Paths

In real world the forward and reverse paths can be asymmetric, comprising different set of SFs or SFs in different orders. The following figure illustrates an example. The forward path is composed of SF1, SF2, SF4 and SF5, while the reverse path skips SF5 and has SF3 in place of SF2.

```
                    ..........        .........
                   .          .      .         .
                  .     249    .    .    246    .
                 .              .  .             .
                .       .-.      ..      .-.      .
  ..............       /   \            /   \      ....SFP1 Forward....>
                      ( SF2 )   247    ( SF5 )
  Forward SI   250   / \   / \        / \   /\
                    /   `-'   \      /   `-'  \
                   /           \    /          \
 +---+         .-./             `-./            \               +---+
 |   |        /   \            /   \             \              |   |
 | A +-------( SF1 )----------( SF4 )-------------+-------------+ B |
 |   |        \   /            \   /                            |   |
 +---+         `-'\             ,-'                             +---+
                   \           /
                    \   .-.   /
   Reverse SI  251   \ /   \ /  254
  <...........        ( SF3 )         .................SFP2 Reverse.....
              .        \   /         .
               .        `-'         .
                .                  .
                 .                .
                  .     253      .
                   ..............
```

SFP1 Forward -> SF1 : SF2 : SF4 : SF5
SFP2 Reverse <- SF1 : SF3 : SF4

Figure 3: SFC example with asymmetric paths

An asymmetric SFC can have completely independent forward and reverse paths. An SF's location in the forward path can be different from that in the reverse path. An SF may appear only in the forward path but not reverse (and vice-versa). In order to use the same algorithm to calculate the service index generated by an SF, one design option is to insert special NOP SFs in the rendered service paths so that each SF is positioned symmetrically in the forward and reverse rendered paths. The SFP corresponding to the example above is:

SFP1 Forward -> SF1 : SF2 : NOP : SF4 : SF5

SFP2 Reverse <- SF1 : NOP : SF3 : SF4 : NOP

The NOP SF is assigned with a sequential service index the same way as a regular SF. The SFF receiving a packet with the service path ID and service index corresponding to a NOP SF should advance the

service index till the service index points to a regular SF. Implementation can use a loopback interface or other methods on the SFF to skip the NOP SFs.

Once the NOP SF is inserted in the rendered service paths, the forward and reverse paths become symmetric. The same algorithm can be applied by the SFs to generate service indexes in the opposite directional path. The following tables list the service indexes corresponding to the example above.

Fwd SI = forward Service Index
Cur SI = Current Service Index
Gen SI = Service Index for Generated packets

RSP1 Forward -
Number of hops: 5
Forward Starting Index: 250
Reverse Starting Index: 255

SF	SF1	SF2	NOP	SF4	SF5
Fwd SI	250	249	248	247	246
Cur SI	249	248	247	246	245
Gen SI	250	251	N/A	253	254

RSP1 Reverse -
Number of hops: 5
Reverse Starting Index: 255
Forward Starting Index: 250

SF	SF1	NOP	SF3	SF4	NOP
Rev SI	251	252	253	254	255
Cur SI	250	251	252	253	254
Gen SI	249	N/A	247	246	N/A

This symmetrization of asymmetric paths could be performed by a controller during path creation.

7. Metadata

A crucial consideration when generating a packet is which metadata should be included in the context headers. In some scenarios if the metadata is not present the packet will not reach its intended destination. Although one could think of many different ways to convey this information, we believe the solution should be simple and require little or no new Service Function functionality.

We assume that a Service Function normally needs to know the semantics of the context headers in order to perform its functions. But clearly knowing the semantics of the metadata is not enough. The issue is that although the SF knows the semantics of the metadata when it receives a packet, it might not be able to generate or retrieve the correct metadata values to insert in the context headers when generating a packet. It is usually the classifier that insert the metadata in the context headers.

In order to solve this problem we propose the notion of service-path-invariant metadata. This is metadata that is the same for all packets traversing a certain path. For example, if all packets exiting a service-path need to be routed to a certain VPN, the VPN id would be a path-invariant metadata. Since the controller needs to send the semantics of the metadata present in the context headers to each Service Function, it is straightforward to send along the values of the path-invariant metadata. Therefore when the Service Function generates a packet in can insert the minimum required metadata for a packet to reach its destination.

There is a second type of metadata that the Service Function can provide the appropriate values, the one that it would be responsible for inserting anyway as part of packet processing.

Finally if the packet needs crucial metadata values that can not be supplied by the two methods above then a reclassification is needed. This reclassification would need to be done by the classifier that would normally process packets in the reverse path or a SFF that had the same rules and capabilities. Ideally the first SFF that processes the generated packet.

8. Other solutions

We explored other solution that we deemed to complex or that would bring a severe performance penalty:

- o An out-of-band request-response protocol between SF-SFF. Given that some service functions need to be able to generate packets quite often this will would create a considerable performance

penalty. Specially given the fact that path-ids (and their symmetric counterpart) might change and SF would not be notified, therefore caching benefits will be limited.

- An out-of-band request-response protocol between SF-Controller. Given that admin or network conditions can trigger service path creation, update or deletions a SF would not be aware of new path attributes. The controller should be able to push new information as it becomes available to the interested parties.

- SF (or SFF) punts the packet back to the controller. This solution obviously has severe scaling limitations.

9. Implementation

The solutions "Reversed Path derived using Forward Path ID and Index Method" and "SF receives Reverse Forwarding Information" were implemented in Opendaylight

10. IANA Considerations

TBD

11. Security Considerations

12. Acknowledgements

Paul Quinn, Jim Guichard

13. Changes

14. References

14.1. Normative References

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <http://www.rfc-editor.org/info/rfc2616>.

14.2. Informative References

[I-D.ietf-nvo3-vxlan-gpe]
Quinn, P., Manur, R., Kreeger, L., Lewis, D., Maino, F., Smith, M., Agarwal, P., Yong, L., Xu, X., Elzur, U., Garg, P., and D. Melman, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-00 (work in progress), May 2015.

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-11 (work in progress), July 2015.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-01 (work in progress), July 2015.

[I-D.penno-sfc-trace]
Penno, R., Quinn, P., Pignataro, C., and D. Zhou, "Services Function Chaining Traceroute", draft-penno-sfc-trace-03 (work in progress), September 2015.

[I-D.penno-sfc-yang]
Penno, R., Quinn, P., Zhou, D., and J. Li, "Yang Data Model for Service Function Chaining", draft-penno-sfc-yang-13 (work in progress), March 2015.

[RSPYang] Opendaylight, , "Rendered Service Path Yang Model", February 2011, <https://github.com/opendaylight/sfc/blob/master/sfc-model/src/main/yang/rendered-service-path.yang>.

[SymmetricPaths]
IETF, , "Symmetric Paths", February 2011, <https://tools.ietf.org/html/draft-ietf-sfc-architecture-11#section-2.2>.

Authors' Addresses

Reinaldo Penno
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: repenno@cisco.com

Carlos Pignataro
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: cpignata@cisco.com

Chui-Tin Yen
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: tin@cisco.com

Eric Wang
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: ejwang@cisco.com

Kent Leung
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: kleung@cisco.com