Internet Draft
SFC Working Group
Category: Informational
Expires: December 30, 2016

Anil Kumar S N
Gaurav Agrawal
Vinod Kumar S
Huawei Technologies
C. Jacquenet
Orange

June 28, 2016

Performance Measurement Architecture for SFC
draft-agv-sfc-performance-measurement-architecture-02

Abstract

This document describes performance measurement(PM) architecture for Service Function Chains (SFCs) to assess the operational status and behavior of a service function, a subset of service function’s, a whole service function chain as a function of the actual deterministic service function / service function chain. This document does not propose solutions, protocols, nor any extension to existing protocols.

Status of this Memo

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

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

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

This Internet-Draft will expire on December 30, 2016

Table of Contents

1. Introduction

The delivery of end-to-end services often requires various service functions. These include traditional network service functions such as firewalls and traditional IP network address translators (NATs), as well as application-specific functions. The definition and instantiation of an ordered set of service functions and subsequent "steering" of traffic through them is termed service function chaining (SFC).

This document describes an architecture used for assessing and monitoring SFC operation. It includes architectural concepts, principles, and components, with a focus on assessing the operational status and behavior of a SF, a subset of SFs, a whole SFC as a function of the actual deterministic SF/SFC.

Proper operation of a SFC depends on the ability to monitor and quickly identify faults and focus attention on the root cause of the problem. To achieve this we need to monitor the performance parameters of SFC like packet loss, delay, jitter etc.

For instance a SFC is composed of diffServ (e.g., traffic conditioning capabilities) and NAT functions, with OOP as a traffic type. To ensure the proper operation of a OOP it’s required to ensure that traffic is properly re-marked before invoking the NAT function which selects a specific pool for such OOP traffic. Incorrect re-marking of this traffic will lead to performance deterioration in terms of increase in one way delay.

For another instance a SFC is composed of firewall which is supposed to controls the incoming and outgoing network traffic based on predetermined security rules. To ensure the proper functioning of this SF it’s required to ensure that a specific set of traffic must be dropped and another specific set of traffic must be allowed to pass.

Continuous measurement and monitoring of packet delay/loss for a SF, a subset of SFs, a whole SFC will help to find the problem. Also performance measurement results can be used to locate the root cause. which inturn can also be used to possibly take required action as far as SFC structuring is concerned.

Service provider’s service level agreements (SLAs) usually include performance indicators like packet loss or inter-packet delay variation that need to be measured over a given period of time. SFC Performance measurement enables operators with greater visibility into the performance characteristics of their SFCs, thereby ensuring service SLAs.

SFC is at service layer which is independent of transport layer technology. For a SFC underlying layer can be a mix of multiple transport layer tunneling technology like MPLS, VXLAN etc. SLA measurements of independent transport layer will not be capable to assess the operational status and behavior of a SF, a subset of SFs, a whole SFC as a function of the actual deterministic SF/SFC which leads to a requirement for performance measurement architecture specific to service function chain.

Transport layer PM and SFC PM can go hand in hand, SFC PM can be used to obtain the exact segment of problem if that segment is one of the Transport layer Tunnel, transport layer PM can be used to locate exact fault location.

Requirements of SFC performance measurement architecture includes:

1) Localization of performance degradation occurred due to a unit in a service function chain.

2) Localization of performance degradation for a specific service traffic type

3) Enable service providers with a tool to offer high value services whose performance degradation can be proactively located and corrected.

4) ECMP scenarios can result in out of order packets, performance measurement should consider this while measurement.

5) Validation of service function chain performance at the time of deployment/fault correction verification (using probe packets)

6) Runtime Validation of service function chain performance using traffic packets itself to get accurate performance data without additional probe packets.

7) Performance measurement under induced traffic level. The motivation for this can be related to the rate and the amount of traffic that can influence the accuracy of the measurement results

8) Optimized measurement by keeping minimal performance measurement load.

Following capabilities are required to be supported by performance measurement architecture to address above requirements.

1) Capability to assess and monitor at
   a) SF
   b) SFF
   c) Set of SF/SFF
   d) Segment(s) between any two SF/SFF in a SFC.
   e) SFC as a whole.

2) Capability to measure performance for fine-grained and coarse-grained flow.

3) Capability for Continuous/proactive & selective/on-demand measurement.

4) Support for all three measurement methods:

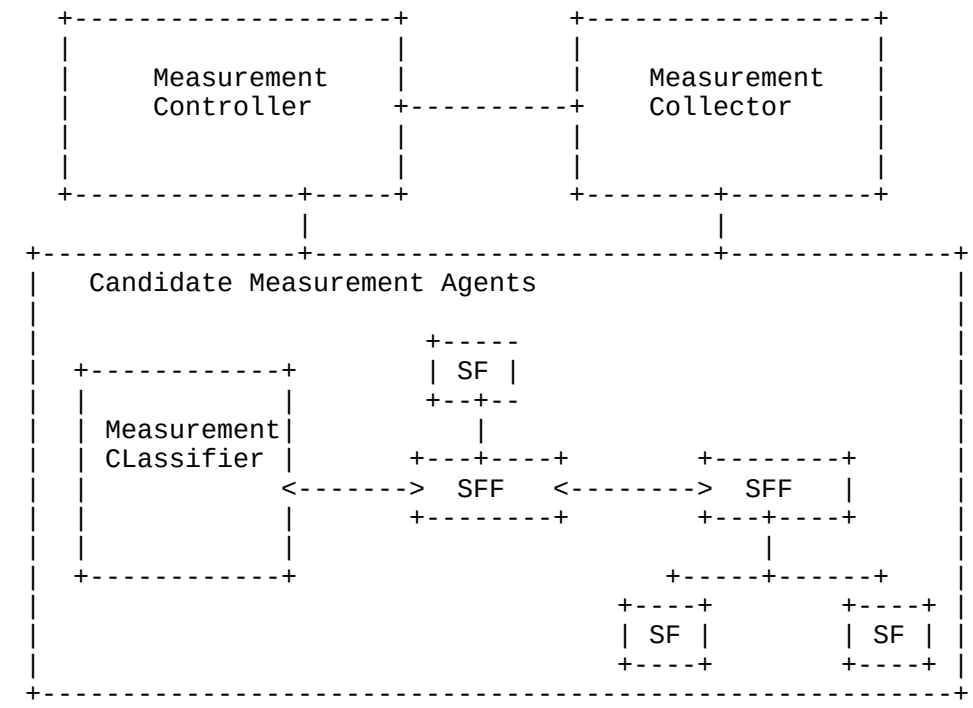
   Active Measurement method:
   Active method measures performance and other parameters by injecting test traffic into the network from specific locations (e.g., a Point of Presence).

   Passive measurement method:
   Passive method measures performance and other parameters based on the observation of live traffic forwarded in the network.

   Hybrid measurement method:
   Hybrid measurement method uses both active and passive measurement methods.

5) Capability to measure performance even in case of out of order packets.

2. Architecture Overview



The SFC performance measurement architecture relies upon the following components:

Measurement Controller: An entity that controls and coordinates a set of measurement functions. The measurement controller is responsible for programming the performance measurement instance at measurement classifier and updating the same to measurement collector, additionally it can inject probe packets in SFP for measurement.

Measurement Collector: An entity that collects measurement data from a measurement agents (MAs). A measurement collector functionality could be integrated with one of the MAs deployed in the network or with the measurement controller itself.

Measurement Classifier: An entity responsible for encoding measurement control information based on PMI programmed by measurement controller. SFC Classifier MUST incorporate the measurement classifier functionality to support SFC performance measurement.

Measurement Agent: An entity that supports one or more Measurement functions. It is responsible for performance data collection and reporting the same to Measurement collector.

The following elements MUST incorporate the MA functionality to support SFC performance measurement.

- o SF
- o SFF
- o Classifier
- o NSH Proxy Agent.

3 PM Measurements Methods:

a) Active measurement : Measurement controller induce probe packets with encoded performance measurement data or programs PMI at measurement classifier which will in turn induce probe packets with encoded performance measurement data. measurement controller updates the PMI to measurement collector. Participating measurement agents based on received encoded information carry out measurements and reports the collected data to measurement collector. measurement collector co-relates received data and provides measurement results.

   Note: Current draft only covers the Measurement attributes programming for probe packets. Construction of base probe packet is outside the scope of this RFC.

b) Passive measurement : Measurement controller programming PMI at measurement classifier and updating the same to measurement collector. measurement classifier encodes the performance measurement data in classified packets. Participating measurement agents based on received encoded information carry out measurement and reports the collected data to measurement collector. measurement collector co- relates received data and provides measurement results.

c) Hybrid measurement : Measurement controller induce probe packets to pre-setup load condition for a SFC using active measurement. measurement controller than uses passive measurement to carry out performance measurement under load condition.

4 Performance Measurement Attributes

There are two types of PM attributes

a) Metadata Attributes: These attributes are to instruct the MA to carry out the required performance measurement. Measurement classifier encodes these attributes in the traffic packets based on instructions provided by controller or these attributes are encoded directly in test packet by measurement controller/measurement classifier.

b) PM reporting attributes: These attributes are used by MA(s) to report the collected performance data to Collector

4.1 Metadata Attributes

The following attributes must be encoded as metadata for selected traffic or probe packets

- PMF Identifier
- Window Identifier
- List of MA(s) identifier with PM Type

4.2 PM Reporting Attributes

The following Information should be sent by each MA to the collector:

- PMF identifier
- Window identifier
- MA identifier with PM type
- Performance statistics

4.3 Performance Measurement Attributes Description

Several performance attributes are introduced in this memo to carry performance control information. Each attribute has a unique purpose; they are interpreted based upon a specific context and the corresponding performance measurement is performed by the respective MA.

4.3.1 PMF Identifier

Purpose : To unambiguously identify a measurement flowin the SFC Domain. When a MA reports its performance data to collector, collector has to associate the data to a particular instance of measurement, as controller would have created multiple instance of measurements. By using service path index + service index only coarse-grained flows can be identified. To identify a fine-grained flows (Ex: SFC classification is based on destination IP, so flow classified based on this destination IP is coarse-grained flow. Sub flow corresponding to a particular destination IP within this flow is an example of fine-grained flows) within a SFC domain SPI + SI is not enough. PMF identifier is used to uniquely identify the performance measurement instance corresponding to fine grained flows.

Value : Unique value within current SFC domain

Processing :

- At Classifier : The PMF Identifier is encoded in metadata
  Note: For probe packets this information is encoded by entity (controller/classifier) constructing probe packet.

- At MA : Used as a key while collecting, maintaining & reporting PM statistics to collector.

- At Collector : Collector correlates the performance data received from a MA using this PMF Identifier.

4.3.2 Window Identifier

Purpose : Window identifier is to handle out of order packet scenario.

Example:

```
SFC classifier-----------SF1-------------SF2-------------SF3
Packet 1-->-------------------------------------------->Packet 2
Packet 2-->-------------------------------------------->Packet 1
```

Packet 1 belongs to earlier cycle of measurement but SF3 receives Packet 2 before packet 1 due to out of order problem. Now, when this data is co-related at Collector, if there is no window identifier, packet 2 will be treated as packet 1 and result will be incorrect. Window identifier is of local scope to a MA.

Sender (Classifier/Controller) divides the flows into multiple windows. Packets with PM information in a window will have the

same window identifier and consecutive windows will have a different identifier. This enables MA to collect & accumulate statistics corresponding to each window & report it to collector. Size of the window is programmable.

Value : Integer (Max/Min Value: Programmable to context header at classifier, once Max value reached then value = Min Value). Value increments with each PM interval.

Processing :
- At Classifier : The window identifier is encoded in NSH context header.

- At MA : Used as a key while collecting, maintaining & reporting PM statistics to collector.

- At Collector : Collector correlates the performance data received from MA by using this window identifier.

4.3.3 Measurement Agents Identifier with PM type

Purpose : To identify participating measurement agents with context and type of performance measurement.

Value : Measurement agent for a given SFP is identified using MA identifier + service index

MA identifier has two parts
1) Node identifier - 24 bit
   a) For SFF: MUST be unique number assigned by controller
   b) For SF: All zero. Context identifier itself identifies SF node.
2) Order identifier - 8 bit
   a) For SFF: Service index of next SF.
   b) For SF: Service index

MA identifier SHOULD be in decreasing order of the SI for optimized traversal of the SI participation.

PM Type (IANA assigned values)

Processing :
- At Classifier : Encode the participating MA(s) with PM type.

- At MA : Presence of self service index triggers the PM collection & reporting

- At Collector : Identifies the reporting MA

4.3.4 Performance Statistics

Purpose : Computation of performance.
Value : Collected statistics

Processing :
- At Classifier : None.
- At MA : Depends on performance measurement type

  For packet loss: Accumulates received & sent packets counter for a given flow + window and reports it to collector

  For packet delay: Record the time for sending and receiving a packet of a given flow + window and reports it to collector.

- At Collector : Correlates and maintains received data

5 Measurement Controller Operation

The Measurement Controller has the following responsibilities:

1) Programs the unique MA identifier for SF/SFF/SFC proxy in SFC domain.
2) Programs the PM instance at the classifier

   PM Instance MUST contain the following information:
   a) PM flow classification rule with PMF identifier (unique across the SFC domain). For coarse grained flow PM flow classification rule is optional.
   b) List of participating MA with PM type.
   c) Window size and PM schedule.

3) Provides the PM Instance to the collector (if required)
4) Ensures the uniqueness of MA identifier & PMF identifier across the SFC domain
5) Programs the reporting interval at the MAs (optional).
6) For active measurement creates and send test probe packet.

6 Measurement Classifier Operation

The classifier classifies packets for the PM instance based on the instructions provided by the controller. In the classified packets it encodes the following information in context header of NSH metadata:

```
PMF identifier      : As programmed by controller
Window identifier   : Locally generated number which changes
                      based on the window size.
MA(s)               : List of MA(s) with PM type
```

For active measurement measurement classifier MAY need to generate probe packet as instructed by measurement controller.
Note: Selection of packets in a flow to participate in a PM is decided by controller and it is outside the scope of this document.

7 MA Operation

MA carries out the following operations upon receipt of a packet with NSH header:

- Detection of PM context header in a packet.
- Processing of context header information.
- Check the presence of self index in context header.
- Identification of the PM type.
- Performance measurement based on the identified type.
  * For packet loss: Accumulates statistics for received & sent packets for a given PMF + window
  * For packet delay: Record the time when the packet was received and sent for a given PMF + window
- Reporting of accumulated statistics at configured interval to the collector. This interval should be consistent across all MAs.

Note:
1) A MA does not maintain the context of the window, the statistical information of a single window can be sent in more than one report, it is the collector’s responsibility to map and accumulate statistics of a window from different reports.
2) If the classifier itself embeds a MA, it also needs to do performance measurement and reporting.

8 Collector Operation

Collector collects data from the MA and performs the following operations for data correlation purposes:

- Collector uses PMF identifier to group statistics related to a given PM flow. Collector maintains the statistics related to a given PM flow from multiple MAs to assess the forwarding performance.

- Collector uses the window identifier to group statistics received from multiple MAs for a single window for a given PM flow.

- Since the MA does not maintain the context of the block interval, statistical information of a single block can be received in more than one report; collector maps and accumulates the statistics of a block from different reports.

- Collector performs the PM computation based on the PM type & statistics collected from the MA; the identification of PM segment is done in the collector, based on the PM types received from the segment end point MAs.

9 Measurement Steps

Step 1: Programming of MA identifier at SF/SFF/SFC proxy by controller. Programming of PM instance at classifier by controller

Step 2: Classifier classifies & select packets for a PM Flow.

Step 3: Classifier encapsulates the PM attributes in PM context header for selected packets.

Step 4: Packet is sent out of classifier.

Step 5: MA receives packet and check presence of PM context header (If context header is not present move to step 9.)

Step 6: MA checks presence of its service index in PM context header (If its own index is not present, then move to Step 9)

Step 7: MA obtains PM type to be carried out and according accumulates PM statistics for a given PMF + window for received and sent packets.

Step 8: MA reports the accumulated PM statistics to collector at reporting intervals.

Step 9: Regular packet processing continues.

Step 5 to step 9 is repeated at every MA.

10 Hop by Hop Performance Measurement

In case PM needs to be performed on all the SFs invoked along the SFP, as per the described NSH programming in the classifier, all the SFs MA identifier needs to be added in the context header. This will ensure all the SFs/SFFs participate in the PM. This mechanism will require all the SFs/SFFs to traverse the context header until the self SI is found.

Alternatively, we can optimize this process by defining a new context type where all the SFs need to perform the specified PM. Thus, the classifier can skip the MA identifier encoding in the context header, and the MA can skip the MA identifier processing.

Extension for SFF participation in hop-by-hop PM. As mentioned in the previous section, we can define a special context types which means SFF and/or needs participate in the PM.

11 SFC as a whole Performance Measurement

In case PM needs to be performed on the endpoint SFs along the SFP, these 2 MA identifier needs to be encoded in the context header, and will follow the procedure to perform E2E SF’s PM.

In case the PM needs to be performed between the classifier and SFC domain boundary SFF, a special context type will be encoded in the NSH, which needs to be processed by the boundary SFF to report the PM data to the collector and then strip the NSH.

12 Acknowledgements

Special Thanks to Nobin Mathew for his review and comments.

13 Security Considerations

There are no security considerations relevant to this document.

14 IANA Considerations

No actions are required from IANA as result of the publication of this document.

15 References

15.1 Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[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>.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.

[RFC2678] Mahdavi, J. and V. Paxson, "IPPM Metrics for Measuring Connectivity", RFC 2678, September 1999.

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, September 1999.

[RFC2680] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, September 1999.

[RFC3148] Mathis, M. and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics", RFC 3148, July 2001.

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, November 2002.

[RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network performance measurement with periodic streams", RFC 3432, November 2002.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M.Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.

[RFC5226] Narten, T. and H. Alvestrand,"Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC7498] Quinn, P., Ed. and T. Nadeau,Ed.,"Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/

RFC7498, April 2015,
<http://www.rfc-editor.org/info/rfc7498>.

[SFC-arch]
Quinn, P., Ed. and J. Halpern, Ed., "Service Function Chaining (SFC) Architecture", 2014,
<http://datatracker.ietf.org/doc/draft-quinn-sfc-arch>.

Authors' Addresses

Anil Kumar S N
Huawei Technologies India Pvt. Ltd,
Near EPIP Industrial Area,
Kundalahalli Village,
Whitefield,
Bangalore - 560066

EMail: anil.sn@huawei.com

Gaurav Agrawal
Huawei Technologies India Pvt. Ltd,
Near EPIP Industrial Area,
Kundalahalli Village,
Whitefield,
Bangalore - 560066

EMail: gaurav.agrawal@huawei.com

Vinod Kumar S
Huawei Technologies India Pvt. Ltd,
Near EPIP Industrial Area,
Kundalahalli Village,
Whitefield,
Bangalore - 560066

EMail: vinods.kumar@huawei.com

Christian Jacquenet
Orange
Rennes 35000
France

Email: christian.jacquenet@orange.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 9, 2017

F. Brockners
S. Bhandari
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.

July 8, 2016

Data Formats for In-band OAM
draft-brockners-inband-oam-data-00

Abstract

In-band operation, administration and maintenance (OAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document discusses the data types and data formats for in-band OAM data records. In-band OAM data records can be embedded into a variety of transports such as NSH, Segment Routing, VXLAN-GPE, native IPv6 (via extension header), or IPv4. In-band OAM is to complement current out-of-band OAM mechanisms based on ICMP or other types of probe packets.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

Table of Contents

1. Introduction

This document defines data record types for "in-band" operation, administration, and maintenance (OAM). In-band OAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-band" 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. A discussion of the motivation and requirements for in-band OAM can be found in [draft-brockners-inband-oam-requirements]. In-band OAM is to complement "out-of-band" or "active" mechanisms such as ping or traceroute, or more recent active probing mechanisms as described in [I-D.lapukhov-dataplane-probe]. In-band OAM mechanisms can be leveraged where current out-of-band mechanisms do not apply or do not offer the desired results, such as proving that a certain set of traffic takes a pre-defined path, SLA verification for the live data traffic, detailed statistics on traffic distribution paths in networks that distribute traffic across multiple paths, or scenarios where probe traffic is potentially handled differently from regular data traffic by the network devices.

This document defines the data types and data formats for in-band OAM data records. The in-band OAM data records can be transported by a variety of transport protocols, including NSH, Segment Routing, VXLAN-GPE, IPv6, IPv4. Encapsulation details for these different transport protocols are outside the scope of this document.

2. Conventions

Abbreviations used in this document:

MTU: Maximum Transmit Unit

OAM: Operations, Administration, and Maintenance

SR: Segment Routing

SID: Segment Identifier

NSH: Network Service Header

SFC: Service Function Chain

TLV: Type-Length-Value

VXLAN-GPE: Virtual eXtensible Local Area Network, Generic Protocol Extension

3. In-band OAM Data Types and Data Format

This section defines in-band OAM data types and data formats of the data records required for in-band OAM. The different uses of in-band OAM require the definition of different types of data. The in-band OAM data format for the data being carried corresponds to the three main categories of in-band OAM data defined in [draft-brockners-inband-oam-requirements], which are edge-to-edge, per node, and for selected nodes only.

Transport options for in-band OAM data are found in [draft-brockners-inband-oam-transport]. In-band OAM data is defined as options in Type-Length-Value (TLV) format. The TLV format for each of the three different types of in-band OAM data is defined in this document.

In-band OAM is expected to be deployed in a specific domain rather than on the overall Internet. The part of the network which employs in-band OAM is referred to as "in-band OAM-domain". In-band OAM data is added to a packet on entering the in-band OAM-domain and is removed from the packet when exiting the domain. Within the in-band

OAM-domain, the in-band OAM data may be updated by network nodes that the packet traverses. The device which adds in-band OAM data to the packet is called the "in-band OAM encapsulating node", whereas the device which removed the in-band OAM data is referred to as the "in-band OAM decapsulating node". Nodes within the domain which are aware of in-band OAM data and read and/or write or process the in-band OAM data are called "in-band OAM transit nodes". Note that not every node in an in-band OAM domain needs to be an in-band OAM transit node. For example, a Segment Routing deployment might require the segment routing path to be verified. In that case, only the SR nodes would also be in-band OAM transit nodes rather than all nodes.

3.1. In-band OAM Tracing Option

"In-band OAM tracing data" is expected to be collected at every hop that a packet traverses, i.e., in a typical deployment all nodes in an in-band OAM-domain would participate in in-band OAM and thus be in-band OAM transit nodes, in-band OAM encapsulating or in-band OAM decapsulating nodes. The network diameter of the in-band OAM domain is assumed to be known. For in-band OAM tracing, the in-band OAM encapsulating node allocates an array which is to store operational data retrieved from every node while the packet traverses the domain. Every entry is to hold information for a particular in-band OAM transit node that is traversed by a packet. In-band OAM transit nodes update the content of the array. A pointer which is part of the in-band OAM trace data points to the next empty slot in the array, which is where the next in-band OAM transit node fills in its data. The in-band OAM decapsulating node removes the in-band OAM data and process and/or export the metadata. In-band OAM data uses its own name-space for information such as node identifier or interface identifier. This allows for a domain-specific definition and interpretation. For example: In one case an interface-id could point to a physical interface (e.g., to understand which physical interface of an aggregated link is used when receiving or transmitting a packet) whereas in another case it could refer to a logical interface (e.g., in case of tunnels).

The following in-band OAM data is defined for in-band OAM tracing:

- Identification of the in-band OAM node. An in-band OAM node identifier can match to a device identifier or a particular control point or subsystem within a device.

- Identification of the interface that a packet was received on.

- Identification of the interface that a packet was sent out on.

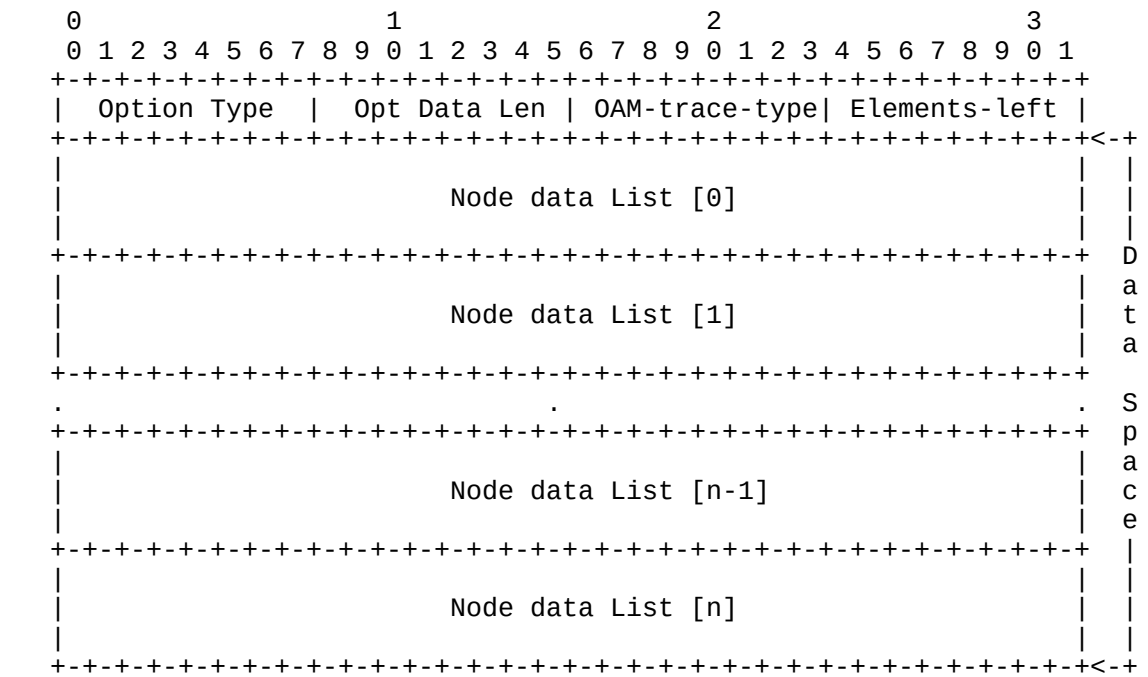
- Time of day when the packet was processed by the node. Different definitions of processing time are feasible and expected, though it is important that all devices of an in-band OAM domain follow the same definition.

- Generic data: Format-free information where syntax and semantic of the information is defined by the operator in a specific deployment. For a specific deployment, all in-band OAM nodes should interpret the generic data the same way. Examples for generic in-band OAM data include geo-location information (location of the node at the time the packet was processed), buffer queue fill level or cache fill level at the time the packet was processed, or even a battery charge level.

- A mechanism to detect whether in-band OAM trace data was added at every hop or whether certain hops in the domain weren't in-band OAM transit nodes.

The "Node data List" array in the packet is populated iteratively as the packet traverses the network, starting with the last entry of the array, i.e., "Node data List [n]" is the first entry to be populated, "Node data List [n-1]" is the second one, etc.

In-band OAM Tracing Option:



Option Type: 8-bit identifier of the type of option. Option number is defined based on the encapsulation protocol.

Opt Data Len: 8-bit unsigned integer. Length of the Option Data field of this option, in octets.

OAM-trace-type: 8-bit identifier of a particular trace element variant.

The trace type value can be interpreted as a bit field. The following bit fields are defined in this document, with details on each field described in the next section. The order of packing the trace data in each Node-data element follows the bit order for setting each trace data element. Only a valid combination of these fields defined in this document are valid in-band OAM-trace-types.

Bit 0 When set indicates presence of node_id in the Node data.

Bit 1 When set indicates presence of ingress_if_id in the Node data.

Bit 2 When set indicates presence of egress_if_id in the Node data.

Bit 3 When set indicates presence of timestamp in the Node data.

Bit 4 When set indicates presence of app_data in the Node data.

Bit 5-7 Undefined in this document.

Section 3.1.1 describes the format of a number of trace types. Specifically, it exemplifies OAM-trace-types 0x00011111, 0x00000111, 0x00001001, 0x00010001, and 0x00011001.

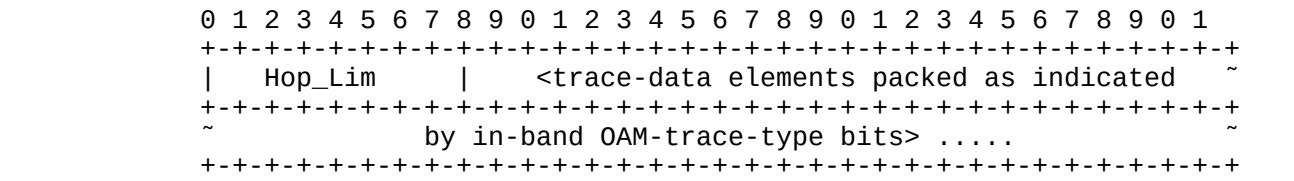
Elements-left: 8-bit unsigned integer. A pointer that indicates the next data recording point in the data space of the packet in octets. It is the index into the "Node data List" array shown above.

Node data List [n]: Variable-length field. The format of which is determined by the OAM Type representing the n-th Node data in the Node data List. The Node data List is encoded starting from the last Node data of the path. The first element of the node data list (Node data List [0]) contains the last node of the path while the last node data of the Node data List (Node data List[n]) contains the first Node data of the path traced. The index contained in "Elements-left" identifies the current active Node data to be populated.

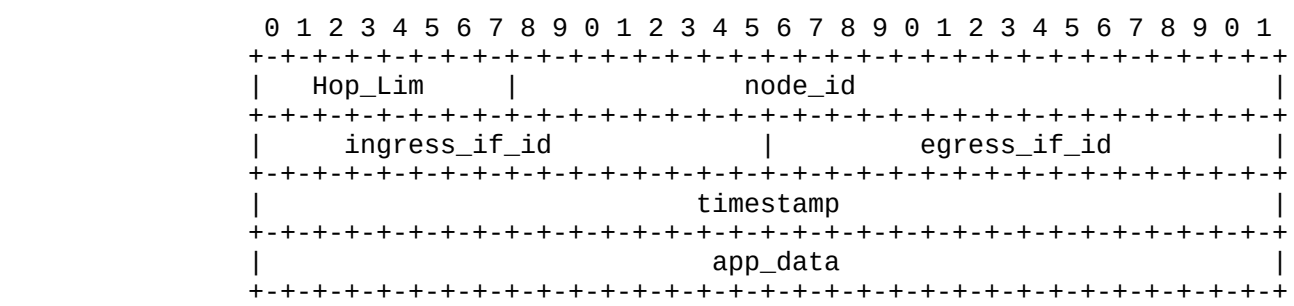
3.1.1. In-band OAM Trace Type and Node Data Element

An entry in the "Node data List" array can have different formats, following the needs of the a deployment. Some deployments might only be interested in recording the node identifiers, whereas others might be interested in recording node identifier and timestamp. The section defines different formats that an entry in "Node data List" can take.

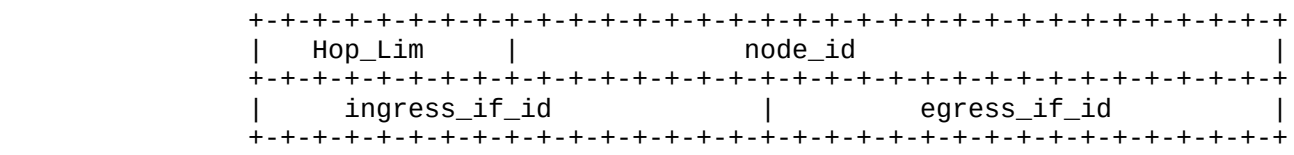
Node data has the following format:



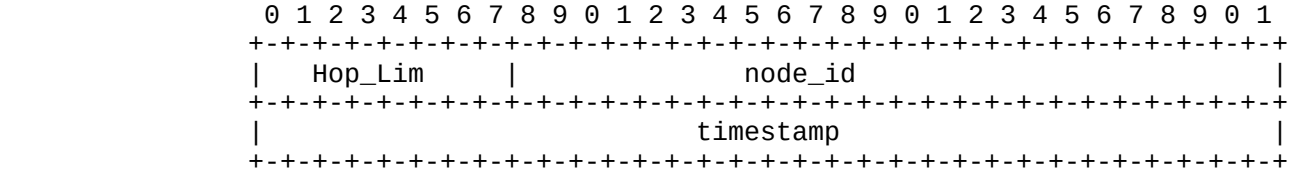
0x00011111: In-band OAM-trace-type is 0x00011111 then the format of node data is:



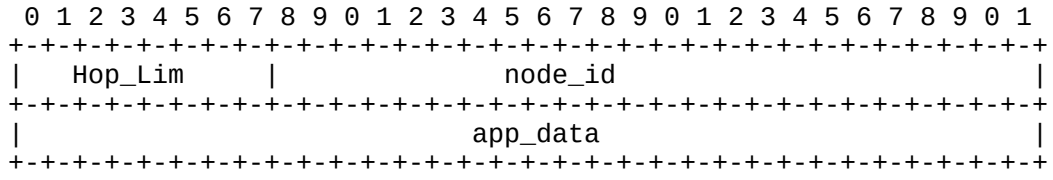
0x00000111: In-band OAM-trace-type is 0x00000111 then the format is:



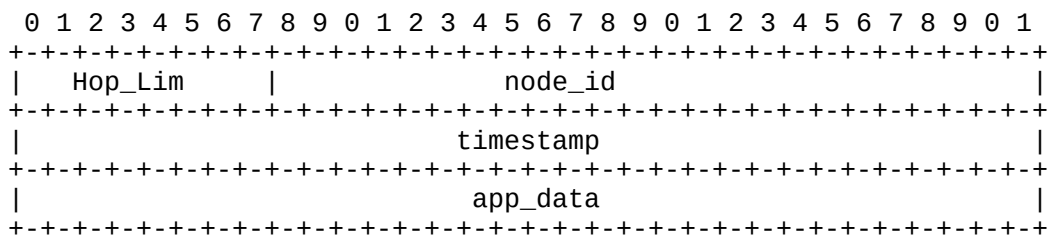
0x00001001: In-band OAM-trace-type is 0x00001001 then the format is:



0x00010001: In-band OAM-trace-type is 0x00010001 then the format is:



0x00011001: In-band OAM-trace-type is 0x00011001 then the format is:



Trace data elements in Node data are defined as follows:

Hop_Lim: 1 octet Hop limit that is set to the TTL value in the packet at the node that records this data.

node_id: Node identifier node_id is a 3 octet field to uniquely identify a node within in-band OAM domain. The procedure to allocate, manage and map the node_ids is beyond the scope of this document.

ingress_if_id: 2 octet interface identifier to record the ingress interface the packet was received on.

egress_if_id: 2 octet interface identifier to record the egress interface the packet is forwarded out of.

timestamp: 4 octet timestamp when packet has been processed by the node.

app_data: 4 octet placeholder which can be used by the node to add application specific data.

Hop Limit information is used to identify the location of the node in the communication path.

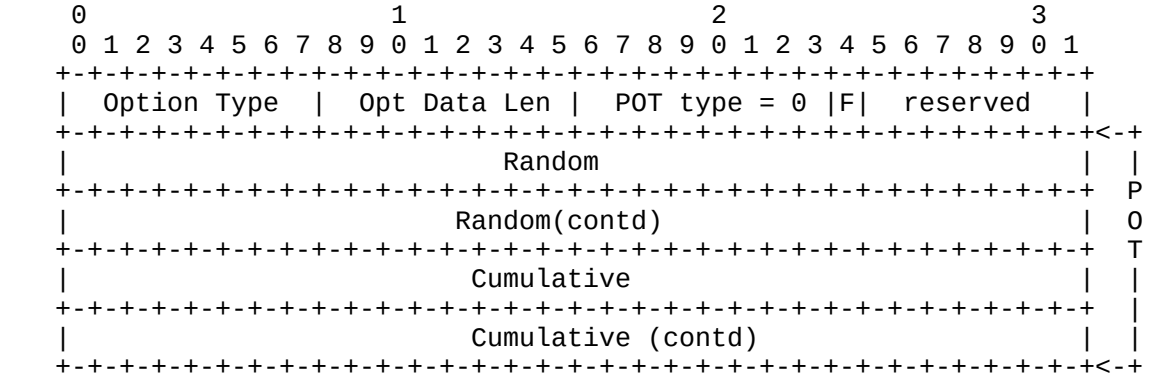
3.2. In-band OAM Proof of Transit Option

In-band OAM Proof of Transit data is to support the path or service function chain [RFC7665] verification use cases. Proof-of-transit uses methods like nested hashing or nested encryption of the in-band OAM data or mechanisms such as Shamir's Secret Sharing Schema (SSSS). While details on how the in-band OAM data for the proof of transit option is processed at in-band OAM encapsulating, decapsulating and transit nodes are outside the scope of the document, all of these approaches share the need to uniquely identify a packet as well as iteratively operate on a set of information that is handed from node to node. Correspondingly, two pieces of information are added as in-band OAM data to the packet:

- Random: Unique identifier for the packet (e.g., 64-bits allow for the unique identification of 2^64 packets).

- Cumulative: Information which is handed from node to node and updated by every node according to a verification algorithm.

In-band OAM Proof of Transit option:



Option Type: 8-bit identifier of the type of option.

Opt Data Len: 8-bit unsigned integer. Length of the Option Data field of this option, in octets.

POT Type: 8-bit identifier of a particular POT variant that dictates the POT data that is included.

* 16 Octet field as described below

Flag (F): 1-bit. Indicates which POT-profile is active. 0 means the even POT-profile is active, 1 means the odd POT-profile is active.

Reserved: 7-bit. (Reserved Octet) Reserved octet for future use.

Random: 64-bit Per packet Random number.

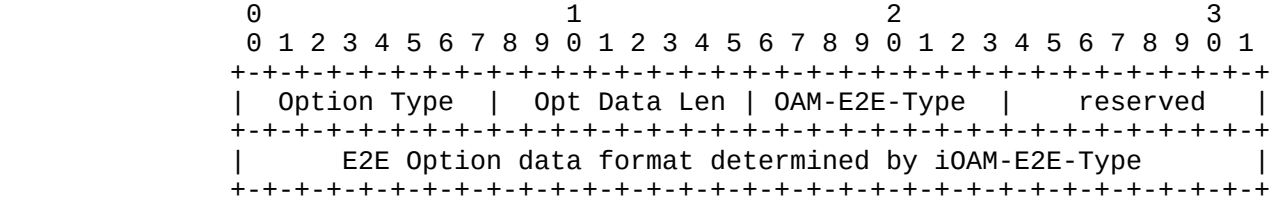
Cumulative: 64-bit Cumulative that is updated at specific nodes by processing per packet Random number field and configured parameters.

Note: Larger or smaller sizes of "Random" and "Cumulative" data are feasible and could be required for certain deployments (e.g. in case of space constraints in the transport protocol used). Future versions of this document will address different sizes of data for "proof of transit".

3.3. In-band OAM Edge-to-Edge Option

The in-band OAM Edge-to-Edge Option is to carry data which is to be interpreted only by the in-band OAM encapsulating and in-band OAM decapsulating node, but not by in-band OAM transit nodes.

Currently only sequence numbers use the in-band OAM Edge-to-Edge option. In order to detect packet loss, packet reordering, or packet duplication in an in-band OAM-domain, sequence numbers can be added to packets of a particular tube (see [I-D.hildebrand-spud-prototype]). Each tube leverages a dedicated namespace for its sequence numbers.



Option Type: 8-bit identifier of the type of option.

Opt Data Len: 8-bit unsigned integer. Length of the Option Data field of this option, in octets.

iOAM-E2E-Type: 8-bit identifier of a particular in-band OAM E2E variant.

0: E2E option data is a 64-bit sequence number added to a specific tube which is used to identify packet loss and reordering for that tube.

Reserved: 8-bit. (Reserved Octet) Reserved octet for future use.

4. In-band OAM Data Export

In-band OAM nodes collect information for packets traversing a domain that supports in-band OAM. The device at the domain edge (which could also be an end-host) which receives a packet with in-band OAM information chooses how to process the in-band OAM data collected within the packet. This decapsulating node can simply discard the information collected, can process the information further, or export the information using e.g., IPFIX.

The discussion of in-band OAM data processing and export is left for a future version of this document.

5. IANA Considerations

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

6. Manageability Considerations

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

7. Security Considerations

Security considerations will be addressed in a later version of this document. For a discussion of security requirements of in-band OAM, please refer to [draft-brockners-inband-oam-requirements].

8. Acknowledgements

The authors would like to thank Steve Youell, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, and Andrew Yourtchenko for the comments and advice. This document leverages and builds on top of several concepts described in [draft-kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

9. References

9.1. Normative References

[draft-brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., and S. Dara, "Requirements for in-band OAM", July 2016.

9.2. Informative References

[draft-brockners-inband-oam-transport]
Brockners, F., Bhandari, S., Pignataro, C., and H. Gredler, "Encapsulations for in-band OAM", July 2016.

[draft-brockners-proof-of-transit]
Brockners, F., Bhandari, S., and S. Dara, "Proof of transit", July 2016.

[draft-kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6),Hop-by-Hop Option Extension", November 2000.

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

[I-D.hildebrand-spud-prototype]
Hildebrand, J. and B. Trammell, "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in progress), March 2015.

[I-D.lapukhov-dataplane-probe]
Lapukhov, P. and r. remy@barefootnetworks.com, "Data-plane probe for in-band telemetry collection", draft-lapukhov-dataplane-probe-01 (work in progress), June 2016.

[P4] Kim, , "P4: In-band Network Telemetry (INT)", September 2015.

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

Authors’ Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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 Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

F. Brockners
S. Bhandari
S. Dara
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.

July 8, 2016

Requirements for In-band OAM
draft-brockners-inband-oam-requirements-00

Abstract

This document discusses the motivation and requirements for including specific operational and telemetry information into data packets while the data packet traverses a path between two points in the network. This method is referred to as "in-band" Operations, Administration, and Maintenance (OAM), given that the OAM information is carried with the data packets as opposed to in "out-of-band" packets dedicated to OAM. In-band OAM complements other OAM mechanisms which use dedicated probe packets to convey OAM information.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document discusses requirements for "in-band" Operations, Administration, and Maintenance (OAM) mechanisms. "In-band" OAM means to record OAM and telemetry information within the data packet

while the data packet traverses a network or a particular network domain. The term "in-band" refers to the fact that the OAM and telemetry data is carried within data packets rather than being sent within packets specifically dedicated to OAM. In-band OAM mechanisms, which are sometimes also referred to as embedded network telemetry are a current topic of discussion. In-band network telemetry has been defined for P4 [P4]. The SPUD prototype [I-D.hildebrand-spud-prototype] uses a similar logic that allows network devices on the path between endpoints to participate explicitly in the tube outside the end-to-end context. Even the IPv4 route-record option defined in [RFC0791] can be considered an in-band OAM mechanism. In-band OAM complements "out-of-band" mechanisms such as ping or traceroute, or more recent active probing mechanisms, as described in [I-D.lapukhov-dataplane-probe]. In-band OAM mechanisms can be leveraged where current out-of-band mechanisms do not apply or do not offer the desired characteristics or requirements, such as proving that a certain set of traffic takes a pre-defined path, strict congruency is desired, checking service level agreements for the live data traffic, detailed statistics on traffic distribution paths in networks that distribute traffic across multiple paths, or scenarios where probe traffic is potentially handled differently from regular data traffic by the network devices. [RFC7276] presents an overview of OAM tools.

Compared to probably the most basic example of "in-band OAM" which is IPv4 route recording [RFC0791], an in-band OAM approach has the following capabilities:

a. A flexible data format to allow different types of information to be captured as part of an in-band OAM operation, including not only path tracing information, but additional operational and telemetry information such as timestamps, sequence numbers, or even generic data such as queue size, geo-location of the node that forwarded the packet, etc.

b. A data format to express node as well as link identifiers to record the path a packet takes with a fixed amount of added data.

c. The ability to detect whether any nodes were skipped while recording in-band OAM information (i.e., in-band OAM is not supported or not enabled on those nodes).

d. The ability to actively process information in the packet, for example to prove in a cryptographically secure way that a packet really took a pre-defined path using some traffic steering method such as service chaining or traffic engineering.

e. The ability to include OAM data beyond simple path information, such as timestamps or even generic data of a particular use case.

f. The ability to include OAM data in various different transport protocols.

2. Conventions

Abbreviations used in this document:

ECMP: Equal Cost Multi-Path

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

OAM: Operations, Administration, and Maintenance

PMTU: Path MTU

SLA: Service Level Agreement

SFC: Service Function Chain

SR: Segment Routing

This document defines in-band Operations, Administration, and Maintenance (in-band OAM), as the subset in which OAM information is carried along with data packets. This is as opposed to "out-of-band OAM", where specific packets are dedicated to carrying OAM information.

3. Motivation for In-band OAM

In several scenarios it is beneficial to make information about which path a packet took through the network available to the operator. This includes not only tasks like debugging, troubleshooting, as well as network planning and network optimization but also policy or service level agreement compliance checks. This section discusses the motivation to introduce new methods for enhanced in-band network diagnostics.

3.1. Path Congruency Issues with Dedicated OAM Packets

Mechanisms which add tracing information to the regular data traffic, sometimes also referred to as "in-band" or "passive OAM" can complement active, probe-based mechanisms such as ping or traceroute, which are sometimes considered as "out-of-band", because the messages

are transported independently from regular data traffic. "In-band" mechanisms do not require extra packets to be sent and hence don't change the packet traffic mix within the network. Traceroute and ping for example use ICMP messages: New packets are injected to get tracing information. Those add to the number of messages in a network, which already might be highly loaded or suffering performance issues for a particular path or traffic type.

Packet scheduling algorithms, especially for balancing traffic across equal cost paths or links, often leverage information contained within the packet, such as protocol number, IP-address or MAC-address. Probe packets would thus either need to be sent from the exact same endpoints with the exact same parameters, or probe packets would need to be artificially constructed as "fake" packets and inserted along the path. Both approaches are often not feasible from an operational perspective, be it that access to the end-system is not feasible, or that the diversity of parameters and associated probe packets to be created is simply too large. An in-band mechanism is an alternative in those cases.

In-band mechanisms also don't suffer from implementations, where probe traffic is handled differently (and potentially forwarded differently) by a router than regular data traffic.

3.2. Results Sent to a System Other Than the Sender

Traditional ping and traceroute tools return the OAM results to the sender of the probe. Even when the ICMP messages that are used with these tools are enhanced, and additional telemetry is collected (e.g., ICMP Multi-Part [RFC4884] supporting MPLS information [RFC4950], Interface and Next-Hop Identification [RFC5837], etc.), it would be advantageous to separate the sending of an OAM probe from the receiving of the telemetry data. In this context, it is desired to not assume there is a bidirectional working path.

3.3. Overlay and Underlay Correlation

Several network deployments leverage tunneling mechanisms to create overlay or service-layer networks. Examples include VXLAN-GPE, GRE, or LISP. One often observed attribute of overlay networks is that they do not offer the user of the overlay any insight into the underlay network. This means that the path that a particular tunneled packet takes, nor other operational details such as the per-hop delay/jitter in the underlay are visible to the user of the overlay network, giving rise to diagnosis and debugging challenges in case of connectivity or performance issues. The scope of OAM tools like ping or traceroute is limited to either the overlay or the underlay which means that the user of the overlay has typically no

access to OAM in the underlay, unless specific operational procedures are put in place. With in-band OAM the operator of the underlay can offer details of the connectivity in the underlay to the user of the overlay. The operator of the egress tunnel router could choose to share the recorded information about the path with the user of the overlay.

Coupled with mechanisms such as Segment Routing (SR) [I-D.ietf-spring-segment-routing], overlay network and underlay network can be more tightly coupled: The user of the overlay has detailed diagnostic information available in case of failure conditions. The user of the overlay can also use the path recording information as input to traffic steering or traffic engineering mechanisms, to for example achieve path symmetry for the traffic between two endpoints. [I-D.brockners-lisp-sr] is an example for how these methods can be applied to LISP.

3.4. SLA Verification

In-band OAM can help users of an overlay-service to verify that negotiated SLAs for the real traffic are met by the underlay network provider. Different from solutions which rely on active probes to test an SLA, in-band OAM based mechanisms avoid wrong interpretations and "cheating", which can happen if the probe traffic that is used to perform SLA-check is prioritized by the network provider of the underlay.

3.5. Analytics and Diagnostics

Network planners and operators benefit from knowledge of the actual traffic distribution in the network. When deriving an overall network connectivity traffic matrix one typically needs to correlate data gathered from each individual devices in the network. If the path of a packet is recorded while the packet is forwarded, the entire path that a packet took through the network is available to the egress system. This obviates the need to retrieve individual traffic statistics from every device in the network and correlate those statistics, or employ other mechanisms such as leveraging traffic engineering with null-bandwidth tunnels just to retrieve the appropriate statistics to generate the traffic matrix.

In addition, with individual path tracing, information is available at packet level granularity, rather than only at aggregate level - as is usually the case with IPFIX-style methods which employ flow-filters at the network elements. Data-center networks which use equal-cost multipath (ECMP) forwarding are one example where detailed statistics on flow distribution in the network are highly desired. If a network supports ECMP, one can create detailed statistics for

the different paths packets take through the network at the egress system, without a need to correlate/aggregate statistics from every router in the system. Transit devices are off-loaded from the task of gathering packet statistics.

3.6. Frame Replication/Elimination Decision for Bi-casting/Active-active Networks

Bandwidth- and power-constrained, time-sensitive, or loss-intolerant networks (e.g., networks for industry automation/control, health care) require efficient OAM methods to decide when to replicate packets to a secondary path in order to keep the loss/error-rate for the receiver at a tolerable level - and also when to stop replication and eliminate the redundant flow. Many IoT networks are time sensitive and cannot leverage automatic retransmission requests (ARQ) to cope with transmission errors or lost packets. Transmitting the data over multiple disparate paths (often called bi-casting or live-live) is a method used to reduce the error rate observed by the receiver. TSN receive a lot of attention from the manufacturing industry as shown by a various standardization activities and industry forums being formed (see e.g., IETF 6TiSCH, IEEE P802.1CB, AVnu).

3.7. Proof of Transit

Several deployments use traffic engineering, policy routing, segment routing or Service Function Chaining (SFC) [RFC7665] to steer packets through a specific set of nodes. In certain cases regulatory obligations or a compliance policy require to prove that all packets that are supposed to follow a specific path are indeed being forwarded across the exact set of nodes specified. If a packet flow is supposed to go through a series of service functions or network nodes, it has to be proven that all packets of the flow actually went through the service chain or collection of nodes specified by the policy. In case the packets of a flow weren’t appropriately processed, a verification device would be required to identify the policy violation and take corresponding actions (e.g., drop or redirect the packet, send an alert etc.) corresponding to the policy. In today’s deployments, the proof that a packet traversed a particular 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 trusted. The evolution to Network Function Virtualization (NFV) and modern service chaining concepts (using

technologies such as LISP, NSH, Segment Routing, 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. Because of that very reason, networks operators require that different trust layers not to be mixed in the same device. For an NFV scenario a different proof is required. Offering a proof that a packet traversed a specific set of service functions would allow network operators to move away from the above described indirect methods of proving that a service chain is in place for a particular application.

A solution approach could be based on OAM data which is added to every packet for achieving Proof Of Transit. The OAM data is updated at every hop and is used to verify whether a packet traversed all required nodes. When the verifier receives each packet, it can validate whether the packet traversed the service chain correctly. The detailed mechanisms used for path verification along with the procedures applied to the OAM data carried in the packet for path verification are beyond the scope of this document. Details are addressed in [draft-brockners-proof-of-transit]. In this document the term "proof" refers to a discrete set of bits that represents an integer or string carried as OAM data. The OAM data is used to verify whether a packet traversed the nodes it is supposed to traverse.

3.8. Use Cases

In-band OAM could be leveraged for several use cases, including:

- Traffic Matrix: Derive the network traffic matrix: Traffic for a given time interval between any two edge nodes of a given domain. Could be performed for all traffic or per QoS-class.

- Flow Debugging: Discover which path(s) a particular set of traffic (identified by an n-tuple) takes in the network. Such a procedure is particularly useful in case traffic is balanced across multiple paths, like with link aggregation (LACP) or equal cost multi-pathing (ECMP).

- Loss Statistics per Path: Retrieve loss statistics per flow and path in the network.

- Path Heat Maps: Discover highly utilized links in the network.

- Trend Analysis on Traffic Patterns: Analyze if (and if so how) the forwarding path for a specific set of traffic changes over time (can give hints to routing issues, unstable links etc.).

- Network Delay Distribution: Show delay distribution across network by node or links. If enabled per application or for a specific flow then display the path taken along with the delay incurred at every hop.

- SLA Verification: Verify that a negotiated service level agreement (SLA), e.g., for packet drop rates or delay/jitter is conformed to by the actual traffic.

- Low-power Networks: Include application level OAM information (e.g., battery charge level, cache or buffer fill level) into data traffic to avoid sending extra OAM traffic which incur an extra cost on the devices. Using the battery charge level as example, one could avoid sending extra OAM packets just to communicate battery health, and as such would save battery on sensors.

- Path Verification or Service Function Path Verification: Proof and verification of packets traversing check points in the network, where check points can be nodes in the network or service functions.

- Geo-location Policy: Network policy implemented based on which path packets took. Example: Only if packets originated and stayed within the trading-floor department, access to specific applications or servers is granted.

4. Considerations for In-band OAM

The implementation of an in-band OAM mechanism needs to take several considerations into account, including administrative boundaries, how information is recorded, Maximum Transfer Unit (MTU), Path MTU discovery and packet size, etc.

4.1. Type of Information to Be Recorded

The information gathered for in-band OAM can be categorized into three main categories: Information with a per-hop scope, such as path tracing; information which applies to a specific set of nodes, such as path or service chain verification; information which only applies to the edges of a domain, such as sequence numbers.

- "edge to edge": Information that needs to be shared between network edges (the "edge" of a network could either be a host or a domain edge device): Edge to edge data e.g., packet and octet count of data entering a well-defined domain and leaving it is helpful in building traffic matrix, sequence number (also called "path packet counters") is useful for the flow to detect packet loss.

- "selected hops": Information that applies to a specific set of nodes only. In case of path verification, only the nodes which are "check points" are required to interpret and update the information in the packet.

- "per hop": Information that is gathered at every hop along the path a packet traverses within an administrative domain:

  * Hop by Hop information e.g., Nodes visited for path tracing, Timestamps at each hop to find delays along the path

  * Stats collection at each hop to optimize communication in resource constrained networks e.g., Battery, CPU, memory status of each node piggy backed in a data packet is useful in low power lossy networks where network nodes are mostly asleep and communication is expensive

4.2. MTU and Packet Size

The recorded data at every hop may lead to packet size exceeding the Maximum Transmit Unit (MTU). Based on the transport protocol used MTU is discovered as a configuration parameter or Path MTU (PMTU) is discovered dynamically. Example: IPv6 recommends PMTU discovery before data packets are sent to prevent packet fragmentation. It specifies 1280 octets as the default PDU to be carried in a IPv6 datagram. A detailed discussion of the implications of oversized IPv6 header chains if found in [RFC7112].

The Path MTU restricts the amount of data that can be recorded for purpose of OAM within a data packet. The total size of data to be recorded needs to be preset to avoid packet size exceeding the MTU. It is recommended to pre-calculate and configures network devices to limit the in-band OAM data that is attached to a packet.

4.3. Administrative Boundaries

There are challenges in enabling in-band OAM in the public Internet across administrative domains:

- Deployment dependent, the data fields that in-band OAM requires as part of a specific transport protocol may not be supported across administrative boundaries.

- Current OAM implementations are often done in the slow path, i.e., OAM packets are punted to router’s CPU for processing. This leads to performance and scaling issues and opens up routers for attacks such as Denial of Service (DoS) attacks.

- Discovery of network topology and details of the network devices across administrative boundaries may open up attack vectors compromising network security.

- Specifically on IPv6: At the administrative boundaries IPv6 packets with extension headers are dropped for several reasons described in [RFC7872]

The following considerations will be discussed in a future version of this document: If the packet is dropped due to the presence of the in-band OAM; If the policy failure is treated as feature disablement and any further recording is stopped but the packet itself is not dropped, it may lead to every node in the path to make this policy decision.

4.4. Selective Enablement

Deployment dependent, in-band OAM could either be used for all, or only a subset of the overall traffic. While it might be desirable to apply in-band OAM to all traffic and then selectively use the data gathered in case needed, it might not always be feasible. Depending on the forwarding infrastructure used, in-band OAM can have an impact on forwarding performance. The SPUD prototype for example uses the notion of "pipes" to describe the portion of the traffic that could be subject to in-path inspection. Mechanisms to decide which traffic would be subject to in-band OAM are outside the scope of this document.

4.5. Optimization of Node and Interface Identifiers

Since packets have a finite maximum size, the data recording or carrying capacity of one packet in which the in-band OAM meta data is present is limited. In-band OAM should use its own dedicated namespace (confined to the domain in-band OAM operates in) to represent node and interface IDs to save space in the header. Generic representations of node and interface identifiers which are globally unique (such as a UUID) would consume significantly more bits of in-band OAM data.

4.6. Loop Communication Path (IPv6-specifics)

When recorded data is required to be analyzed on a source node that issues a packet and inserts in-band OAM data, the recorded data needs to be carried back to the source node.

One way to carry the in-band OAM data back to the source is to utilize an ICMP Echo Request/Reply (ping) or ICMPv6 Echo Request/ Reply (ping6) mechanism. In order to run the in-band OAM mechanism

appropriately on the ping/ping6 mechanism, the following two operations should be implemented by the ping/ping6 target node:

1. All of the in-band OAM fields would be copied from an Echo Request message to an Echo Reply message.

2. The Hop Limit field of the IPv6 header of these messages would be copied as a continuous sequence. Further considerations are addressed in a future version of this document.

5. Requirements for In-band OAM Data Types

The above discussed use cases require different types of in-band OAM data. This section details requirements for in-band OAM derived from the discussion above.

5.1. Generic Requirements

REQ-G1: Classification: It should be possible to enable in-band OAM on a selected set of traffic. The selected set of traffic can also be all traffic.

REQ-G2: Scope: If in-band OAM is used only within a specific domain, provisions need to be put in place to ensure that in-band OAM data stays within the specific domain only.

REQ-G3: Transport independence: Data formats for in-band OAM shall be defined in a transport independent way. In-band OAM applies to a variety of transport protocols. Encapsulations should be defined how the generic data formats are carried by a specific protocol.

REQ-G4: Layering: It should be possible to have in-band OAM information for different transport protocol layers be present in several fields within a single packet. This could for example be the case when tunnels are employed and in-band OAM information is to be gathered for both the underlay as well as the overlay network.

REQ-G5: MTU size: With in-band OAM information added, packets should not become larger than the path MTU.

REQ-G6: Data Structure Reusability: The data types and data formats defined and used for in-band OAM ought to be reusable for out-of-band OAM telemetry as well.

5.2. In-band OAM Data with Per-hop Scope

REQ-H1: Missing nodes detection: Data shall be present that allows a node to detect whether all nodes that should participate in in-band OAM operations have indeed participated.

REQ-H2: Node, instance or device identifier: Data shall be present that allows to retrieve the identity of the entity reporting telemetry information. The entity can be a device, or a subsystem/component within a device. The latter will allow for packet tracing within a device in much the same way as between devices.

REQ-H3: Ingress interface identifier: Data shall be present that allows the identification of the interface a particular packet was received from. The interface can be a logical or physical entity.

REQ-H4: Egress interface identifier: Data shall be present that allows the identification of the interface a particular packet was forwarded to. Interface can be a logical or physical entity.

REQ-H5: Time-related requirements

- REQ-H5.1: Delay: Data shall be present that allows to retrieve the delay between two or more points of interest within the system. Those points can be within the same device or on different devices.
- REQ-H5.2: Jitter: Data shall be present that allows to retrieve the jitter between two or more points of interest within the system. Those points can be within the same device or on different devices.
- REQ-H5.3: Wall-clock time: Data shall be present that allows to retrieve the wall-clock time visited a particular point of interest in the system.
- REQ-H5.4: Time precision: The precision of the time related data should be configurable. Use-case dependent, the required precision could e.g., be nano-seconds, micro-seconds, milli-seconds, or seconds.

REQ-H6: Generic data records (like e.g., GPS/Geo-location information): It should be possible to add user-defined OAM

data at select hops to the packet. The semantics of the data are defined by the user.

5.3. In-band OAM with Selected Hop Scope

REQ-S1: Proof of transit: Data shall be present which allows to securely prove that a packet has visited or ore several particular points of interest (i.e., a particular set of nodes).

REQ-S1.1: In case "Shamir's secret sharing scheme" is used for proof of transit, two data records, "random" and "cumulative" shall be present. The number of bits used for "random" and "cumulative" data records can vary between deployments and should thus be configurable.

5.4. In-band OAM with End-to-end Scope

REQ-E1: Sequence numbering:

REQ-E1.1: Reordering detection: It should be possible to detect whether packets have been reordered while traversing an in-band OAM domain.

REQ-E1.2: Duplicates detection: It should be possible to detect whether packets have been duplicated while traversing an in-band OAM domain.

REQ-E1.3: Detection of packet drops: It should be possible to detect whether packets have been dropped while traversing an in-band OAM domain.

6. Security Considerations and Requirements

General Security considerations will be addressed in a later version of this document. Security considerations for Proof of Transit alone are discussed below.

6.1. Proof of Transit

Threat Model: Attacks on the deployments could be due to malicious administrators or accidental misconfigurations resulting in bypassing of certain nodes. The solution approach should meet the following requirements:

REQ-SEC1: Sound Proof of Transit: A valid and verifiable proof that the packet definitively traversed through all the nodes as

expected. Probabilistic methods to achieve this should be avoided, as the same could be exploited by an attacker.

REQ-SEC2: Tampering of meta data: An active attacker should not be able to insert or modify or delete meta data in whole or in parts and bypass few (or all) nodes. Any deviation from the expected path should be accurately determined.

REQ-SEC3: Replay Attacks: A attacker (active/passive) should not be able to reuse the proof of transit bits in the packet by observing the OAM data in the packet, packet characteristics (like IP addresses, octets transferred, timestamps) or even the proof bits themselves. The solution approach should consider usage of these parameters for deriving any secrets cautiously. Mitigating replay attacks beyond a window of longer duration could be intractable to achieve with fixed number of bits allocated for proof.

REQ-SEC4: Recycle Secrets: Any configuration of the secrets (like cryptographic keys, initialisation vectors etc.) either in the controller or service functions should be reconfigurable. Solution approach should enable controls, API calls etc. needed in order to perform such recycling. It is desirable to provide recommendations on the duration of rotation cycles needed for the secure functioning of the overall system.

REQ-SEC5: Secret storage and distribution: Secrets should be shared with the devices over secure channels. Methods should be put in place so that secrets cannot be retrieved by non authorized personnel from the devices.

7. IANA Considerations

[RFC Editor: please remove this section prior to publication.]

This document has no IANA actions.

8. Acknowledgements

The authors would like to thank Steve Youell, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, and Andrew Yourtchenko for the comments and advice. This document leverages and builds on top of several concepts described in [draft-kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

9. Informative References

[draft-brockners-proof-of-transit]
Brockners, F., Bhandari, S., and S. Dara, "Proof of transit", July 2016.

[draft-kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6),Hop-by-Hop Option Extension", November 2000.

[I-D.brockners-lisp-sr]
Brockners, F., Bhandari, S., Maino, F., and D. Lewis, "LISP Extensions for Segment Routing", draft-brockners-lisp-sr-01 (work in progress), February 2014.

[I-D.hildebrand-spud-prototype]
Hildebrand, J. and B. Trammell, "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in progress), March 2015.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-09 (work in progress), July 2016.

[I-D.lapukhov-dataplane-probe]
Lapukhov, P. and r. remy@barefootnetworks.com, "Data-plane probe for in-band telemetry collection", draft-lapukhov-dataplane-probe-01 (work in progress), June 2016.

[P4] Kim, , "P4: In-band Network Telemetry (INT)", September 2015.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <http://www.rfc-editor.org/info/rfc4884>.

[RFC4950] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "ICMP Extensions for Multiprotocol Label Switching", RFC 4950, DOI 10.17487/RFC4950, August 2007, <http://www.rfc-editor.org/info/rfc4950>.

[RFC5837] Atlas, A., Ed., Bonica, R., Ed., Pignataro, C., Ed., Shen, N., and JR. Rivers, "Extending ICMP for Interface and Next-Hop Identification", RFC 5837, DOI 10.17487/RFC5837, April 2010, <http://www.rfc-editor.org/info/rfc5837>.

[RFC7112] Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, DOI 10.17487/RFC7112, January 2014, <http://www.rfc-editor.org/info/rfc7112>.

[RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <http://www.rfc-editor.org/info/rfc7276>.

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

[RFC7872] Gont, F., Linkova, J., Chown, T., and W. Liu, "Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World", RFC 7872, DOI 10.17487/RFC7872, June 2016, <http://www.rfc-editor.org/info/rfc7872>.

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: sadara@cisco.com

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

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

F. Brockners
S. Bhandari
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.

July 8, 2016

Encapsulations for In-band OAM Data
draft-brockners-inband-oam-transport-00

Abstract

In-band operation, administration and maintenance (OAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. In-band OAM is to complement current out-of-band OAM mechanisms based on ICMP or other types of probe packets. This document outlines how in-band OAM data records can be transported in protocols such as NSH, Segment Routing, VXLAN-GPE, native IPv6 (via extension header), and IPv4. Transport options are currently investigated as part of an implementation study. This document is intended to only serve informational purposes.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document discusses transport mechanisms for "in-band" operation, administration, and maintenance (OAM) data records. In-band OAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-band" 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. A discussion of the motivation and requirements for in-band OAM can be found in [draft-brockners-inband-oam-requirements]. Data types and data formats for in-band OAM are defined in [draft-brockners-inband-oam-data].

This document outlines transport encapsulations for the in-band OAM data defined in [draft-brockners-inband-oam-data]. This document is to serve informational purposes only. As part of an in-band OAM implementation study different protocol encapsulations for in-band OAM data are being explored. Once data formats and encapsulation approaches are settled, protocol specific specifications for in-band OAM data transport will address the standardization aspect.

The data for in-band OAM defined in [draft-brockners-inband-oam-data] can be carried in a variety of protocols based on the deployment needs. This document discusses transport of in-band OAM data for the following protocols:

- IPv6
- VXLAN-GPE
- NSH
- Segment Routing (IPv6 and MPLS)

This list is non-exhaustive, as it is possible to carry the in-band OAM data in several other protocols and transports.

A feasibility study of in-band OAM is currently underway as part of the FD.io project [FD.io]. The in-band OAM implementation study should be considered as a "tool box" to showcase how "in-band" OAM can complement probe-packet based OAM mechanisms for different deployments and packet transport formats. For details, see the open source code in the FD.io [FD.io].

2. Conventions

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

Abbreviations used in this document:

MTU: Maximum Transmit Unit

OAM: Operations, Administration, and Maintenance

SR: Segment Routing

SID: Segment Identifier

NSH: Network Service Header

POT: Proof of Transit

SFC: Service Function Chain

VXLAN-GPE: Virtual eXtensible Local Area Network, Generic Protocol Extension

3. In-Band OAM Metadata Transport in IPv6

This mechanisms of in-band OAM in IPv6 complement others proposed to enhance diagnostics of IPv6 networks, such as the IPv6 Performance and Diagnostic Metrics Destination Option described in [I-D.ietf-ippm-6man-pdm-option]. The IP Performance and Diagnostic Metrics Destination Option is destination focused and specific to IPv6, whereas in-band OAM is performed between end-points of the network or a network domain where it is enabled and used.

A historical note: The idea of IPv6 route recording was originally introduced by [draft-kitamura-ipv6-record-route] back in year 2000. With IPv6 now being generally deployed and new concepts such as Segment Routing [I-D.ietf-spring-segment-routing] being introduced, it is imperative to further mature the operations, administration, and maintenance mechanisms available to IPv6 networks.

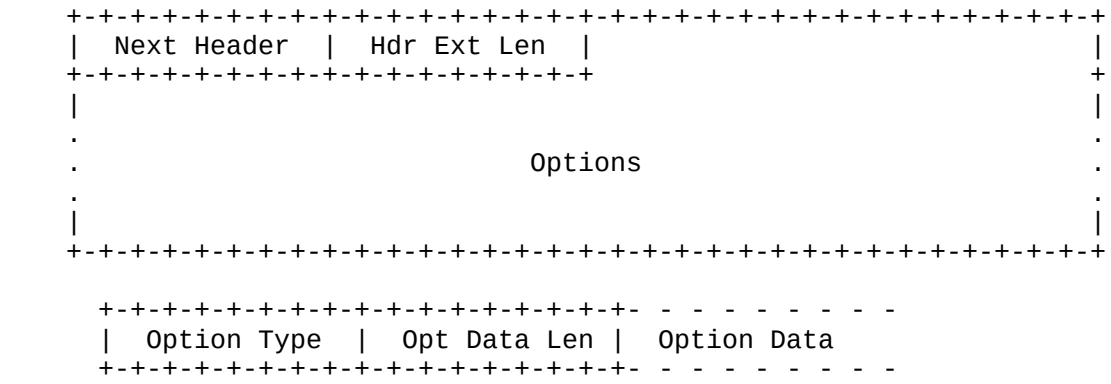
The in-band OAM options translate into options for an IPv6 extension header. The extension header would be inserted by either a host source of the packet, or by a transit/domain-edge node.

3.1. In-band OAM in IPv6 Hop by Hop Extension Header

This section defines in-band OAM for IPv6 transport. In-band OAM data is transported as an IPv6 hop-by-hop extension header.

3.1.1. In-band OAM Hop by Hop Options

Brief recap of the IPv6 hop-by-hop header as well as the options used for carrying in-band OAM data:



With 2 highest order bits of Option Type indicating the following:

00 - skip over this option and continue processing the header.

01 - discard the packet.

10 - discard the packet and, regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

11 - discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

3rd highest bit:

0 - Option Data does not change en-route

1 - Option Data may change en-route

In-band OAM data records are inserted as options in an IPv6 hop-by-hop extension header:

1. Tracing Option: The in-band OAM Tracing option defined in [draft-brockners-inband-oam-data] is represented as a IPv6 option in hop by hop extension header by allocating following type:

   Option Type: 001xxxxx 8-bit identifier of the type of option.
   xxxxxx=TBD_IANA_TRACE_OPTION_IPV6.

2. Proof of Transit Option: The in-band OAM POT option defined in [draft-brockners-inband-oam-data] is represented as a IPv6 option in hop by hop extension header by allocating following type:

   Option Type: 001xxxxxx 8-bit identifier of the type of option.
   xxxxxx=TBD_IANA_POT_OPTION_IPV6.

3. Edge to Edge Option: The in-band OAM E2E option defined in [draft-brockners-inband-oam-data] is represented as a IPv6 option in hop by hop extension header by allocating following type:

   Option Type: 000xxxxxx 8-bit identifier of the type of option.
   xxxxxx=TBD_IANA_E2E_OPTION_IPV6.

3.1.2. Procedure at the Ingress Edge to Insert the In-band OAM Header

In an administrative domain where in-band OAM is used, insertion of the in-band OAM header is enabled at the required edge nodes by means of configuration.

Such a config SHOULD allow selective enablement of in-band OAM header insertion for a subset of traffic (e.g., one or several "pipes").

Further the ingress edge node should be aware of maximum size of the header that can be inserted. Details on how the maximum size/size of the in-band OAM domain are retrieved are outside the scope of this document.

```
Let n = max number of nodes to be allocated;
(Based on PMTU advertised in the domain)

Let k = number of node data that can be allocated by this node
Let node_data_size = size of each node_data based on in-band OAM type

if (packet matches traffic for which in-band OAM is enabled) {
    Create in-band OAM hbyh ext header with k node data preallocated
    Increment payload length in IPv6 header :
                      with size of in-band OAM hbyh ext header
    Populate node data at :
        (size of in-band OAM hbyh header = 8) + k * node_data_size
    from the beginning of the header
    Set segments left to : k - 1

 }
```

3.1.3. Procedure at Intermediate Nodes

If a network node receives a packet with an in-band OAM header and it is enabled to process in-band OAM data it performs the following:

```
k = number of node data that this node can allocate
if (in-band OAM ext hbyh header is present) {
    if (Segments Left > 0)) {
      populate node data at :
         node_data_start[Segments Left]
      Segments Left = Segments Left - 1
    }
}
```

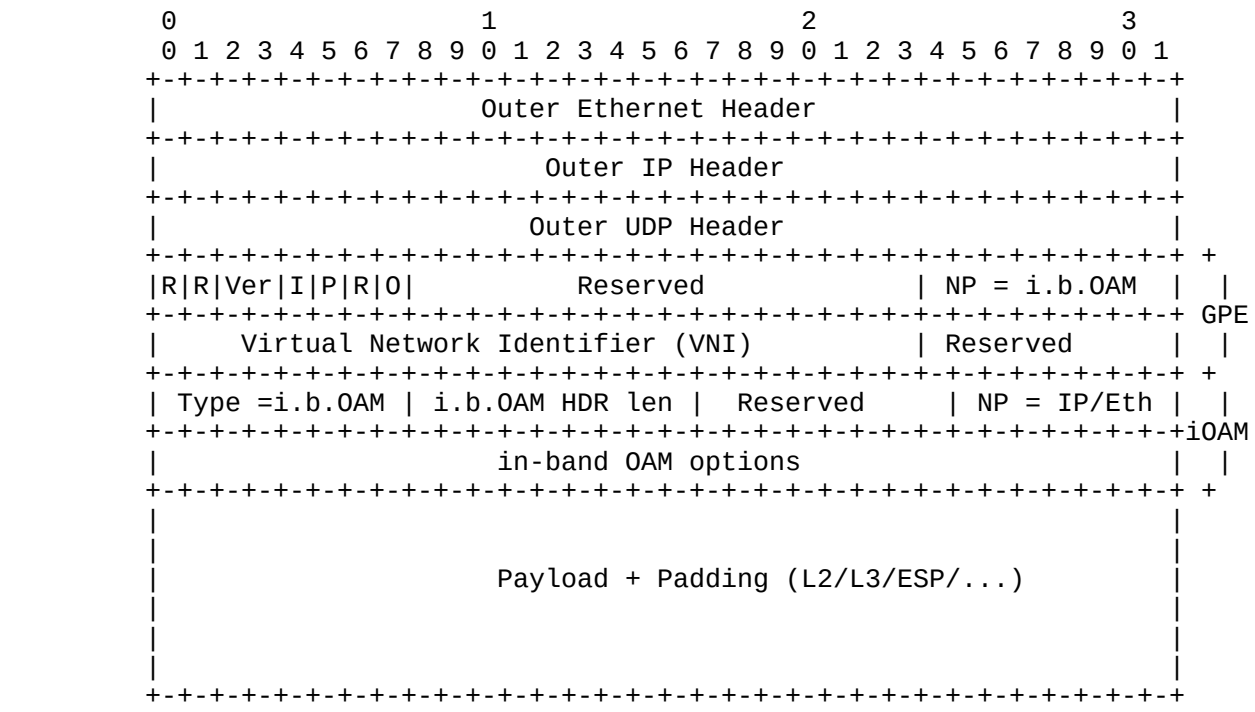
3.1.4. Procedure at the Egress Edge to Remove the In-band OAM Header

```
egress_edge = list of interfaces where in-band OAM hbyh ext
               header is to be stripped
Before forwarding packet out of interfaces in egress_edge list:
if (in-band OAM hbyh ext header is present) {
   remove the in-band OAM hbyh ext header,
   possibly store the record along with additional
   fields for analysis and export
   Decrement Payload Length in IPv6 header
   by size of in-band OAM ext header
}
```

4. In-band OAM Metadata Transport in VXLAN-GPE

VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe] encapsulation is somewhat similar to IPv6 extension headers in that a series of headers can be contained in the header as a linked list. The different in-band OAM types are added as options within a new in-band OAM protocol header in VXLAN GPE.

In-band OAM header in VXLAN GPE header:

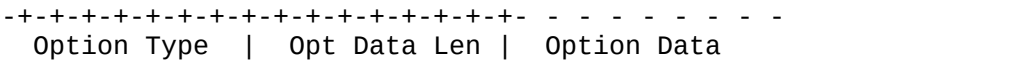


The VXLAN-GPE header and fields are defined in [I-D.ietf-nvo3-vxlan-gpe]. in-band OAM specific fields and header are defined here:

Type: 8-bit unsigned integer defining in-band OAM header type

in-band OAM HDR len: 8-bit unsigned integer. Length of the in-band OAM HDR in 8-octet units

in-band OAM options: Variable-length field, of length such that the complete in-band OAM header is an integer multiple of 8 octets long. Contains one or more TLV-encoded options of the format:



Option Type 8-bit identifier of the type of option.

Opt Data Len 8-bit unsigned integer. Length of the Option Data field of this option, in octets.

Option Data Variable-length field. Option-Type-specific data.

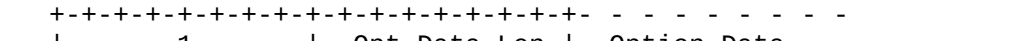
The in-band OAM options defined in [draft-brockners-inband-oam-data] are encoded with an option type allocated in the new in-band OAM IANA registry - in-band OAM_PROTOCOL_OPTION_REGISTRY_IANA_TBD. In addition the following padding options are defined to be used when necessary to align subsequent options and to pad out the containing header to a multiple of 8 octets in length.

Pad1 option (alignment requirement: none)

NOTE: The format of the Pad1 option is a special case -- it does not have length and value fields.

The Pad1 option is used to insert one octet of padding into the Options area of a header. If more than one octet of padding is required, the PadN option, described next, should be used, rather than multiple Pad1 options.

PadN option (alignment requirement: none)

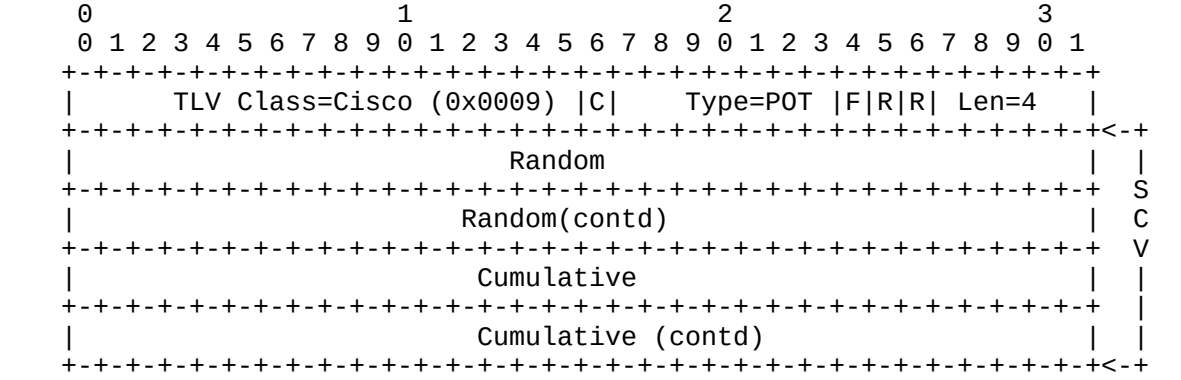


The PadN option is used to insert two or more octets of padding into the Options area of a header. For N octets of padding, the Opt Data Len field contains the value N-2, and the Option Data consists of N-2 zero-valued octets.

5. In-band OAM Metadata Transport in NSH

In Service Function Chaining (SFC) [RFC7665], the Network Service Header (NSH) [I-D.ietf-sfc-nsh] already includes path tracing capabilities [I-D.penno-sfc-trace], but currently does not offer a solution to securely prove that packets really traversed the service

chain. The "Proof of Transit" capabilities (see [draft-brockners-inband-oam-requirements] and [draft-brockners-proof-of-transit]) of in-band OAM can be leveraged within NSH. Proof of transit in-band OAM data is added as NSH Type 2 metadata:



TLV Class: Describes the scope of the "Type" field. In some cases, the TLV Class will identify a specific vendor, in others, the TLV Class will identify specific standards body allocated types. POT is currently defined using the Cisco (0x0009) TLV class.

Type: The specific type of information being carried, within the scope of a given TLV Class. Value allocation is the responsibility of the TLV Class owner. Currently a type value of 0x94 is used for proof of transit

Reserved bits: Two reserved bit are present for future use. The reserved bits MUST be set to 0x0.

F: One bit. Indicates which POT-profile is active. 0 means the even POT-profile is active, 1 means the odd POT-profile is active.

Length: Length of the variable metadata, in 4-octet words. Here the length is 4.

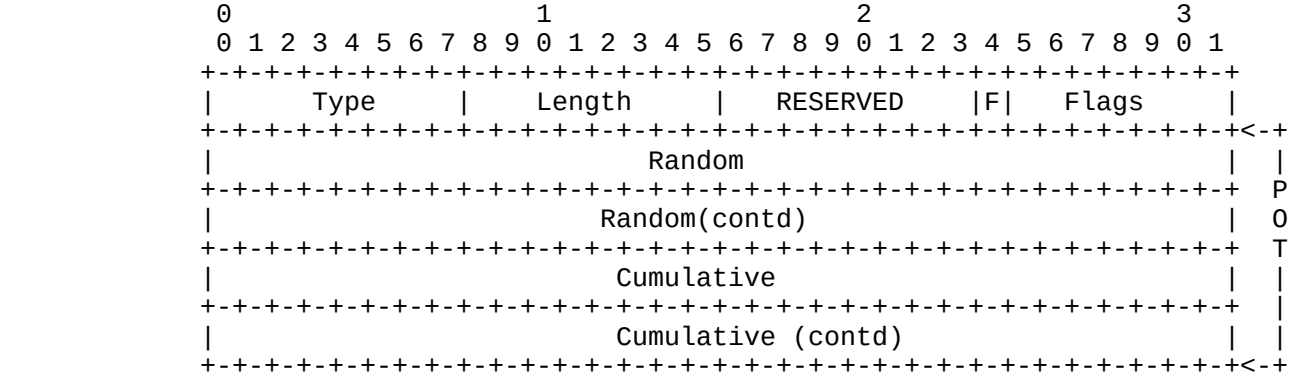
Random: 64-bit Per packet Random number.

Cumulative: 64-bit Cumulative that is updated by the Service Functions.

6. In-band OAM Metadata Transport in Segment Routing

6.1. In-band OAM in SR with IPv6 Transport

Similar to NSH, a service chain or path defined using Segment Routing for IPv6 can be verified using the in-band OAM "Proof of Transit" approach. The Segment Routing Header (SRH) for IPv6 offers the ability to transport TLV structured data, similar to what NSH does (see [I-D.ietf-6man-segment-routing-header]). A new "POT TLV" is defined for the SRH which is to carry proof of transit in-band OAM data.



Type: To be assigned by IANA.

Length: 18.

RESERVED: 8 bits. SHOULD be unset on transmission and MUST be ignored on receipt.

F: 1 bit. Indicates which POT-profile is active. 0 means the even POT-profile is active, 1 means the odd POT-profile is active.

Flags: 8 bits. No flags are defined in this document.

Random: 64-bit per packet random number.

Cumulative: 64-bit cumulative value that is updated at specific nodes that form the service path to be verified.

6.2. In-band OAM in SR with MPLS Transport

In-band OAM "Proof of Transit" data can also be carried as part of the MPLS label stack. Details will be addressed in a future version of this document.

7. IANA Considerations

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

8. Manageability Considerations

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

9. Security Considerations

Security considerations will be addressed in a later version of this document. For a discussion of security requirements of in-band OAM, please refer to [draft-brockners-inband-oam-requirements].

10. Acknowledgements

The authors would like to thank Steve Youell, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, and Andrew Yourtchenko for the comments and advice. For the IPv6 encapsulation, this document leverages and builds on top of several concepts described in [draft-kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

11. References

11.1. Normative References

[draft-brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., and S. Dara, "Requirements for in-band OAM", July 2016.

11.2. Informative References

[draft-brockners-inband-oam-data]
Brockners, F., Bhandari, S., Pignataro, C., and H. Gredler, "Data Formats for in-band OAM", July 2016.

[draft-brockners-proof-of-transit]
Brockners, F., Bhandari, S., and S. Dara, "Proof of transit", July 2016.

[draft-kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6),Hop-by-Hop Option Extension", November 2000.

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

[I-D.hildebrand-spud-prototype]
Hildebrand, J. and B. Trammell, "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in progress), March 2015.

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Field, B., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., and D. Lebrun, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-01 (work in progress), March 2016.

[I-D.ietf-ippm-6man-pdm-option]
Elkins, N., Hamilton, R., and m. mackermann@bcbsm.com, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", draft-ietf-ippm-6man-pdm-option-03 (work in progress), June 2016.

[I-D.ietf-nvo3-vxlan-gpe]
Kreeger, L. and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-02 (work in progress), April 2016.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-05 (work in progress), May 2016.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-09 (work in progress), July 2016.

[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.

[P4] Kim, , "P4: In-band Network Telemetry (INT)", September 2015.

[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>.

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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 Working Group
Internet-Draft
Intended status: Experimental
Expires: January 9, 2017

F. Brockners
S. Bhandari
S. Dara
C. Pignataro
Cisco

July 8, 2016

Proof of Transit
draft-brockners-proof-of-transit-00

Abstract

Several technologies such as traffic engineering, service function chaining, or policy based routing, are used to steer traffic through a specific, user-defined path. This document defines mechanisms to securely prove that traffic transited the defined path. The mechanisms allow to securely verify whether all packets traversed all those nodes of a given path that they are supposed to visit.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction

Several deployments use traffic engineering, policy routing, segment routing or 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 LISP, 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-band" operational data is also referred to as "proof of transit data", or POT data. The POT data is updated at every required node and is used to verify whether a packet traversed all required nodes. A particular set of nodes "to be verified" is either described by a set of secret keys, or a set of shares of a single secret. Nodes on the path retrieve their individual keys or shares of a key (using for e.g., Shamir's Secret Sharing scheme) from a central controller. The complete key set is only known to the controller and a verifier node, which is typically the ultimate node on a path that performs

verification. Each node in the path uses its secret or share of the secret to update the POT data of the packets as the packets pass through the node. When the verifier receives a packet, it uses its key(s) along with data found in the packet to validate whether the packet traversed the path correctly.

2. Conventions

Abbreviations used in this document:

MTU: Maximum Transmit Unit

SR: Segment Routing

NSH: Network Service Header

SFC: Service Function Chain

POT: Proof of Transit

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

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-tupel classifier), or an identifier which is already present in the packet (e.g., path or service identifier, flow-label, etc.).

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

3.1. Basic Idea

The method relies on adding POT data to all packets that traverse a path. The added POT data allows a verifying node (egress node) to check whether a packet traversed the identified set of nodes on a path correctly or not. Security mechanisms are natively built into

the generation of the POT data to protect against misuse (i.e. configuration mistakes, malicious administrators playing tricks with routing, capturing, spoofing and replaying packets). The mechanism for POT leverages "Shamir's secret sharing scheme" [SSS].

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

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

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

3.2. Solution Approach

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

solution leverages finite field arithmetic in a field of size "prime number".

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

3.2.1. Setup

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

3.2.2. In Transit

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

3.2.3. Verification

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

3.3. Example for Illustration

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

3.3.1. Basic Version

Assumption: We like to verify that packets pass through 3 nodes. Consequently we need a polynomial of degree 2.

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

3.3.1.1. Secret Shares

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

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

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

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

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

3.3.1.2. Lagrange Polynomials

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

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

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

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

3.3.1.3. LPC Computation

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

LPC(x0) = (10/3) mod 53 = 21

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

LPC(x2) = (8/3) mod 53 = 38

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

3.3.1.4. Reconstruction

Reconstruction of the polynomial is well defined as

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

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

SECRET = (y0*LPC(l0)+y1*LPC(l1)+y2*LPC(l2)) mod 53.

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

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

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

3.3.1.5. Verification

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

3.3.2. Enhanced Version

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

3.3.2.1. Random Polynomial

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

3.3.2.2. Reconstruction

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

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

Lets 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 source: The fields RND = 45. CML = 0.

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

CML = 0 + ((28 + 46)* 21) mod 53 = 17.

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

CML = 17 + ((17 + 21)*48) mod 53 = 17 + 22 = 39.

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

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

The verification using CML is discussed in next section.

3.3.2.3. Verification

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

VERIFY = (SECRET + RND) mod Prime, with Prime = 53 here.

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

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

3.4. 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).

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. The 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.

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.

4. Sizing the Data for Proof of Transit

Proof of transit requires transport of two data records 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 records 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 records, the time between a

renewal of secrets could be as long as 3,100 years, even when running at 100 Gbps.

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

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

5. Node Configuration

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

5.1. Procedure

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

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

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

5.2. YANG Model

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

```
module ietf-pot-profile {

  yang-version 1;

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

  prefix ietf-pot-profile;

  organization "IETF xxx Working Group";

  contact "";

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

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

```
}

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


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

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

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

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

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

    leaf lpc {
```

```
          type uint64;
          mandatory true;
          description
            "Lagrange Polynomial Coefficient";
        }

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

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

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

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

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

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

        leaf active-profile-index {
```

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

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

6. IANA Considerations

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

7. Manageability Considerations

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

8. Security Considerations

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

8.1. Proof of Transit

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

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

- The Shares of the SECRET (i.e., points on POLY-1) are kept secret by individual nodes.

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.

8.2. Anti Replay

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 as the operations are done modulo prime. The solution approach is flexible, one could use different points on POLY-1 or different polynomials as POLY-1 across different paths, traffic profiles or service chains.

Doing differential analysis across packets could be mitigated with POLY-2 being be random. Further an attacker could reuse a set of RND and all the intermediate CML values to bypass certain nodes in later packets. Such attacks could be avoided by carefully choosing POLY-2 as a timestamp concatenated with a random string. The verifier could use the timestamp to mitigate reuse within a time window.

8.3. Anti Tampering

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

8.4. Recycling

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

8.5. 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.

8.6. 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-records "CML" and "RND" which are carried within the data packet, the regular intervals are expected to be quite long (e.g., at 100 Gbps, a profile would only be used up after 3100 years) - see Section 4 above, thus even a "headless" operation without a Controller can be considered feasible. In such a case, the Controller would only be used for the initial configuration of the POT-profiles.

8.7. 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.

8.7.1. Node Ordering

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

8.7.2. 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 Steve Youell, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, and Andrew Yourtchenko for the comments and advice.

10. Normative References

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

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: sadara@cisco.com

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

Email: cpignata@cisco.com

Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: December 29, 2016

D. Dolson
Sandvine
S. Homma
NTT
D. Lopez
Telefonica I+D
M. Boucadair
Orange
D. Liu
Alibaba Group
T. Ao
ZTE Corporation
V. Vu
SSU
June 27, 2016

Hierarchical Service Function Chaining (hSFC)
draft-dolson-sfc-hierarchical-06

Abstract

Hierarchical Service Function Chaining (hSFC) is a network architecture allowing an organization to compartmentalize a large-scale network into multiple domains of administration.

The goals of hSFC are to make a large-scale network easier to reason about, simpler to control and to able support independent functional groups within large operators.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 29, 2016.

Table of Contents

1. Introduction

Service Function Chaining (SFC) is a technique for prescribing differentiated traffic forwarding policies within an SFC-enabled domain. SFC is described in detail in the SFC architecture document [RFC7665], and is not repeated here.

In this document we consider the difficult problem of implementing SFC across a large, geographically dispersed network comprised of millions of hosts and thousands of network forwarding elements, involving multiple operational teams (with varying functional responsibilities). We expect asymmetrical routing is inherent in the network, while recognizing that some Service Functions (SFs) require bidirectional traffic for transport-layer sessions (e.g., NATs, firewalls). We assume that some Service Function Paths (SFPs) need to be selected on the basis of application-specific data visible to the network, with transport-layer coordinate (typically, 5-tuple) stickiness to specific SF instances.

Note: in this document, the notion of the "path" of a packet is the series of SF instances traversed by a packet. The means of delivering packets between SFs (the forwarding mechanisms enforced in the underlying network) is not relevant to the discussion.

Difficult problems are often made easier by decomposing them in a hierarchical (nested) manner. So instead of considering an omniscient SFC Control Plane that can manage (create, withdraw, supervise, etc.) complete SFPs from one end of the network to the other, we decompose the network into smaller sub-domains. Each sub-domain may support a subset of the network applications or a subset of the users. The criteria for determining decomposition into SFC-enabled sub-domains are beyond the scope of this document.

Note that decomposing a network into multiple SFC-enabled domains should permit end-to-end visibility of SFs and SFPs. Decomposition should also be implemented with special care to ease monitoring and troubleshooting of the network and services as a whole.

An example of simplifying a network by using multiple SF domains is further discussed in [I-D.ietf-sfc-dc-use-cases].

We assume the SFC-aware nodes use NSH [I-D.ietf-sfc-nsh] or a similar labeling mechanism.

The "domains" discussed in this document are assumed to be under control of a single organization, such that there is a strong trust relationship between the domains. The intention of creating multiple domains is to improve the ability to operate a network. It is

outside of the scope of the document to consider domains operated by different organizations.

2. Hierarchical Service Function Chaining (hSFC)

A hierarchy has multiple levels. The top-most level encompasses the entire network domain to be managed, and lower levels encompass portions of the network.

2.1. Top Level

Considering the example depicted in Figure 1, a top-level network domain includes SFC data plane components distributed over a wide area, including:

- Classifiers (CFs),
- Service Function Forwarders (SFFs) and
- Sub-domains.

For the sake of clarity, components of the underlay network are not shown; an underlay network is assumed to provide connectivity between SFC data plane components.

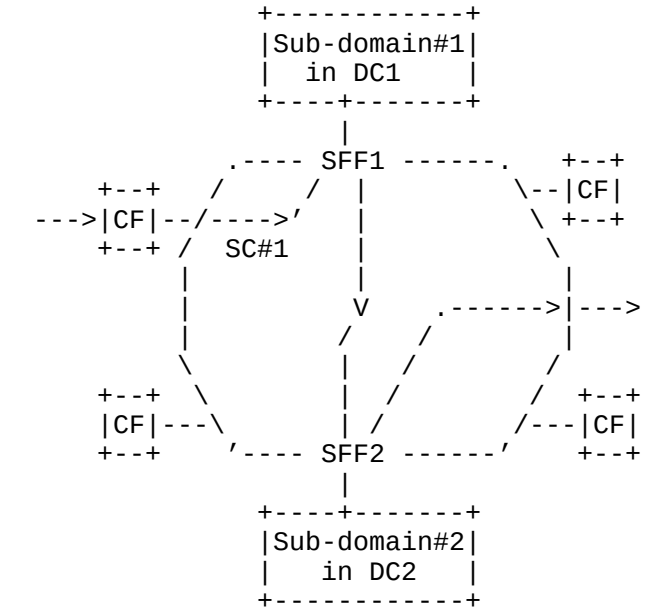
Top-level SFPs carry packets from classifiers through a series of SFFs and sub-domains, with the operations within sub-domains being opaque to the higher levels.

We expect the system to include a top-level control-plane having responsibility for configuring forwarding and classification (see [I-D.ietf-sfc-control-plane]). The top-level Service Chaining control-plane manages end-to-end service chains and associated service function paths from network edge points to sub-domains and configuring top-level classifiers at a coarse level (e.g., based on source or destination host) to forward traffic along paths that will transit appropriate sub-domains. Figure 1 shows one possible service chain passing from edge, through two sub-domains, to network egress. The top-level control plane does not configure classification or forwarding within the sub-domains.

At this network-wide level, the number of SFPs required is a linear function of the number of ways in which a packet is required to traverse different sub-domains and egress the network. Note that the various paths which may be taken within a sub-domain are not represented by distinct network-wide SFPs; specific policies at the ingress nodes of each sub-domain bind flows to sub-domain paths.

Packets are classified at the edge of the network to select the paths by which sub-domains are to be traversed. At the ingress of each sub-domain, paths are reclassified to select the paths by which SFs in the sub-domain are to be traversed. At the egress of each sub-domain, packets are returned to the top-level paths. Contrast this with an approach requiring the top-level classifier to select paths to specify all of the SFs in each sub-domain.

It should be assumed that some SFs require bidirectional symmetry of paths (see more in Section 4). Therefore the classifiers at the top level must be configured with policies ensuring outgoing packets take the reverse path of incoming packets through sub-domains.



One path is shown from edge classifier to SFF1 to Sub-domain#1 (residing in data-center1) to SFF1 to SFF2 (residing in data-center 2) to Sub-domain#2 to SFF2 to network egress.

Figure 1: Network-wide view of top level of hierarchy

2.2. Lower Levels

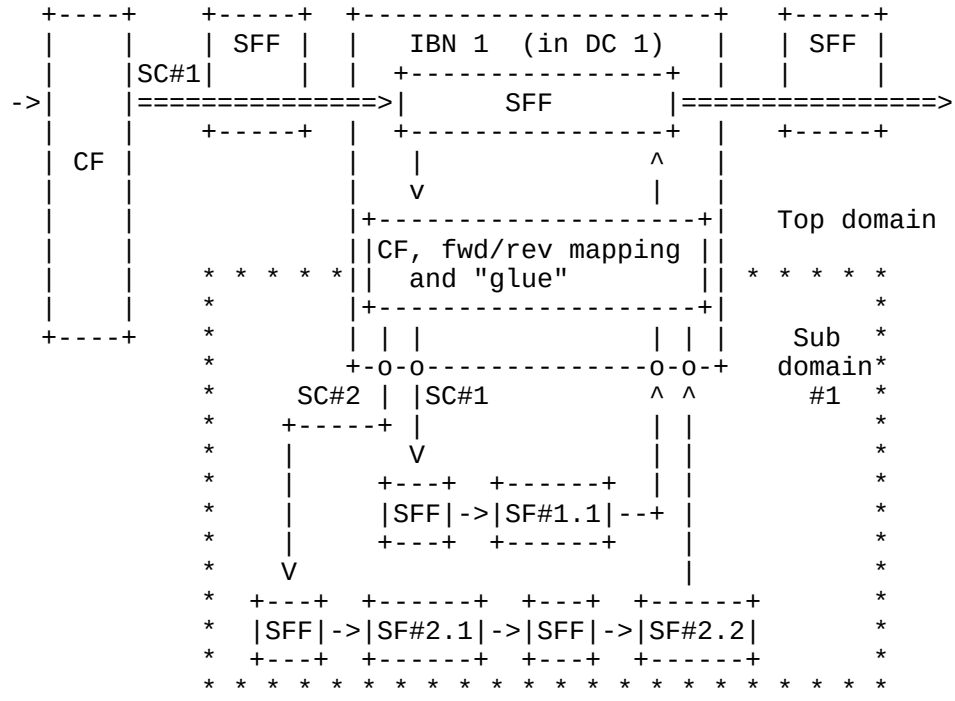
Each of the sub-domains in Figure 1 is an SFC-enabled domain.

Unlike the top level, data packets entering the sub-domain are already SFC-encapsulated. Figure 2 shows a sub-domain interfaced with a higher-level domain by means of an Internal Boundary Node

(IBN). It is the purpose of the IBN to apply classification rules and direct the packets to the selected local SFPs terminating at an egress IBN. The egress IBN finally restores packets to the original SFC shim and hands them off to SFFs.

Each sub-domain intersects a subset of the total paths that are possible in the higher-level domain. An IBN is concerned with higher-level paths, but only those traversing its sub-domain. A top-level control element may configure the IBN as an SF (i.e., the IBN plays the SF role in the top-level domain).

Each sub-domain is likely to have a control-plane that can operate independently of the top-level control-plane. The sub-domain control-plane configures the classification and forwarding rules in the sub-domain. The classification rules reside in the IBN, where SFC encapsulation of the top-level domain is converted to/from SFC encapsulation of the lower-level domain.



*** Sub-domain boundary; === top-level chain; --- low-level chain.

Figure 2: Sub-domain within a higher-level domain

If desired, the pattern can be applied recursively. For example, SF#1.1 in Figure 2 could be a sub-domain of the sub-domain.

3. Internal Boundary Node (IBN)

A network element termed "Internal Boundary Node" (IBN) bridges packets between domains. It behaves as an SF to the higher level, and looks like a classifier and end-of-chain to the lower level.

To achieve the benefits of hierarchy, the IBN should be applying more granular traffic classification rules at the lower level than the traffic passed to it. This means that the number of SFPs within the lower level is greater than the number of SFPs arriving to the IBN.

The IBN is also the termination of lower-level SFPs. This is because the packets exiting lower-level SF paths must be returned to the higher-level SF paths and forwarded to the next hop in the higher-level domain.

3.1. IBN Path Configuration

An operator of a lower-level domain may be aware of which high-level paths transit their domain, or they may wish to accept any paths.

When packets enter the sub-domain, the Service Path Identifier (SPI) and Service Index (SI) are re-marked according to the path selected by the classifier.

After exiting a path in the sub-domain, packets can be restored to an original upper-level SFP by these methods:

1. Saving SPI and SI in transport-layer flow state,

2. Pushing SPI and SI into metadata,

3. Using unique lower-level paths per upper-level path coordinates,

4. Nesting NSH headers, encapsulating the higher-level NSH headers within the lower-level NSH headers,

5. Saving upper-level by a flow ID and placing an hSFC flow ID into metadata,

3.1.1. Flow-Stateful IBN

An IBN can be flow-aware, returning packets to the correct higher-level SFP on the basis of the transport-layer coordinates (typically, a 5-tuple) of packets exiting the lower-level SFPs.

When packets are received by the IBN on a higher-level path, the encapsulated packets are parsed for IP and transport-layer (TCP, UDP, etc.) coordinates. State is created, indexed by these coordinates ({source-IP, destination-IP, source-port, destination-port and transport protocol} typically). The state contains at least critical fields of the encapsulating SFC header (or perhaps the entire header).

The simplest approach has the packets return to the same IBN at the end of the chain that classified the packet at the start of the chain. This is because the required transport-coordinates state is rapidly changing and most efficiently kept locally. If the packet is returned to a different IBN for egress, transport-coordinates state must be synchronized between the IBNs.

When a packet returns to the IBN at the end of a chain, the SFC header is removed, the packet is parsed for IP and transport-layer coordinates, and state is retrieved from them. The state contains the information required to forward the packet within the higher-level service chain.

State cannot be created by packets arriving from the lower-level chain; when state cannot be found for such packets, they must be dropped.

This stateful approach is limited to use with SFs that retain the transport coordinates of the packet. This approach cannot be used with SFs that modify those coordinates (e.g., NATs) or otherwise create packets for new coordinates other than those received (e.g., as an HTTP cache might do to retrieve content on behalf of the original flow). In both cases, the fundamental problem is the inability to forward packets when state cannot be found for the packet transport-layer coordinates.

In the stateful approach, there are issues caused by having state, such as how long the state should be maintained (it must time out eventually), as well as whether the state needs to be replicated to other devices to create a highly available network.

It is valid to consider the state to be disposable after failure, since it can be re-created by each new packet arriving from the higher-level domain. For example, if an IBN loses all flow state, the state is re-created by an end-point retransmitting a TCP packet.

If an SFC domain handles multiple network regions (e.g., multiple private networks), the coordinates may be augmented with additional parameters, perhaps using some metadata to identify the network region.

In this stateful approach, it is not necessary for the sub-domain's control-plane to modify paths when higher-level paths are changed. The complexity of the higher-level domain does not cause complexity in the lower-level domain.

Since it doesn't depend on NSH in the lower domain, this flow-stateful approach can be applied to translation methods of converting NSH to other forwarding techniques. (Refer to Section 6.)

3.1.2. Encoding Upper-Level Paths in Metadata

An IBN can push the upper-level Service Path Identifier (SPI) and Service Index (SI) (or encoding thereof) into a metadata field of the lower-level encapsulation (e.g., placing upper-level path information into a metadata field of NSH). When packets exit the lower-level path, the upper-level SPI and SI can be restored from the metadata retrieved from the packet.

This approach requires the SFs in the path to be capable of forwarding the metadata and appropriately attaching metadata to any packets injected for a flow.

Using new metadata may inflate packet size when variable-length metadata (type 2 from NSH [I-D.ietf-sfc-nsh]) is used.

It is conceivable that the MD-type 1 Mandatory Context Header fields of NSH [I-D.ietf-sfc-nsh] are not all relevant to the lower-level domain. In this case, one of the metadata slots of the Mandatory Context Header could be repurposed within the lower-level domain, and restored when leaving.

In this metadata approach, it is not necessary for the sub-domain's control element to modify paths when higher-level paths are changed. The complexity of the higher-level domain does not cause complexity in the lower-level domain.

3.1.3. Using Unique Paths per Upper-Level Path

In this approach, paths within the sub-domain are constrained so that a SPI (of the sub-domain) unambiguously indicates the egress SPI and SI (of the upper domain). This allows the original path information to be restored at sub-domain egress from a look-up table using the sub-domain SPI.

Whenever the upper-level domain provisions a path via the lower-level domain, the lower-level domain controller must provision corresponding paths to traverse the lower-level domain.

A down-side of this approach is that the number of paths in the lower-level domain is multiplied by the number of paths in the higher-level domain that traverse the lower-level domain. I.e., a sub-path must be created for each combination of upper SPI/SI and lower chain.

3.1.4. Nesting Upper-Level NSH within Lower-Level NSH

In this approach, when packets arrive at the IBN in the top-level domain, the classifier in the IBN determines the path for the lower-level domain and pushes the new NSH header in front of the original NSH header.

As shown in Figure 3 the Lower-NSH Header used to forward packets in the lower-level domain precedes the Upper-NSH Header from the top-level domain.

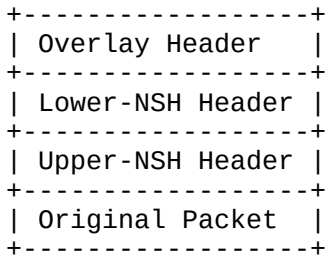


Figure 3: Encapsulation of NSH within NSH

The traffic with the above stack of two-layer-NSH header is to be forwarded according to the Lower-NSH header in the lower-level SFC domain. The Upper-NSH header is preserved in the packets but not used for forwarding. At the last SFF of the chain of the lower-level domain (which resides in the IBN), the Lower-NSH header is removed from the packet, and then the packet is forwarded by the IBN to an SFF of the upper-level domain, which will be forwarded according to the Upper-NSH header.

With such encapsulation, Upper-NSH information is carried along the extent of the lower-level chain without modification.

A benefit of this approach is that it does not require state in the IBN or configuration to encode fields in meta-data.

However, the down-side is it does require SFs in the lower-level domain to be able to parse multiple layers of NSH. If the SF injects

packets, it must also be able to deal with adding appropriate multiple layers of headers to injected packets.

3.1.5. Stateful / Metadata Hybrid

The basic idea of this approach is for the IBN to save upper domain encapsulation information such that it can be retrieved by a unique identifier, termed an "hSFC Flow ID". An example ID is shown in Table 1.

hSFC Flow ID	SPI	SI	Context1	Context2	Context3	Context4
1	45	254	100	2112	12345	7

Table 1: Example Mapping of an hSFC Flow ID to Upper-Level Header

The ID is placed in the metadata in NSH headers of the packet in the lower domain, as shown in Figure 4. When packets exit the lower domain, the IBN uses the ID to retrieve the appropriate NSH encapsulation for returning the packet to the upper domain.

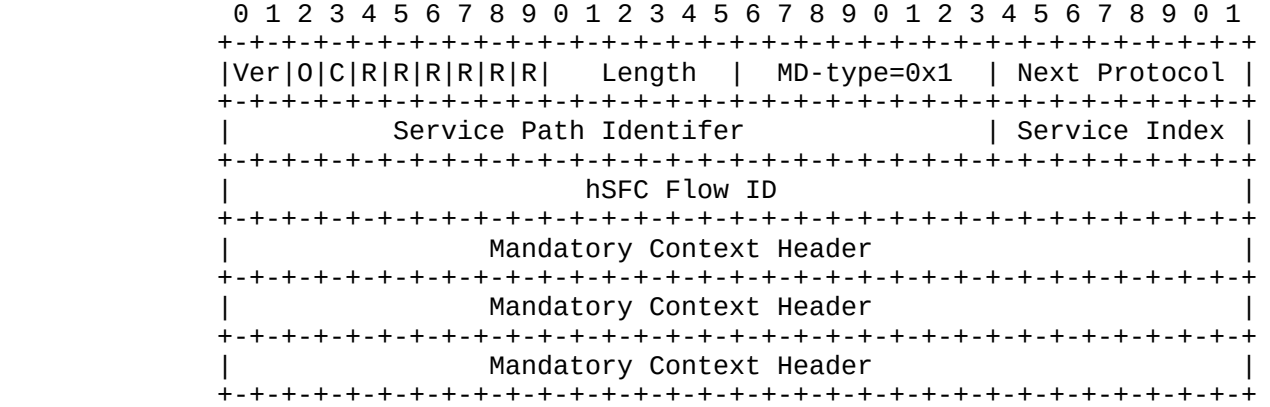


Figure 4: Storing hSFC Flow ID in lower-level metadata

Advantages of this approach include:

- Does not require state based on 5-tuple, so it works with functions that change the IP addresses or ports of a packet such as NATs,

- Does not require all domains to have the same metadata scheme,
- Can be used to restore any upper-domain information, not just service path,
- The lower domain only requires a single item of metadata regardless of the number of items of metadata used in the upper domain. (For MD-Type 1, this leaves 3 slots for use in the lower domain.)
- No special functionality is required of the SF, other than the usual ability to preserve metadata and to apply metadata to injected packets.

Disadvantages include those of other stateful approaches, including state timeout and replication mentioned in Section 3.1.1.

There may be a large number of unique NSH encapsulations to be stored, given that the hSFC Flow ID must represent all of the bits in the upper-level encapsulation. This might consume a lot of memory or create out-of-memory situations in which IDs cannot be created or old IDs are discarded while still in use.

3.2. Gluing Levels Together

The SPI or metadata on a packet received by the IBN may be used as input to reclassification and path selection within the lower-level domain.

In some cases the meanings of the various path IDs and metadata must be coordinated between domains.

One approach is to use well-known identifier values in metadata, communicated by some organizational registry.

Another approach is to use well-known labels for chain identifiers or metadata, as an indirection to the actual identifiers. The actual identifiers can be assigned by control-plane systems. For example, a sub-domain classifier could have a policy, "if pathID=classA then chain packet to path 1234"; the higher-level controller would be expected to configure the concrete higher-level pathID for classA.

3.3. Decrementing Service Index

Because the IBN acts as a Service Function to the higher-level domain, it must decrement the Service Index in the NSH headers of the higher-level path.

A good strategy seems to be to do this when the packet is first received by the IBN, before applying any of the strategies of Section 3.1, immediately prior to classification.

4. Sub-domain Classifier

Within the sub-domain (referring to Figure 2), after the IBN removes higher-level encapsulation from incoming packets, it sends the packets to the classifier, which selects the encapsulation for the packet within the sub-domain.

One of the goals of the hierarchical approach is to make it easy to have transport-flow-aware service chaining with bidirectional paths. For example, it is desired that for each TCP flow, the client-to-server packets traverse the same SFs as the server-to-client packets, but in the opposite sequence. We call this bidirectional symmetry. If bidirectional symmetry is required, it is the responsibility of the control-plane to be aware of symmetric paths and configure the classifier to chain the traffic in a symmetric manner.

Another goal of the hierarchical approach is to simplify the mechanisms of scaling in and scaling out service functions. All of the complexities of load-balancing among multiple SFs can be handled within a sub-domain, under control of the classifier, allowing the higher-level domain to be oblivious to the existence of multiple SF instances.

Considering the requirements of bidirectional symmetry and load-balancing, it is useful to have all packets entering a sub-domain to be received by the same classifier or a coordinated cluster of classifiers. There are both stateful and stateless approaches to ensuring bidirectional symmetry.

5. Control Plane Elements

Although control protocols have not yet been standardized, from the point of view of hierarchical service function chaining we have these expectations:

- Each control-plane instance manages a single level of hierarchy of a single domain.
- Each control-plane is agnostic about other levels of hierarchy. This aspect allows humans to reason about the system within a single domain and allows control-plane algorithms to use only domain-local inputs. Top-level control does not need visibility to sub-domain policies, nor does sub-domain control need visibility to higher-level policies.

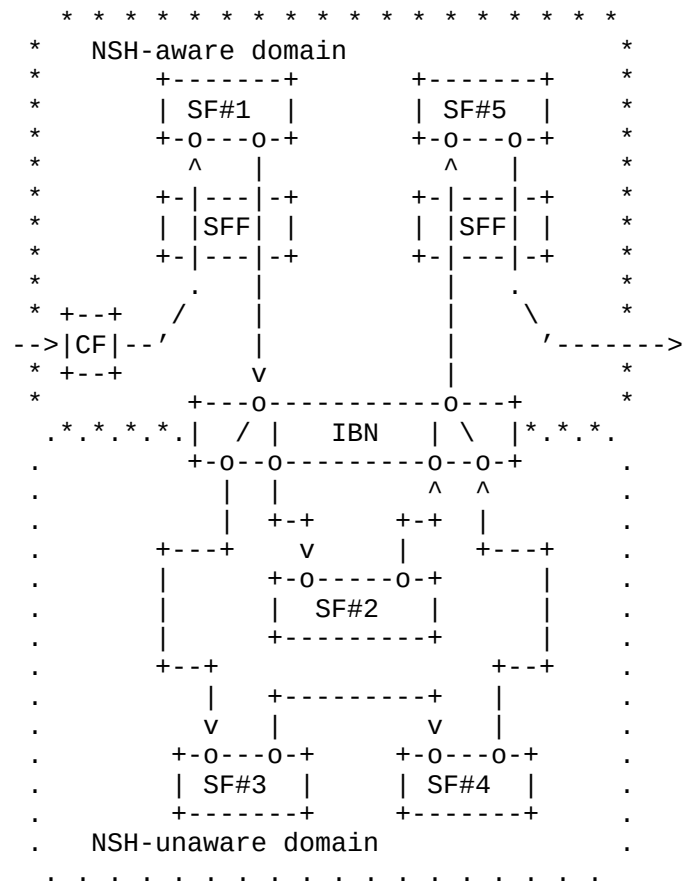
- o Sub-domain control-planes are agnostic about control-planes of other sub-domains. This allows both humans and machines to manipulate sub-domain policy without considering policies of other domains.

Recall that the IBN acts as an SF in the higher-level domain (receiving SF instructions from the higher-level control-plane) and as a classifier in the lower-level domain (receiving classification rules from the sub-domain control-plane). In this view, it is the IBN that glues the layers together.

The above expectations are not intended to prohibit network-wide control. A control hierarchy can be envisaged to distribute information and instructions to multiple domains and sub-domains. Control hierarchy is outside the scope of this document.

6. Extension for Adopting to NSH-Unaware Service Functions

The hierarchical approach can be used for dividing networks into NSH-aware and NSH-unaware domains by converting NSH encapsulation to other forwarding techniques (e.g., 5-tuple-based routing with OpenFlow), as shown in Figure 5.



SF#1 and SF#5 are NSH-aware and SF#2, SF#3 and SF#4 are NSH-unaware. In the NSH-unaware domain, packets are conveyed in a format supported by SFs which are deployed there.

Figure 5: Dividing NSH-aware and NSH-unaware domains

6.1. Purpose

This approach is expected to facilitate service chaining in networks in which NSH-aware and NSH-unaware SFs coexist. Some examples of such situations are:

- In a period of transition from legacy SFs to NSH-aware SFs and
- Supporting multi-tenancy.

6.2. Requirements for IBN

In this usage, an IBN classifier is required to have an NSH conversion table for applying packets to appropriate lower-level paths and returning packets to the correct higher-level paths. For example, the following methods would be used for saving/restoring upper-level path information:

- Saving SPI and SI in transport-layer flow state (refer to Section 3.1.1) and
- Using unique lower-level paths per upper-level NSH coordinates (refer to Section 3.1.3).

Especially, the use of unique paths approach would be good for translating NSH to a different forwarding technique in the lower level. A single path in the upper level may be branched to multiple paths in the lower level such that any lower-level path is only used by one upper-level path. This allows unambiguous restoration to the upper-level path.

In addition, an IBN might be required to convert metadata contained in NSH to the format appropriate to the packet in the lower-level path. For example, some legacy SFs identify subscriber based on information of network topology, such as VID, and IBN would be required to create VLAN to packets from metadata if subscriber identifier is conveyed as metadata in higher-level domains.

Other fundamental functions required as IBN (e.g., maintaining metadata of upper level or decrementing Service Index) are same as normal usage.

7. Acknowledgements

The concept of Hierarchical Service Path Domains was introduced in [I-D.homma-sfc-forwarding-methods-analysis] as a means to improve scalability of service chaining in large networks.

The concept of nested NSH headers was introduced in [I-D.ao-sfc-for-dc-interconnect] as a means of creating hierarchical SFC in a data center.

The authors would like to thank the following individuals for providing valuable feedback:

Ron Parker

Christian Jacquenet

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

Hierarchical service function chaining makes use of service chaining architecture, and hence inherits the security considerations described in the architecture document.

Furthermore, hierarchical service function chaining inherits security considerations of the data-plane protocols (e.g., NSH) and control-plane protocols used to realize the solution.

The systems described in this document bear responsibility for forwarding internet traffic. In some cases the systems are responsible for maintaining separation of traffic in private networks.

This document describes systems within different domains of administration that must have consistent configurations in order to properly forward traffic and to maintain private network separation. Any protocol designed to distribute the configurations must be secure from tampering.

All of the systems and protocols must be secure from modification by untrusted agents.

Security considerations related to the control plane are discussed in [I-D.ietf-sfc-control-plane].

10. References

10.1. Normative References

[I-D.ietf-sfc-control-plane]
Boucadair, M., "Service Function Chaining (SFC) Control Plane Components & Requirements", draft-ietf-sfc-control-plane-06 (work in progress), May 2016.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-05 (work in progress), May 2016.

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

10.2. Informative References

[I-D.ao-sfc-for-dc-interconnect]
Ao, T. and W. Bo, "Hierarchical SFC for DC Interconnection", draft-ao-sfc-for-dc-interconnect-01 (work in progress), October 2015.

[I-D.homma-sfc-forwarding-methods-analysis]
Homma, S., Naito, K., Lopez, D., Stiemerling, M., Dolson, D., Gorbunov, A., Leymann, N., Bottorff, P., and d. don.fedyk@hpe.com, "Analysis on Forwarding Methods for Service Chaining", draft-homma-sfc-forwarding-methods-analysis-05 (work in progress), January 2016.

[I-D.ietf-sfc-dc-use-cases]
Surendra, S., Tufail, M., Majee, S., Captari, C., and S. Homma, "Service Function Chaining Use Cases In Data Centers", draft-ietf-sfc-dc-use-cases-02 (work in progress), January 2015.

Appendix A. Examples of Hierarchical Service Function Chaining

The advantage of hierarchical service function chaining compared with normal or flat service function chaining is that it can reduce the management complexity significantly. This section discusses examples that show those advantages.

A.1. Reducing the Number of Service Function Paths

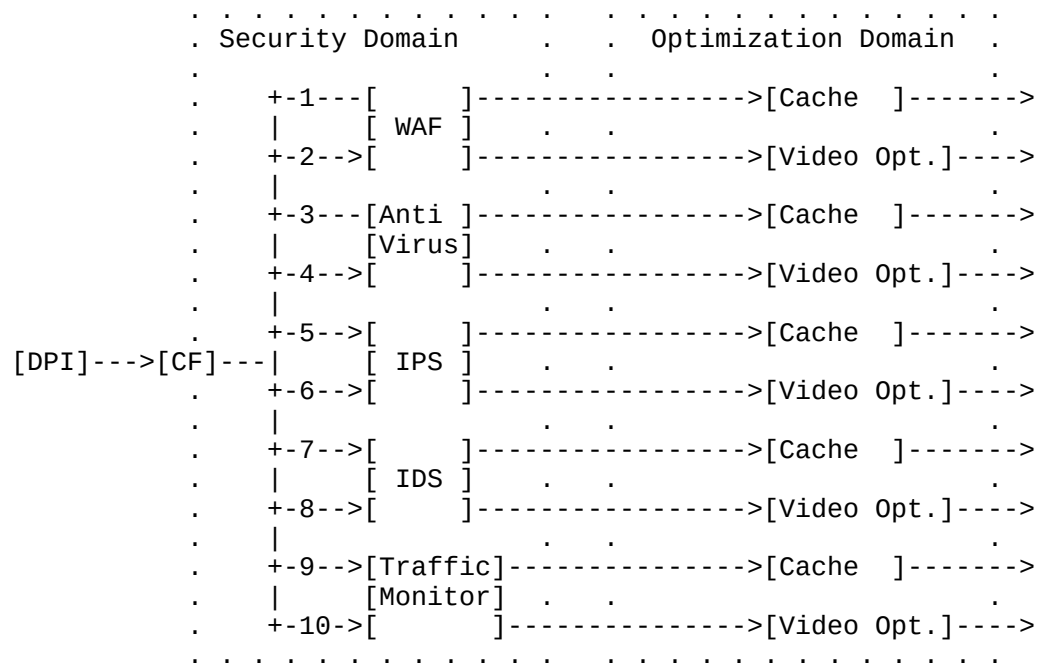
In this case, hierarchical service function chaining is used to simplify service function chaining management by reducing the number of Service Function Paths.

As shown in Figure 6, there are two domains, each with different concerns: a Security Domain that selects Service Functions based on network conditions and an Optimization Domain that selects Service Functions based on traffic protocol.

In this example there are five security functions deployed in the Security Domain. The Security Domain operator wants to enforce the five different security policies, and the Optimization Domain operator wants to apply different optimizations (either cache or video optimization) to each of these two types of traffic. If we use

flat SFC (normal branching), 10 SFPs are needed in each domain. In contrast, if we use hierarchical SFC, only 5 SFPs in Security Domain and 2 SFPs in Optimization Domain will be required, as shown in Figure 7.

In the flat model, the number of SFPs is the product of the number of functions in all of the domains. In the hSFC model, the number of SFPs is the sum of the number of functions. For example, adding a "bypass" path in the Optimization Domain would cause the flat model to require 15 paths (5 more), but cause the hSFC model to require one more path in the Optimization Domain.



The classifier must select paths that determine the combination of Security and Optimization concerns. 1:WAF+Cache, 2:WAF+VideoOpt, 3:AntiVirus+Cache, 4:AntiVirus+VideoOpt, 5: IPS+Cache, 6:IPS+VideoOpt, 7:IDS+Cache, 8:IDS+VideoOpt, 9:TrafficMonitor+Cache, 10:TrafficMonitor+VideoOpt

Figure 6: Flat SFC (normal branching)

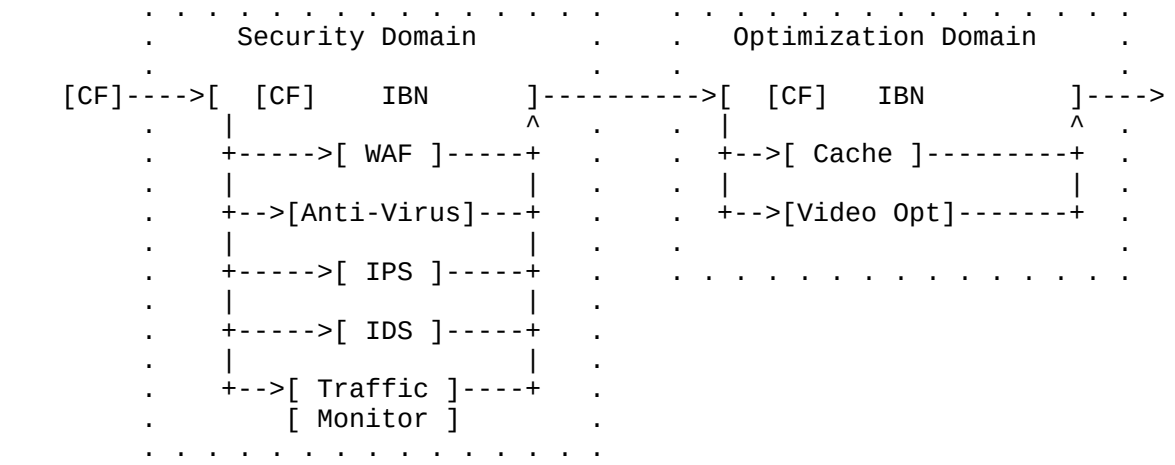


Figure 7: Simplified path management with Hierarchical SFC

A.2. Managing a Distributed Data-Center Network

Hierarchical service function chaining can be used to simplify inter-data-center SFC management. In the example of Figure 8, shown below, there is a central data center (Central DC) and multiple local data centers (Local DC#1, #2, #3) that are deployed in a geographically distributed manner. All of the data centers are under a single administrative domain.

The central DC may have some service functions that the local DC needs, such that the local DC needs to chain traffic via the central DC. This could be because:

- Some service functions are deployed as dedicated hardware appliances, and there is a desire to lower the cost (both CAPEX and OPEX) of deploying such service functions in all data centers.

- Some service functions are being trialed, introduced or otherwise handle a relatively small amount of traffic. It may be cheaper to manage these service functions in a single central data center and steer packets to the central data center than to manage these service functions in all data centers.

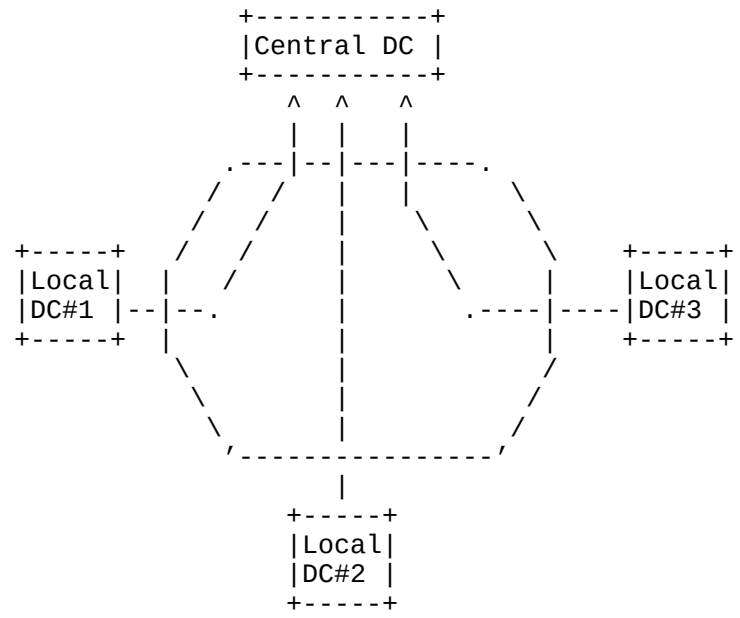


Figure 8: Simplify inter-DC SFC management

For large data center operators, one local DC may have tens of thousands of servers and hundred of thousands of virtual machines. SFC can be used to manage user traffic. For example, SFC can be used to classify user traffic based on service type, DDoS state etc.

In such large scale data center, using flat SFC is very complex, requiring a super-controller to configure all data centers. For example, any changes to Service Functions or Service Function Paths in the central DC (e.g., deploying a new SF) would require updates to all of the Service Function Paths in the local DCs accordingly. Furthermore, requirements for symmetric paths add additional complexity when flat SFC is used in this scenario.

Conversely, if using hierarchical SFC, each data center can be managed independently to significantly reduce management complexity. Service Function Paths between data centers can represent abstract notions without regard to details within data centers. Independent controllers can be used for the top level (getting packets to pass the correct data centers) and local levels (getting packets to specific SF instances).

Authors' Addresses

David Dolson
Sandvine
408 Albert Street
Waterloo, ON N2L 3V3
Canada

Phone: +1 519 880 2400
Email: ddolson@sandvine.com

Shunsuke Homma
NTT, Corp.
3-9-11, Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Email: homma.shunsuke@lab.ntt.co.jp

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Dapeng Liu
Alibaba Group
Beijing 100022
China

Email: max.ldp@alibaba-inc.com

Ting Ao
ZTE Corporation
No.889,Bibo Rd.,Zhangjiang Hi-tech Park
Shanghai 201203
China

Phone: +86-21-688976442
Email: ao.ting@zte.com.cn

Vu Anh Vu
Soongsil University
369 Sangdo-ro
Seoul, Dongjak-gu 06978
Korea

Email: vuva@dcn.ssu.ac.kr

rtgwg
Internet-Draft
Intended status: Informational
Expires: January 8, 2017

N. Kumar
C. Pignataro
D. Kumar
Cisco Systems, Inc.
G. Mirsky
Ericsson
M. Chen
Huawei Technologies
E. Nordmark
Arista Networks
S. Pallagatti
Juniper Networks
D. Mozes
Mellanox Technologies Ltd

July 7, 2016

Overlay OAM Requirements
draft-ooamdt-rtgwg-ooam-requirement-01

Abstract

This document describes a list of functional requirements for Operations Administration and Maintenance (OAM) in various Overlay and Service networks like Service Function Chaining (SFC), Bit Index Explicit Replication (BIER), Network Virtualization over Layer 3 (NVO3).

Status of This Memo

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

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

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

This Internet-Draft will expire on January 8, 2017.

Table of Contents

1. Introduction

We have witnessed and participated in design of new paradigms in the networking that are aimed to address network virtualization, service function chaining, and multicast services. New paradigms require new architectural concepts, principles and components. [RFC7365] defines a framework for Data Center Network Virtualization over Layer 3 (NVO3). [RFC7665] describes the architecture for creating and maintaining Service Function Chains (SFCs) in a network. [I-D.ietf-bier-architecture] defines a stateless multicast architecture for optimal multicast packet forwarding using "Bit Index Explicit Replication" (BIER). These frameworks are defined in a

flexible manner that they are transport agnostic and may be deployed on various underlay networks such as IPv4, IPv6 and MPLS.

The above mentioned new architectural concepts and principles have been combined into new network layers with distinct encapsulation headers. For example, [I-D.ietf-sfc-nsh] defines an encapsulation header as Network Service Header (NSH) to realize Service Function Path. While [RFC7348] (VxLAN) and [RFC7637] (NVGRE) are different encapsulation header proposed for NVO3, [I-D.ietf-nvo3-vxlan-gpe] extends VxLAN further to be used for Service Function Chain (SFC). Similarly, [I-D.ietf-bier-mpls-encapsulation] defines the BIER encapsulation header over MPLS network and [I-D.xu-bier-encapsulation] describes the BIER encapsulation header over IP network.

Introduction of the new Overlay networks, sets forth new Operations, Administration and Maintenance (OAM) requirements that can be addressed by enhancing the existing toolset or developing new protocols. For example, [I-D.ietf-sfc-oam-framework] defines the framework for SFC OAM, [I-D.nordmark-nvo3-transcending-traceroute] proposes a way to perform traceroute in NVO3 networks and [I-D.kumarzheng-bier-ping] proposes on-demand connectivity verification and fault isolation procedure (Ping and Trace) on BIER network.

The goal of this document is to identify and list the OAM requirements commonly applicable to new Overlay networks which can further be used to analyze the existing OAM tools. The identified gaps can be addressed, either through enhancing existing OAM tools and if necessary, constructing new OAM tools, that can be used as a common unified OAM toolset to support and perform various OAM functions including proactive and on-demand path monitoring and service validation on the new Overlay network.

2. Requirements notation

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].

3. Terminology

ECMP: Equal Cost Multipath

SFC: Service Function Chaining

BIER: Bit Index Explicit Replication

NVO3: Network Virtualization over L3

OAM: Operations, Administration and Maintenance

MPLS: Multiprotocol Label Switching

VxLAN: Virtual Extensible Local Area Network

NVGRE: Network Virtualization Using Generic Routing Encapsulation

Centralized Controller: An external standalone or virtual entity with topology awareness and with an ability to interact with network devices for OAM functionality.

Overlay nodes: Network nodes participating in the Overlay network.

Underlay Network or Underlay Layer: The network that provides connectivity between the Overlay nodes. MPLS network providing LSP connectivity between BIER nodes is an example for underlay layer.

Overlay Network or Overlay Layer: A network layer that is built on top another network layer. VxLAN-GPE over IP network is an example for Overlay layer.

4. Detailed Requirement List

This section lists the OAM requirement for different Overlay networks. The below listed requirement MUST be supported with any underlay transport network:

REQ#1: The listed requirements MUST be supported with any type of transport layer over which the overlay network can be realized.

REQ#2: It MUST be possible to initialize Overlay OAM between any node towards any node(s) in the overlay network.

REQ#3: It SHOULD be possible to initialize an Overlay OAM from a centralized controller.

REQ#4: Overlay OAM MUST support proactive and on-demand OAM monitoring and measurement methods.

REQ#5: Overlay OAM MUST support unidirectional OAM methods for continuity check, connectivity verification and performance measurement.

REQ#6: Overlay OAM packets SHOULD be fate sharing with data traffic, i.e. in-band with the monitored traffic, i.e. follow exactly the same overlay and transport path as data plane traffic, in forward direction, i.e. from ingress toward egress end point(s) of the OAM test.

REQ#7: Overlay OAM MUST support bi-directional OAM methods. Such OAM methods MAY combine in-band monitoring or measurement in forward direction and out-of-band notification in the reverse direction, i.e. from egress to ingress end point of the OAM test.

REQ#8: Overlay OAM MUST support Path Maximum Transmission Unit (MTU) Discovery from the overlay layer over any transport layer.

4.1. Fault Management

4.1.1. Pro-active Fault Management

Availability, not as performance metric, is understood as ability to reach the node, i.e. the fact that path between ingress and egress does exist. Such OAM mechanism also referred as Continuity Check.

REQ#9: Overlay OAM MUST support pro-active monitoring of any virtual node availability in the given overlay network.

REQ#10: Overlay OAM MUST support Remote Defect Indication (RDI) notification by egress to the ingress, i.e. source of continuity checking.

REQ#11: Overlay OAM MUST support connectivity verification. Definition of mis-connectivity defect entry and exit criteria are outside the scope of this document.

4.1.2. On-demand Fault Management

REQ#12: Overlay OAM MUST support fault localization of Loss of Continuity check at Overlay layer.

REQ#13: Overlay OAM MAY support fault localization of Loss of Continuity check at transport layer.

REQ#14: Overlay OAM MUST support tracing path in overlay network through the overlay nodes.

REQ#15: Overlay OAM MAY support tracing path in underlay network connecting overlay border nodes.

REQ#16: Overlay OAM MAY support verification of the mapping between its data plane state and client layer services.

REQ#17: Overlay OAM MUST have the ability to discover and exercise equal cost multipath (ECMP) paths in its underlay network.

REQ#18: Overlay OAM MUST be able to trigger on-demand FM with responses being directed towards initiator of such proxy request.

4.2. Performance Management

Section 3.4 and Section 3.5 of [RFC7799] defines the definition for Active and Passive mode of Performance Measurement (PM) methods. This section lists the requirements for both active and passive PM methods. Passive PM is a measurement method that should not modify the actual data packet processing behavior on underlay and overlay network. Accordingly, it should be supported by the Overlay nodes.

REQ#19: Overlay OAM MUST support active one-way packet delay measurement.

REQ#20: Overlay OAM MUST support passive one-way packet delay measurement.

REQ#21: Overlay OAM MUST support active two-way packet delay measurement.

REQ#22: Overlay OAM MUST support packet delay variation measurement.

REQ#23: Overlay OAM MUST support active end to end packet loss measurement.

REQ#24: Overlay OAM MUST support passive end to end packet loss measurement.

REQ#25: Overlay OAM SHOULD support active per-segment packet delay measurement.

REQ#26: Overlay OAM SHOULD support passive per-segment packet delay measurement.

REQ#27: Overlay OAM SHOULD support active per-segment packet loss measurement.

REQ#28: Overlay OAM SHOULD support passive per-segment packet loss measurement.

REQ#29: Overlay OAM MUST support delivered packet throughput measurement.

4.3. Alarm Indication Suppression

REQ#30: Overlay OAM MUST support defect notification mechanism, like Alarm Indication Signal.

REQ#31: Any virtual node in the given overlay network MAY originate a defect notification addressed to any node in that network.

4.4. Overlay Network Resiliency

REQ#32: Overlay OAM MUST support methods to enable survivability of an overlay network. These recovery methods MAY use protection switching and restoration.

5. IANA Considerations

This document does not propose any IANA consideration.

6. Security Considerations

This document list the OAM requirement for various Overlay encapsulations and may have security implications. For example, if proactive FM is required, the security implication is that a passive eavesdropper can know when the session is down. Or, proactive FM may be used either to launch DoS or to highjack session and impact state, e.g. cause protection switchover. These security implications are natural results of the requirements, and do not depend on the particular implementation. Whether existing security mechanisms of existing protocols proposed to be re-used in OAM for overlay networks are adequate or require enhancements is for further study. New OAM protocols for overlay networks must consider their security mechanism to on per-solution basis.

7. Acknowledgement

The Authors would like to thank Ron Bonico, Tal Mizrahi, Alia Atlas and Saumya Dikshit for their review and comments.

8. References

8.1. Normative References

[I-D.ietf-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-03 (work in progress), January 2016.

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-ietf-bier-mpls-encapsulation-04 (work in progress), April 2016.

[I-D.ietf-nvo3-vxlan-gpe]
Kreeger, L. and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-02 (work in progress), April 2016.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-05 (work in progress), May 2016.

[I-D.ietf-sfc-oam-framework]
Aldrin, S., Krishnan, R., Akiya, N., Pignataro, C., and A. Ghanwani, "Service Function Chaining Operation, Administration and Maintenance Framework", draft-ietf-sfc-oam-framework-01 (work in progress), February 2016.

[I-D.kumarzheng-bier-ping]
Kumar, N., Pignataro, C., Akiya, N., Zheng, L., Chen, M., and G. Mirsky, "BIER Ping and Trace", draft-kumarzheng-bier-ping-03 (work in progress), July 2016.

[I-D.nordmark-nvo3-transcending-traceroute]
Nordmark, E., Appanna, C., and A. Lo, "Layer-Transcending Traceroute for Overlay Networks like VXLAN", draft-nordmark-nvo3-transcending-traceroute-02 (work in progress), March 2016.

[I-D.xu-bier-encapsulation]
Xu, X., somasundaram.s@alcatel-lucent.com, s., Jacquenet, C., and R. Raszuk, "A Transport-Independent Bit Index Explicit Replication (BIER) Encapsulation Header", draft-xu-bier-encapsulation-05 (work in progress), June 2016.

[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>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, DOI 10.17487/RFC7365, October 2014, <http://www.rfc-editor.org/info/rfc7365>.

[RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <http://www.rfc-editor.org/info/rfc7637>.

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

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <http://www.rfc-editor.org/info/rfc7799>.

8.2. Informative References

[I-D.ietf-bier-oam-requirements] Mirsky, G., Nordmark, E., Pignataro, C., Kumar, N., Aldrin, S., Zheng, L., Chen, M., Akiya, N., and S. Pallagatti, "Operations, Administration and Maintenance (OAM) Requirements for Bit Index Explicit Replication (BIER) Layer", draft-ietf-bier-oam-requirements-01 (work in progress), March 2016.

Authors' Addresses

Nagendra Kumar
Cisco Systems, Inc.
7200 Kit Creek Road
Research Triangle Park, NC 27709
US

Email: naikumar@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
7200 Kit Creek Road
Research Triangle Park, NC 27709-4987
US

Email: cpignata@cisco.com

Deepak Kumar
Cisco Systems, Inc.
3700 Cisco Way
SJ
US

Email: dekumar@cisco.com

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Mach Chen
Huawei Technologies

Email: mach.chen@huawei.com

Eric Nordmark
Arista Networks

Email: nordmark@acm.org

Santosh Pallagatti
Juniper Networks

Email: santosh.pallagatti@gmail.com

David Mozes
Mellanox Technologies Ltd

Email: davidm@mellanox.com

SFC
Internet-Draft
Intended status: Standards Track
Expires: October 31, 2016

R. Penno
C. Pignataro
C. Yen
E. Wang
K. Leung
Cisco Systems
D. Dolson
Sandvine

April 29, 2016

Packet Generation in Service Function Chains
draft-penno-sfc-packet-03

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

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

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

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

This Internet-Draft will expire on October 31, 2016.

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. In some cases, devices generate packets without any in-process packet. Packet generation 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:

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

- The SF needs to be able to 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]

- 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

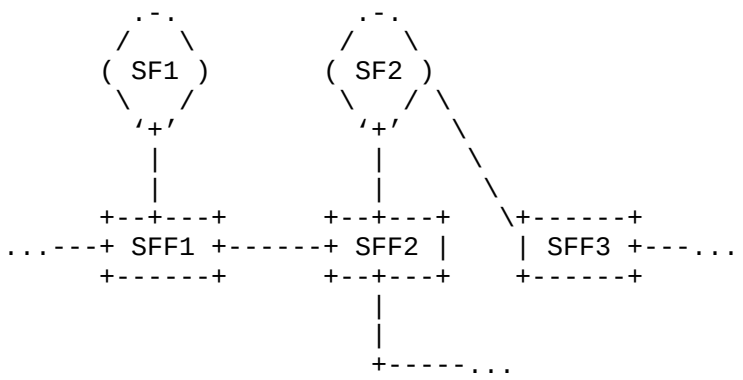
The reader should be familiar with the terms contained in [I-D.ietf-sfc-nsh] ,[I-D.ietf-sfc-architecture] and [I-D.ietf-nvo3-vxlan-gpe]

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.

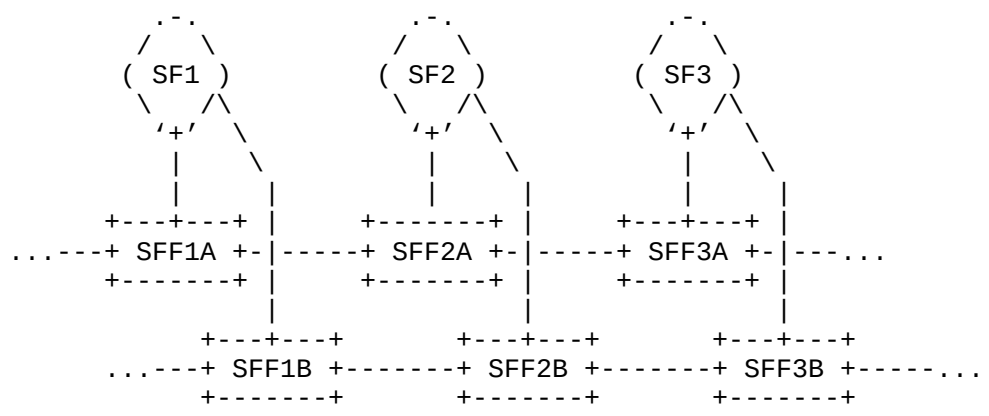
2. After forwarding a packet to an SF, the SFF always has connectivity to the next hop SFF to complete the path. This means the following Figure 1 scenario is not permitted. (SFF2 cannot complete the forward path which contains SFF3 and potentially SFs connected to SFF3.)



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

Figure 1: Arrangement not supported

3. Forward and reverse paths may be required to utilize different service function forwarders. In the Figure 2 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.



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
```

Figure 2: Supported SFF arrangement

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 are free of the constraints in #3, which will simplify the SF logic in handling generated traffic.

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:

- 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.

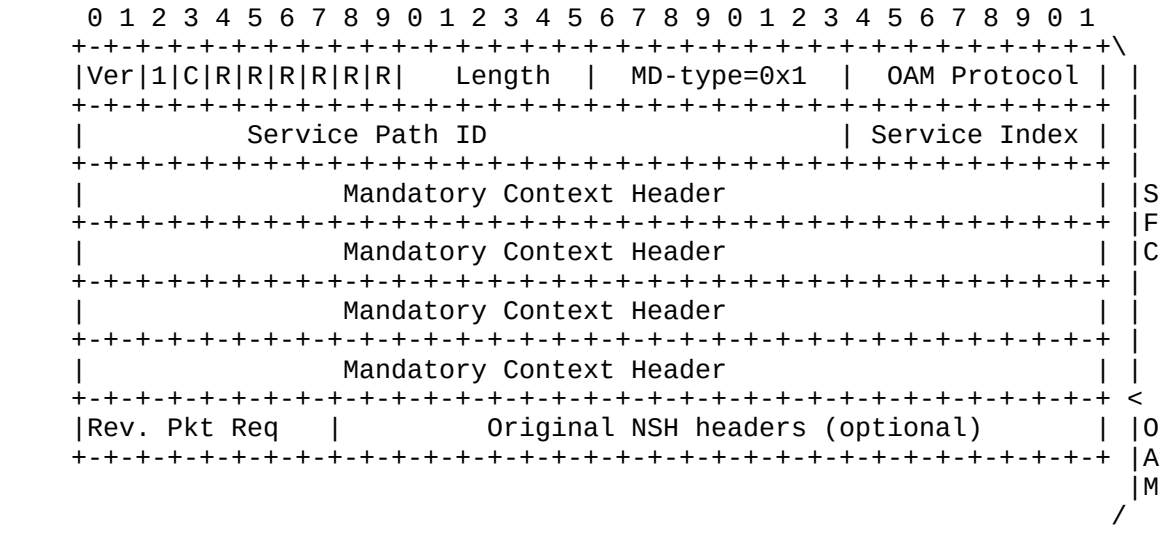
5.2. SF requests SFF cooperation

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



(postamble)

Ver: 1

OAM Bit: 1

Length: 6

MD-Type: 1

Next Protocol: OAM Protocol

Rev. Pkt Req: 1 Reverse packet request

Advantages:

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

Disadvantages:

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

- Process/decode OAM message
- Examine and act on any metadata present in the NSH header
- 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

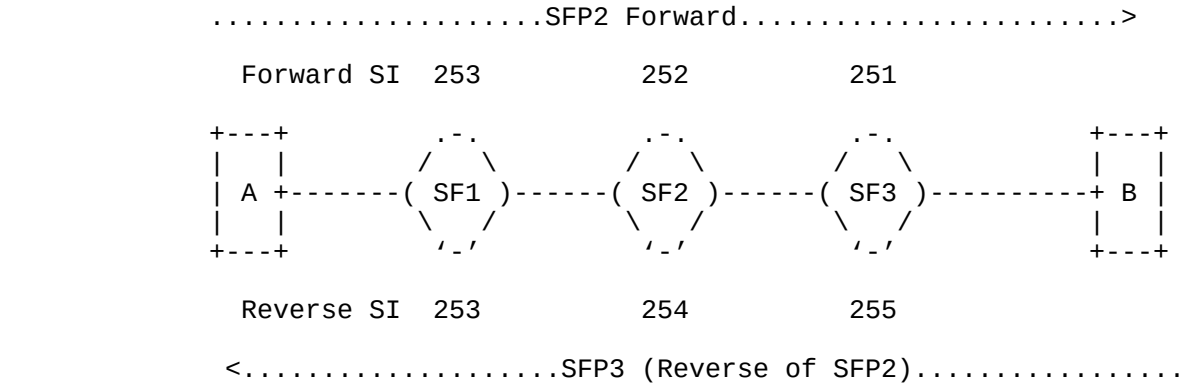
5.3. Classifier Encodes Information

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 cannot 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

Figure 3 below shows a simple SFC with symmetric service paths comprising three SFs.



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

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

Figure 3: 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 in Figure 4 summarize the service indexes as calculated by each SF in the forward and reverse paths respectively.

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

5.3.2. Symmetric Service Paths, Optimized

This approach is effectively the same as Section 5.3.1, but with redundant information removed such that the reverse-path information can be packed into 32 bits. This approach is obtained by observing that the same arithmetic is always done on the same constants of starting_index, symmetric_starting_index and symmetric_number_hops.

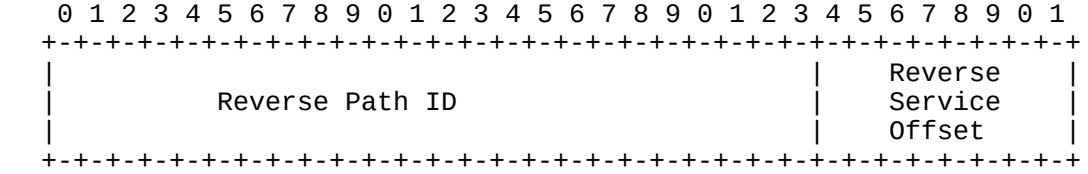
As before, we require symmetric paths, meaning there are two paths that are exactly the reverse of each other. We assume that the classifier at each end has available the following information:

- symmetric path-id (24 bits)
- starting index (8 bits)
- symmetric starting index (8 bits)
- symmetric number of hops, which is the same in both directions (8 bits)

The classifier computes, for each path, a "reverse service offset":

```
# Compute using 8-bit, two's-complement arithmetic:
# (Overflow or underflow are okay)
reverse_service_offset = symmetric_starting_index
                         + starting_index
                         - symmetric_number_of_hops
```

This reverse_service_offset is an 8-bit value that is encoded in metadata along with the 24 bits of reverse_path_id.



Metadata format of reverse_info_metadata (32 bits)

We'll refer to the 32-bit value as reverse_info_metadata. Any Service Function may compute the NSH fields of a reverse packet as follows from the NSH fields of a forward packet.

```
reverse.NSH.Service_Path_ID =
    forward.NSH.reverse_info_metadata.Reverse_Path_ID
# Compute using 8-bit two's-complement arithmetic:
# (Overflow or underflow are okay)
reverse.NSH.Service_Index :=
    forward.NSH.reverse_info_metadata.Reverse_Service_Offset
    - forward.NSH.Service_Index - 1
reverse.NSH.reverse_info_metadata.Reverse_Service_Offset =
    forward.NSH.reverse_info_metadata.Reverse_Service_Offset
reverse.NSH.reverse_info_metadata.Reverse_Path_ID =
    forward.NSH.Service_Path_ID
```

As you can see, this approach has the convenient property that the reverse_info_metadata can be determined by a Service Function while being agnostic about both forward and reverse paths.

Using the example of Section 5.3.1, these values are used for the SFP2 Forward path:

- starting_index=253
- symmetric_starting_index=255
- symmetric_number_of_hops=3
- reverse_service_offset=(253+255-3)=249 in 8-bit two’s complement arithmetic

At SF2 on the SFP2 Forward path, where the service index is 251 after decrementing the index, the reverse service index is calculated as:

- reverse_service_index = 249-251-1 = 253 using 8-bit two’s complement arithmetic

This is the correct index to forward to SF1 on SFP3.

5.3.3. Analysis

Advantages of encoding information in the NSH frame:

- 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 or use of a metadata slot in case MD-Type 1 is used.
- Relies on classifier to encode metadata information
- Requires perfectly symmetrical paths. E.g., one direction cannot have more SFs than the other direction.
- If classifier will encode information it needs to receive and process rendered service path information

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.

Note that this mechanism is a special case of Section 5.3.2 wherein Reverse_Path_ID is the forward path ID and Reverse_Service_Offset=255.

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 is the recommended approach.

Advantages:

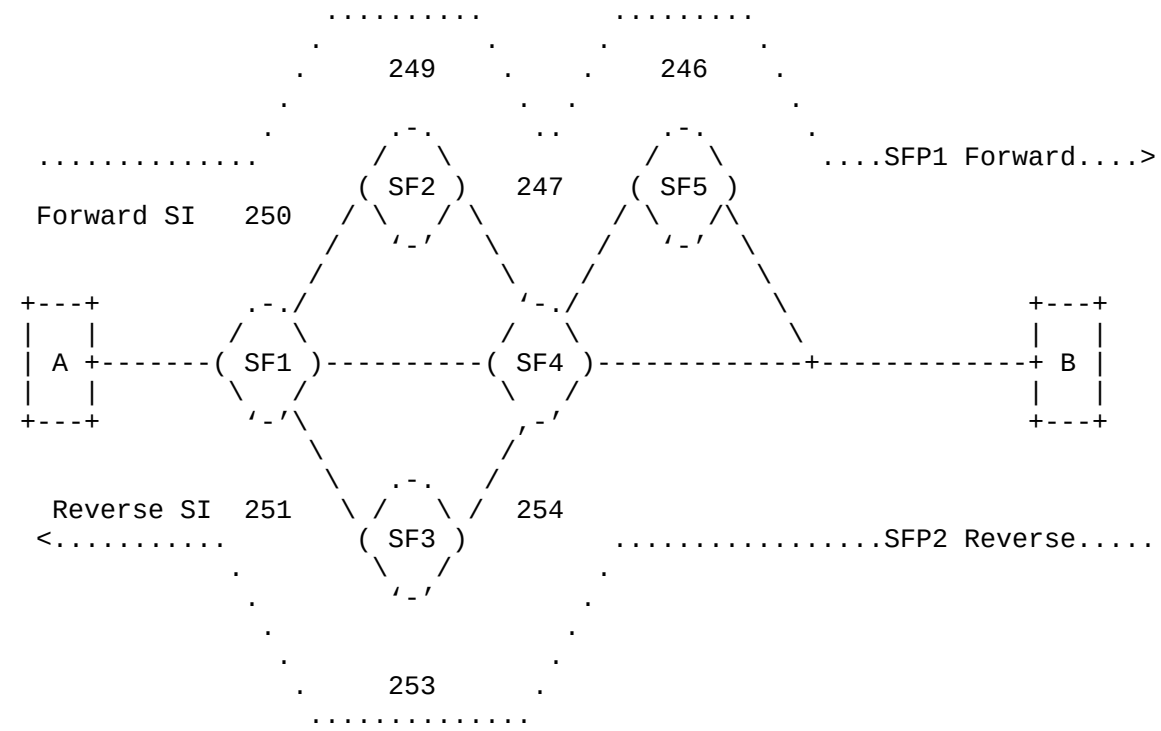
- No precious NSH area is consumed
- SF self-contained solution
- No SFF performance impact and no cooperation needed
- No Special Classification required

Disadvantages:

- Assumption that the reverse path Service Functions are the same as forward path, only in reverse
- Forward and Reverse Path IDs are algorithmically linked and can not be chosen arbitrarily.

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 5 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.



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

Figure 5: 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 inserts the metadata in the context headers.

7.1. Service-Path-Invariant Metadata

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.

To implement this, the controller needs to configure appropriate fixed values of the metadata present in the context headers for each path identifier in each Service Function that needs to inject packets. The Service Function must store this information so that when the Service Function generates a packet it can insert the minimum required metadata for a packet to reach its destination.

A disadvantage to path-invariant metadata is that it is a type of metadata that adds no information beyond the information available in the path identifier itself. The corollary is that if different metadata is required, a different service paths must be created.

7.2. Service-Path-Default Metadata

We also propose the notion of service-path-default metadata. This is metadata that could vary for different packets on a path but has a default value acceptable for any packet injected onto a certain path. For example, metadata might indicate a quality-of-service (QoS) treatment, and an operator considers it acceptable for injected

packets to have a default QoS treatment. It might also be considered acceptable to not send a particular type of metadata.

To implement this, the controller configures appropriate default metadata values for each path identifier in Service Functions that need to inject packets. The controller may also indicate a particular type may be omitted. The Service Function must store this information so that it can insert the minimum required metadata for a packet to reach its destination.

The disadvantage of this approach is that it relies on the assumption that there is a meaningful default metadata value, which may not exist.

7.3. Bidirectional Clonable Metadata

Some types of metadata may use values applicable to both directions of traffic. An example is routing domain, for which an identifier indicates a private network such that the value is the same for both directions of traffic and may be copied from one packet to another.

To implement this, the controller must indicate to each Service Function that a particular metadata type is bidirectional-clonable. The Service Function can therefore clone the metadata value from one packet to a new packet that it creates, even in the reverse direction. For this type, it is also considered safe to save a copy of metadata for the transport flow. (E.g., to retransmit a TCP packet using metadata cloned from another TCP packet of the same connection.)

Note that the Service Function need not know the meaning of the metadata; it just needs to know it is safe to clone in this manner.

7.4. Unidirectional Clonable Metadata

Some types of metadata may use values applicable to only one direction of traffic, but a value may be cloned from one packet to another in the same direction. An example is a destination identifier, in which meatadata indicates a network egress point. Another example is metadata indicating a property of either the source or destination end-point of the packet.

To implement this, the controller must indicate to each Service Function that a particular metadata type is unidirectional-clonable. A transport-layer-stateful Service Function can therefore save away metadata values that it has witnessed. An injected packet can therefore be assigned a clone of metadata taken from an earlier packet going in the same direction. For example, a Service Function

can send a TCP packet using metadata cloned from another TCP packet of the same connection and direction.

Note that the Service Function need not know the meaning of the metadata; it just needs to know it is safe to clone in this manner.

A disadvantage of unidirectional clonable metadata is that a device cannot respond to a packet unless it has previously witnessed a packet for the same connection in the opposite direction. For example, a firewall cannot respond to the first packet of a connection (since both directions have not been witnessed). However, having seen a full hand-shake, a cache or optimizing proxy can inject or retransmit packets.

7.5. Service-Function-Mastered Metadata

The easiest case to reason about is a type of metadata for which the Service Function can provide the appropriate values: specifically the metadata that it would be responsible for inserting for all packets as part of packet processing. We can assume this is configured by Service-Function-Specific methods.

7.6. Metadata from Reclassification

Finally if the packet needs crucial metadata values that cannot be supplied by the 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.

If a packet needs to be sent to classifier then it should be carried inside a NSH OAM packet that in turn is tunneled with a protocol such as VXLAN-GPE with the classifier as its tunnel endpoint.

8. Other solutions

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

- 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 "Flip Path-Id and Index High Order bits" and "SF receives Reverse Forwarding Information" were implemented in Opendaylight.

10. IANA Considerations

TBD

11. Security Considerations

Service Functions must be trusted entities, being permitted to rewrite service path headers.

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]
Kreeger, L. and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-02 (work in progress), April 2016.

[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-04 (work in progress), March 2016.

[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-14 (work in progress), January 2016.

[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

David Dolson
Sandvine
408 Albert Street
Waterloo, ON N2L 3V3
Canada

Phone: +1 519 880 2400
Email: ddolson@sandvine.com