

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: May 3, 2017

F. Brockners
S. Bhandari
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JMPC
T. Mizrahi
Marvell
D. Mozes
Mellanox Technologies Ltd.
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
October 30, 2016

Data Formats for In-situ OAM
draft-brockners-inband-oam-data-02

Abstract

In-situ Operations, Administration, and Maintenance (OAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document discusses the data types and data formats for in-situ OAM data records. In-situ 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-situ 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 May 3, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Conventions	3
3. In-situ OAM Data Types and Data Format	4
3.1. In-situ OAM Tracing Options	4
3.1.1. Pre-allocated Trace Option	6
3.1.2. Incremental Trace Option	9
3.1.3. In-situ OAM node data element format	11
3.1.4. Examples of In-situ OAM node data	14
3.2. In-situ OAM Proof of Transit Option	16
3.3. In-situ OAM Edge-to-Edge Option	18
4. In-situ OAM Data Export	18
5. IANA Considerations	19
6. Manageability Considerations	19
7. Security Considerations	19
8. Acknowledgements	19
9. References	19
9.1. Normative References	19
9.2. Informative References	20
Authors' Addresses	20

1. Introduction

This document defines data record types for "in-situ" Operations, Administration, and Maintenance (OAM). In-situ OAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM

data is added to the data packets rather than is being sent within packets specifically dedicated to OAM. A discussion of the motivation and requirements for in-situ OAM can be found in [I-D.brockners-inband-oam-requirements]. In-situ 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-situ 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-situ OAM data records. The in-situ 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

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
NSH:	Network Service Header
OAM:	Operations, Administration, and Maintenance
SFC:	Service Function Chain
SID:	Segment Identifier
SR:	Segment Routing
TLV:	Type-Length-Value
VXLAN-GPE:	Virtual eXtensible Local Area Network, Generic Protocol Extension

3. In-situ OAM Data Types and Data Format

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

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

In-situ OAM is expected to be deployed in a specific domain rather than on the overall Internet. The part of the network which employs in situ OAM is referred to as the "in-situ OAM-domain". In-situ OAM data is added to a packet upon entering the in-situ OAM-domain and is removed from the packet when exiting the domain. Within the in-situ OAM-domain, the in-situ OAM data may be updated by network nodes that the packet traverses. The device which adds in-situ OAM data container to the packet to capture in-situ OAM data is called the "in-situ OAM encapsulating node", whereas the device which removes the in-situ OAM data container is referred to as the "in-situ OAM decapsulating node". Nodes within the domain which are aware of in-situ OAM data and read and/or write or process the in-situ OAM data are called "in-situ OAM transit nodes". Note that not every node in an in-situ OAM domain needs to be an in-situ 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-situ OAM transit nodes rather than all nodes.

3.1. In-situ OAM Tracing Options

"In-situ OAM tracing data" is expected to be collected at every node that a packet traverses, i.e., in a typical deployment all nodes in an in-situ OAM-domain would participate in in-situ OAM and thus be in-situ OAM transit nodes, in-situ OAM encapsulating or in-situ OAM decapsulating nodes. The maximum network diameter of the in-situ OAM domain is assumed to be known.

To optimize hardware and software implementations tracing is defined as two separate options. Any deployment MAY choose to configure and support one or both of the following options. An implementation of the transport protocol that carries these in-situ OAM data MAY choose to support only one of the options. In the event that both options

are utilized at the same time, the Incremental Trace Option MUST be placed before the Pre-allocated Trace Option.

Pre-allocated Trace Option: This trace option is defined as a container of node-data elements with pre-allocated space for each node to populate its information. This option is useful for software implementations where it is efficient to allocate the space once and index into the array to populate the data during transit. The in-situ OAM encapsulating node allocates the option header and sets the fields in the option header. The in situ OAM encapsulating node allocates an array which is to store operational data retrieved from every node while the packet traverses the domain. In-situ OAM transit nodes update the content of the array. A pointer which is part of the in-situ OAM trace data points to the next empty slot in the array, which is where the next in-situ OAM transit node fills in its data.

Incremental Trace Option: This trace options is defined as a container of node-data elements where each node allocates and pushes its node data immediately following the option header. The number of node-data recorded and maximum number of node data that can be recorded are written into the option header. This format of trace recording is useful for some of the hardware implementations as this eliminates the need for the transit network elements to read the full array in the option and allows for arbitrarily long packets as the MTU allows. The in-situ OAM encapsulating node allocates the option header. The in-situ OAM encapsulating node based on operational state and configuration sets the fields in the header to control how large the node data list can grow. In-situ OAM transit nodes pushes its node data to the node data list and increments the number of node data records in the header.

Every node data entry is to hold information for a particular in situ OAM transit node that is traversed by a packet. The in-situ OAM decapsulating node removes the in-situ OAM data and process and/or export the metadata. In-situ 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).

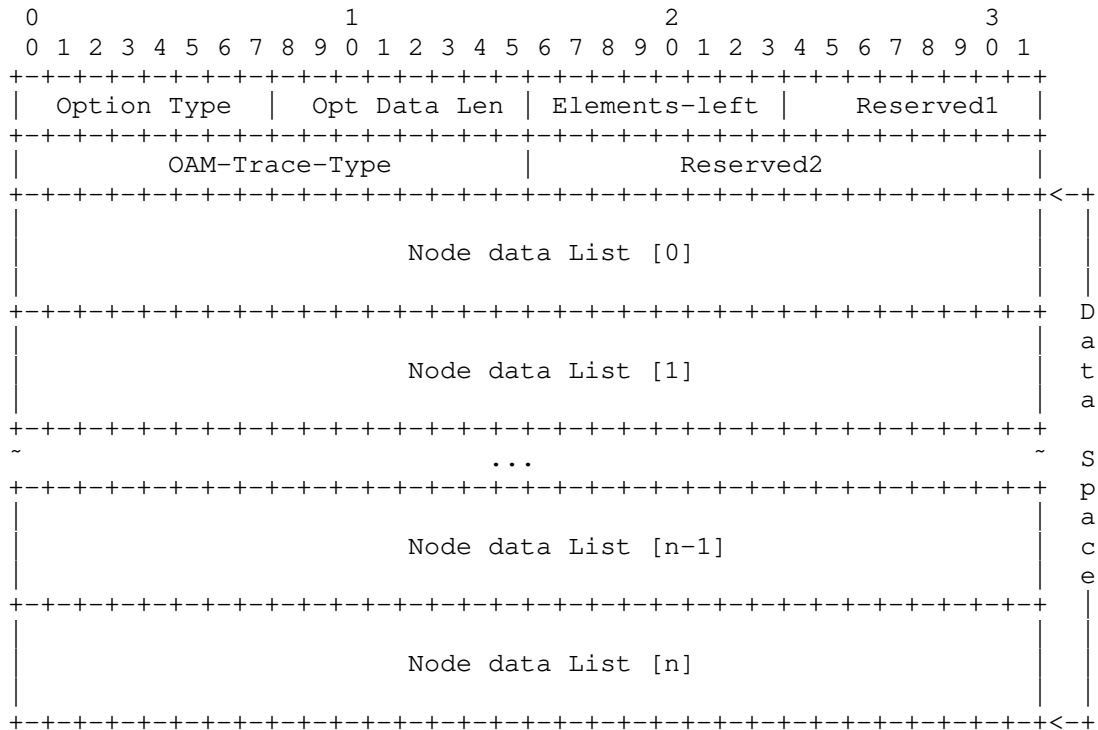
The following in-situ OAM data is defined for in-situ OAM tracing:

- o Identification of the in-situ OAM node. An in-situ OAM node identifier can match to a device identifier or a particular control point or subsystem within a device.
- o Identification of the interface that a packet was received on.
- o Identification of the interface that a packet was sent out on.
- o 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-situ OAM domain follow the same definition.
- o 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-situ OAM nodes should interpret the generic data the same way. Examples for generic in-situ 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.
- o A mechanism to detect whether in-situ OAM trace data was added at every hop or whether certain hops in the domain weren't in-situ 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.

3.1.1. Pre-allocated Trace Option

In-situ OAM Pre-allocated Trace 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.

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.

Reserved1: 8-bit unused field in this document and MUST be set to zero.

OAM-trace-type: 16-bit identifier of a particular trace element variant.

The trace type value is a bit field. The following bit fields are defined in this document, with details on each field described in the Section 3.1.3. The order of packing the trace data in each Node-data element follows the bit order for setting each trace data element.

- Bit 0 When set indicates presence of Hop_Lim and node_id in the Node data.
- Bit 1 When set indicates presence of ingress_if_id and egress_if_id in the Node data.
- Bit 2 When set indicates presence of timestamp seconds in the Node data
- Bit 3 When set indicates presence of timestamp nanoseconds in the Node data.
- Bit 4 When set indicates presence of transit delay in the Node data.
- Bit 5 When set indicates presence of app_data in the Node data.
- Bit 6 When set indicates presence of queue depth in the Node data.
- Bit 7 - 14 Undefined in this document.
- Bit 15 When set indicates wide data format for all the node data elements that are present. When unset indicates short data format for all the node data elements that are present.

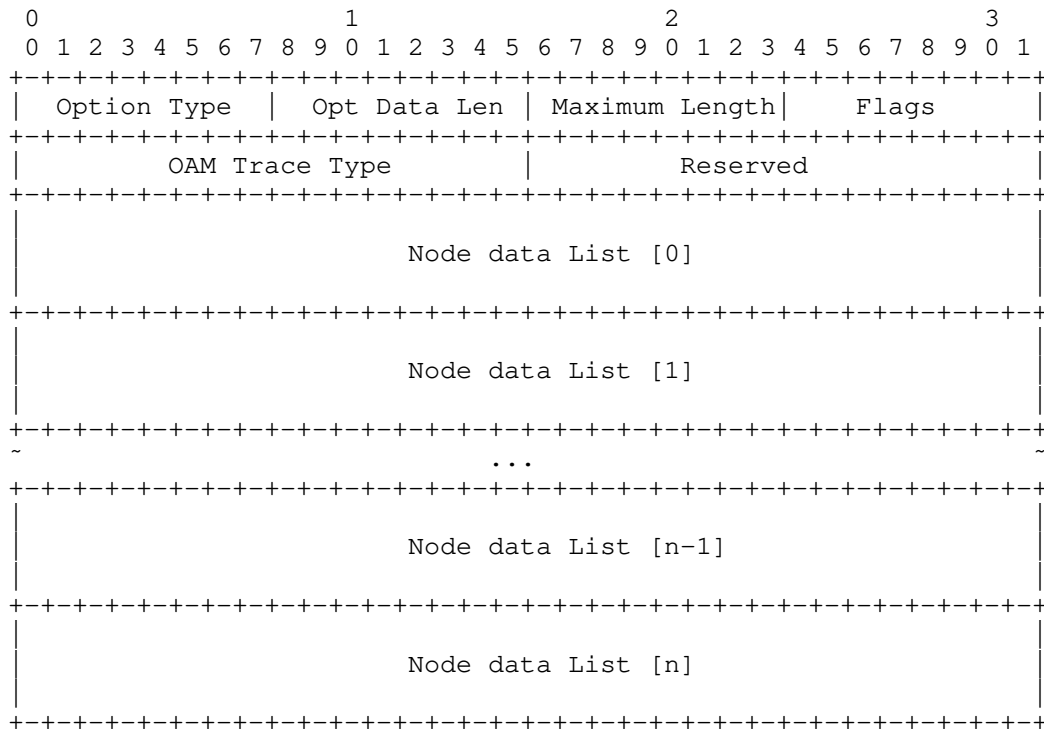
Section 3.1.4 describes the format of a number of trace types.

Reserved2: 16-bit unused field in this document and MUST be set to zero.

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.

3.1.2. Incremental Trace Option

In-situ OAM Incremental Trace 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.

Maximum Length: 8-bit unsigned integer. This field specifies the maximum length of the node data list in octets. Given that the sender knows the minimum path MTU, the sender can set the maximum of node data bytes allowed before exceeding the MTU. Thus, a simple comparison between "Opt data Len" and "Max Length" allows to decide whether or not data could be added.

Flags 8-bit field. Following flags are defined:

- 1 "Overflow" (O-bit) (least significant bit). This bit is set by the network element if the number of records on the packet is at the maximum limit as specified by the packet: i.e., the packet is already "full" of telemetry information. This is useful for transit nodes to ignore further processing of the option. If inserting a new node data record would cause "Opt Data Len" to exceed "Max Length", no record is added and the overflow "O-bit" must be set to "1" in the header.

OAM-trace-type: 16-bit identifier of a particular trace element variant.

The trace type value is a bit field. The following bit fields are defined in this document, with details on each field described in the Section 3.1.3. The order of packing the trace data in each Node-data element follows the bit order for setting each trace data element.

- | | |
|----------|--|
| Bit 0 | When set indicates presence of Hop_Lim and node_id in the Node data. |
| Bit 1 | When set indicates presence of ingress_if_id and egress_if_id in the Node data. |
| Bit 2 | When set indicates presence of timestamp seconds in the Node data |
| Bit 3 | When set indicates presence of timestamp nanoseconds in the Node data. |
| Bit 4 | When set indicates presence of transit delay in the Node data. |
| Bit 5 | When set indicates presence of app_data in the Node data. |
| Bit 6 | When set indicates presence of queue depth in the Node data. |
| Bit 7 | When set indicates presence of variable length Opaque State Snapshot field. |
| Bit 8-14 | Undefined in this draft. |
| Bit 15 | When set indicates wide data format for all the node data elements that are present. When unset indicates short data format for all the node data elements that are present. |

Section 3.1.4 describes the format of a number of trace types.

Reserved: 2 bytes unused field in this document and MUST be set to zero.

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.

3.1.3. In-situ OAM node data element format

The in-situ OAM node data elements are defined in 2 formats - short and wide that is selected by bit 15 in the OAM-trace-type field. All the data records MUST be 4-byte aligned in both the formats.

Data type and format for each of the data records in short format is shown below:

Hop_Lim and node_id: 4-octet field defined as follows:

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Hop_Lim | node_id |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop Limit information is used to identify the location of the node in the communication path. This is copied from the lower layer for e.g. TTL value in IPv4 header or hop limit field from IPv6 header of the packet.

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

ingress_if_id and egress_if_id: 4-octet field defined as follows:

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ingress_if_id | egress_if_id |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

ingress_if_id: 2-octet unsigned integer. Interface identifier to record the ingress interface the packet was received on.

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

timestamp seconds: 4-octet unsigned integer. Absolute timestamp in seconds that specifies the time at which the packet was received by the node. The format of this field is identical to the most significant 32 bits of 64 least significant bits of the [IEEE1588v2]. This truncated format consists of a 32-bit seconds field. As defined in [IEEE1588v2], the timestamp specifies the number of seconds elapsed since 1 January 1970 00:00:00 according to the International Atomic Time (TAI).

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     timestamp seconds                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

timestamp nanoseconds: 4-octet unsigned integer in the range 0 to 10^9-1 . This timestamp specifies the fractional part of the wall clock time at which the packet was received by the node in units of nanoseconds. It is nanoseconds that are recorded in 32 least significant bits of absolute time as per [IEEE1588v2]. This fields allows for delay computation between any two nodes in the network when the nodes are time synchronized.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     timestamp nanoseconds                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

transit delay: 4-octet unsigned integer in the range 0 to $2^{30}-1$. It is the time in nanoseconds packet spent in transiting a node. This can serve to give an indication of queuing delay at the node. If the transit delay exceeds $2^{30}-1$ nanoseconds then the top bit 'O' is set to indicate overflow.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|O|                                     transit delay                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

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

queue depth: 4-octet unsigned integer field. This field indicates the length of the egress interface queue of the interface where the packet is forwarded out of.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               queue depth                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Data type and format for each of the elements in wide format follows when Most Significant Bit (MSB) i.e., bit 15 of OAM-Trace-Type is set:

Hop_Lim and node_id: 8-octet field defined as follows:

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Hop_Lim | node_id | ~ |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
~ | node_id (contd) |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop Limit information is used to identify the location of the node in the communication path. This is copied from the lower layer for e.g. TTL value in IPv4 header or hop limit field from IPv6 header of the packet.

node_id: 7-octet unsigned integer. Node identifier field to uniquely identify a node within in-situ OAM domain. The procedure to allocate, manage and map the node_ids is beyond the scope of this document.

ingress_if_id and egress_if_id: 8-octet field defined as follows:

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ingress_if_id |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| egress_if_id |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

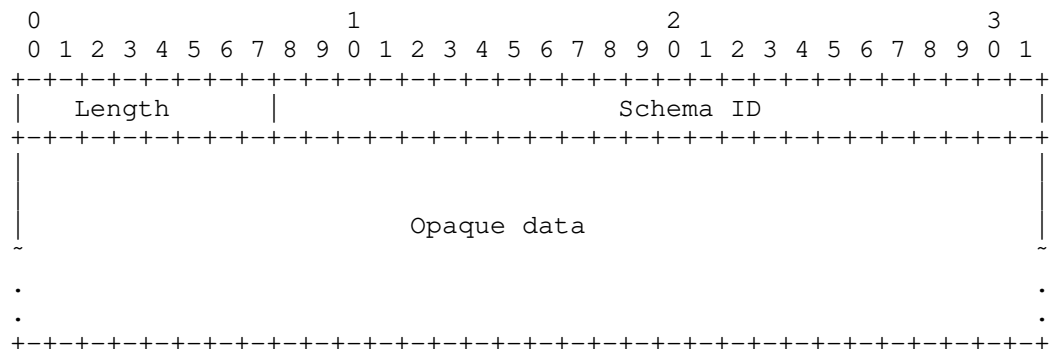
```

ingress_if_id: 4-octet unsigned integer. Interface identifier to record the ingress interface the packet was received on.

egress_if_id: 4-octet unsigned integer. Interface identifier to record the egress interface the packet is forwarded out of.

app_data: 8-octet placeholder which can be used by the node to add application specific data.

Opaque State Snapshot: Variable length field. It allows the network element to store arbitrary state in the node data record, without a pre-defined schema. The schema needs to be made known to the analyzer by some out-of-band means. The 24-bit "Schema Id" field in the record is supposed to let the analyzer know which particular schema to use, and it is expected to be configured on the network element by the operator. This ID is expected to be configured on the device by the network operator.



Length: 1-octet unsigned integer. It is the length of the Opaque data field that follows Schema Id. It MUST always be a multiple of 4.

Schema ID: 3-octet unsigned integer identifying the schema of Opaque data.

Opaque data: Variable length field. This field is interpreted as specified by the schema identified by the Schema ID.

The fields - timestamp seconds, timestamp nanoseconds and transit delay have the same format as defined in short format.

3.1.4. Examples of In-situ OAM node data

An entry in the "Node data List" array can have different formats, following the needs of the 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.

0x002B: In-situ OAM-trace-type is 0x2B then the format of node data is:

```

 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Hop_Lim | node_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ingress_if_id | egress_if_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| timestamp nanoseconds |
+-----+-----+-----+-----+-----+-----+-----+-----+
| app_data |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

0x0003: In-situ OAM-trace-type is 0x0003 then the format is:

```

 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Hop_Lim | node_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ingress_if_id | egress_if_id |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

0x0009: In-situ OAM-trace-type is 0x0009 then the format is:

```

 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Hop_Lim | node_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| timestamp nanoseconds |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

0x0021: In-situ OAM-trace-type is 0x0021 then the format is:

```

 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Hop_Lim | node_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| app_data |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

0x0029: In-situ OAM-trace-type is 0x0029 then the format is:

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Hop_Lim | node_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| timestamp nanoseconds |
+-----+-----+-----+-----+-----+-----+-----+-----+
| app_data |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

0x104D: In-situ OAM-trace-type is 0x104D then the format is:

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Hop_Lim | node_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| node_id(contd) |
+-----+-----+-----+-----+-----+-----+-----+-----+
| timestamp seconds |
+-----+-----+-----+-----+-----+-----+-----+-----+
| timestamp nanoseconds |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Length | Schema Id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Opaque data |
+-----+-----+-----+-----+-----+-----+-----+-----+
.
.
.
+-----+-----+-----+-----+-----+-----+-----+-----+

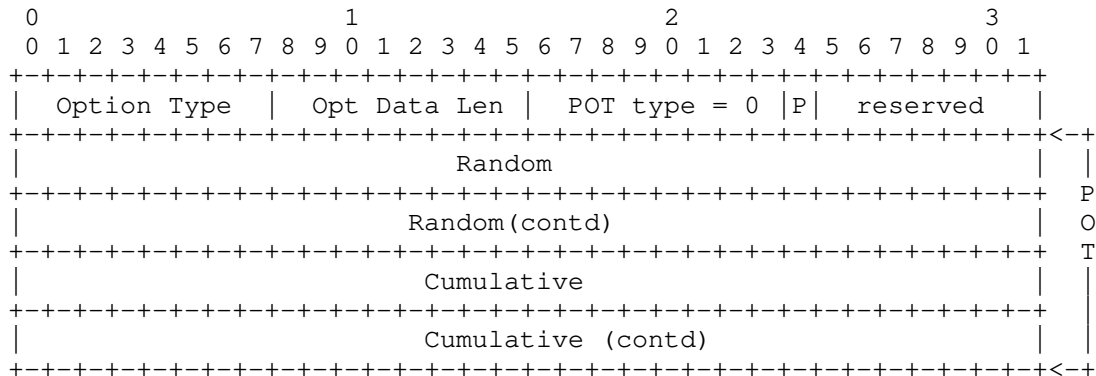
```

3.2. In-situ OAM Proof of Transit Option

In-situ 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-situ OAM data or mechanisms such as Shamir's Secret Sharing Schema (SSSS). While details on how the in-situ OAM data for the proof of transit option is processed at in-situ 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-situ OAM data to the packet:

- o Random: Unique identifier for the packet (e.g., 64-bits allow for the unique identification of 2^{64} packets).
- o Cumulative: Information which is handed from node to node and updated by every node according to a verification algorithm.

In-situ 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. This document defines POT Type 0:

0: POT data is a 16 Octet field as described below.

Profile to use (P): 1-bit. Indicates which POT-profile is used to generate the Cumulative. Any node participating in POT will have a maximum of 2 profiles configured that drive the computation of cumulative. The two profiles are numbered 0, 1. This bit conveys whether profile 0 or profile 1 is used to compute the Cumulative.

Reserved: 7-bit. Reserved for future use.

Random: 64-bit Per packet Random number.

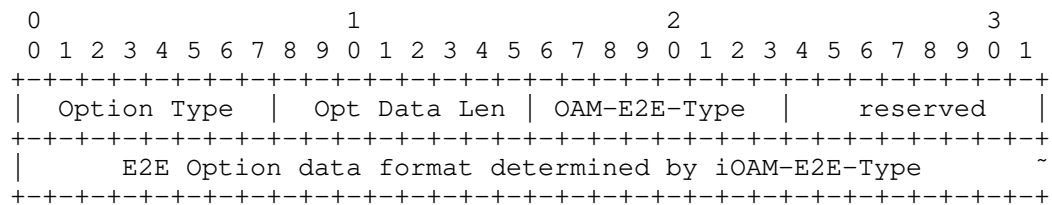
Cumulative: 64-bit Cumulative that is updated at specific nodes by processing per packet Random number field and configured parameters.

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-situ OAM Edge-to-Edge Option

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

Currently only sequence numbers use the in-situ OAM Edge-to-Edge option. In order to detect packet loss, packet reordering, or packet duplication in an in-situ 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.

OAM-E2E-Type: 8-bit identifier of a particular in situ 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-situ OAM Data Export

In-situ OAM nodes collect information for packets traversing a domain that supports in-situ OAM. The device at the domain edge (which could also be an end-host) which receives a packet with in-situ OAM information chooses how to process the in-situ 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-situ 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-situ OAM, please refer to [I-D.brockners-inband-oam-requirements].

8. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, LJ Wobker, Erik Nordmark, and Andrew Yourtchenko for the comments and advice. This document leverages and builds on top of several concepts described in [I-D.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

[I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C.,
Gredler, H., Leddy, J., and S. Youell, "Requirements for
In-band OAM", draft-brockners-inband-oam-requirements-01
(work in progress), July 2016.

- [IEEE1588v2]
Institute of Electrical and Electronics Engineers,
"1588-2008 - IEEE Standard for a Precision Clock
Synchronization Protocol for Networked Measurement and
Control Systems", IEEE Std 1588-2008, 2008,
<[http://standards.ieee.org/findstds/
standard/1588-2008.html](http://standards.ieee.org/findstds/standard/1588-2008.html)>.
- [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>>.

9.2. Informative References

- [I-D.brockners-inband-oam-transport]
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H.,
Leddy, J., and S. Youell, "Encapsulations for In-band OAM
Data", draft-brockners-inband-oam-transport-01 (work in
progress), July 2016.
- [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.kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop
Option Extension", draft-kitamura-ipv6-record-route-00
(work in progress), November 2000.
- [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.
- [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

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

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

Email: petr@fb.com

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

ippm
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2018

F. Brockners
S. Bhandari
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JPMC
T. Mizrahi
Marvell
D. Mozes
Mellanox Technologies Ltd.
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
D. Bernier
Bell Canada
July 2, 2017

Data Fields for In-situ OAM
draft-brockners-inband-oam-data-07

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document discusses the data fields and associated data types for in-situ OAM. In-situ OAM data fields can be embedded into a variety of transports such as NSH, Segment Routing, Geneve, native IPv6 (via extension header), or IPv4. In-situ OAM can be used to complement OAM mechanisms based on e.g. 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 3, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Conventions	3
3. Scope, Applicability, and Assumptions	4
4. IOAM Data Types and Formats	5
4.1. IOAM Tracing Options	6
4.1.1. Pre-allocated Trace Option	8
4.1.2. Incremental Trace Option	11
4.1.3. IOAM node data fields and associated formats	14
4.1.4. Examples of IOAM node data	19
4.2. IOAM Proof of Transit Option	21
4.3. IOAM Edge-to-Edge Option	23
5. IOAM Data Export	23
6. IANA Considerations	24
6.1. Creation of a New In-Situ OAM (IOAM) Protocol Parameters IANA registry	24
6.2. IOAM Trace Type Registry	24
6.3. IOAM Trace Flags Registry	24
6.4. IOAM POT Type Registry	25
6.5. IOAM E2E Type Registry	25
7. Manageability Considerations	25
8. Security Considerations	25
9. Acknowledgements	25
10. References	25
10.1. Normative References	25

10.2. Informative References	26
Authors' Addresses	27

1. Introduction

This document defines data fields for "in-situ" Operations, Administration, and Maintenance (IOAM). In-situ OAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM data is added to the data packets rather than is being sent within packets specifically dedicated to OAM. A discussion of the motivation and requirements for in-situ OAM can be found in [I-D.brockners-inband-oam-requirements]. IOAM is to complement mechanisms such as Ping or Traceroute, or more recent active probing mechanisms as described in [I-D.lapukhov-dataplane-probe]. In terms of "active" or "passive" OAM, "in-situ" OAM can be considered a hybrid OAM type. While no extra packets are sent, IOAM adds information to the packets therefore cannot be considered passive. In terms of the classification given in [RFC7799] IOAM could be portrayed as Hybrid Type 1. "In-situ" mechanisms do not require extra packets to be sent and hence don't change the packet traffic mix within the network. IOAM mechanisms can be leveraged where mechanisms using e.g. ICMP do not apply or do not offer the desired results, such as proving that a certain traffic flow 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 in which probe traffic is potentially handled differently from regular data traffic by the network devices.

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:

E2E	Edge to Edge
Geneve:	Generic Network Virtualization Encapsulation [I-D.ietf-nvo3-geneve]
IOAM:	In-situ Operations, Administration, and Maintenance
MTU:	Maximum Transmit Unit
NSH:	Network Service Header [I-D.ietf-sfc-nsh]

OAM: Operations, Administration, and Maintenance

POT: Proof of Transit

SFC: Service Function Chain

SID: Segment Identifier

SR: Segment Routing

VXLAN-GPE: Virtual eXtensible Local Area Network, Generic Protocol Extension [I-D.ietf-nvo3-vxlan-gpe]

3. Scope, Applicability, and Assumptions

IOAM deployment assumes a set of constraints, requirements, and guiding principles which are described in this section.

Scope: This document defines the data fields and associated data types for in-situ OAM. The in-situ OAM data field can be transported by a variety of transport protocols, including NSH, Segment Routing, Geneve, IPv6, or IPv4. Specification details for these different transport protocols are outside the scope of this document.

Deployment domain (or scope) of in-situ OAM deployment: IOAM is a network domain focused feature, with "network domain" being a set of network devices or entities within a single administration. For example, a network domain can include an enterprise campus using physical connections between devices or an overlay network using virtual connections / tunnels for connectivity between said devices. A network domain is defined by its perimeter or edge. Designers of carrier protocols for IOAM must specify mechanisms to ensure that IOAM data stays within an IOAM domain. In addition, the operator of such a domain is expected to put provisions in place to ensure that IOAM data does not leak beyond the edge of an IOAM domain, e.g. using for example packet filtering methods. The operator should consider potential operational impact of IOAM to mechanisms such as ECMP processing (e.g. load-balancing schemes based on packet length could be impacted by the increased packet size due to IOAM), path MTU (i.e. ensure that the MTU of all links within a domain is sufficiently large to support the increased packet size due to IOAM) and ICMP message handling (i.e. in case of a native IPv6 transport, IOAM support for ICMPv6 Echo Request/Reply could be desired which would translate into ICMPv6 extensions to enable IOAM data fields to be copied from an Echo Request message to an Echo Reply message).

IOAM control points: IOAM data fields are added to or removed from the live user traffic by the devices which form the edge of a domain.

Devices within an IOAM domain can update and/or add IOAM data-fields. Domain edge devices can be hosts or network devices.

Traffic-sets that IOAM is applied to: IOAM can be deployed on all or only on subsets of the live user traffic. It SHOULD be possible to enable IOAM on a selected set of traffic (e.g., per interface, based on an access control list or flow specification defining a specific set of traffic, etc.) The selected set of traffic can also be all traffic.

Encapsulation independence: Data formats for IOAM SHOULD be defined in a transport-independent manner. IOAM applies to a variety of encapsulating protocols. A definition of how IOAM data fields are carried by different transport protocols is outside the scope of this document.

Layering: If several encapsulation protocols (e.g., in case of tunneling) are stacked on top of each other, IOAM data-records could be present at every layer. The behavior follows the ships-in-the-night model.

Combination with active OAM mechanisms: IOAM should be usable for active network probing, enabling for example a customized version of traceroute. Decapsulating IOAM nodes may have an ability to send the IOAM information retrieved from the packet back to the source address of the packet or to the encapsulating node.

IOAM implementation: The IOAM data-field definitions take the specifics of devices with hardware data-plane and software data-plane into account.

4. IOAM Data Types and Formats

This section defines IOAM data types and data fields and associated data types required for IOAM. The different uses of IOAM require the definition of different types of data. The IOAM data fields for the data being carried corresponds to the three main categories of IOAM data defined in [I-D.brockners-inband-oam-requirements], which are: edge-to-edge, per node, and for selected nodes only.

Transport options for IOAM data are outside the scope of this memo, and are discussed in [I-D.brockners-inband-oam-transport]. IOAM data fields are fixed length data fields. A bit field determines the set of OAM data fields embedded in a packet. Depending on the type of the encapsulation, a counter field indicates how many data fields are included in a particular packet.

IOAM is expected to be deployed in a specific domain rather than on the overall Internet. The part of the network which employs IOAM is referred to as the "IOAM-domain". IOAM data is added to a packet upon entering the IOAM-domain and is removed from the packet when exiting the domain. Within the IOAM-domain, the IOAM data may be updated by network nodes that the packet traverses. The device which adds an IOAM data container to the packet to capture IOAM data is called the "IOAM encapsulating node", whereas the device which removes the IOAM data container is referred to as the "IOAM decapsulating node". Nodes within the domain which are aware of IOAM data and read and/or write or process the IOAM data are called "IOAM transit nodes". IOAM nodes which add or remove the IOAM data container can also update the IOAM data fields at the same time. Or in other words, IOAM encapsulation or decapsulating nodes can also serve as IOAM transit nodes at the same time. Note that not every node in an IOAM domain needs to be an IOAM 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 IOAM transit nodes rather than all nodes.

4.1. IOAM Tracing Options

"IOAM tracing data" is expected to be collected at every node that a packet traverses to ensure visibility into the entire path a packet takes within an IOAM domain, i.e., in a typical deployment all nodes in an in-situ OAM-domain would participate in IOAM and thus be IOAM transit nodes, IOAM encapsulating or IOAM decapsulating nodes. If not all nodes within a domain are IOAM capable, IOAM tracing information will only be collected on those nodes which are IOAM capable. Nodes which are not IOAM capable will forward the packet without any changes to the IOAM data fields. The maximum number of hops and the minimum path MTU of the IOAM domain is assumed to be known.

To optimize hardware and software implementations tracing is defined as two separate options. Any deployment MAY choose to configure and support one or both of the following options. An implementation of the transport protocol that carries these in-situ OAM data MAY choose to support only one of the options. In the event that both options are utilized at the same time, the Incremental Trace Option MUST be placed before the Pre-allocated Trace Option. Given that the operator knows which equipment is deployed in a particular IOAM, the operator will decide by means of configuration which type(s) of trace options will be enabled for a particular domain.

Pre-allocated Trace Option: This trace option is defined as a container of node data fields with pre-allocated space for each node to populate its information. This option is useful for

software implementations where it is efficient to allocate the space once and index into the array to populate the data during transit. The IOAM encapsulating node allocates the option header and sets the fields in the option header. The in situ OAM encapsulating node allocates an array which is used to store operational data retrieved from every node while the packet traverses the domain. IOAM transit nodes update the content of the array. A pointer which is part of the IOAM trace data points to the next empty slot in the array, which is where the next IOAM transit node fills in its data.

Incremental Trace Option: This trace option is defined as a container of node data fields where each node allocates and pushes its node data immediately following the option header. The maximum length of the node data list is written into the option header. This type of trace recording is useful for some of the hardware implementations as this eliminates the need for the transit network elements to read the full array in the option and allows for arbitrarily long packets as the MTU allows. The in-situ OAM encapsulating node allocates the option header. The in-situ OAM encapsulating node based on operational state and configuration sets the fields in the header to control how large the node data list can grow. IOAM transit nodes push their node data to the node data list and increment the number of node data fields in the header.

Every node data entry is to hold information for a particular IOAM transit node that is traversed by a packet. The in-situ OAM decapsulating node removes the IOAM data and processes and/or exports the metadata. IOAM 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 IOAM data is defined for IOAM tracing:

- o Identification of the IOAM node. An IOAM node identifier can match to a device identifier or a particular control point or subsystem within a device.
- o Identification of the interface that a packet was received on, i.e. ingress interface.
- o Identification of the interface that a packet was sent out on, i.e. egress interface.

- o 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-situ OAM domain follow the same definition.
- o 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 IOAM nodes should interpret the generic data the same way. Examples for generic IOAM 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.
- o A mechanism to detect whether IOAM trace data was added at every hop or whether certain hops in the domain weren't in-situ 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.

4.1.1. Pre-allocated Trace Option

In-situ OAM pre-allocated trace option:

Pre-allocated trace option header:

[illegible]

Pre-allocated Trace Option Data MUST be 4-octet aligned:

[illegible]

IOAM-Trace-Type: A 16-bit identifier which specifies which data types are used in this node data list.

The IOAM-Trace-Type value is a bit field. The following bit fields are defined in this document, with details on each field described in the Section 4.1.3. The order of packing the data fields in each node data element follows the bit order of the IOAM-Trace-Type field, as follows:

Bit 0 (Most significant bit) When set indicates presence of Hop_Lim and node_id in the node data.

Bit 1	When set indicates presence of ingress_if_id and egress_if_id (short format) in the node data.
-------	--

- Bit 2 When set indicates presence of timestamp seconds in the node data
- Bit 3 When set indicates presence of timestamp nanoseconds in the node data.
- Bit 4 When set indicates presence of transit delay in the node data.
- Bit 5 When set indicates presence of app_data (short format) in the node data.
- Bit 6 When set indicates presence of queue depth in the node data.
- Bit 7 When set indicates presence of variable length Opaque State Snapshot field.
- Bit 8 When set indicates presence of Hop_Lim and node_id in wide format in the node data.
- Bit 9 When set indicates presence of ingress_if_id and egress_if_id in wide format in the node data.
- Bit 10 When set indicates presence of app_data wide in the node data.
- Bit 11 When set indicates presence of the Checksum Complement node data.
- Bit 12-15 Undefined in this draft.

Section 4.1.3 describes the IOAM data types and their formats. Within an in-situ OAM domain possible combinations of these bits making the IOAM-Trace-Type can be restricted by configuration knobs.

Node Data Length: 4-bit unsigned integer. This field specifies the length of data added by each node in multiples of 4-octets. For example, if 3 IOAM-Trace-Type bits are set and none of them is wide, then the Node Data Length would be 3. If 3 IOAM-Trace-Type bits are set and 2 of them are wide, then the Node Data Length would be 5.

Flags 5-bit field. Following flags are defined:

- Bit 0 "Overflow" (O-bit) (most significant bit). This bit is set by the network element if there is not enough number of octets

left to record node data, no field is added and the overflow "O-bit" must be set to "1" in the header. This is useful for transit nodes to ignore further processing of the option.

Bit 1 "Loopback" (L-bit). Loopback mode is used to send a copy of a packet back towards the source. Loopback mode assumes that a return path from transit nodes and destination nodes towards the source exists. The encapsulating node decides (e.g. using a filter) which packets loopback mode is enabled for by setting the loopback bit. The encapsulating node also needs to ensure that sufficient space is available in the IOAM header for loopback operation. The loopback bit when set indicates to the transit nodes processing this option to create a copy of the packet received and send this copy of the packet back to the source of the packet while it continues to forward the original packet towards the destination. The source address of the original packet is used as destination address in the copied packet. The address of the node performing the copy operation is used as the source address. The L-bit MUST be cleared in the copy of the packet a nodes sends it back towards the source. On its way back towards the source, the packet is processed like a regular packet with IOAM information. Once the return packet reaches the IOAM domain boundary IOAM decapsulation occurs as with any other packet containing IOAM information.

Bit 2-4 Reserved: Must be zero.

Octets-left: 7-bit unsigned integer. It is the data space in multiples of 4-octets remaining for recording the node data. This is used as an offset in data space to record the node data element.

Node data List [n]: Variable-length field. The type of which is determined by the IOAM-Trace-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 "Octets-left" identifies the offset for current active node data to be populated.

4.1.2. Incremental Trace Option

In-situ OAM incremental trace option:

In-situ OAM incremental trace option Header:

0																1																2																3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9																								
IOAM-Trace-Type																NodeLen																Flags																Max Length															

IOAM Incremental Trace Option Data MUST be 4-octet aligned:

node data list [0]																																																															
node data list [1]																																																															
...																																																															
node data list [n-1]																																																															
node data list [n]																																																															

IOAM-trace-type: A 16-bit identifier which specifies which data types are used in this node data list.

The IOAM-Trace-Type value is a bit field. The following bit fields are defined in this document, with details on each field described in the Section 4.1.3. The order of packing the data fields in each node data element follows the bit order of the IOAM-Trace-Type field, as follows:

Bit 0 (Most significant bit) When set indicates presence of Hop_Lim and node_id in the node data.

Bit 1 When set indicates presence of ingress_if_id and egress_if_id (short format) in the node data.

- Bit 2 When set indicates presence of timestamp seconds in the node data.
- Bit 3 When set indicates presence of timestamp nanoseconds in the node data.
- Bit 4 When set indicates presence of transit delay in the node data.
- Bit 5 When set indicates presence of app_data in the node data.
- Bit 6 When set indicates presence of queue depth in the node data.
- Bit 7 When set indicates presence of variable length Opaque State Snapshot field.
- Bit 8 When set indicates presence of Hop_Lim and node_id wide in the node data.
- Bit 9 When set indicates presence of ingress_if_id and egress_if_id in wide format in the node data.
- Bit 10 When set indicates presence of app_data wide in the node data.
- Bit 11 When set indicates presence of the Checksum Complement node data.
- Bit 12-15 Undefined in this draft.

Section 4.1.3 describes the IOAM data types and their formats.

Node Data Length: 4-bit unsigned integer. This field specifies the length of data added by each node in multiples of 4-octets. For example, if 3 IOAM-Trace-Type bits are set and none of them is wide, then the Node Data Length would be 3. If 3 IOAM-Trace-Type bits are set and 2 of them are wide, then the Node Data Length would be 5.

Flags 5-bit field. Following flags are defined:

- Bit 0 "Overflow" (O-bit) (least significant bit). This bit is set by the network element if there is not enough number of octets left to record node data, no field is added and the overflow "O-bit" must be set to "1" in the header. This is useful for transit nodes to ignore further processing of the option.

Bit 1 "Loopback" (L-bit). This bit when set indicates to the transit nodes processing this option to send a copy of the packet back to the source of the packet while it continues to forward the original packet towards the destination. The L-bit MUST be cleared in the copy of the packet before sending it.

Bit 2-4 Reserved. Must be zero.

Maximum Length: 7-bit unsigned integer. This field specifies the maximum length of the node data list in multiples of 4-octets. Given that the sender knows the minimum path MTU, the sender can set the maximum length according to the number of node data bytes allowed before exceeding the MTU. Thus, a simple comparison between "Opt data Len" and "Max Length" allows to decide whether or not data could be added.

Node data List [n]: Variable-length field. The type 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.

4.1.3. IOAM node data fields and associated formats

All the data fields MUST be 4-octet aligned. The IOAM encapsulating node MUST initialize data fields that it adds to the packet to zero. If a node which is supposed to update an IOAM data field is not capable of populating the value of a field set in the IOAM-Trace-Type, the field value MUST be left unaltered except when explicitly specified in the field description below. In the description of data below if zero is valid value then a non-zero value to mean not populated is specified.

Data field and associated data type for each of the data field is shown below:

Hop_Lim and node_id: 4-octet field defined as follows:

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Hop_Lim   |           node_id           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop Limit information is used to identify the location of the node

in the communication path. This is copied from the lower layer, e.g., TTL value in IPv4 header or hop limit field from IPv6 header of the packet when the packet is ready for transmission. The semantics of the Hop_Lim field depend on the lower layer protocol that IOAM is encapsulated over, and therefore its specific semantics are outside the scope of this memo.

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

ingress_if_id and egress_if_id: 4-octet field defined as follows:
When this field is part of the data field but a node populating the field is not able to fill it, the position in the field must be filled with value 0xFFFFFFFF to mean not populated.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      ingress_if_id      |      egress_if_id      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

ingress_if_id: 2-octet unsigned integer. Interface identifier to record the ingress interface the packet was received on.

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

timestamp seconds: 4-octet unsigned integer. Absolute timestamp in seconds that specifies the time at which the packet was received by the node. The structure of this field is identical to the most significant 32 bits of the 64 least significant bits of the [IEEE1588v2] timestamp. This truncated field consists of a 32-bit seconds field. As defined in [IEEE1588v2], the timestamp specifies the number of seconds elapsed since 1 January 1970 00:00:00 according to the International Atomic Time (TAI).

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     timestamp seconds                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

timestamp nanoseconds: 4-octet unsigned integer in the range 0 to 10^9-1 . This timestamp specifies the fractional part of the wall clock time at which the packet was received by the node in units of nanoseconds. This field is identical to the 32 least significant bits of the [IEEE1588v2] timestamp. This fields

allows for delay computation between any two nodes in the network when the nodes are time synchronized. When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     timestamp nanoseconds                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

transit delay: 4-octet unsigned integer in the range 0 to $2^{30}-1$.

It is the time in nanoseconds the packet spent in the transit node. This can serve as an indication of the queuing delay at the node. If the transit delay exceeds $2^{30}-1$ nanoseconds then the top bit 'O' is set to indicate overflow. When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|O|                                     transit delay                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

app_data: 4-octet placeholder which can be used by the node to add application specific data. App_data represents a "free-format" 4-octet bit field with its semantics defined by a specific deployment.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     app_data                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

queue depth: 4-octet unsigned integer field. This field indicates the current length of the egress interface queue of the interface from where the packet is forwarded out. The queue depth is expressed as the current number of memory buffers used by the queue (a packet may consume one or more memory buffers, depending on its size). When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               queue depth               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Opaque State Snapshot: Variable length field. It allows the network element to store an arbitrary state in the node data field, without a pre-defined schema. The schema needs to be made known to the analyzer by some out-of-band mechanism. The specification of this mechanism is beyond the scope of this document. The 24-bit "Schema Id" field in the field indicates which particular schema is used, and should be configured on the network element by the operator.

```

0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Length   |               Schema ID               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|               Opaque data
|
~
.
.
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Length: 1-octet unsigned integer. It is the length in octets of the Opaque data field that follows Schema Id. It MUST always be a multiple of 4.

Schema ID: 3-octet unsigned integer identifying the schema of Opaque data.

Opaque data: Variable length field. This field is interpreted as specified by the schema identified by the Schema ID.

Hop_Lim and node_id wide: 8-octet field defined as follows:

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Hop_Lim   |               node_id               ~
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
~
~               node_id (contd)               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop

Limit information is used to identify the location of the node in the communication path. This is copied from the lower layer for e.g. TTL value in IPv4 header or hop limit field from IPv6 header of the packet. The semantics of the Hop_Lim field depend on the lower layer protocol that IOAM is encapsulated over, and therefore its specific semantics are outside the scope of this memo.

node_id: 7-octet unsigned integer. Node identifier field to uniquely identify a node within in-situ OAM domain. The procedure to allocate, manage and map the node_ids is beyond the scope of this document.

ingress_if_id and egress_if_id wide: 8-octet field defined as follows: When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFFFFFFFFFF to mean not populated.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     ingress_if_id                      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     egress_if_id                       |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

ingress_if_id: 4-octet unsigned integer. Interface identifier to record the ingress interface the packet was received on.

egress_if_id: 4-octet unsigned integer. Interface identifier to record the egress interface the packet is forwarded out of.

app_data wide: 8-octet placeholder which can be used by the node to add application specific data. App data represents a "free-format" 8-octet bit field with its semantics defined by a specific deployment.

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     app data                          ~
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
~                                     app data (contd)                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Checksum Complement: 4-octet node data which contains a two-octet Checksum Complement field, and a 2-octet reserved field. The Checksum Complement can be used when IOAM is transported over encapsulations that make use of a UDP transport, such as VXLAN-GPE

or Geneve. In this case, incorporating the IOAM node data requires the UDP Checksum field to be updated. Rather than to recompute the Checksum field, a node can use the Checksum Complement to make a checksum-neutral update in the UDP payload; the Checksum Complement is assigned a value that complements the rest of the node data fields that were added by the current node, causing the existing UDP Checksum field to remain correct. Checksum Complement fields are used in a similar manner in [RFC7820] and [RFC7821].

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Checksum Complement           |           Reserved           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

4.1.4. Examples of IOAM node data

An entry in the "node data list" array can have different formats, following the needs of the 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 types that an entry in "node data list" can take.

0x002B: IOAM-Trace-Type is 0x2B then the format of node data is:

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| Hop_Lim |           node_id           |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ingress_if_id |           egress_if_id           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           timestamp nanoseconds           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           app_data           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

0x0003: IOAM-Trace-Type is 0x0003 then the format is:

```

 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Hop_Lim | node_id |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ingress_if_id | egress_if_id |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

0x0009: IOAM-Trace-Type is 0x0009 then the format is:

```

 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Hop_Lim | node_id |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| timestamp nanoseconds |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

0x0021: IOAM-Trace-Type is 0x0021 then the format is:

```

 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Hop_Lim | node_id |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| app_data |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

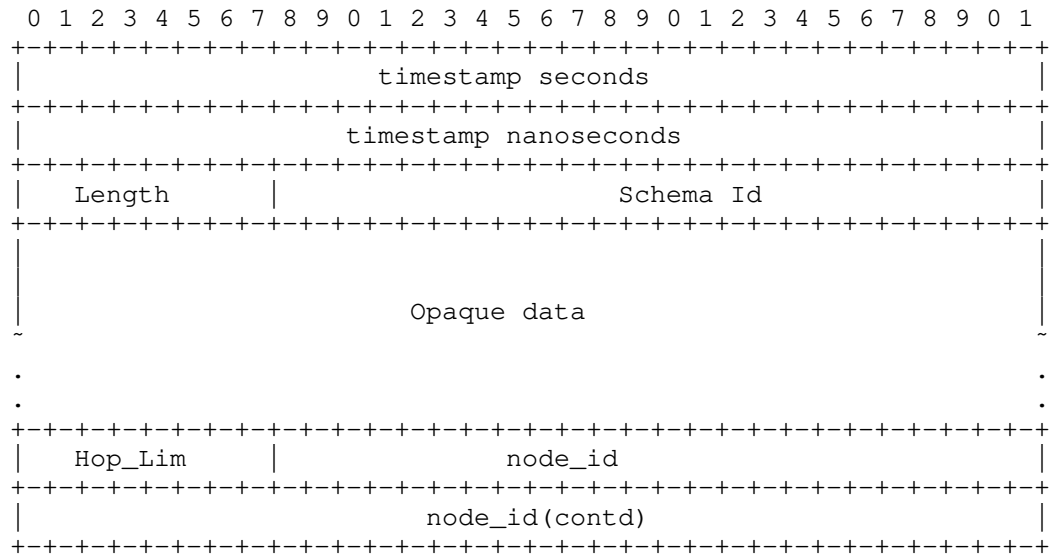
0x0029: IOAM-Trace-Type is 0x0029 then the format is:

```

 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Hop_Lim | node_id |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| timestamp nanoseconds |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| app_data |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

0x018C: IOAM-Trace-Type is 0x104D then the format is:



4.2. IOAM Proof of Transit Option

IOAM 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 IOAM data or mechanisms such as Shamir's Secret Sharing Schema (SSSS). While details on how the IOAM data for the proof of transit option is processed at IOAM 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 IOAM data to the packet:

- o Random: Unique identifier for the packet (e.g., 64-bits allow for the unique identification of 2^{64} packets).
- o Cumulative: Information which is handed from node to node and updated by every node according to a verification algorithm.

4.3. IOAM Edge-to-Edge Option

The IOAM edge-to-edge option is to carry data that is added by the IOAM encapsulating node and interpreted by IOAM decapsulating node. The IOAM transit nodes MAY process the data without modifying it.

Currently only sequence numbers use the IOAM edge-to-edge option. In order to detect packet loss, packet reordering, or packet duplication in an in-situ 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.

IOAM edge-to-edge option:

IOAM edge-to-edge option header:

```

  0 1 2 3 4 5 6 7
+-----+
| IOAM-E2E-Type |
+-----+
```

IOAM edge-to-edge option data MUST be 4-octet aligned:

```

      0              1              2              3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+
|           E2E Option data field determined by IOAM-E2E-Type           |
+-----+-----+-----+-----+
```

IOAM-E2E-Type: 8-bit identifier of a particular in situ 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.

5. IOAM Data Export

IOAM nodes collect information for packets traversing a domain that supports IOAM. IOAM decapsulating nodes as well as IOAM transit nodes can choose to retrieve IOAM information from the packet, process the information further and export the information using e.g., IPFIX.

The discussion of IOAM data processing and export is left for a future version of this document.

6. IANA Considerations

This document requests the following IANA Actions.

6.1. Creation of a New In-Situ OAM (IOAM) Protocol Parameters IANA registry

IANA is requested to create a new protocol registry for "In-Situ OAM (IOAM) Protocol Parameters". This is the common registry that will include registrations for all IOAM namespaces. Each Registry, whose names are listed below:

IOAM Trace Type

IOAM Trace flags

IOAM POT Type

IOAM E2E Type

will contain the current set of possibilities defined in this document. New registries in this name space are created via RFC Required process as per [RFC8126].

The subsequent sub-sections detail the registries herein contained.

6.2. IOAM Trace Type Registry

This registry defines code point for each bit in the 16-bit IOAM-Trace-Type field for Pre-allocated trace option and Incremental trace option defined in Section 4.1. The meaning of Bit 0 - 11 for trace type are defined in this document in Paragraph 1 of (Section 4.1.1). The meaning for Bit 12 - 15 are available for assignment via RFC Required process as per [RFC8126].

6.3. IOAM Trace Flags Registry

This registry defines code point for each bit in the 5 bit flags for Pre-allocated trace option and Incremental trace option defined in Section 4.1. The meaning of Bit 0 - 1 for trace flags are defined in this document in Paragraph 5 of Section 4.1.1. The meaning for Bit 2 - 4 are available for assignment via RFC Required process as per [RFC8126].

6.4. IOAM POT Type Registry

This registry defines 128 code points to define IOAM POT Type for IOAM proof of transit option Section 4.2. The code point value 0 is defined in this document, 1 - 127 are available for assignment via RFC Required process as per [RFC8126].

6.5. IOAM E2E Type Registry

This registry defines 256 code points to define IOAM-E2E-Type for IOAM E2E option Section 4.3. The code point value 0 is defined in this document, 1 - 255 are available for assignments via RFC Required process as per [RFC8126].

7. Manageability Considerations

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

8. Security Considerations

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

9. Acknowledgements

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

This document leverages and builds on top of several concepts described in [I-D.kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

The authors would like to gracefully acknowledge useful review and insightful comments received from Joe Clarke, Al Morton, and Mickey Spiegel.

10. References

10.1. Normative References

- [IEEE1588v2]
Institute of Electrical and Electronics Engineers,
"1588-2008 - IEEE Standard for a Precision Clock
Synchronization Protocol for Networked Measurement and
Control Systems", IEEE Std 1588-2008, 2008,
<[http://standards.ieee.org/findstds/
standard/1588-2008.html](http://standards.ieee.org/findstds/standard/1588-2008.html)>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for
Writing an IANA Considerations Section in RFCs", BCP 26,
RFC 8126, DOI 10.17487/RFC8126, June 2017,
<<http://www.rfc-editor.org/info/rfc8126>>.

10.2. Informative References

- [I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C.,
Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi,
T., <>, P., and r. remy@barefootnetworks.com,
"Requirements for In-situ OAM", draft-brockners-inband-
oam-requirements-03 (work in progress), March 2017.
- [I-D.brockners-inband-oam-transport]
Brockners, F., Bhandari, S., Govindan, V., Pignataro, C.,
Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes,
D., Lapukhov, P., and R. <>, "Encapsulations for In-situ
OAM Data", draft-brockners-inband-oam-transport-03 (work
in progress), March 2017.
- [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-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic
Network Virtualization Encapsulation", draft-ietf-
nvo3-geneve-04 (work in progress), March 2017.
- [I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol
Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work
in progress), April 2017.

- [I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-13 (work in progress), June 2017.
- [I-D.kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop Option Extension", draft-kitamura-ipv6-record-route-00 (work in progress), November 2000.
- [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.
- [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>>.
- [RFC7820] Mizrahi, T., "UDP Checksum Complement in the One-Way Active Measurement Protocol (OWAMP) and Two-Way Active Measurement Protocol (TWAMP)", RFC 7820, DOI 10.17487/RFC7820, March 2016, <<http://www.rfc-editor.org/info/rfc7820>>.
- [RFC7821] Mizrahi, T., "UDP Checksum Complement in the Network Time Protocol (NTP)", RFC 7821, DOI 10.17487/RFC7821, March 2016, <<http://www.rfc-editor.org/info/rfc7821>>.

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

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 2066721
Israel

Email: talmi@marvell.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

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

Email: petr@fb.com

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

Daniel
Bell Canada

Email: daniel.bernier@bell.ca

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2017

F. Brockners
S. Bhandari
S. Dara
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JMPC
D. Mozes
Mellanox Technologies Ltd.
T. Mizrahi
Marvell
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
October 30, 2016

Requirements for In-situ OAM
draft-brockners-inband-oam-requirements-02

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-situ" 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 situ 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 May 3, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Conventions	4
3. Motivation for in-situ OAM	5
3.1. Path Congruency Issues with Dedicated OAM Packets	5
3.2. Results Sent to a System Other Than the Sender	6
3.3. Overlay and Underlay Correlation	6
3.4. SLA Verification	7
3.5. Analytics and Diagnostics	7
3.6. Frame Replication/Elimination Decision for Bi-casting /Active-active Networks	8
3.7. Proof of Transit	8
3.8. Use Cases	9
4. Considerations for In-situ OAM	11
4.1. Type of Information to Be Recorded	11
4.2. MTU and Packet Size	12
4.3. Administrative Boundaries	12
4.4. Selective Enablement	13
4.5. Optimization of Node and Interface Identifiers	13
4.6. Loop Communication Path (IPv6-specifics)	14
5. Requirements for In-situ OAM Data Types	14
5.1. Generic Requirements	14
5.2. In-situ OAM Data with Per-hop Scope	16
5.3. In-situ OAM with Selected Hop Scope	17
5.4. In-situ OAM with End-to-end Scope	17

6. Security Considerations and Requirements	17
6.1. General considerations	17
6.2. Proof of Transit	18
7. IANA Considerations	19
8. Acknowledgements	19
9. References	19
9.1. Normative References	19
9.2. Informative References	19
Authors' Addresses	21

1. Introduction

This document discusses requirements for "in-situ" Operations, Administration, and Maintenance (OAM) mechanisms. In this context, "in-situ OAM" refers to the concept of directly encoding telemetry information within the data packet as it traverses the network or telemetry domain. Mechanisms which add tracing or other types of telemetry information to the regular data traffic, sometimes also referred to as "in-band" 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 terms of "active" or "passive" OAM, "in-situ" OAM can be considered a hybrid OAM type. While no extra packets are sent, in-situ OAM adds information to the packets therefore cannot be considered passive. In terms of the classification given in [RFC7799] in-situ OAM could be portrayed as "hybrid OAM, type 1". "In-situ" 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.

A number of in-situ as well as in-band OAM mechanisms have been discussed, such as the INT spec for the P4 programming language [P4] or the SPUD prototype [I-D.hildebrand-spud-prototype]. The 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-situ OAM mechanism. Per what was already stated, in-situ 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-situ 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 between overlay and underlay transports is in

place, checking service level agreements for the live data traffic, detailed statistics or verification of path selections within a domain, 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-situ OAM" which is IPv4 route recording [RFC0791], an in-situ 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-situ OAM operation, including but not limited to path tracing information, 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 determine whether any nodes were skipped while recording in-situ OAM information (i.e., in-situ 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 carry in-situ OAM data in various different transport protocols.

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:

ECMP:	Equal Cost Multi-Path
LISP:	Locator/ID Separation Protocol
MTU:	Maximum Transmit Unit

NSH:	Network Service Header
NFV:	Network Function Virtualization
OAM:	Operations, Administration, and Maintenance
PMTU:	Path MTU
SFC:	Service Function Chain
SLA:	Service Level Agreement
SR:	Segment Routing

This document defines in-situ Operations, Administration, and Maintenance (in-situ 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-situ OAM

In several scenarios it is beneficial to make information about the path a packet took through the network or through a network device as well as associated telemetry information 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-situ network diagnostics.

3.1. Path Congruency Issues with Dedicated OAM Packets

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-situ mechanism is an alternative in those cases.

In-situ mechanisms are not impacted by differences in the handling of probe traffic compared to other data packets, where probe traffic is handled differently (and potentially forwarded differently) by a

router than regular data traffic. This obviously assumes that the addition of in-situ information does not change the forwarding behavior of the packet. Note that in certain implementations, the addition information to a transport protocol changes the forwarding behavior. IPv6 extension header processing is one example. Some implementations process IPv6 packets with extension headers in the "slow" path of a router, as opposed to the "fast" path.

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 helpful to eliminate the requirement that there be a working bidirectional 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-situ OAM the operator of the underlay can offer details of the connectivity in the underlay to the user of the overlay. This could include the ability to find out which underlay elements are shared by overlays and ability to know which overlays are mapped to the same underlay elements. Deployment dependent underlay transit nodes can be configured to update OAM information in the overlay transport encapsulation. 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-situ 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-situ 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. In active/standby deployments in-situ OAM would only allow for SLA verification of the active path.

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 device 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.

In high-speed networks one can leverage and benefit from packet-accurate measurements with for example hardware-accurate timestamping (i.e., nanosecond-level verification) to support optimized packet scheduling and queuing mechanisms.

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 Internet of Things (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. Time sensitive networks (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 Locator/ID Separation Protocol (LISP), Network Service Header (NSH), Segment Routing (SR), etc.) blurs the line between the different trust domains, because the hand-off-points are no longer clearly defined physical interfaces, but are virtual interfaces. 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.

Deployed service chains without the presence of a "proof of transit" mechanism are typically operated as fail-open system: The packets that arrive at the end of a service chain are processed. Adding "proof of transit" capabilities to a service chain allows an operator to turn a fail-open system into a fail-close system, i.e. packets that did not properly traverse the service chain can be blocked.

A solution approach could be based on OAM data which is added to every packet for achieving Proof Of Transit (POT). 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 [I-D.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-situ OAM could be leveraged for several use cases, including:

- o 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 on a per Quality of Service (QoS) class.
- o 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).
- o Loss Statistics per Path: Retrieve loss statistics per flow and path in the network.
- o Path Heat Maps: Discover highly utilized links in the network.

- o 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.)
- o 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.
- o 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.
- o 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.
- o 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.
- o 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.
- o Device-level Troubleshooting and Optimization: In many cases, network operators could benefit from information specific to a single device. A non-exhaustive list of useful information includes: queue-depths, buffer utilization (either shared or per-port), packet latency measured from a known starting point, packet latency introduced by a single device, and resource utilization (CPU, memory, link bandwidth) of a given device or link. In some cases, this information changes over per-packet timescales (i.e., nanoseconds) and as such it is extremely challenging to collect and report this info in an accurate and scalable manner. By encoding the information from the forwarding element directly within a data packet (i.e., within the 'fast-path') this information can be added to some or all data packets and then collected and analyzed by human or machine tools. This type of information is particularly valuable for troubleshooting low-level device errors as well as providing a knowledge feedback loop for network and device optimization.

- o Custom Network Probing: Active network probing and in-situ OAM can be combined for customized and efficient network probing. This could for example be a customized traceroute.

4. Considerations for In-situ OAM

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

4.1. Type of Information to Be Recorded

The information gathered for in-situ 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 hops, such as path or service chain verification; information which only applies to the edges of a domain, such as sequence numbers. Note that a single network device could comprise several in-situ OAM hops, for example in case one wants to trace the path of a packet through that device.

- o "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.
- o "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.
- o "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 might lead to packet size exceeding the Maximum Transmit Unit (MTU). A detailed discussion of the implications of oversized IPv6 header chains is found in [RFC7112]. The Path MTU restricts the amount of data that can be recorded for purpose of OAM within a data packet.

If in-situ OAM data is inserted at the edge of the domain (e.g., by intermediate routers) then the MTU on all interfaces with the domain (MTU_INT) MUST be \geq the maximum MTU on any "external" facing interfaces (MTU_EXT) and the total size of in-situ OAM data to be recorded MUST be \leq (MTU_INT - MTU_EXT).

In-situ OAM comprises two approaches to insert OAM data-records in the packets:

- o Pre-allocated: In this case, the encapsulating node inserts empty data records into the packet to cover the entire domain. The data records will be incrementally updated/filled as the packet progresses through the network. With pre-allocation the packet size is only changed at the encapsulating node and is kept constant throughout the domain. The pre-allocated approach is beneficial for software data-plane implementations where allocating the required space only once and index into the array to populate the data during transit avoids copy operations at every hop.
- o Incremental: Every node that desires to include in-situ OAM information extends the packet as needed. The incremental approach is beneficial for hardware data-plane implementations as it eliminates the need for the transit nodes to read the full array and lookup the pointer in the option prior to updating the data record contents.

The "incremental" or the "pre-allocated" approaches could even be combined in the same deployment - in which case two in-situ OAM headers would be present in the packet: One for the incremental approach and one for the pre-allocated approach. In such a case one would expect that nodes with a hardware data-plane would update the incremental header, whereas nodes with a software data-plane would process the pre-allocated header.

4.3. Administrative Boundaries

There are several challenges in enabling in-situ OAM in the public Internet as well as in corporate/enterprise networks across administrative domains, which include but are not limited to:

- o Deployment dependent, the data fields that in-situ OAM requires as part of a specific transport protocol may not be supported across administrative boundaries.
- o 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.
- o Discovery of network topology and details of the network devices across administrative boundaries may open up attack vectors compromising network security.
- o 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-situ 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

The ability to selectively enable in-situ OAM is valuable. While it may be desirable to enable data collection on all traffic or devices, this may not always be feasible. In-situ OAM collection may also come with a performance impact to forwarding rates or feature capabilities, which may be acceptable in only some locations. For example, the SPUD prototype 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-situ 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-situ OAM metadata is present is limited. In-situ OAM should use its own dedicated namespace (confined to the domain in-situ 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-situ 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-situ OAM data, the recorded data needs to be carried back to the source node.

One way to carry the in-situ 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-situ 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-situ 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-situ OAM Data Types

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

5.1. Generic Requirements

- REQ-G1: Classification: It should be possible to enable in-situ OAM on a selected set of traffic (e.g., per interface, based on an access control list specifying a specific set of traffic, etc.) The selected set of traffic can also be all traffic.
- REQ-G2: Scope: If in-situ OAM is used only within a specific domain, provisions need to be put in place to ensure that in-situ OAM data stays within the specific domain only.
- REQ-G3: Transport independence: Data formats for in-situ OAM shall be defined in a transport independent way. In-situ 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-situ 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-situ OAM information is to be gathered for both the underlay as well as the overlay network. Layering support

should not be limited to just underlay and overlay, but include more than two layers.

REQ-G5: MTU size: With in-situ OAM information added, packets MUST NOT become larger than the path MTU.

REQ-G5.1: If due to some reason a packet which contains in situ OAM data record cannot be forwarded due to the presence of in-situ OAM data records, the node SHOULD remove the in situ OAM data records and forward the packet, rather than drop the entire packet.

REQ-G5.2: If the encapsulating router is unable to insert in-situ OAM data records into a packet, e.g., due to MTU issues, even though it is configured to do so, it should use some operational means to inform the operator (e.g., syslog) about the inability to add in-situ OAM data records. Even if the in-situ OAM encapsulating node fails to add in-situ OAM data records, it should forward the packet normally.

REQ-G5.3: MTU size consideration for in-situ OAM MUST take domain specifics into account, e.g., changes of the domain topology due to path protection mechanisms might extend the hop count of a path etc.

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

REQ-G7: Data records format: It is desirable that the format of in-situ OAM data-records leverages already defined data formats for OAM as much as feasible.

REQ-G8: Combination with active OAM mechanisms: In-situ OAM should be useable for active network probing, like for example a customized version of traceroute. Decapsulating in-situ OAM nodes may have an ability to send the in-situ OAM information retrieved from the packet back to the source address of the packet or to the encapsulating node.

5.2. In-situ 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 might participate in in-situ 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 and/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. Jitter can be derived from the different timestamps gathered and does not necessarily need to be an explicit data record.
 - 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: Time with different precision should be supported. Use-case dependent, the required precision could e.g., be nanoseconds, microseconds, milliseconds, 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-situ OAM with Selected Hop Scope

REQ-S1: Proof of transit: Data shall be present which allows to securely prove that a packet has visited one or 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.

REQ-S1.2: Enable a fail-open service chaining system to be converted into a fail-closed service chaining system.

5.4. In-situ 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 situ OAM domain.

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

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

6. Security Considerations and Requirements

6.1. General considerations

General Security considerations will be expanded on in a later version of this document.

In-situ OAM is considered a "per domain" feature, where one or several operators decide on leveraging and configuring in-situ OAM

according to their needs. Still operators need to properly secure the in-situ OAM domain to avoid malicious configuration and use, which could include injecting malicious in-situ OAM packets into a domain.

6.2. Proof of Transit

Threat Model: Attacks on the deployments could be due to malicious administrators or accidental misconfiguration 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 POT 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: **Pre-play Attacks:** A active attacker should not be able to generate or reuse valid POT bits from legitimate packets, in order to prove to the verifier as valid packets. This slight variant of replay attacks. The attacker extracts POT bits from legitimate packets and ensure they do not reach the verifier. Subsequently reuse those POT bits in crafted packets.
- REQ-SEC5: **Recycle Secrets:** Any configuration of the secrets (like cryptographic keys, initialization vectors etc.) either in the controller or service functions should be re-configurable. 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-SEC6: 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 Jen Linkova, LJ Wobker, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Ignas Bagdonas, LJ Wobker, Erik Nordmark, and Andrew Yourtchenko for the comments and advice. This document leverages and builds on top of several concepts described in [I-D.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

[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>>.

9.2. Informative References

[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.brockners-proof-of-transit]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Leddy, J., and S. Youell, "Proof of Transit", draft-brockners-proof-of-transit-01 (work in progress), July 2016.

[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.kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop
Option Extension", draft-kitamura-ipv6-record-route-00
(work in progress), November 2000.
- [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>>.
- [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>>.
- [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

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

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

URI: petr@fb.com

Remy Chang
Barefoot Networks

Email: remy@barefootnetworks.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

F. Brockners
S. Bhandari
S. Dara
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JMPC
D. Mozes
Mellanox Technologies Ltd.
T. Mizrahi
Marvell
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
March 13, 2017

Requirements for In-situ OAM
draft-brockners-inband-oam-requirements-03

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-situ" 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 situ 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 September 14, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Conventions	4
3. Motivation for in-situ OAM	5
3.1. Path Congruency Issues with Dedicated OAM Packets	5
3.2. Results Sent to a System Other Than the Sender	6
3.3. Overlay and Underlay Correlation	6
3.4. SLA Verification	7
3.5. Analytics and Diagnostics	7
3.6. Frame Replication/Elimination Decision for Bi-casting /Active-active Networks	8
3.7. Proof of Transit	8
3.8. Use Cases	9
4. Considerations for In-situ OAM	11
4.1. Type of Information to be Recorded	11
4.2. MTU and Packet Size	12
4.3. Administrative Boundaries	13
4.3.1. Layered In-Situ OAM Domains	13
4.4. Selective Enablement	14
4.5. Forwarding Behavior	14
4.6. Optimization of Node and Interface Identifiers	14
4.7. Loop Communication Path (IPv6-specifics)	15
5. Requirements for In-situ OAM Data Types	15
5.1. Generic Requirements	15
5.2. In-situ OAM Data with Per-hop Scope	17

5.3. In-situ OAM with Selected Hop Scope	18
5.4. In-situ OAM with End-to-end Scope	18
6. Security Considerations and Requirements	19
6.1. General considerations	19
6.2. Proof of Transit	19
7. IANA Considerations	20
8. Acknowledgements	20
9. References	20
9.1. Normative References	20
9.2. Informative References	21
Authors' Addresses	22

1. Introduction

This document discusses requirements for "in-situ" Operations, Administration, and Maintenance (OAM) mechanisms. In this context, "in-situ OAM" refers to the concept of directly encoding telemetry information within the data packet as it traverses the network or telemetry domain. Mechanisms which add tracing or other types of telemetry information to the regular data traffic, sometimes also referred to as "in-band" 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 terms of "active" or "passive" OAM, "in-situ" OAM can be considered a hybrid OAM type. While no extra packets are sent, in-situ OAM adds information to the packets therefore cannot be considered passive. In terms of the classification given in [RFC7799] in-situ OAM could be portrayed as "hybrid OAM, type 1". "In-situ" 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.

A number of in-situ as well as in-band OAM mechanisms have been discussed, such as the INT spec for the P4 programming language [P4] or the SPUD prototype [I-D.hildebrand-spud-prototype]. The 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-situ OAM mechanism. Per what was already stated, in-situ 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-situ 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 between overlay and underlay transports is in place, checking service level agreements for the live data traffic, detailed statistics or verification of path selections within a domain, 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-situ OAM" which is IPv4 route recording [RFC0791], an in-situ 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-situ OAM operation, including but not limited to path tracing information, 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 determine whether any nodes were skipped while recording in-situ OAM information (i.e., in-situ 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 carry in-situ OAM data in various different transport protocols.

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:

ECMP: Equal Cost Multi-Path

IOAM: In-situ Operations, Administration, and Maintenance

LIISP:	Locator/ID Separation Protocol
MTU:	Maximum Transmit Unit
NSH:	Network Service Header
NFV:	Network Function Virtualization
OAM:	Operations, Administration, and Maintenance
PMTU:	Path MTU
SFC:	Service Function Chain
SLA:	Service Level Agreement
SR:	Segment Routing
SID:	Segment Identifier
VXLAN-GPE:	Virtual eXtensible Local Area Network, Generic Protocol Extension

This document defines in-situ Operations, Administration, and Maintenance (in-situ 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-situ OAM

In several scenarios it is beneficial to make information about the path a packet took through the network or through a network device as well as associated telemetry information 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-situ network diagnostics.

3.1. Path Congruency Issues with Dedicated OAM Packets

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-situ mechanism is an alternative in those cases.

In-situ mechanisms are not impacted by differences in the handling of probe traffic compared to other data packets, where probe traffic is handled differently (and potentially forwarded differently) by a router than regular data traffic. This obviously assumes that the addition of in-situ information does not change the forwarding behavior of the packet. Note that in certain implementations, the addition information to a transport protocol changes the forwarding behavior. IPv6 extension header processing is one example. Some implementations process IPv6 packets with extension headers in the "slow" path of a router, as opposed to the "fast" path.

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 helpful to eliminate the requirement that there be a working bidirectional 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-situ OAM the operator of the underlay can offer details of the connectivity in the underlay to the user of the overlay. This could include the ability to find out which underlay elements are shared by overlays and ability to know which overlays are mapped to the same underlay elements. Deployment dependent

underlay transit nodes can be configured to update OAM information in the overlay transport encapsulation. 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-situ 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-situ 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. In active/standby deployments in-situ OAM would only allow for SLA verification of the active path.

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 device 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.

In high-speed networks one can leverage and benefit from packet-accurate measurements with for example hardware-accurate timestamping (i.e., nanosecond-level verification) to support optimized packet scheduling and queuing mechanisms.

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 Internet of Things (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. Time sensitive networks (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 Locator/ID Separation Protocol (LISP), Network Service Header (NSH), Segment Routing (SR), etc.) blurs the line between the different trust domains, because the hand-off-points are no longer clearly defined physical interfaces, but are virtual interfaces. 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.

Deployed service chains without the presence of a "proof of transit" mechanism are typically operated as fail-open system: The packets that arrive at the end of a service chain are processed. Adding "proof of transit" capabilities to a service chain allows an operator to turn a fail-open system into a fail-close system, i.e. packets that did not properly traverse the service chain can be blocked.

A solution approach could be based on OAM data which is added to every packet for achieving Proof Of Transit (POT). 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 [I-D.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-situ OAM could be leveraged for several use cases, including:

- o 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 on a per Quality of Service (QoS) class.
- o 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).

- o Loss Statistics per Path: Retrieve loss statistics per flow and path in the network.
- o Path Heat Maps: Discover highly utilized links in the network.
- o 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.)
- o 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.
- o 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.
- o 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.
- o 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.
- o 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.
- o Device-level Troubleshooting and Optimization: In many cases, network operators could benefit from information specific to a single device. A non-exhaustive list of useful information includes: queue-depths, buffer utilization (either shared or per-port), packet latency measured from a known starting point, packet latency introduced by a single device, and resource utilization (CPU, memory, link bandwidth) of a given device or link. In some cases, this information changes over per-packet timescales (i.e., nanoseconds) and as such it is extremely challenging to collect

and report this info in an accurate and scalable manner. By encoding the information from the forwarding element directly within a data packet (i.e., within the 'fast-path') this information can be added to some or all data packets and then collected and analyzed by human or machine tools. This type of information is particularly valuable for troubleshooting low-level device errors as well as providing a knowledge feedback loop for network and device optimization.

- o Custom Network Probing: Active network probing and in-situ OAM can be combined for customized and efficient network probing. This could for example be a customized traceroute.

4. Considerations for In-situ OAM

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

4.1. Type of Information to be Recorded

The information gathered for in-situ 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 hops, such as path or service chain verification; information which only applies to the edges of a domain, such as sequence numbers. Note that a single network device could comprise several in-situ OAM hops, for example in case one wants to trace the path of a packet through that device.

- o "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.
- o "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.
- o "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 might lead to packet size exceeding the Maximum Transmit Unit (MTU). A detailed discussion of the implications of oversized IPv6 header chains is found in [RFC7112]. The Path MTU restricts the amount of data that can be recorded for purpose of OAM within a data packet.

If in-situ OAM data is inserted at the edge of the domain (e.g., by intermediate routers) then the MTU on all interfaces with the domain (MTU_INT) MUST be \geq the maximum MTU on any "external" facing interfaces (MTU_EXT) and the total size of in-situ OAM data to be recorded MUST be \leq (MTU_INT - MTU_EXT).

In-situ OAM comprises two approaches to insert OAM data fields in the packets:

- o Pre-allocated: In this case, the encapsulating node inserts empty data fields into the packet to cover the entire domain. The data fields will be incrementally updated/filled as the packet progresses through the network. With pre-allocation the packet size is only changed at the encapsulating node and is kept constant throughout the domain. The pre-allocated approach is beneficial for software data-plane implementations where allocating the required space only once and index into the array to populate the data during transit avoids copy operations at every hop.
- o Incremental: Every node that desires to include in-situ OAM information extends the packet as needed. The incremental approach is beneficial for hardware data-plane implementations as it eliminates the need for the transit nodes to read the full array and lookup the pointer in the option prior to updating the data fields contents.

The "incremental" or the "pre-allocated" approaches could even be combined in the same deployment - in which case two in-situ OAM headers would be present in the packet: One for the incremental approach and one for the pre-allocated approach. In such a case one would expect that nodes with a hardware data-plane would update the incremental header, whereas nodes with a software data-plane would process the pre-allocated header.

4.3. Administrative Boundaries

There are several challenges in enabling in-situ OAM in the public Internet as well as in corporate/enterprise networks across administrative domains, which include but are not limited to:

- o Deployment dependent, the data fields that in-situ OAM requires as part of a specific transport protocol may not be supported across administrative boundaries.
- o 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.
- o Discovery of network topology and details of the network devices across administrative boundaries may open up attack vectors compromising network security.
- o 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-situ 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.

4.3.1. Layered In-Situ OAM Domains

Like any OAM domain, in-situ OAM domains could also be layered/nested. Layering/nesting of in-situ OAM follows the general approach of OAM layering: An in-situ OAM domain consists of maintenance end-points (MEP) and maintenance intermediate points (MIP). MEP add to or remove the entire set of in-situ OAM data fields from the traffic, while only MIP update or add in-situ OAM data fields. When in-situ OAM layering is employed, a MEP of one layer becomes a MIP in the layer above, while MIP of the lower layer are not visible to the layer above – unless specifically configured otherwise.

Consider the following examples:

- o NSH over IPv6: In-situ OAM data fields could be present in both transport protocols: NSH and IPv6, with NSH forming the overlay network and IPv6 forming the underlay network. The network which deploys NSH would form an in-situ OAM domain. In addition each

IPv6 underlay network which connects two NSH nodes forms an in-situ OAM domain. The in-situ OAM domain with NSH as transport could be considered as layered on top of the different in-situ OAM domains which use IPv6 as transport.

- o NSH using an in-situ OAM aware transport: Consider a case where the underlay network would not natively support in-situ OAM, still the individual transport nodes would have the capability to "look deep into the packet" and update/add in-situ OAM information in the NSH header. The in-situ OAM domain with NSH as transport could be considered as layered on top of the different in-situ OAM domains which are in-situ OAM aware and connect the individual NSH nodes.

4.4. Selective Enablement

The ability to selectively enable in-situ OAM is valuable. While it may be desirable to enable data collection on all traffic or devices, this may not always be feasible. In-situ OAM collection may also come with a performance impact to forwarding rates or feature capabilities, which may be acceptable in only some locations. For example, the SPUD prototype 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-situ OAM are outside the scope of this document.

4.5. Forwarding Behavior

In-situ OAM adds additional data fields to live user traffic and as such changes the packet which is also why in-situ OAM is characterized as "hybrid, type 1" OAM. The effectiveness of in-situ OAM as a tool for operations depends on forwarding nodes not altering their forwarding behavior in case of in-situ OAM data fields being present in the packet. As a consequence, an implementation of in-situ OAM should not change the forwarding behavior of the packet, i.e. packets with or without in-situ OAM data fields should be handled the same way by a forwarding node (see also the associated requirement further below). Note that there are implementations where the addition of meta-data to live user traffic might cause the forwarding behavior of the packet to change, e.g. certain implementations handle IPv6 packets with or without extension headers differently (see [RFC7872]).

4.6. 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-situ OAM metadata is present is limited. In-situ OAM should use its own dedicated

namespace (confined to the domain in-situ 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-situ OAM data.

4.7. Loop Communication Path (IPv6-specifics)

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

One way to carry the in-situ 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-situ 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-situ 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-situ OAM Data Types

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

5.1. Generic Requirements

- REQ-G1: Classification: It should be possible to enable in-situ OAM on a selected set of traffic (e.g., per interface, based on an access control list specifying a specific set of traffic, etc.) The selected set of traffic can also be all traffic.
- REQ-G2: Scope: If in-situ OAM is used only within a specific domain, provisions need to be put in place to ensure that in-situ OAM data stays within the specific domain only.
- REQ-G3: Transport independence: Data formats for in-situ OAM shall be defined in a transport independent way. In-situ 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-situ 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-situ OAM information is to be gathered for both the underlay as well as the overlay network. Layering support should not be limited to just underlay and overlay, but include more than two layers.
- REQ-G5: MTU size: With in-situ OAM information added, packets MUST NOT become larger than the path MTU.
- REQ-G5.1: If due to some reason a packet which contains in situ OAM data fields cannot be forwarded due to the presence of in-situ OAM data fields, the node SHOULD remove the in situ OAM data fields and forward the packet, rather than drop the entire packet.
- REQ-G5.2: If the encapsulating router is unable to insert in-situ OAM data fields into a packet, e.g., due to MTU issues, even though it is configured to do so, it should use some operational means to inform the operator (e.g., syslog) about the inability to add in-situ OAM data fields. Even if the in-situ OAM encapsulating node fails to add in-situ OAM data fields, it should forward the packet normally.
- REQ-G5.3: MTU size consideration for in-situ OAM MUST take domain specifics into account, e.g., changes of the domain topology due to path protection mechanisms might extend the hop count of a path etc.
- REQ-G6: Data structure reuse: The data fields and associated types defined and used for in-situ OAM ought to be reusable for out-of-band OAM telemetry as well.
- REQ-G7: Data fields: It is desirable that the format of in-situ OAM data fields leverages already defined data formats for OAM as much as feasible.
- REQ-G8: Combination with active OAM mechanisms: In-situ OAM should be usable for active network probing, like for example a customized version of traceroute. Decapsulating in-situ OAM nodes may have an ability to send the in-situ OAM

information retrieved from the packet back to the source address of the packet or to the encapsulating node.

REQ-G9: Unaltered forwarding behavior of in-situ OAM nodes: The addition of in-situ OAM data fields should not change the way packets are forwarded within the in-situ OAM domain.

REQ-G10: Layering of in-situ OAM domains: It should be possible to layer in-situ OAM domains on each other. Layering should be supported within the same, as well as with different transport protocols which carry in-situ OAM data fields.

5.2. In-situ 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 might participate in in-situ 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 and/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. Jitter can be derived from the different

timestamps gathered and does not necessarily need to be an explicit data field.

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: Time with different precision should be supported. Use-case dependent, the required precision could e.g., be nanoseconds, microseconds, milliseconds, or seconds.

REQ-H6: Generic data fields (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-situ 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 fields, "random" and "cumulative" shall be present. The number of bits used for "random" and "cumulative" data fields can vary between deployments and should thus be configurable.

REQ-S1.2: Enable a fail-open service chaining system to be converted into a fail-closed service chaining system.

5.4. In-situ 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 situ OAM domain.

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

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

6. Security Considerations and Requirements

6.1. General considerations

General Security considerations will be expanded on in a later version of this document.

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

6.2. Proof of Transit

Threat Model: Attacks on the deployments could be due to malicious administrators or accidental misconfiguration 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 POT 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: Pre-play Attacks: A active attacker should not be able to generate or reuse valid POT bits from legitimate packets, in order to prove to the verifier as valid packets. This

slight variant of replay attacks. The attacker extracts POT bits from legitimate packets and ensure they do not reach the verifier. Subsequently reuse those POT bits in crafted packets.

REQ-SEC5: Recycle Secrets: Any configuration of the secrets (like cryptographic keys, initialization vectors etc.) either in the controller or service functions should be re-configurable. 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-SEC6: 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 Jen Linkova, LJ Wobker, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Ignas Bagdonas, LJ Wobker, Erik Nordmark, Vengada Prasad Govindan, and Andrew Yourtchenko for the comments and advice. This document leverages and builds on top of several concepts described in [I-D.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

[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>>.

9.2. Informative References

- [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.brockners-proof-of-transit]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C.,
Leddy, J., Youell, S., Mozes, D., and T. Mizrahi, "Proof
of Transit", draft-brockners-proof-of-transit-02 (work in
progress), October 2016.
- [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-10 (work in progress), November
2016.
- [I-D.kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop
Option Extension", draft-kitamura-ipv6-record-route-00
(work in progress), November 2000.
- [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>>.
- [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>>.
- [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

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

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

URI: petr@fb.com

Remy Chang
Barefoot Networks

Email: remy@barefootnetworks.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2017

F. Brockners
S. Bhandari
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JMPC
T. Mizrahi
Marvell
D. Mozes
Mellanox Technologies Ltd.
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
October 30, 2016

Encapsulations for In-situ OAM Data
draft-brockners-inband-oam-transport-02

Abstract

In-situ Operations, Administration, and Maintenance (OAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. In-situ OAM is to complement current out-of-band OAM mechanisms based on ICMP or other types of probe packets. This document outlines how in-situ OAM data records can be transported in protocols such as NSH, Segment Routing, VXLAN-GPE, native IPv6 (via extension headers), 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 May 3, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Conventions	4
3. In-Situ OAM Metadata Transport in IPv6	4
3.1. In-situ OAM in IPv6 Hop by Hop Extension Header	5
3.1.1. In-situ OAM Hop by Hop Options	5
3.1.2. Procedure at the Ingress Edge to Insert the In-situ OAM Header	7
3.1.3. Procedure at Transit Nodes	8
3.1.4. Procedure at the Egress Edge to Remove the In-situ OAM Header	8
4. In-situ OAM Metadata Transport in IPv4	9
5. In-situ OAM Metadata Transport in VXLAN-GPE	9
6. In-situ OAM Metadata Transport in NSH	11
7. In-situ OAM Metadata Transport in Segment Routing	13
7.1. In-situ OAM in SR with IPv6 Transport	13
7.2. In-situ OAM in SR with MPLS Transport	14
8. IANA Considerations	14
9. Manageability Considerations	14
10. Security Considerations	14
11. Acknowledgements	14
12. References	14
12.1. Normative References	14
12.2. Informative References	14
Authors' Addresses	16

1. Introduction

This document discusses transport mechanisms for "in-situ" Operations, Administration, and Maintenance (OAM) data records. In-situ OAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM data is added to the data packets rather than is being sent within packets specifically dedicated to OAM. A discussion of the motivation and requirements for in-situ OAM can be found in [I-D.brockners-inband-oam-requirements]. Data types and data formats for in-situ OAM are defined in [I-D.brockners-inband-oam-data].

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

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

- o IPv6
- o IPv4
- o VXLAN-GPE
- o NSH
- o Segment Routing (IPv6 and MPLS)

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

A feasibility study of in-situ OAM is currently underway as part of the FD.io project [FD.io]. The in-situ OAM implementation study should be considered as a "tool box" to showcase how "in-situ" 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

NSH: Network Service Header

OAM: Operations, Administration, and Maintenance

POT: Proof of Transit

SFC: Service Function Chain

SID: Segment Identifier

SR: Segment Routing

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

3. In-Situ OAM Metadata Transport in IPv6

This mechanisms of in-situ 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-situ 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 [I-D.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-situ 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. If the addition of the in-situ OAM Hop-by-Hop Option header would lead to the packet exceeding the MTU of the domain an error should be reported. The methods and procedures of how the error is reported

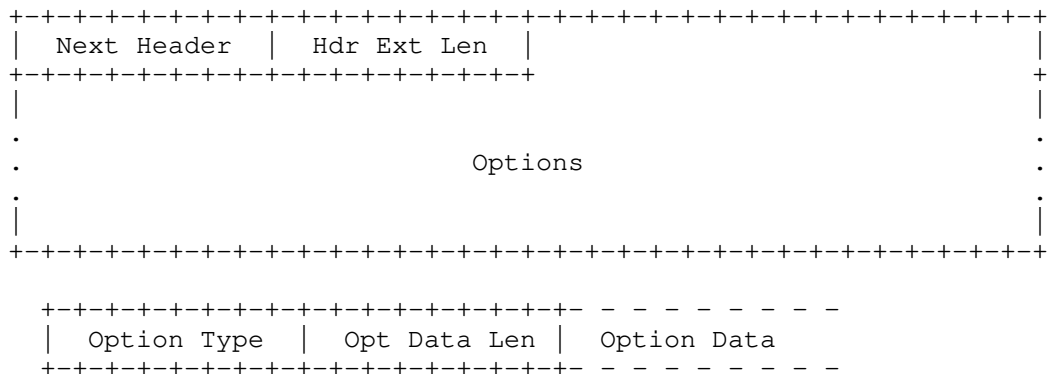
are outside the scope of this document. Likewise if an ICMPv6 forwarding error occurs between encapsulating and decapsulating nodes, the node generating the ICMPv6 error should strip the in-situ OAM Hop-by-Hop Option header before sending the ICMPv6 message to the source.

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

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

3.1.1. In-situ OAM Hop by Hop Options

Brief recap of the IPv6 hop-by-hop header as well as the options used for carrying in-situ 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-situ OAM data records are inserted as options in an IPv6 hop-by-hop extension header:

1. Tracing Option: The in-situ OAM Tracing option defined in [I-D.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_TRACE_OPTION_IPV6.

2. Proof of Transit Option: The in-situ OAM POT option defined in [I-D.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-situ OAM E2E option defined in [I-D.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-situ OAM Header

In an administrative domain where in-situ OAM is used, insertion of the in-situ OAM header is enabled at the required edge nodes (i.e. at the encapsulating/decapsulating nodes) by means of configuration.

Such a configuration SHOULD allow selective enablement of in-situ 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-situ OAM domain are retrieved are outside the scope of this document.

Let n = max number of nodes updating in-situ OAM data;
(calculated based on the packet size and the minimal MTU on all
links within the OAM domain)

Let k = number of node data records that can be allocated
by this node.

Let `node_data_size` = size of each `node_data` based on
in-situ OAM type.

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

  Update "Next Header" field in main IPv6 header and
  set "Next Header" field of OAM hbyh extension header
  appropriately.
}
```

3.1.3. Procedure at Transit Nodes

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

k = number of node data that this node can allocate

```
if (in-situ OAM ext hbyh ext-header is present) {
  if (Elements-left > 0) {
    populate node data at :
      node_data_start[Elements-left]
    Elements-left = Elements-left - 1
  }
}
```

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

egress_edge = list of interfaces where in-situ OAM hbyh ext header is to be stripped

Before forwarding packet out of interfaces in egress_edge list:

```
if (in-situ OAM hbyh ext-header is present) {  
    remove the in-situ 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-situ OAM ext header  
  
    Update "Next Header" field in main IPv6 header and  
    set "Next Header" field of OAM hbyh extension header  
    appropriately.  
}
```

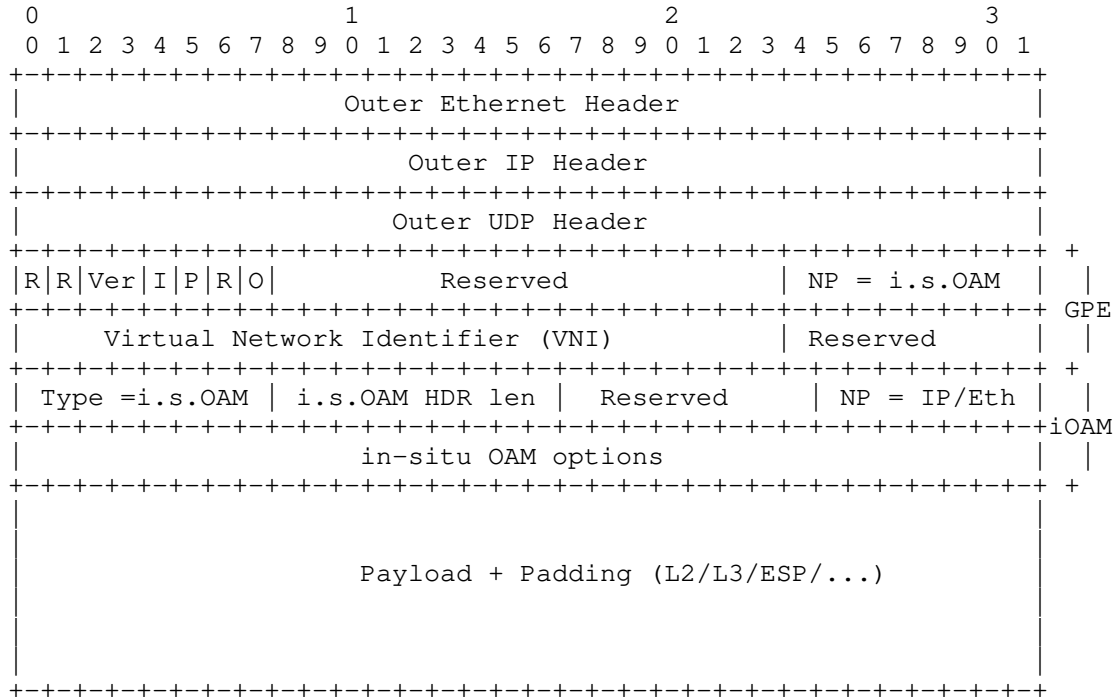
4. In-situ OAM Metadata Transport in IPv4

Transport of in-situ OAM data in IPv4 will be detailed in a future version of this document.

5. In-situ 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-situ OAM types are added as options within a new in-situ OAM protocol header in VXLAN GPE. In an administrative domain where in-situ OAM is used, insertion of the in-situ OAM protocol header in VXLAN GPE is enabled at the VXLAN GPE tunnel endpoint which also serve as in-situ OAM encapsulating/decapsulating nodes by means of configuration.

In-situ OAM header in VXLAN GPE header:



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

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

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

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

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Option Type | Opt Data Len | Option Data |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

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-situ OAM options defined in [I-D.brockners-inband-oam-data] are encoded with an option type allocated in the new in-situ OAM IANA registry - in-situ 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)

```

+-----+-----+-----+-----+-----+
|           0           |
+-----+-----+-----+-----+-----+

```

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)

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|           1           | Opt Data Len | Option Data |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

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.

6. In-situ 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 [I-D.brockners-inband-oam-requirements] and [I-D.brockners-proof-of-transit]) of in-situ OAM can be leveraged within NSH. In an administrative domain where in-situ OAM is used, insertion of the in-situ OAM data into the NSH header is enabled at the required nodes (i.e. at the in-situ OAM encapsulating/decapsulating nodes) by means of configuration.

Proof of transit in-situ OAM data is added as NSH Type 2 metadata:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      TLV Class=Cisco (0x0009) |C|      Type=POT |R|      Len=4      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+<--+
|                                     Random                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ P
|                                     Random(contd.)                             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ O
|                                     Cumulative                                 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ T
|                                     Cumulative (contd.)                         |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+<--+

```

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.

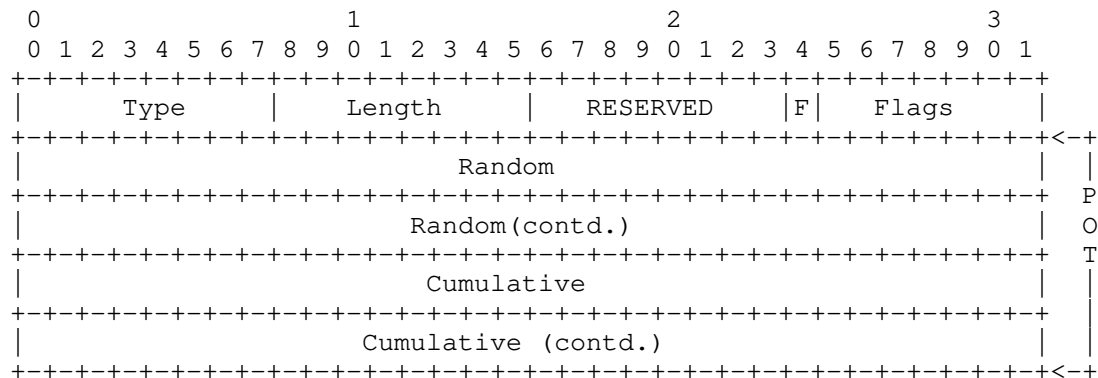
Cumulative: 64-bit Cumulative that is updated by the Service Functions.

7. In-situ OAM Metadata Transport in Segment Routing

7.1. In-situ 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-situ 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]). In an domain where in-situ OAM is used, insertion of the in-situ OAM data is enabled at the required edge nodes (i.e. at the in-situ OAM encapsulating/decapsulating nodes) by means of configuration.

A new "POT TLV" is defined for the SRH which is to carry proof of transit in situ 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.

7.2. In-situ OAM in SR with MPLS Transport

In-situ 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.

8. IANA Considerations

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

9. Manageability Considerations

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

10. Security Considerations

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

11. Acknowledgements

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

12. References

12.1. Normative References

[I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., and S. Youell, "Requirements for In-band OAM", draft-brockners-inband-oam-requirements-01 (work in progress), July 2016.

12.2. Informative References

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

- [I-D.brockners-inband-oam-data]
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., and S. Youell, "Data Formats for In-band OAM", draft-brockners-inband-oam-data-01 (work in progress), July 2016.
- [I-D.brockners-proof-of-transit]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Leddy, J., and S. Youell, "Proof of Transit", draft-brockners-proof-of-transit-01 (work in progress), July 2016.
- [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-02 (work in progress), September 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-06 (work in progress), September 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-10 (work in progress), September 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.kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop Option Extension", draft-kitamura-ipv6-record-route-00 (work in progress), November 2000.
- [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.

- [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

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

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

Email: petr@fb.com

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

ippm
Internet-Draft
Intended status: Informational
Expires: January 3, 2018

F. Brockners
S. Bhandari
V. Govindan
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JMPC
T. Mizrahi
Marvell
D. Mozes
Mellanox Technologies Ltd.
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
July 02, 2017

Encapsulations for In-situ OAM Data
draft-brockners-inband-oam-transport-05

Abstract

In-situ Operations, Administration, and Maintenance (OAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. In-situ OAM is to complement current out-of-band OAM mechanisms based on ICMP or other types of probe packets. This document outlines how in-situ OAM data fields can be transported in protocols such as NSH, Segment Routing, VXLAN-GPE, native IPv6 (via extension headers), 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 3, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Conventions	4
3. In-Situ OAM Metadata Transport in IPv6	4
3.1. In-situ OAM in IPv6 Hop by Hop Extension Header	5
3.1.1. In-situ OAM Hop by Hop Options	5
4. In-situ OAM Metadata Transport in IPv4	7
4.1. In-situ OAM Tracing in GRE	7
4.2. In-situ OAM POT in GRE	10
4.3. In-situ OAM End-to-End in GRE	12
5. In-situ OAM Metadata Transport in VXLAN-GPE	12
5.1. In-situ OAM Tracing in VXLAN-GPE	13
5.2. In-situ OAM POT in VXLAN-GPE	16
5.3. In-situ OAM Edge-to-Edge in VXLAN-GPE	18
6. In-situ OAM Metadata Transport in NSH	18
6.1. In-situ OAM Tracing in NSH	18
6.2. In-situ OAM POT in NSH	22
6.3. In-situ OAM Edge-to-Edge in NSH	24
7. In-situ OAM Metadata Transport in Segment Routing	25
7.1. In-situ OAM in SR with IPv6 Transport	25
7.2. In-situ OAM in SR with MPLS Transport	26
8. IANA Considerations	26
9. Manageability Considerations	26
10. Security Considerations	26
11. Acknowledgements	26

12. References	27
12.1. Normative References	27
12.2. Informative References	28
Authors' Addresses	28

1. Introduction

This document discusses transport mechanisms for "in-situ" Operations, Administration, and Maintenance (OAM) data fields. In-situ OAM records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM data is added to the data packets rather than is being sent within packets specifically dedicated to OAM. A discussion of the motivation and requirements for in-situ OAM can be found in [I-D.brockners-inband-oam-requirements]. Data types and data formats for in-situ OAM are defined in [I-D.brockners-inband-oam-data].

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

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

- o IPv6
- o IPv4
- o VXLAN-GPE
- o NSH
- o Segment Routing (IPv6 and MPLS)

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

A feasibility study of in-situ OAM is currently underway as part of the FD.io project [FD.io]. The in-situ OAM implementation study should be considered as a "tool box" to showcase how "in-situ" 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:

IOAM:	In-situ Operations, Administration, and Maintenance
MTU:	Maximum Transmit Unit
NSH:	Network Service Header
OAM:	Operations, Administration, and Maintenance
POT:	Proof of Transit
SFC:	Service Function Chain
SID:	Segment Identifier
SR:	Segment Routing
VXLAN-GPE:	Virtual eXtensible Local Area Network, Generic Protocol Extension

3. In-Situ OAM Metadata Transport in IPv6

This mechanisms of in-situ 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-situ 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 [I-D.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.

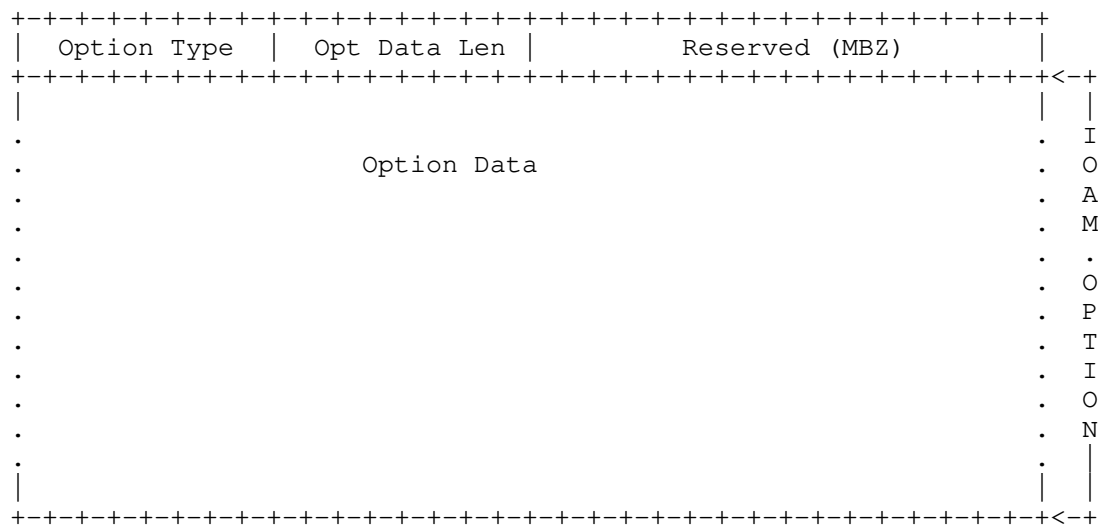
The in-situ OAM options translate into options for an IPv6 hop by hop extension header. The extension header would be inserted by either a host source of the packet, or by a transit/domain-edge node. If the addition of the in-situ OAM Hop-by-Hop Option header would lead to the packet exceeding the MTU of the domain an error should be reported. The methods and procedures of how the error is reported are outside the scope of this document. Likewise if an ICMPv6 forwarding error occurs between encapsulating and decapsulating nodes, the node generating the ICMPv6 error should strip the in-situ OAM Hop-by-Hop Option header before sending the ICMPv6 message to the source.

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

This section defines in-situ OAM for IPv6 transport. In-situ OAM Options are transported in IPv6 hop-by-hop extension header.

3.1.1. In-situ OAM Hop by Hop Options

IPv6 hop-by-hop option format for carrying in-situ OAM data fields:



Option Type	8-bit identifier of the type of option.
Opt Data Len	8-bit unsigned integer. Length of the Reserved and Option Data field of this option, in octets.
Reserved (MBZ)	16-bit field MUST be filled with zeroes.
Option Data	Variable-length field. Option-Type-specific data.

In-situ OAM Options are inserted as Option data as follows:

1. Pre-allocated Tracing Option: The in-situ OAM Preallocated Tracing option defined in [I-D.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_PRE_TRACE_OPTION_IPV6.
2. Incremental Tracing Option: The in-situ OAM Incremental Tracing option defined in [I-D.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_INCR_TRACE_OPTION_IPV6.

3. Proof of Transit Option: The in-situ OAM POT option defined in [I-D.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.

4. Edge to Edge Option: The in-situ OAM E2E option defined in [I-D.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.

4. In-situ OAM Metadata Transport in IPv4

Transport of in-situ OAM data in IPv4 will use GRE encapsulation.

GRE encapsulation is defined in [RFC2784]. IOAM is defined as a "set of Protocol Types" TBD_IANA_ETHERNET_NUMBER_IOAM_* and follows GRE header. These Protocol Types are defined in [RFC3232] as "ETHER TYPES" and in [ETYPES].

The different IOAM data fields defined in [I-D.brockners-inband-oam-data] are added as TLVs following the GRE header. In an administrative domain where IOAM is used, insertion of the IOAM protocol header in GRE is enabled at the GRE tunnel endpoints which also serve as IOAM encapsulating/decapsulating nodes by means of configuration.

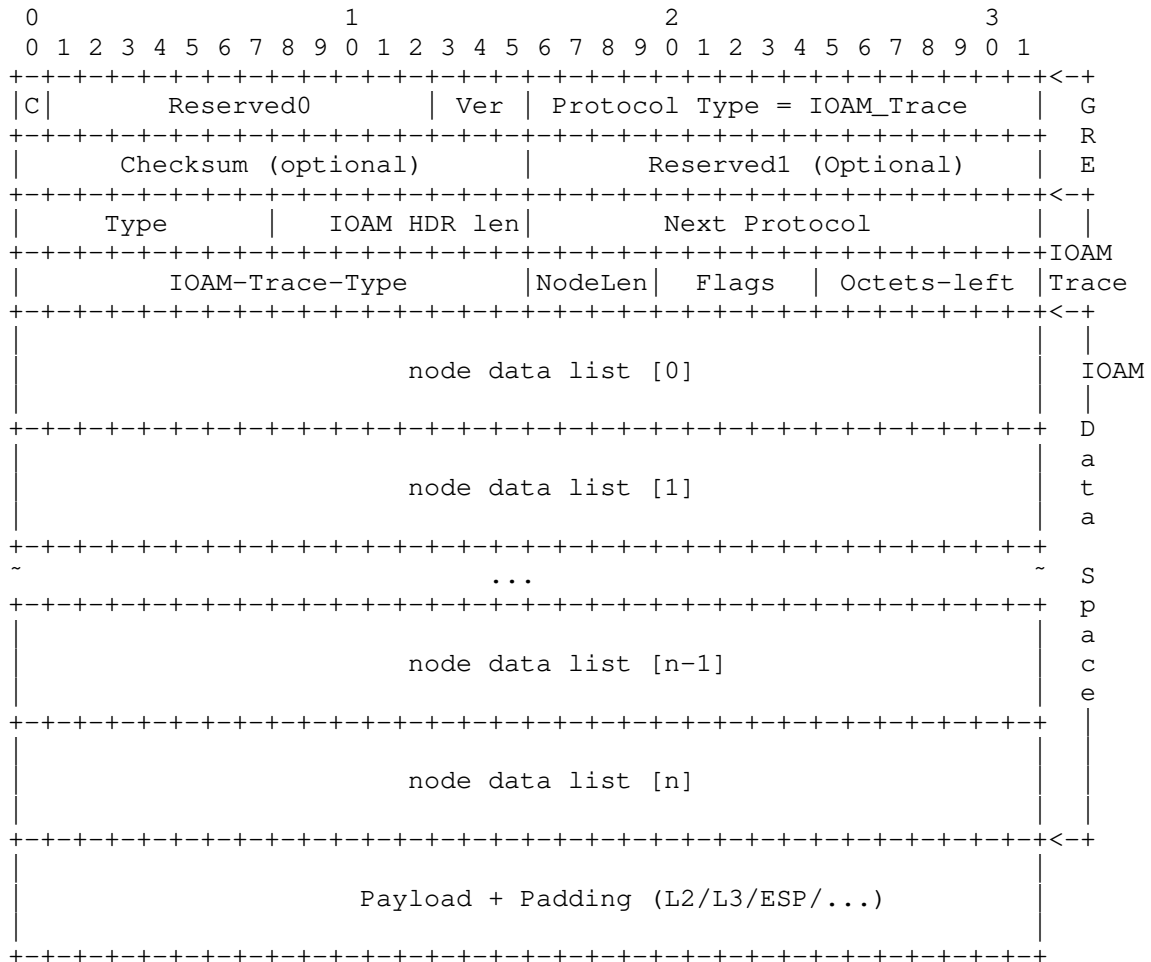
For IOAM the following new GRE protocol types are requested:

1. IOAM_Trace_Preallocated:
TBD_IANA_ETHERNET_NUMBER_IOAM_TRACE_PREALLOCATED
2. IOAM_Trace_Incremental:
TBD_IANA_ETHERNET_NUMBER_IOAM_TRACE_INCREMENTAL
3. IOAM_POT: TBD_IANA_ETHERNET_NUMBER_IOAM_POT
4. IOAM_End-to_End: TBD_IANA_ETHERNET_NUMBER_IOAM_E2E

4.1. In-situ OAM Tracing in GRE

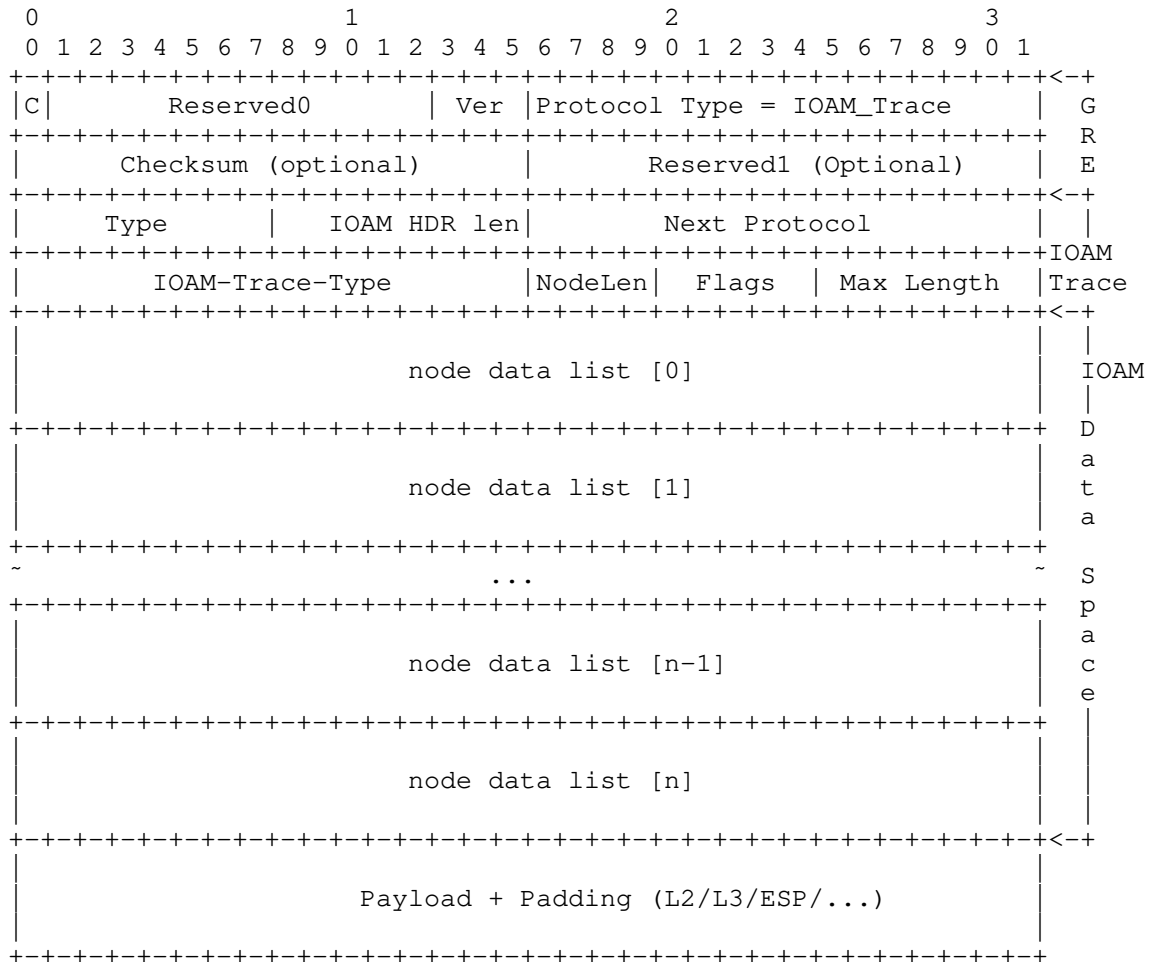
The packet formats of the pre-allocated IOAM trace and incremental IOAM trace when transported using GRE are defined as below. See [I-D.brockners-inband-oam-data] for details about pre-allocated and incremental IOAM trace options.

In-situ OAM Trace header following GRE header (Preallocated IOAM trace):



Pre-allocated Trace Option Data MUST be 4-octet aligned:

In-situ OAM Trace header following GRE header(Incremental IOAM trace):



In-situ OAM Incremental Trace Option Data MUST be 4-octet aligned:

The GRE header and fields are defined in [RFC2784] with Protocol Type set to TBD_IANA_ETHERNET_NUMBER_IOAM_TRACE. IOAM specific fields and header are defined here:

Type: 8-bit unsigned integer defining IOAM header type
 IOAM_TRACE_Preallocated or IOAM_Trace_Incremental are defined here.

IOAM HDR Len: 8 bits Length field contains the length of the variable metadata octets.

Next Protocol: 16 bits Next Protocol Type field contains the protocol type of the packet following IOAM protocol header. These Protocol Types are defined in [RFC3232] as "ETHER TYPES" and in [ETYPES]. An implementation receiving a packet containing a Protocol Type which is not listed in [RFC3232] or [ETYPES] SHOULD discard the packet.

IOAM-Trace-Type: 16-bit identifier of IOAM Trace Type as defined in [I-D.brockners-inband-oam-data] IOAM-Trace-Types.

Node Data Length: 4-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Flags: 5-bit field as defined in [I-D.brockners-inband-oam-data].

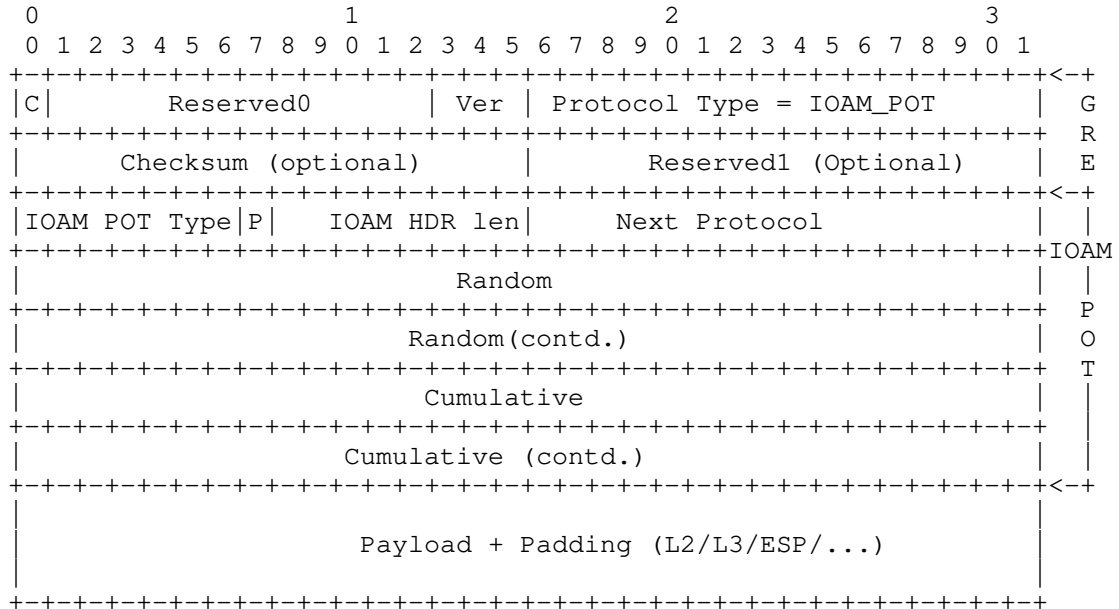
Octets-left: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Maximum-length: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Node data List [n]: Variable-length field as defined in [I-D.brockners-inband-oam-data].

4.2. In-situ OAM POT in GRE

In-situ OAM POT header following GRE header:



The GRE header and fields are defined in [RFC2784] with Protocol Type set to TBD_IANA_ETHERNET_NUMBER_IOAM_POT. IOAM specific fields and header are defined here:

IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included as defined in [I-D.brockners-inband-oam-data].

Profile to use (P): 1-bit as defined in [I-D.brockners-inband-oam-data] IOAM POT Option.

IOAM HDR Len: 8 bits Length field contains the length of the variable metadata octets.

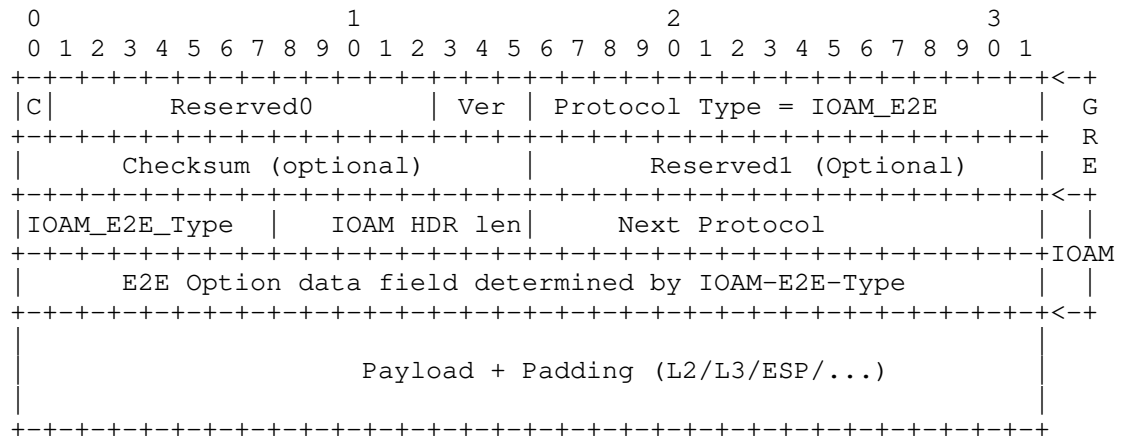
Next Protocol: 16 bits Next Protocol Type field contains the protocol type of the packet following IOAM protocol header. These Protocol Types are defined in [RFC3232] as "ETHER TYPES" and in [ETYPES]. An implementation receiving a packet containing a Protocol Type which is not listed in [RFC3232] or [ETYPES] SHOULD discard the packet.

Random: 64-bit Per-packet random number.

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

4.3. In-situ OAM End-to-End in GRE

In-situ OAM End-to-End header following GRE header:



IOAM E2E Type: 8-bit identifier of a particular E2E variant that dictates the E2E data that is included as defined in [I-D.brockners-inband-oam-data].

IOAM HDR Len: 8 bits Length field contains the length of the variable metadata octets.

Next Protocol: 16 bits Next Protocol Type field contains the protocol type of the packet following IOAM protocol header. These Protocol Types are defined in [RFC3232] as "ETHER TYPES" and in [ETYPES]. An implementation receiving a packet containing a Protocol Type which is not listed in [RFC3232] or [ETYPES] SHOULD discard the packet.

E2E Option data field: Variable length field as defined in [I-D.brockners-inband-oam-data] IOAM E2E Option.

5. In-situ 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 iIOAM types are added as options within a new IOAM protocol header in VXLAN GPE. In an administrative domain where IOAM is used, insertion of the IOAM

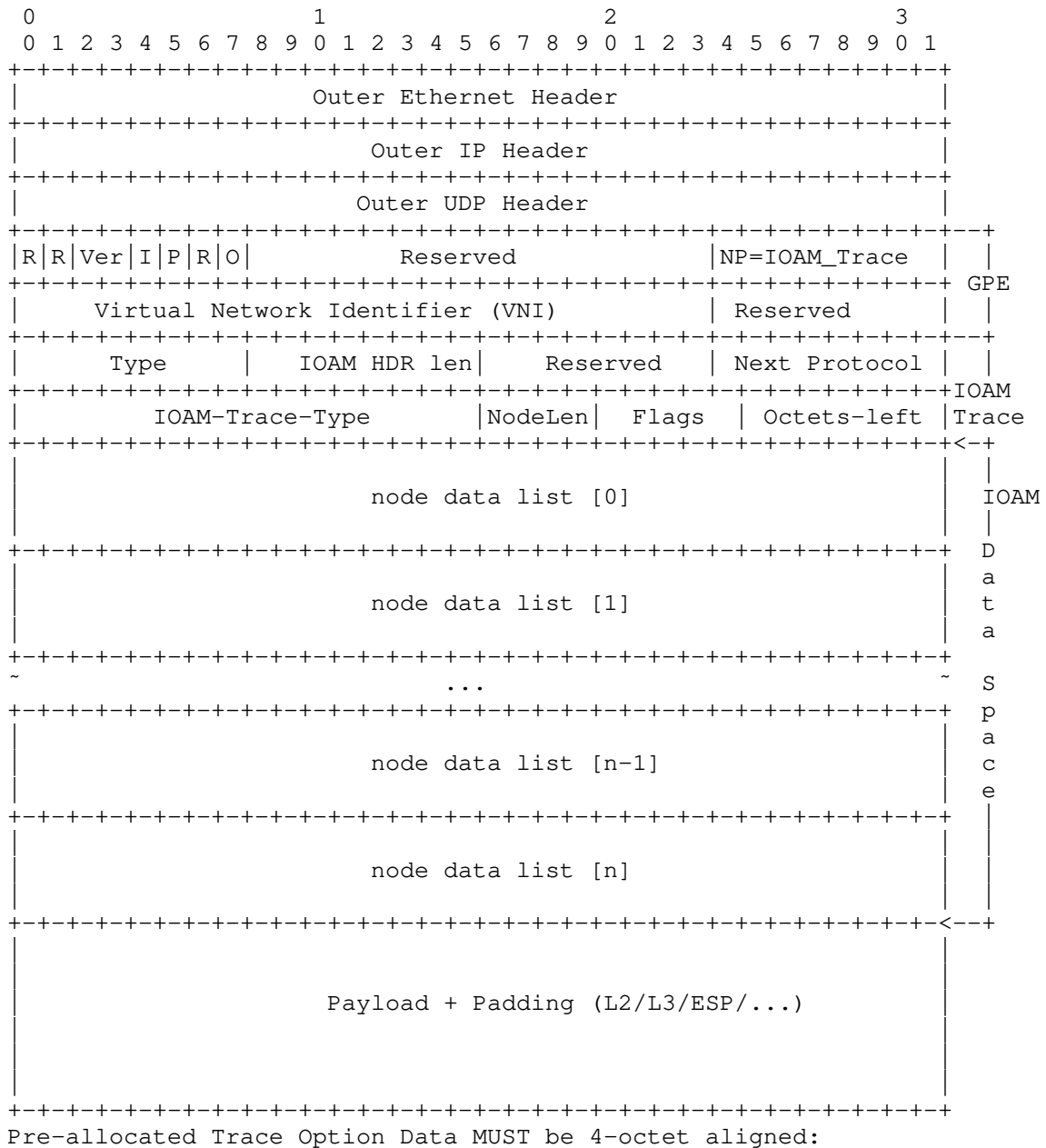
protocol header in VXLAN GPE is enabled at the VXLAN GPE tunnel endpoint which also serve as IOAM encapsulating/decapsulating nodes by means of configuration.

5.1. In-situ OAM Tracing in VXLAN-GPE

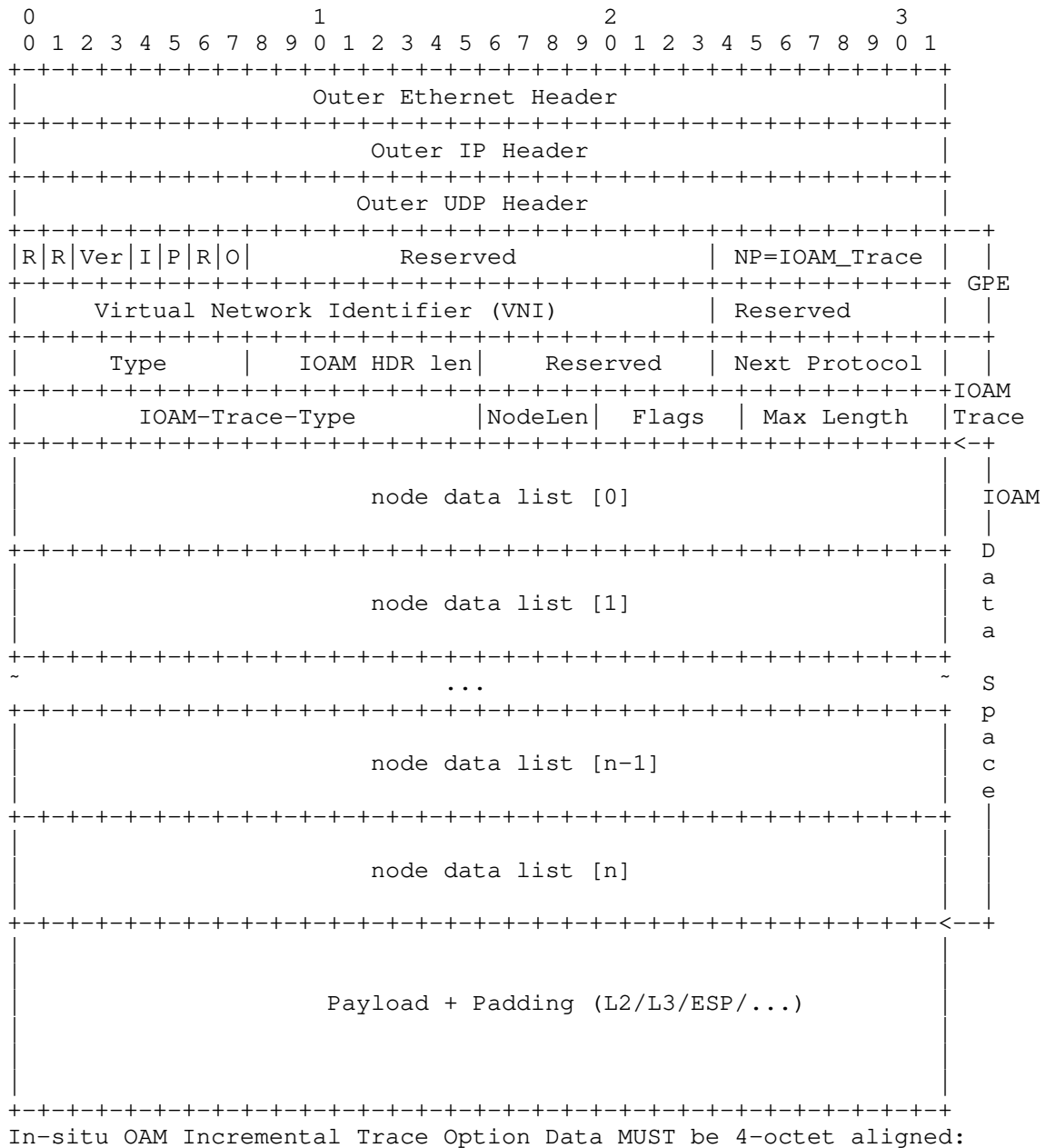
The packet formats of the pre-allocated IOAM trace and incremental IOAM trace when transported in VXLAN-GPE are defined as below. See [I-D.brockners-inband-oam-data] for details about pre-allocated and incremental IOAM trace options.

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

In-situ OAM Trace header following VXLAN GPE header
(Pre-allocated trace):



In-situ OAM Trace header following VXLAN GPE header
(Incremental IOAM trace):



Type: 8-bit unsigned integer defining IOAM header type
IOAM_TRACE_Preallocated or IOAM_Trace_Incremental are defined here.

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

Reserved: 8-bit reserved field MUST be set to zero.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The value is from the IANA registry setup for VXLAN GPE Next Protocol defined in [I-D.ietf-nvo3-vxlan-gpe].

IOAM-Trace-Type: 16-bit identifier of IOAM Trace Type as defined in [I-D.brockners-inband-oam-data] IOAM-Trace-Types.

Node Data Length: 4-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Flags: 5-bit field as defined in [I-D.brockners-inband-oam-data].

Octets-left: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

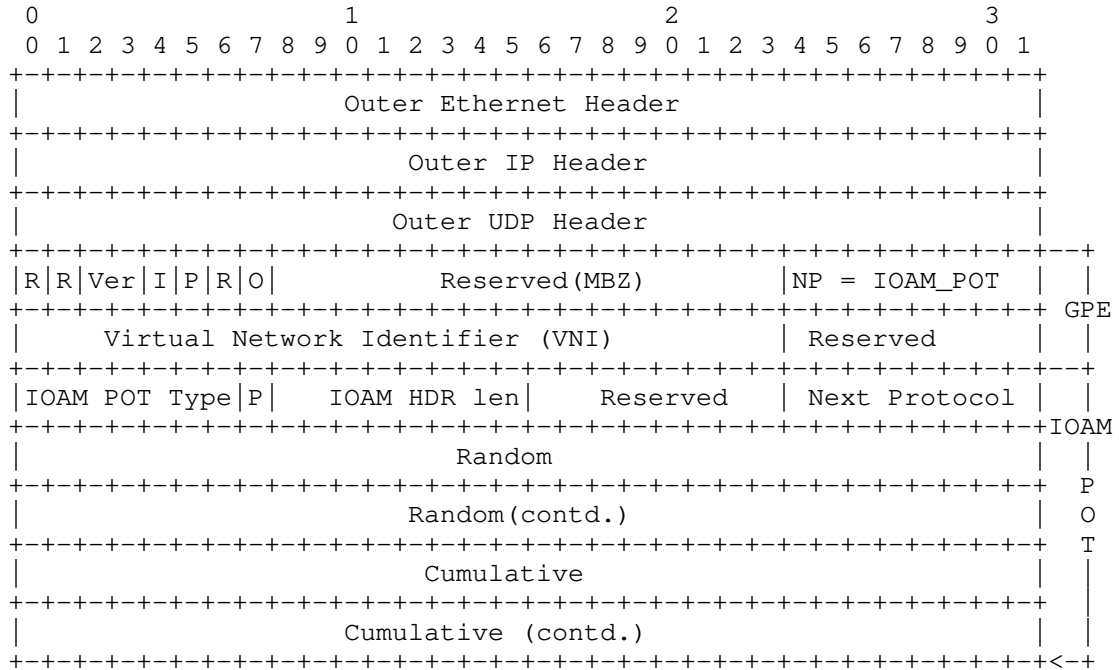
Maximum-length: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Node data List [n]: Variable-length field as defined in [I-D.brockners-inband-oam-data].

5.2. In-situ OAM POT in VXLAN-GPE

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

In-situ OAM POT header following VXLAN GPE header:



IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included as defined in [I-D.brockners-inband-oam-data].

Profile to use (P): 1-bit as defined in [I-D.brockners-inband-oam-data] IOAM POT Option.

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

Reserved: 8-bit reserved field MUST be set to zero.

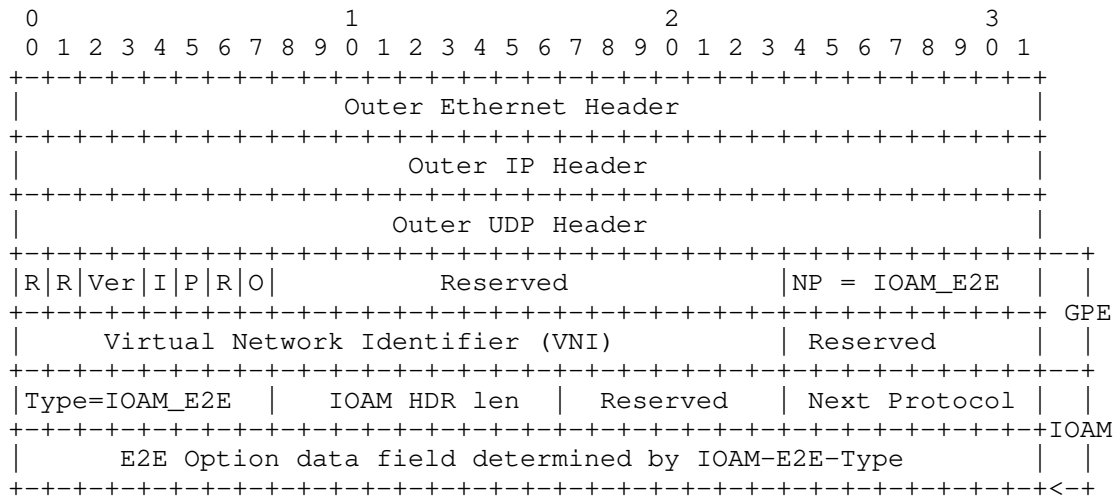
Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The value is from the IANA registry setup for VXLAN GPE Next Protocol defined in [I-D.ietf-nvo3-vxlan-gpe].

Random: 64-bit Per-packet random number.

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

5.3. In-situ OAM Edge-to-Edge in VXLAN-GPE

In-situ OAM Edge-to-Edge in VXLAN GPE header:



Type: 8-bit identifier of a particular E2E variant that dictates the E2E data that is included as defined in [I-D.brockners-inband-oam-data].

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

Reserved: 8-bit reserved field MUST be set to zero.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The value is from the IANA registry setup for VXLAN GPE Next Protocol defined in [I-D.ietf-nvo3-vxlan-gpe].

E2E Option data field: Variable length field as defined in [I-D.brockners-inband-oam-data] IOAM E2E Option.

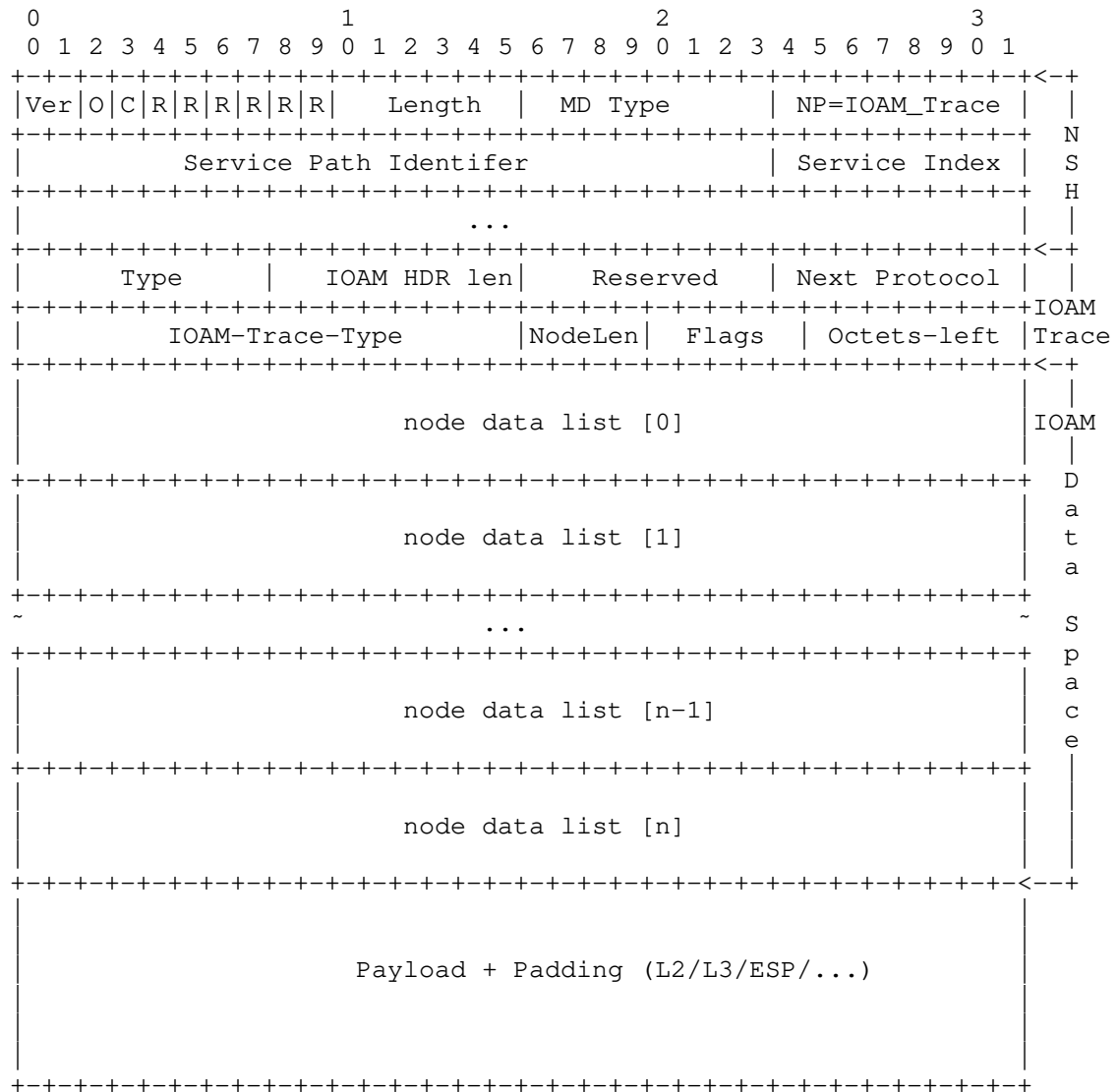
6. In-situ OAM Metadata Transport in NSH

6.1. In-situ OAM Tracing in NSH

The packet formats of the pre-allocated IOAM trace and incremental IOAM trace when transported in NSH are defined as below. See [I-D.brockners-inband-oam-data] for details about pre-allocated and incremental IOAM trace options.

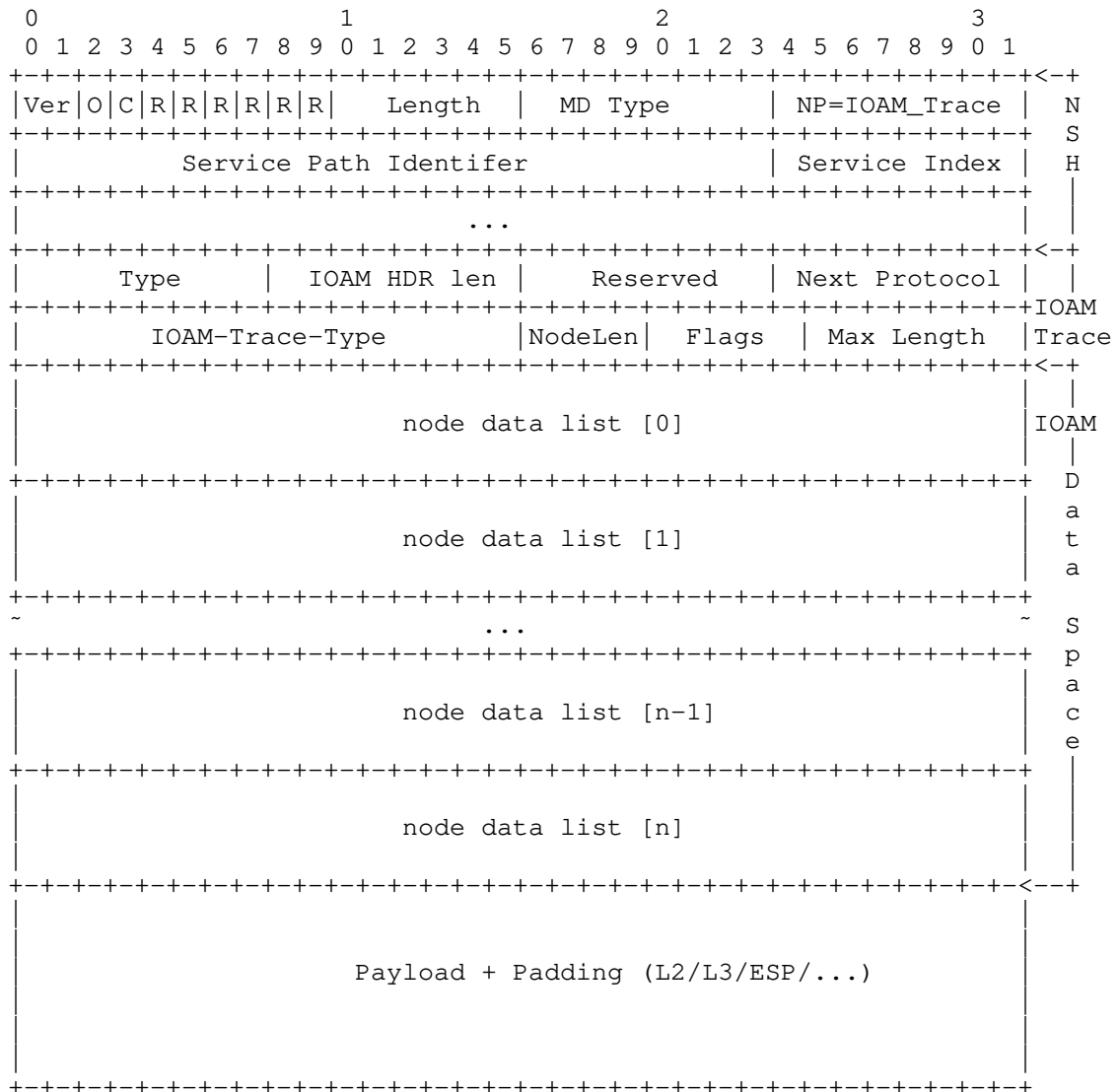
In Service Function Chaining (SFC) [RFC7665], the Network Service Header (NSH) [I-D.ietf-sfc-nsh] already includes path tracing capabilities [I-D.penno-sfc-trace]. Tracing information can be carried in-situ as IOAM data fields following NSH MDx metadata TLVs.

In-situ OAM Trace header following NSH MDx header
(Pre-allocated IOAM trace):



In-situ OAM Pre-allocated Trace Option Data MUST be 4-octet aligned:

In-situ OAM Trace header following NSH MDx header
(Incremental IOAM trace):



In-situ OAM Incremental Trace Option Data MUST be 4-octet aligned:

Next Protocol of NSH: TBD value for IOAM_Trace.

Type: 8-bit unsigned integer defining IOAM header type
IOAM_TRACE_Preallocated or IOAM_Trace_Incremental are defined here.

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

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

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol.

IOAM-Trace-Type: 16-bit identifier of IOAM Trace Type as defined in [I-D.brockners-inband-oam-data] IOAM-Trace-Types.

Node Data Length: 4-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Flags: 5-bit field as defined in [I-D.brockners-inband-oam-data].

Octets-left: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

Maximum-length: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

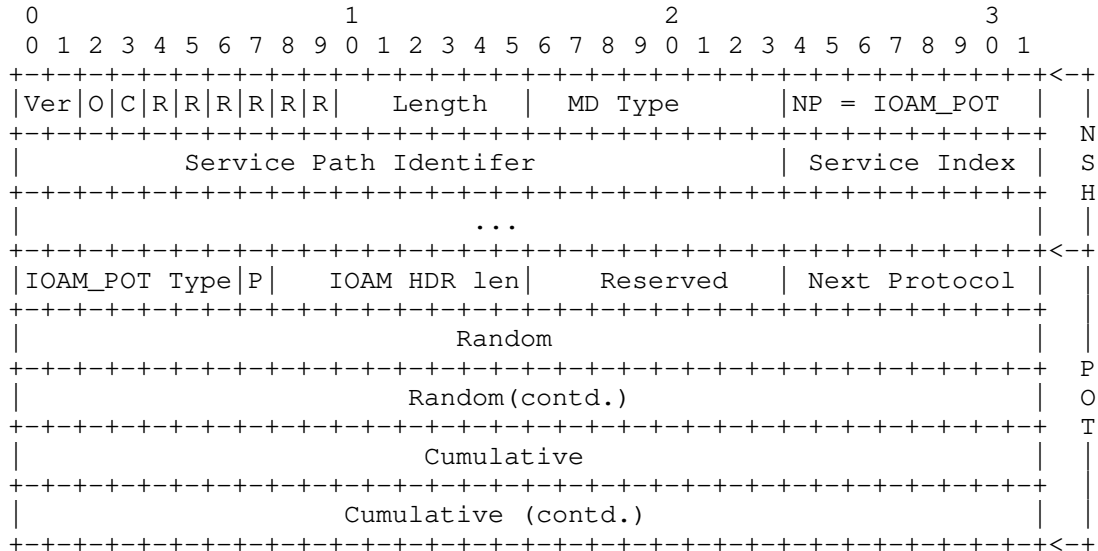
Node data List [n]: Variable-length field as defined in [I-D.brockners-inband-oam-data].

6.2. In-situ OAM POT in NSH

The "Proof of Transit" capabilities (see [I-D.brockners-inband-oam-requirements] and [I-D.brockners-proof-of-transit]) of in-situ OAM can be leveraged within NSH. In an administrative domain where in-situ OAM is used, insertion of the in-situ OAM data into the NSH header is enabled at the required nodes (i.e. at the in-situ OAM encapsulating/decapsulating nodes) by means of configuration.

Proof of transit in-situ OAM data is added as NSH Type 2 metadata:

In-situ OAM POT header following NSH MDx header:



Next Protocol of NSH: TBD value for IOAM_POT.

IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included as defined in [I-D.brockners-inband-oam-data].

Profile to use (P): 1-bit as defined in [I-D.brockners-inband-oam-data] IOAM POT Option.

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

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

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol.

Random: 64-bit Per-packet random number.

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

6.3. In-situ OAM Edge-to-Edge in NSH

The "Edge-to-Edge" capabilities (see [I-D.brockners-inband-oam-requirements]) of in-situ OAM can be leveraged within NSH. In an administrative domain where in-situ OAM is used, insertion of the in-situ OAM data into the NSH header is enabled at the required nodes (i.e. at the in-situ OAM encapsulating/decapsulating nodes) by means of configuration.

Edge-to-Edge in-situ OAM data is added as a TLV following NSH MDx metadata:

In-situ OAM E2E header following NSH MDx header:

[illegible]

Next Protocol of NSH: TBD value for IOAM_E2E.

IOAM E2E Type: 8-bit identifier of a particular E2E variant that dictates the IOAM E2E data that is included as defined in [I-D.brockners-inband-oam-data].

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

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

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol.

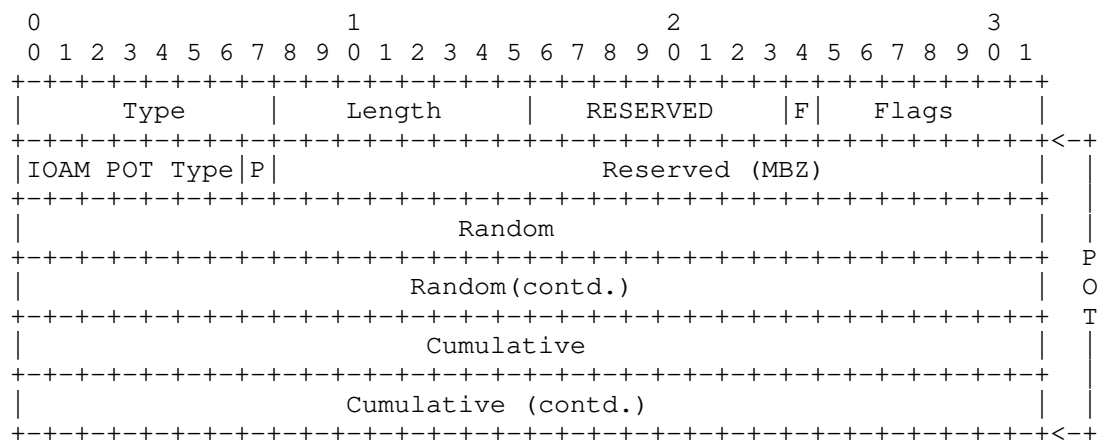
E2E Option data field: Variable length field as defined in [I-D.brockners-inband-oam-data] IOAM E2E Option.

7. In-situ OAM Metadata Transport in Segment Routing

7.1. In-situ OAM in SR with IPv6 Transport

Similar to NSH, a policy defined using Segment Routing for IPv6 can be verified using the in-situ 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]). In an domain where in-situ OAM is used, insertion of the in-situ OAM data is enabled at the required edge nodes (i.e. at the in-situ OAM encapsulating/decapsulating nodes) by means of configuration.

A new "POT TLV" is defined for the SRH which is to carry proof of transit in situ OAM data.



Type: To be assigned by IANA.

Length: 20.

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.

IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included as defined in [I-D.brockners-inband-oam-data].

Profile to use (P): 1-bit as defined in
[I-D.brockners-inband-oam-data] IOAM POT Option.

Reserved (MBZ): 24-bit field MUST be filled with zeroes.

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.

7.2. In-situ OAM in SR with MPLS Transport

In-situ 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.

8. IANA Considerations

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

9. Manageability Considerations

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

10. Security Considerations

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

11. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari
Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya
Nadahalli, Stefano Previdi, Hemant Singh, Erik Nordmark, LJ Wobker,
and Andrew Yourtchenko for the comments and advice. The authors
would like to acknowledge Craig Hill for contributing GRE IOAM
encapsulation. For the IPv6 encapsulation, this document leverages
and builds on top of several concepts described in
[I-D.kitamura-ipv6-record-route]. The authors would like to
acknowledge the work done by the author Hiroshi Kitamura and people
involved in writing it.

12. References

12.1. Normative References

- [ETYPES] "IANA Ethernet Numbers",
<<https://www.iana.org/assignments/ethernet-numbers/ethernet-numbers.xhtml>>.
- [I-D.brockners-inband-oam-data]
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., <>, R., and d. daniel.bernier@bell.ca, "Data Fields for In-situ OAM", draft-brockners-inband-oam-data-05 (work in progress), May 2017.
- [I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., <>, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.
- [I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-06 (work in progress), March 2017.
- [I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work in progress), April 2017.
- [I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-13 (work in progress), June 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <<http://www.rfc-editor.org/info/rfc2784>>.

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

12.2. Informative References

- [FD.io] "Fast Data Project: FD.io", <<https://fd.io/>>.
- [I-D.brockners-proof-of-transit]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Leddy, J., Youell, S., Mozes, D., and T. Mizrahi, "Proof of Transit", draft-brockners-proof-of-transit-03 (work in progress), March 2017.
- [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-13 (work in progress), June 2017.
- [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-12 (work in progress), June 2017.
- [I-D.kitamura-ipv6-record-route]
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop Option Extension", draft-kitamura-ipv6-record-route-00 (work in progress), November 2000.
- [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.
- [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

Vengada Prasad Govindan
Cisco Systems, Inc.

Email: venggovi@cisco.com

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

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

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

Email: petr@fb.com

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

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: May 3, 2017

F. Brockners
S. Bhandari
S. Dara
C. Pignataro
Cisco
J. Leddy
Comcast
S. Youell
JMPC
D. Mozes
Mellanox Technologies Ltd.
T. Mizrahi
Marvell
October 30, 2016

Proof of Transit
draft-brockners-proof-of-transit-02

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on May 3, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

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

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

1. Introduction

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

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

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

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

2. Conventions

Abbreviations used in this document:

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

LISP: Locator/ID Separation Protocol

LPC: Lagrange Polynomial Constants

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

NSH: Network Service Header

POT: Proof of Transit

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

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

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

SFC: Service Function Chain

SR: Segment Routing

3. Proof of Transit

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

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

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

3.1. Basic Idea

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

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

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

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

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

3.2. Solution Approach

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

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

3.2.1. Setup

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

3.2.2. In Transit

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

3.2.3. Verification

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

3.3. Illustrative Example

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

3.3.1. Basic Version

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

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

3.3.1.1. Secret Shares

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

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

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

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

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

3.3.1.2. Lagrange Polynomials

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

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

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

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

3.3.1.3. LPC Computation

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

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

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

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

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

3.3.1.4. Reconstruction

Reconstruction of the polynomial is well-defined as

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

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

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

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

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

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

3.3.1.5. Verification

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

3.3.2. Enhanced Version

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

3.3.2.1. Random Polynomial

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

3.3.2.2. Reconstruction

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

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

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

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

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

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

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

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

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

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

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

The verification using CML is discussed in next section.

3.3.2.3. Verification

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

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

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

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

3.3.3. Final Version

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

3.4. Operational Aspects

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

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

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

3.5. Alternative Approach

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

3.5.1. Basic Idea

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

3.5.2. Pros

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

3.5.3. Cons

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

4. Sizing the Data for Proof of Transit

Proof of transit requires transport of two data 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	$2^{64} = \text{approx. } 2 \times 10^{19}$	approx. 310,000 years
10 Gbps	64	$2^{64} = \text{approx. } 2 \times 10^{19}$	approx. 31,000 years
100 Gbps	64	$2^{64} = \text{approx. } 2 \times 10^{19}$	approx. 3,100 years
1 Gbps	32	$2^{32} = \text{approx. } 4 \times 10^9$	2,200 seconds
10 Gbps	32	$2^{32} = \text{approx. } 4 \times 10^9$	220 seconds
100 Gbps	32	$2^{32} = \text{approx. } 4 \times 10^9$	22 seconds

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

5. Node Configuration

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

5.1. Procedure

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

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

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

5.2. YANG Model

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

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

  yang-version 1;

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

  prefix ietf-pot-profile;

  organization "IETF xxx Working Group";
```

```
contact "";

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

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

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

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

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

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

    leaf secret-share {
```



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

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

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

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

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

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

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

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

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

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

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

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

6. IANA Considerations

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

7. Manageability Considerations

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

8. Security Considerations

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

8.1. Proof of Transit

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

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

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

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

8.2. Cryptanalysis

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

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

8.3. Anti-Replay

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

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

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

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

8.4. Anti-Preplay

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

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

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

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

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

8.5. Anti-Tampering

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

8.6. Recycling

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

8.7. Redundant Nodes and Failover

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

8.8. Controller Operation

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

The Controller only interacts with the nodes during the initial configuration and thereafter at regular intervals at which the operator chooses to switch to a new set of secrets. In case 64 bits are used for the data-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.

8.9. Verification Scope

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

8.9.1. Node Ordering

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

8.9.2. Stealth Nodes

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

9. Acknowledgements

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

10. References

10.1. Normative References

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

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

10.2. Informative References

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: sadara@cisco.com

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

Email: cpignata@cisco.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

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

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

T. Mizrahi
Marvell
May 7, 2018

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

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

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

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

Table of Contents

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

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

1. Introduction

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

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

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

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

2. Conventions

Abbreviations used in this document:

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

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

LPC: Lagrange Polynomial Constants

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

NSH: Network Service Header

POT: Proof of Transit

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

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

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

SFC: Service Function Chain

SR: Segment Routing

3. Proof of Transit

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

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

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

3.1. Basic Idea

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

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

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

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

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

3.2. Solution Approach

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

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

3.2.1. Setup

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

3.2.2. In Transit

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

3.2.3. Verification

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

3.3. Illustrative Example

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

3.3.1. Basic Version

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

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

3.3.1.1. Secret Shares

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

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

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

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

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

3.3.1.2. Lagrange Polynomials

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

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

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

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

3.3.1.3. LPC Computation

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

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

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

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

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

3.3.1.4. Reconstruction

Reconstruction of the polynomial is well-defined as

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

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

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

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

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

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

3.3.1.5. Verification

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

3.3.2. Enhanced Version

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

3.3.2.1. Random Polynomial

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

3.3.2.2. Reconstruction

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

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

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

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

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

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

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

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

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

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

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

The verification using CML is discussed in next section.

3.3.2.3. Verification

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

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

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

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

3.3.3. Final Version

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

3.4. Operational Aspects

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

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

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

3.5. Alternative Approach

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

3.5.1. Basic Idea

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

3.5.2. Pros

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

3.5.3. Cons

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

4. Sizing the Data for Proof of Transit

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

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

2. CML: Cumulative

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

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

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

5. Node Configuration

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

5.1. Procedure

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

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

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

5.2. YANG Model

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

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

    yang-version 1;

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

    prefix ietf-pot-profile;

    organization "IETF xxx Working Group";
```

```
contact "";

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

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

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

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

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

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

    leaf secret-share {
```

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

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

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

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

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

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

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

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



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

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

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

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

6. IANA Considerations

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

7. Manageability Considerations

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

8. Security Considerations

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

8.1. Proof of Transit

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

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

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

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

8.2. Cryptanalysis

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

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

8.3. Anti-Replay

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

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

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

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

8.4. Anti-Preplay

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

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

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

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

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

8.5. Anti-Tampering

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

8.6. Recycling

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

8.7. Redundant Nodes and Failover

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

8.8. Controller Operation

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

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

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

8.9. Verification Scope

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

8.9.1. Node Ordering

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

8.9.2. Stealth Nodes

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

9. Acknowledgements

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

10. References

10.1. Normative References

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

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

10.2. Informative References

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: sadara@cisco.com

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

Email: cpignata@cisco.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

David Mozes

Email: mosesster@gmail.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

OPSA Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 27, 2017

R. Chen
L. Zhang
H. Deng
Huawei Technologies
L. Geng
China Mobile
C. Xie
China Telecom
October 24, 2016

YANG Data Model for Composite VPN Service Delivery
draft-chen-opsawg-composite-vpn-dm-00

Abstract

The operator facing data model is valuable to reduce the operations and management. This document describes the YANG data model of the composite VPN for operators to provision, optimize, monitor and diagnose the end to end PE-based VPN services across multiple autonomous systems (ASes). The model from this document are limited to multiple ASes within one service provider.

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 April 27, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Definitions	3
3. Use Cases and Usage	4
4. Data Model Design Requirements	5
5. Design of the Data Model	7
5.1. Composite VPN	7
5.2. Access Point	8
5.2.1. Termination Point Basic Information	10
5.2.2. QoS	10
5.2.3. Routing Protocol	11
5.3. Segmental VPN	11
6. YANG Module	11
7. IANA Considerations	45
8. Security Considerations	45
9. Acknowledgements	45
10. References	46
10.1. Normative References	46
10.2. Informative References	46
Authors' Addresses	46

1. Introduction

Internet Service Providers (ISPs) have significant interest on providing Provider Edge (PE) based virtual private network (VPN) services, in which the tunnel endpoints are the PE devices. In this case, the Customer Edge (CE) devices do not need to have any special VPN capabilities. Customers can reduce support costs by outsourcing VPN operations to ISPs and using the obtained connectivity.

Typically, customers require either layer 2 or layer 3 connectivity services to exchange traffic among a collection of sites. The ISP

gets the requirement and deploys the end to end VPN with an orchestrator across multiple autonomous systems (AS) within its administration.

The YANG data model[RFC7950] described in [I-D.ietf-l3sm-l3vpn-service-model] is used for communication between customers and network operators. It facilitates customers to request the layer 3 VPN service while concealing many provider parameters they do not need to know.

However, the network operators have a different view of the managed network. An operator facing data model is required to reduce the operations and management while still having full control on the network. So that the operators can provision, optimize, monitor and diagnose the VPN deployment based on the existing network infrastructure. Standardization of such a operator facing data model can help operators to operate and manage the VPN deployment in a standard way, and facilitate automation of optimization and diagnosis by standardizing the communication among multi-vendor services such as inventory, provisioning, monitoring, analysis and so on.

This document describes the generic VPN data model from the operators' view for the PE-based VPN service configuration. It aims at providing a simplified configuration on how the requested VPN service is to be deployed over the shared network infrastructure. This data model is limited to PE-Based VPNs as described in RFC 4110 [RFC4110] with the combination of layer 2 and layer 3 VPN services across multiple ASes within one ISP.

2. Definitions

- o Customer Facing Service: The highly abstracted service which can be easily understood and consumed by the customer. With the emerging of cloud computing, customers require fast, easy to use, and on demand service from network operators.
- o Operator Facing Service: The service provided for the operator to analyse, optimize, monitor and diagnose the network. It usually provides more detailed information related to the operations and management.
- o Segmental VPN service: The VPN service deployed for one VPN segment within one AS.
- o Composite VPN service: The VPN service deployed within the ISP administrative domain across one or more segments. It could be a combination of layer 2 and layer 3 VPN services for each segment.

3. Use Cases and Usage

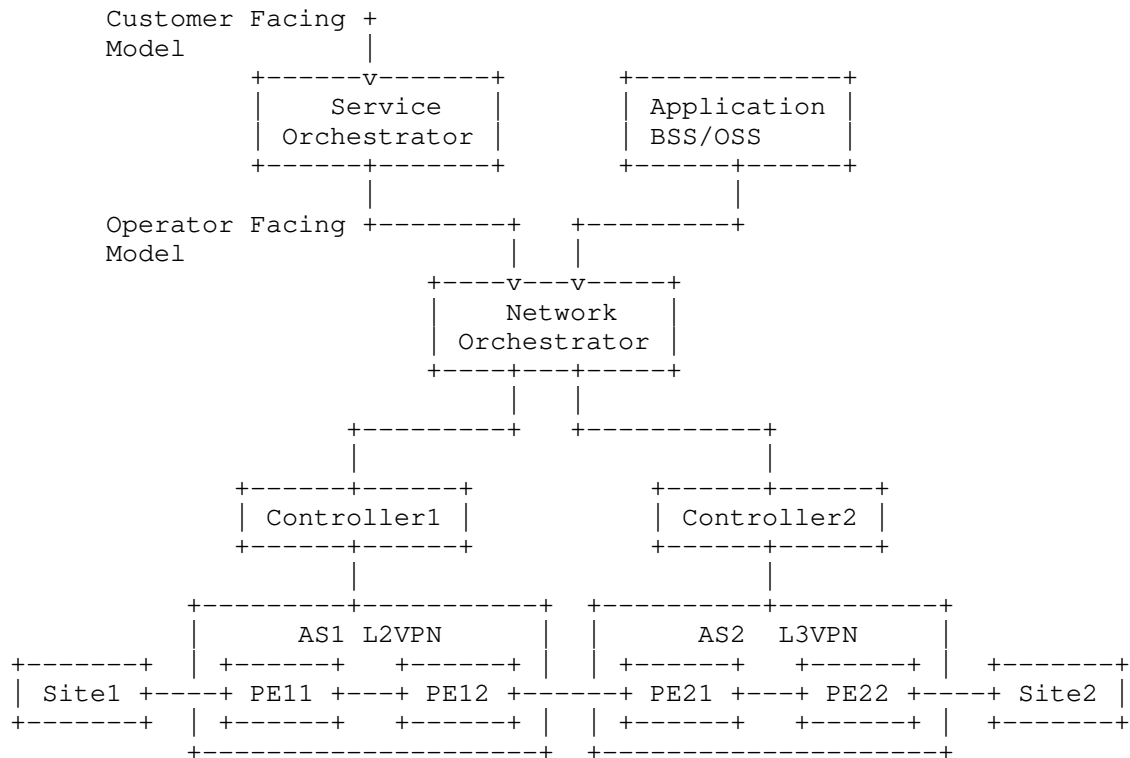
In practice, ISP may have various scenarios for the end to end VPN service deployment depending on the network infrastructure and the customer sites connectivity requirements. It will consequently generate requirements of the generic Composite VPN service delivery model design. The Composite VPN data model described in this document covers the following scenarios:

- o Multi-AS VPN Service: Customer sites are located in different autonomous systems(AS). ISP need to operate and manage the VPN service across multiple ASes. There are several different deployment scenarios in this case, e.g. single provider and multiple providers. This document only considers a single provider has multiple ASes which is less complex than the multi-provider and multi-AS case.
- o Composite L2 and/or L3 VPN Service: Although the customer may request either layer 2 or layer 3 VPN service, the network infrastructure among customer sites may require different VPN service in the corresponding AS. So, an end to end VPN service within the ISP domain may be a composition of multiple segmental layer 2 and layer 3 VPN services. For example, a typical use case is that the enterprise need a layer 3 VPN connection to the remote data center. The end to end VPN connection may go across the metro access network and the IP core network. So the actual deployment of the customer's request in ISP network maybe the Virtual Private Wire Service (VPLS) in the metro access network and the L3VPN in the IP WAN.
- o Dynamic Site Insertion: The customer site that is not in the previously provisioned VPN can be quickly included.

A typical usage of this operator facing model is as a description of the end to end PE based VPN service for a network orchestration layer [I-D.wu-opsawg-service-model-explained] which can then be translated to Segmental VPN information for the configuration of domain controllers. As shown in the following figure, while, for example, users may send highly abstracted layer 3 VPN service requests to the service orchestrator, it cannot provide enough information for operators to operate and manage an end to end VPN service. The operator facing interface enables configuration of VPN deployment by introducing more network knowledge and governance policies. For example :

- o Add the operational requirements for operation visualization, monitoring, and diagnosis. This requires more information on the Segmental VPN deployment in each AS.

- o Optimize the VPN deployment of the customer's requests based on the exiting networking, e.g. deploy the L3VPN request from the customer to multiple VPN segments (IP Radio Access Network, Packet Transport Network, IP Core) in the end to end environment.
- o Manage various policies for different customers.



4. Data Model Design Requirements

In order to describe the operator facing interface for the end to end PE based VPN service, the data model design should address the following requirements:

1. As the operator facing model, the model need to facilitate the operator to provision, optimize, monitor and diagnose the end to end PE-based VPN services across multiple autonomous systems. The model should be described from the operator's view of the VPN, not from the customer's view. For example, when the customer requests a VPN service, this can be expressed as a set of sites with the interconnected VPN. Both the sites and the VPN

model only consist of information that can be provided by the customer. For the same VPN service, the operator need to know information for operations and management, e.g. the access points on the PE, the VPN segment in each AS, and even the inter-connection between two ASes.

2. The model facilitates that the operator can quickly find all the information related to one end-to-end VPN of a particular customer. So that the operator can easily deal with one end to end VPN deployment. For example, the operator can trace how a VPN service request from the customer is really deployed (possibly across several ASes) in the network infrastructure. So when a connection failure happens, the operator can quickly determine where the problem is.
3. The model must be able to express various number of Segmental VPN composition. As described in Section 3, the operator need to operate and manage VPN among customer sites located in different autonomous systems(AS). In this case, an end to end VPN service deployed across multiple ASes, each of which can be described as a Segmental VPN. So the model should have the flexibility to describe both single AS and multi-AS VPN cases.
4. The model should include the basic information about the end to end VPN service. On one hand the operator can easily understand the deployment of one customer request. On the other hand, the overall information contains many parameters that can be referred and reused by many associated Segmental VPN models. This could be an abstract of the customer facing VPN model with information can be reused from operator's view.
5. The model should allow to define one or multiple Segmental VPN information for each AS among the sites. As described in Section 3, an end to end VPN service within the ISP domain may consist of multiple segmental layer 2 and layer 3 VPN services. So the Segmental VPN description should have the capability to address the type of technology in use, and the corresponding parameters that will be used for the operations and management.
6. The model should facilitate operators to know the Access Point (AP) information for both Composite VPN and Segmental VPN. The AP of a Composite VPN is the interface between the provider network and the customer network. The AP of the Segmental VPN is the interface between two adjacent Segmental VPNs. The AP information is crucial because it has to match the remote peer for connectivity, and it usually contains information that is useful for the operations and management. For example, the AP

may indicate what's the peer connected, and the routing protocol that is used for exchanging routing information, and so on.

7. The VPN provisioning policy is optional but recommended in the model. The operator can predefine several VPN provisioning policies based on the offered business. The policy description may include the naming, path selection, VPN concatenation rules, and resource pools, such as route target, route distinguisher, VLAN, and IP address. So when the customer requests a VPN, the system can automatically generate the end to end VPN planning according to the provisioning policies. To be simple, the model can only have an ID index refers to the VPN provisioning policy defined in other service applications.
8. The capability to describe the various QoS requirements should be supported by the model. There can be two kinds of QoS configuration.
 - * The AP based QoS: describes the QoS requirements on the access point. For example, the CAR (committed access rate) definition on the inbound or outbound ports.
 - * The flow based QoS: describes the QoS requirements on a flow. This enables the fine grained QoS control with the capability of identifying the flow.

5. Design of the Data Model

The PE-based VPN service is modeled with a recursive pattern. The VPN service deployed within each AS is modeled as a Segmental VPN object including the VPN description information within this AS and the Access Points (AP) that are used to connect to the peered device or AS. As an end to end VPN service within the ISP domain, it's then modeled as a Composite VPN object with the overall VPN information and the APs that are used to connect to the peered customer sites.

5.1. Composite VPN

The composedVPN top container contains the business type, VPN basic information, a list of segment VPN information, a list of access point information and and some overall information. The id MUST be unique for the reference of this composed VPN service. The tenantID associates this VPN service to a dedicated customer. The businessTypeID can associate composed VPN with a business template. The vpnBasicInfo object here includes basic information for this Composite VPN service. I.e., all the properties (e.g., topology, serviceType) in this object describe the overview that the customer want, no matter with any segment VPN information. The

accessPointList describes a list of APs that are used to connect to the peered customer sites. A Composite VPN includes one or more Segmental VPN.

```

+--rw composedVpns* [id]
  +--rw id                yang:uuid
  +--rw name?             string
  +--rw description?      string
  +--rw tenantId?         yang:uuid
  +--rw businessTypeID?   yang:uuid
  +--rw vpnBasicInfo
  |   ...
  +--ro operStatus?       CommonTypes:OperStatus
  +--ro syncStatus?       CommonTypes:SyncStatus
  +--rw startTime?        yang:date-and-time
  +--rw segVpnList* [index]
  |   ...
  +--rw accessPointList* [id]
  |   ...

```

5.2. Access Point

The accessPointList models a list of APs including the access or attachment information for the remote peer. AP is provided by the PE either in the Composite VPN view or in the Segmental VPN view. The AP uses working layer and corresponding layer information to describe the different configurations in Composite VPN level and the Segmental VPN level.

```

+--rw accessPointList* [id]
  +--rw id                yang:uuid
  +--rw name?             string
  +--rw description?      string
  +--rw neID?             yang:uuid
  +--rw containingMainTPID? yang:uuid
  +--rw tpBasicInfo
  |   +--rw edgePointRole? CommonTypes:EdgePointRole
  |   +--rw topologyRole?  CommonTypes:TopoNodeRole
  |   +--rw Type?          CommonTypes:TpType
  |   +--rw workingLayer?   CommonTypes:LayerRate
  |   +--rw typeSpecList* [layerRate]
  |   |   +--rw layerRate      CommonTypes:LayerRate
  |   |   +--rw (specValue)?
  |   |   |   +--:(LR_Ethernet)
  |   |   |   |   +--rw ethernetSpec
  |   |   |   |   |   +--rw accessType?    CommonTypes:EthernetEncapType
  |   |   |   |   |   +--rw (accessVlanValue)?
  |   |   |   |   |   |   +--:(QinQVlan)

```

```

    +--rw qinqVlan
    |   +--rw cvlanList*   uint64
    |   +--rw svlanList?  uint64
    +--:(DOT1Q)
    |   +--rw dot1q
    |   |   +--rw dot1qVlanList*   uint64
    +--rw vlanAction?   CommonTypes:EthernetAction
    +--rw actionValue?  string
+--:(LR_IP)
|   +--rw ipSpec
|   |   +--rw masterIp?   string
|   |   +--rw mtu?        uint64
+--:(LR_Vxlan)
|   +--rw vxlanSpec
|   |   +--rw vni?        uint32
|   |   +--rw vtepIP?     string
+--rw adminStatus?     CommonTypes:AdminStatus
+--rw tpQosNode
|   +--rw qosConfigType?   QosConfigType
|   +--rw qosDetailType?   QosDetailType
|   +--rw inTpCar* [index]
|   |   +--rw index        uint32
|   |   +--rw dataKind?    dataKind
|   |   +--rw actionType?  ActionType
|   |   +--rw action?      string
|   +--rw outTpCar* [index]
|   |   +--rw index        uint32
|   |   +--rw dataKind?    dataKind
|   |   +--rw actionType?  ActionType
|   |   +--rw action?      string
|   +--rw inQosProfileId?  yang:uuid
|   +--rw outQosProfileId? yang:uuid
+--rw flowServices* [index]
|   +--rw qosConfigType?   QosConfigType
|   +--rw flowQosTemplateID? yang:uuid
|   +--rw flowServices* [flowClassifierId]
|   |   +--rw flowClassifierId  yang:uuid
|   |   +--rw flowBehaviors* [index]
|   |   |   +--rw index        uint32
|   |   |   +--rw dataKind?    dataKind
|   |   |   +--rw actionType?  ActionType
|   |   |   +--rw action?      string
+--rw addtionalInfo* [name]
|   +--rw name      string
|   +--rw value?    string
+--rw peerCeTp
|   +--rw ceID?      yang:uuid
|   +--rw ceDirectNeID? yang:uuid

```



```

|   +---rw ceDirectTPID?   yang:uuid
|   +---rw ceIfmasterIp?   string
|   +---rw location?       string
+---rw routeProtocolSpec* [type]
|   +---rw type             CommonTypes:RouteProtocolType
|   +---rw (para)?
|       +---:(staticRouting)
|           +---rw staticRouteItems* [index]
|               +---rw index             uint32
|               +---rw destinationCidr?   string
|               +---rw egressTP?         yang:uuid
|               +---rw routePreference?   string
|               +---rw nextHopIp?         string
|       +---:(bgp)
|           +---rw bgpProtocols* [index]
|               +---rw index             uint32
|               +---rw peerAsNumber?     uint64
|               +---rw bgpMaxPrefix?     int32
|               +---rw bgpMaxPrefixAlarm? uint32
|               +---rw peerIp?           string
+---ro operStatus?         CommonTypes:OperStatus

```

5.2.1. Termination Point Basic Information

The `tpBasicInfo` describes the basic information that is used to express the design intent of the VPN from the Termination Point (TP) of view. That means the information described here is relative static, no matter which exact peer TP is going to connect.

The `typeSpecList` describes the layered information on the TP. It describes in detail on the ethernet layer information, or the IP layer and VxLan information if any higher layer protocol is enabled.

5.2.2. QoS

This model support two kinds of QoS description as described in the section 4:

- o TP based QoS: describes the QoS requirements on a termination point. For example, the CAR (committed access rate) definition on the inbound or outbound ports.
- o Flow based QoS: describes the QoS requirements on a flow. This enables the fine grained QoS control with the capability of identifying the flow.

5.2.3. Routing Protocol

The routeProtocolSpec object describes information of the routing protocol that is used to exchange the routing information with the remote peer. This object is extensible with any possible routing protocols. The BGP and static routing listed are examples to show how these two widely used solutions are described.

5.3. Segmental VPN

The segVpnList describes a list of Segmental VPN information which is only from the segment point of view. I.e., the description here takes care about how the Segmental VPN looks like and how it can communicate with peered devices outside this Segmental VPN. The segment information is composed of basic VPN information and a list of APs. The set of APs in the description are interfaces that customer sites or other segment VPNs can attach. In different scenarios, each Segmental VPN could be a layer 2 VPN, or layer 3 VPN, or even a termination point.

```
+--rw segVpnList* [index]
  +--rw index      uint32
  +--rw vpnType?   string
  +--rw vpnRole?   VPNTypes:ProtectionRole
  +--rw vpnInfo
    +--rw (vpnType)?
      +--:(wanVpn)
        +--rw vpn
          +--rw id?                yang:uuid
          +--rw name?              string
          +--rw description?       string
          +--rw vpnBasicInfo
            +--rw topology?       CommonTypes:Topology
            +--rw serviceType?    VPNTypes:ServiceType
            +--rw technology?     VPNTypes:VPNTunnelType
            +--rw adminStatus?    CommonTypes:AdminStatus
          +--ro operStatus?        CommonTypes:OperStatus
          +--ro syncStatus?        CommonTypes:SyncStatus
          +--rw accessPointList* [id]
            ...
```

6. YANG Module

```
<CODE BEGINS> file "ietf-nvo-vpn.yang"
module ietf-nvo-vpn {
  namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-vpn" ;
  prefix VPN ;
```

```
import ietf-yang-types {
    prefix yang;
}

import ietf-nvo-common-types {
    prefix CommonTypes;
}
import ietf-nvo-tp {
    prefix TP;
}

import ietf-nvo-vpn-types {
    prefix VPNTypes;
}

organization "";
contact "";
description "ietf-nvo-vpn";
revision 2016-10-24 {
    reference "draft-chen-opsawg-composite-vpn-dm-00";
}

container nvoVPNMgr{
    description "";
    list composedVPNs {
        key "id";
        description "";
        uses VPN:ComposedVPN;
    }
}

grouping ComposedVPN {
    description "ComposedVPN Grouping.";

    leaf id {
        type yang:uuid ;
        description "UUID-STR for service ." ;
    }

    leaf name {
        type string {length "0..200";}
        description "Human-readable name for the service." ;
    }
    leaf description {
        type string {length "0..200";}
        description "Detailed specification for the servcie." ;
    }
    leaf tenantId {
```

```
        type yang:uuid;
        description "UUID-STR for tenant." ;
    }
    leaf businessTypeID {
        type yang:uuid;
        description "business Type Name" ;
    }

    container vpnBasicInfo {
        description "VPN BASIC INFO";
        uses VPNTypes:VPNBasicInfo;
    }

    leaf operStatus {
        type CommonTypes:OperStatus;
        config false;
        description "Operational status." ;
    }

    leaf syncStatus {
        type CommonTypes:SyncStatus;
        config false;
        description "Sync status." ;
    }

    leaf startTime {
        type yang:date-and-time;
        description "Service lifecycle: request for service start
        time." ;
    }

    list segVpnList      {
        key "index";
        description "SegVpn list ";
        uses VPN:SegmentVPN;
    }

    list accessPointList {
        key "id";
        description "TP list of the access links which associated
        with CE and PE";
        uses TP:Tp;
    }
}

grouping SegmentVPN {
    description "SegmentVPN Grouping.";
```

```
    leaf index {
        type uint32;
        description "index of segment VPN in a composed VPN.";
    }

    leaf vpnType {
        type string {length "0..30";}
        description "value: nop/wanVpn";
    }

    leaf vpnRole {
        type VPNTypes:ProtectionRole;
        description "value: nop|vpn";
    }

    container vpnInfo {
        description "vpn information";
        choice vpnType {
            description "vpn type.";
            case wanVpn {
                container vpn {
                    description "vpn.";
                    uses VPN:VPN;
                }
            }
        }
    }
}

grouping VPN {
    description "VPN Grouping.";

    leaf id {
        type yang:uuid ;
        description "UUID-STR for VPN." ;
    }
    leaf name {
        type string {length "0..200";}
        description "Human-readable name for the service." ;
    }

    leaf description {
        type string {length "0..200";}
        description "Detailed specification for the servcie." ;
    }

    container vpnBasicInfo {
        description "vpn basic info" ;
    }
}
```

```
        uses VPNTypes:VPNBasicInfo;
    }

    leaf operStatus {
        type CommonTypes:OperStatus;
        config false;
        description "Operational status." ;
    }

    leaf syncStatus {
        type CommonTypes:SyncStatus;
        config false;
        description "Sync status." ;
    }

    list accessPointList {
        key "id";
        description "TP list of the access links which associated
        with CE and PE";
        uses TP:Tp;
    }
}
<CODE ENDS>

<CODE BEGINS> file "ietf-nvo-common-types.yang"
module ietf-nvo-common-types {
    namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-common-types" ;
    prefix CommonTypes ;
    import ietf-yang-types {
        prefix yang;
    }

    organization "";
    contact "";
    description "ietf-nvo-common-types";
    revision 2016-10-24 {
        reference "draft-chen-opsawg-composite-vpn-dm-00";
    }

    typedef AdminStatus {
        type enumeration {
            enum active {
                description "Active status";
            }
            enum inactive {
                description "Inactive status";
            }
        }
    }
}
```

```
        enum partial {
            description "Partial status";
        }
    }
    description "AdminStatus";
}

typedef OperStatus {
    type enumeration {
        enum up {
            description "Up status";
        }
        enum down {
            description "Down status";
        }
        enum degrade {
            description "Degrade status";
        }
    }
    description "OperStatus";
}

typedef SyncStatus {
    type enumeration {
        enum SYNC {
            description "Sync status";
        }
        enum OUT-SYNC {
            description "Out sync status";
        }
    }
    description "SyncStatus";
}

typedef Topology {
    type enumeration {
        enum full-mesh {
            description "full-mesh";
        }
        enum point_to_multipoint {
            description "point_to_multipoint";
        }
        enum point_to_point {
            description "point_to_point";
        }
        enum complex {
            description "complex";
        }
    }
}
```

```
    }
    description "Topology";
}

typedef Technology {
    type enumeration {
        enum mpls {
            description "mpls";
        }
        enum rosen_multivpn {
            description "rosen_multivpn";
        }
        enum ng_multivpn {
            description "ng_multivpn";
        }
        enum vxlan_overlay_l3vpn {
            description "vxlan_overlay_l3vpn";
        }
        enum eth_oversdh {
            description "eth_oversdh";
        }
    }
    description "Technology";
}

typedef TopoNodeRole {
    type enumeration {
        enum other {
            description "other";
        }
        enum hub {
            description "hub";
        }
        enum spoke {
            description "spoke";
        }
    }
    description "TopoNodeRole";
}

typedef TpType{
    type enumeration {
        enum nop {
            description "nop";
        }
        enum PTP {
            description "PTP";
        }
    }
}
```



```
        enum CTP {
            description "CTP";
        }
        enum TRUNK {
            description "TRUNK";
        }
        enum LoopBack {
            description "LoopBack";
        }
        enum TPPool {
            description "TPPool";
        }
    }
    description "TpType";
}

typedef LayerRate{
    type enumeration {
        enum LR_UNKNOW {
            description "LR_UNKNOW";
        }
        enum LR_IP {
            description "LR_IP";
        }
        enum LR_ETHERNET {
            description "LR_ETHERNET";
        }
        enum LR_VXLAN {
            description "LR_VXLAN";
        }
    }
    description "LayerRate";
}

typedef EthernetEncapType {
    type enumeration {
        enum DEFAULT {
            description "DEFAULT";
        }
        enum DOT1Q {
            description "DOT1Q";
        }
        enum QINQ {
            description "QINQ";
        }
        enum UNTAG {
            description "UNTAG";
        }
    }
}
```

```
    }
    description "EthernetEncapType";
}
typedef EthernetAction {
    type enumeration {
        enum nop {
            description "nop";
        }
        enum UNTAG {
            description "UNTAG";
        }
        enum STACKING {
            description "STACKING";
        }
    }
    description "EthernetAction";
}

typedef RouteProtocolType {
    type enumeration {
        enum staticRouting {
            description "staticRouting";
        }
        enum bgp {
            description "bgp";
        }
        enum rip {
            description "rip";
        }
        enum ospf {
            description "ospf";
        }
        enum isis {
            description "isis";
        }
    }
    description "RouteProtocolType";
}

typedef QosPriorityType {
    type enumeration {
        enum nop {
            description "nop";
        }
        enum 802dot1p {
            description "802dot1p";
        }
    }
}
```

```
        enum dscp {
            description "dscp";
        }
        enum mplsExp {
            description "mplsExp";
        }
        enum cos {
            description "cos";
        }
        enum ipPrecedence {
            description "ipPrecedence";
        }
    }
    description "QosPriorityType";
}

typedef ColorType {
    type enumeration {
        enum nop {
            description "nop";
        }
        enum green {
            description "green";
        }
        enum yellow {
            description "yellow";
        }
        enum red {
            description "red";
        }
    }
    description "ColorType";
}

grouping nvStringList{
    description "nvStringList Grouping.";

    list nvstringList {
        key "name";
        uses CommonTypes:nvstring;
        description "nvStringList";
    }
}

grouping nvstring {
    description "nvstring Grouping.";
```

```
    leaf name {
      type string;
      description "string name ";
    }
    leaf value {
      type string;
      description "string value";
    }
  }

  typedef FlowClassifierType {
    type enumeration {
      enum nop {
        description "nop";
      }
      enum 802dot1p {
        description "802dot1p";
      }
      enum dscp {
        description "dscp";
      }
      enum cos {
        description "cos";
      }
      enum mpls-exp {
        description "mpls-exp";
      }
      enum sourceIP {
        description "sourceIP";
      }
      enum destinationIP {
        description "destinationIP";
      }
    }
    description "FlowClassifierType";
  }

  grouping ObjectIdentifiers {
    description "ObjectIdentifiers Grouping.";

    list ObjectIdentifiers {
      key "obejctId";
      uses CommonTypes:ObjectIdentifier;
      description "ObjectIdentifiers";
    }
  }

  grouping ObjectIdentifier {
```

```
description "ObjectIdentifier Grouping.";

leaf objectType {
    type CommonTypes:ObjectType;
    description "objectType";
}

leaf obejctId {
    type yang:uuid;
    description "obejctId";
}

leaf roleLable {
    type string {length "0..200";}
    description "here is the route role";
}
}

typedef ObjectType {
    type enumeration {
        enum nop {
            description "nop";
        }
        enum SEG-VPN {
            description "SEG-VPN";
        }
        enum TP {
            description "TP";
        }
        enum TPL {
            description "TPL";
        }
        enum NE {
            description "NE";
        }
        enum BUSINESSTYPE {
            description "BUSINESSTYPE";
        }
        enum COMPOSED-VPN {
            description "COMPOSED-VPN";
        }
        enum SUBNETWORK {
            description "SUBNETWORK";
        }
    }
    description "ObjectType";
}
```

```
typedef EdgePointRole {
    type enumeration {
        enum nop {
            description "nop";
        }
        enum PE {
            description "PE";
        }
        enum P {
            description "P";
        }
        enum UNI {
            description "UNI";
        }
        enum NNI {
            description "NNI";
        }
        enum AsbTP {
            description "AsbTP";
        }
    }
    description "EdgePointRole";
}

typedef DomainRole {
    type enumeration {
        enum nop {
            description "nop";
        }
        enum external {
            description "external";
        }
        enum internal {
            description "internal";
        }
        enum asb {
            description "asb";
        }
    }
    description "DomainRole";
}

grouping CommandResult {
    description "this is the common result which is send back by
server by RPC response";
    leaf resultCode {
        type int32;
        description "resultCode";
    }
}
```

```
    }
    leaf-list successResourceList {
        type yang:uuid;
        description "successResourceList";
    }
    list failedResourceList {
        uses CommonTypes:FailedNode;
        description "failedResourceList";
    }
    leaf errorReason {
        type string {length "0..200";}
        description "errorReason";
    }
}

grouping FailedNode {
    description "fail reason for each object.";
    leaf resourceId {
        type yang:uuid;
        description "failed object.";
    }
    leaf errorReason {
        type string {length "0..200";}
        description "errorReason";
    }
}

grouping SLA {
    description "SLA";
    leaf latency {
        type uint32;
        description "delay, unit:ms.";
    }
}

typedef ConnectionDirection {
    type enumeration {
        enum CD-UNI {
            description "CD-UNI";
        }
        enum CD-BI {
            description "CD-BI";
        }
    }
    description "ConnectionDirection";
}
```

```
typedef ObjectDirection {
    type enumeration {
        enum IN {
            description "IN";
        }
        enum OUT {
            description "OUT";
        }
        enum BI-DIRECTION {
            description "BI-DIRECTION";
        }
    }
    description "ObjectDirection";
}

typedef DiversityType {
    type enumeration {
        enum NOP {
            description "NOP";
        }
        enum PE-DIFF {
            description "PE-DIFF";
        }
        enum MAIN-TP-DIFF {
            description "MAIN-TP-DIFF";
        }
    }
    description "DiversityType";
}

}
<CODE ENDS>

<CODE BEGINS> file "ietf-nvo-qos-types.yang"
module ietf-nvo-qos-types {
    namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-qos-types";
    prefix QosTypes;

    import ietf-yang-types { prefix yang; }

    organization "";
    contact "";
    description "ietf-nvo-qos-types";
    revision 2016-10-24 {
        reference "draft-chen-opsawg-composite-vpn-dm-00";
    }

    /*****
    * Type definitions
    *****/
}
```



```

*****/
typedef qosPriorityType {
    type enumeration {
        enum nop{
            description "nop";
        }
        enum 802dot1p{
            description "802dot1p";
        }
        enum dscp{
            description "dscp";
        }
        enum mplsExp{
            description "mplsExp";
        }
        enum cos{
            description "cos";
        }
        enum ipPrecedence{
            description "ipPrecedence";
        }
    }
    description "qosPriorityType";
}

typedef classifierDetailType {
    type enumeration {
        enum nop{
            description "nop";
        }
        enum ipPrefixList{
            description "ipPrefixList";
        }
    }
    description "classifierDetailType";
}

typedef ActionType {
    type enumeration {
        enum nop{
            description "nop";
        }
        enum bandwidth{
            description "bandwidth";
        }
        enum pass{
            description "pass";
        }
    }
}
```

```
        enum discard{
            description "discard";
        }
        enum remark{
            description "remark";
        }
        enum redirect{
            description "redirect";
        }
        enum recolor{
            description "recolor";
        }
        enum addRt{
            description "addRt";
        }
    }
    description "ActionType";
}

typedef QosConfigType {
    type enumeration {
        enum nop{
            description "nop";
        }
        enum template{
            description "template";
        }
        enum agile{
            description "agile";
        }
    }
    description "QosConfigType";
}

typedef flowClassifierType {
    type enumeration {
        enum nop{
            description "nop";
        }
        enum aluFlowClassifier{
            description "aluFlowClassifier";
        }
    }
    description "flowClassifierType";
}

typedef QosDetailType {
    type enumeration {
```

```
        enum nop{
            description "nop";
        }
        enum car{
            description "car";
        }
        enum qosProfile{
            description "qosProfile";
        }
        enum diffServDomain{
            description "diffServDomain";
        }
        enum diffServ{
            description "diffServ";
        }
        enum aluDiffServ{
            description "aluDiffServ";
        }
    }
    description "QosDetailType";
}

typedef ClassifierType {
    type enumeration {
        enum 802dot1p{
            description "802dot1p";
        }
        enum dscp{
            description "dscp";
        }
        enum cos{
            description "cos";
        }
        enum mpls-exp{
            description "mpls-exp";
        }
        enum sourceIP{
            description "sourceIP";
        }
        enum destinationIP{
            description "destinationIP";
        }
    }
    description "ClassifierType";
}

typedef OrchPermitType {
    type enumeration {
```

```
        enum readUse{
            description "readUse";
        }
        enum crud{
            description "crud";
        }
    }
    description "OrchPermitType";
}

typedef ruleType {
    type enumeration {
        enum 802dot1p{
            description "802dot1p";
        }
        enum dscp{
            description "dscp";
        }
        enum cos{
            description "cos";
        }
        enum mpls_exp{
            description "mpls_exp";
        }
        enum source_ip{
            description "source_ip";
        }
        enum destination_ip{
            description "destination_ip";
        }
    }
    description "ruleType";
}

typedef MatchModeType {
    type enumeration {
        enum nop{
            description "nop";
        }
        enum match{
            description "match";
        }
        enum unmatch{
            description "unmatch";
        }
    }
    description "MatchModeType";
}
```

```
typedef qosBehaviorType {
    type enumeration {
        enum qosCarBehavior{
            description "qosCarBehavior";
        }
        enum qosServiceChainBehavior{
            description "qosServiceChainBehavior";
        }
    }
    description "qosBehaviorType";
}

typedef qosBehaviorDirectionType {
    type enumeration {
        enum upstream{
            description "upstream";
        }
        enum downstream{
            description "downstream";
        }
        enum bidirectional{
            description "bidirectional";
        }
    }
    description "qosBehaviorDirectionType";
}

typedef dataKind {
    type enumeration {
        enum green{
            description "green";
        }
        enum yellow{
            description "yellow";
        }
        enum red{
            description "red";
        }
        enum all{
            description "all";
        }
    }
    description "dataKind";
}

typedef profileType {
    type enumeration {
        enum profile{
```

```
        description "profile";
    }
}
description "profileType";
}

typedef ruleOperatorType {
    type enumeration {
        enum and{
            description "and";
        }
        enum or{
            description "or";
        }
    }
    description "ruleOperatorType";
}

/*****
* Groupings
*****/
grouping TPQosNode {
    description "TPQosNode Grouping.";

    leaf qosConfigType {
        type QosConfigType;
        description "qosConfigType";
    }
    leaf qosDetailType {
        type QosDetailType;
        description "qosDetailType";
    }
    list inTpCar {
        key index;
        uses FlowBehavior;
        description "inTpCar";
    }
    list outTpCar {
        key index;
        uses FlowBehavior;
        description "outTpCar";
    }
    leaf inQosProfileId {
        type yang:uuid;
        description "inQosProfileId";
    }
    leaf outQosProfileId {
        type yang:uuid;
    }
}
```

```
        description "outQosProfileId";
    }
}

grouping FlowAndBehavior {
    description "FlowAndBehavior Grouping.";

    leaf flowClassifierId {
        type yang:uuid;
        description "flowClassifierId";
    }
    list flowBehaviors {
        key index;
        uses FlowBehavior;
        description "flowBehaviors";
    }
}

grouping FlowBehavior {
    description "FlowAndBehavior Grouping.";

    leaf index {
        type uint32;
        description "index";
    }
    leaf dataKind {
        type dataKind;
        description "dataKind";
    }
    leaf actionType {
        type ActionType;
        description "actionType";
    }
    leaf action {
        type string;
        description "action";
    }
}

grouping FlowClassifierRule {
    description "FlowClassifierRule Grouping.";

    leaf index {
        type uint32;
        description "index";
    }
    leaf matchMode {
        type MatchModeType;
    }
}
```

```
        description "matchMode";
    }
    leaf type {
        type ClassifierType;
        description "type";
    }
    leaf-list flowClassifierValue {
        type string;
        description "flowClassifierValue";
    }
    leaf appendix {
        type string;
        description "appendix";
    }
}

grouping BandWidthNode {
    description "BandWidthNode Grouping.";

    leaf cir {
        type uint32;
        description "cir";
    }
    leaf pir {
        type uint32;
        description "pir";
    }
    leaf cbs {
        type uint32;
        description "cbs";
    }
    leaf pbs {
        type uint32;
        description "pbs";
    }
}

grouping FlowServices {
    description "FlowServices Grouping.";

    leaf qosConfigType {
        type QosConfigType;
        description "qosConfigType";
    }
    leaf flowQosTemplateID {
        type yang:uuid;
        description "flowQosTemplateID";
    }
}
```



```
        leaf qosDetailType {
            type QosTypes:QosDetailType;
            description "qosDetailType";
        }
        leaf inFlowQosTemplateID {
            type yang:uuid;
            description "inFlowQosTemplateID";
        }
        leaf outFlowQosTemplateID {
            type yang:uuid;
            description "outFlowQosTemplateID";
        }
    }

    list flowServices {
        key flowClassifierId;
        description "default in flow and behaviors";
        uses FlowAndBehavior;
    }
}
<CODE ENDS>

<CODE BEGINS> file "ietf-nvo-tp.yang"
module ietf-nvo-tp {
    namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-tp";
    prefix TP;

    import ietf-yang-types {
        prefix yang;
    }

    import ietf-nvo-common-types {
        prefix CommonTypes;
    }

    import ietf-nvo-vpn-routeprotocol {
        prefix RouteProtocol;
    }

    import ietf-nvo-tp-types {
        prefix TPTypes;
    }

    organization "";
    contact "";
    description "This module contains a collection of YANG definitions
    for tp";
    revision 2016-10-24 {
```

```
        reference "draft-chen-opsawg-composite-vpn-dm-00";
    }

    container nvoTPMgr{
        description "nvo tp management";
        list tps {
            key "id";
            uses TP:Tp;
            description "tp retrieve functions";
        }
    }

    grouping Tp {
        description "model of TP";
        leaf id {
            type yang:uuid;
            description "yang:uuid-str for TP";
        }
        leaf name {
            type string {length "0..200";}
            description "Must abbey to name rule defined in system.
            Example FE0/0/1, GE1/2/1.1, Eth-Trunk1.1, etc";
        }
        leaf description {
            type string {length "0..200";}
            description "description for this tp.";
        }

        leaf neID {
            type yang:uuid;
            description "yang:uuid-str for NE ";
        }

        leaf containingMainTPID {
            type yang:uuid;
            description "uuid-str for main interface";
        }
        container tpBasicInfo {
            description "Tp non-instance basic info";
            uses TPTypes:TPBasicInfo;
        }
        container peerCeTp {
            description "CE TP Information";
            uses TPTypes:CeTp;
        }

        list routeProtocolSpec {
            key "type";
```

```
        description "route protocol spec";
        uses RouteProtocol:RouteProtocolSpec;
    }

    leaf operStatus {
        type CommonTypes:OperStatus;
        config false;
        description "Operational status." ;
    }
}

grouping TpInventory {
    description "inventory model of TP";
    leaf tpID {
        type yang:uuid;
        description "uuid of tp";
    }
    leaf detailType {
        type string {length "0..100";}
        description "tp detail type. reported by controller";
    }
    leaf maxBandWidth {
        type uint64;
        description "max bandwidth";
    }
    leaf-list potentialLayers {
        type CommonTypes:LayerRate;
        description "capability of tp to create VPN, reported by
        controller";
    }
}
}
}
<CODE ENDS>

<CODE BEGINS> file "ietf-nvo-tp-types.yang"
module ietf-nvo-tp-types {
    namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-tp-types";
    prefix TPTypes;

    import ietf-nvo-common-types {prefix CommonTypes;}
    import ietf-yang-types { prefix yang; }
    import ietf-nvo-qos-types { prefix QosTypes;}

    organization "";
    contact "";
    description "ietf-nvo-tp-types";
    revision 2016-10-24 {
        reference "draft-chen-opsawg-composite-vpn-dm-00";
    }
}
```

```
}

//LayerRate parameters.
grouping PwSpec {
    description "PwSpec Grouping.";

    leaf controlWord {
        type boolean;
        default false;
        description "controlWord";
    }
    leaf pwVlanAction {
        type TPTypes:PWTagMode;
        description "pwVlanAction";
    }
}

grouping IpSpec {
    description "IpSpec Grouping.";

    leaf masterIp {
        type string;
        description "master IP address";
    }
    leaf mtu {
        type uint64;
        description "mtu for ip layer,scope:46~9600";
    }
}

grouping EthernetSpec {
    description "EthernetSpec Grouping.";

    leaf accessType {
        type CommonTypes:EthernetEncapType;
        description "access frame type";
    }

    choice accessVlanValue {
        description "accessVlanValue";
        case QinQVlan {
            container qinqVlan {
                description "qinqVlan";
                uses TPTypes:QinQVlan;
            }
        }
    }
}
```

```
        case DOT1Q {
            container dot1q {
                description "dot1q";
                uses TPTypes:Dot1QVlan;
            }
        }
    }

    leaf vlanAction {
        type CommonTypes:EthernetAction;
        description "Frame type that can be accepted. not needed
        now";
    }

    leaf actionValue{
        type string{length "0..100";}
        description "action value";
    }
}

grouping QinQVlan {
    description "QinQVlan Grouping.";

    leaf-list cvlanList {
        type uint64;
        description "cvlanList";
    }
    leaf svlanList {
        type uint64;
        description "svlanList";
    }
}

grouping Dot1QVlan {
    description "Dot1QVlan Grouping.";

    leaf-list dot1qVlanList {
        type uint64;
        description "dot1qVlanList";
    }
}

grouping VxlanSpec {
    description "VxlanSpec Grouping.";

    leaf vni {
        type uint32;
```

```
        description "vni";
    }

    leaf vtepIP {
        type string;
        description "vtep ip";
    }
}

//CE spec
grouping CeTp {
    description "CeTp Grouping.";

    leaf ceID {
        type yang:uuid;
        description "Site router ID";
    }

    leaf ceDirectNeID {
        type yang:uuid;
        description "direction connected NE ID, only valid in
asbr ";
    }

    leaf ceDirectTPID {
        type yang:uuid;
        description "ce Direct TP id, only valid in asbr";
    }

    leaf ceIfmasterIp {
        type string;
        description "ceIfmasterIp";
    }

    leaf location {
        type string {length "0..400";}
        description "CE device location ";
    }
}

//TPBasicInfo
grouping TPBasicInfo {
    description "TPBasicInfo Grouping.";

    leaf edgePointRole {
        type CommonTypes:EdgePointRole;
        description "edge role for TP, for example:UNI/NNI ";
    }
}
```

```
    leaf topologyRole {
        type CommonTypes:TopoNodeRole;
        description "hub/spoke role, etc";
    }

    leaf Type
    {
        type CommonTypes:TpType;
        description "Type";
    }

    leaf workingLayer {
        type CommonTypes:LayerRate;
        description "working layer";
    }

    list typeSpecList {
        key "layerRate";
        uses TPTypes:TpTypeSpec;
        description "typeSpecList";
    }

    leaf adminStatus {
        type CommonTypes:AdminStatus;
        description "administrative status.";
    }

    container tpQosNode {
        description "tpQosNode";
        uses QosTypes:TPQosNode;
    }

    container flowServices{
        description "flow services in one TP";
        uses QosTypes:FlowServices;
    }

    list additionalInfo {
        key "name";
        uses CommonTypes:nvstring;
        description "additionalInfo";
    }
}

//TpTypeSpec
grouping TpTypeSpec{
    description "TpTypeSpec Grouping.";
```

```
    leaf layerRate {
        type CommonTypes:LayerRate;
        description "layerRate";
    }

    choice specValue {
        description "specValue";
        case LR_Ethernet {
            container ethernetSpec {
                description "ethernetSpec";
                uses TPTypes:EthernetSpec;
            }
        }
        case LR_IP {
            container ipSpec {
                description "ipSpec";
                uses TPTypes:IpSpec;
            }
        }
        case LR_Vxlan {
            container vxlanSpec {
                description "vxlanSpec";
                uses TPTypes:VxlanSpec;
            }
        }
    }
}

//-----//
//typedef
//-----//
typedef PWTagMode {
    type enumeration {
        enum RAW{
            description "RAW";
        }
        enum TAGGED{
            description "TAGGED";
        }
    }
    description "PWTagMode";
}

}
<CODE ENDS>

<CODE BEGINS> file "ietf-nvo-vpn-routeprotocol.yang"
module ietf-nvo-vpn-routeprotocol {
    namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-vpn-routeprotocol";
```



```
prefix RouteProtocol;
import ietf-yang-types { prefix yang; }
import ietf-nvo-common-types {
    prefix CommonTypes;
}

organization "";
contact "";
description "ietf-nvo-vpn-routeprotocol";
revision 2016-10-24 {
    reference "draft-chen-opsawg-composite-vpn-dm-00";
}

grouping RouteProtocolSpec {
    description "RouteProtocolSpec Grouping.";

    leaf type {
        type CommonTypes:RouteProtocolType;
        description "Protocol type" ;
    }
    choice para {
        description "para" ;
        case staticRouting {
            list staticRouteItems {
                key "index";
                uses RouteProtocol:StaticRouteItem;
                description "staticRouteItems" ;
            }
        }
        case bgp {
            list bgpProtocols {
                key "index";
                uses RouteProtocol:BGPProtocolItem;
                description "bgpProtocols" ;
            }
        }
    }
}

grouping StaticRouteItem {
    description "StaticRouteItem Grouping.";

    leaf index {
        type uint32;
        description "static item index";
    }
    leaf destinationCidr {
```

```
        type string;
        description "destination ip cidr. ";
    }
    leaf egressTP {
        type yang:uuid;
        description "egress tp";
    }
    leaf routePreference {
        type string;
        description "route priority. Ordinary, work route have
higher priority.";
    }
    leaf nextHopIp {
        type string {length "0..200";}
        description "nextHopIp";
    }
}

grouping BGPProtocolItem {
    description "BGPProtocolItem Grouping.";

    leaf index {
        type uint32;
        description "index of BGP protocol item";
    }
    leaf peerAsNumber {
        type uint64;
        description "";
    }
    leaf bgpMaxPrefix {
        type int32;
        description "";
    }
    leaf bgpMaxPrefixAlarm {
        type uint32;
        description "alarm threshold of BGP rout";
    }
    leaf peerIp {
        type string;
        description "peerIp";
    }
}
}
}
<CODE ENDS>

<CODE BEGINS> file "ietf-nvo-vpn-types.yang"
module ietf-nvo-vpn-types {
    namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-vpn-types" ;
```

```
prefix VPNTypes ;

import ietf-nvo-common-types {
    prefix CommonTypes;
}

organization "";
contact "";
description "ietf-nvo-vpn-types";
revision 2016-10-24 {
    reference "draft-chen-opsawg-composite-vpn-dm-00";
}

typedef ProtectionRole {
    type enumeration {
        enum NOP{
            description "NOP";
        }
        enum MAIN{
            description "MAIN";
        }
    }
    description "ProtectionRole";
}

grouping VPNBasicInfo {
    description "VPNBasicInfo Grouping.";

    leaf topology {
        type CommonTypes:Topology;
        description "current support for full-mesh and
        point_to_multipoint(hub-spoke), others is reserved for
        future extensions." ;
    }

    leaf serviceType {
        type VPNTypes:ServiceType;
        description "current support for mpls l3vpn/vxlan/L2VPN
        overlay, others is reserved for future extensions." ;
    }

    leaf technology {
        type VPNTypes:VPNTunnelType;
        description "mpls|vxlan overlay l3vpn|eth over sdh|nop";
    }

    leaf adminStatus {
        type CommonTypes:AdminStatus;
    }
}
```

```
        description "administrative status." ;
    }
}

typedef VPNTunnelType {
    type enumeration {
        enum NOP{
            description "NOP";
        }
        enum MPLS{
            description "MPLS";
        }
        enum MPLS-TP{
            description "MPLS-TP";
        }
    }
    description "VPNTunnelType";
}

typedef ServiceType {
    type enumeration {
        enum l3vpn {
            description "l3vpn" ;
        }
        enum l2vpn {
            description "l2vpn" ;
        }
    }
    description "ServiceType";
}
}
<CODE ENDS>
```

7. IANA Considerations

TBD

8. Security Considerations

TBD

9. Acknowledgements

TBD

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4110] Callon, R. and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)", RFC 4110, DOI 10.17487/RFC4110, July 2005, <<http://www.rfc-editor.org/info/rfc4110>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<http://www.rfc-editor.org/info/rfc7950>>.

10.2. Informative References

- [I-D.ietf-l3sm-l3vpn-service-model]
Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN service delivery", draft-ietf-l3sm-l3vpn-service-model-18 (work in progress), October 2016.
- [I-D.wu-opsawg-service-model-explained]
Wu, Q., LIU, S., and A. Farrel, "Service Models Explained", draft-wu-opsawg-service-model-explained-03 (work in progress), September 2016.

Authors' Addresses

Rui Chen
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
China

Email: chenrui@huawei.com

Liya Zhang
Huawei Technologies
Wuhan
China

Email: zhangliya@huawei.com

Hui Deng
Huawei Technologies
Beijing
China

Email: denghui02@hotmail.com

Liang Geng
China Mobile
No.32 Xuanwumen West Street
Beijing 100053
China

Email: liang.geng@hotmail.com

Chongfeng Xie
China Telecom
Beijing
China

Email: xiechf@ctbri.com.cn

Network Working Group
Internet-Draft
Intended status: Informational

L. Dunbar
L. Yong
Song Xiao Lin
Huawei

Expires: April 2017

October 31, 2016

Client Defined Private Networks laid over Thin CPEs
draft-dunbar-opsawg-private-networks-over-thin-cpe-01

Abstract

This document specifies a type of private networks that interconnect thin CPEs at multiple client sites by IP tunnels, or more specifically, lay over multiple client sites' Thin CPEs via IP tunnels. Those private overlay networks not only interconnect those sites by secure IP tunnels but can also enforce the client specified policies to govern how applications or hosts within those sites communicate and how to access public internet.

Hosts or applications in those sites can be interconnected by Layer 2 networks or/and by Layer 3 networks. The network that the IP tunnels are traversing can be IPv4 or IPv6 networks. This document describes the special properties of the client defined networks over Thin CPEs.

A separate draft will describes the special features that those IP tunnels need to have in order to interconnect multiple sites as if those sites are directly connected by wires and how communication policies are enforced.

Status of This Document

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 April 31, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction.....	4
2. Terminology.....	4
2.1. Requirements Language.....	4
2.2. Terms defined in this document.....	4
3. Brief Description of the Private networks laid over Thin CPEs..	6
4. Overlay Private Network Configuration from Client Perspective..	8
4.1. Client Defined Overlay Private Networks.....	8
4.2. Client's site Configuration.....	8
4.3. Internet Gateway for each Site.....	9
4.4. Overlay-VPN Gateway.....	9
4.5. Interconnection among Sites.....	9
5. Protocols needed for the Client Defined Overlay Private Networks	10
5.1. Thin CPE Auto Instantiation.....	10
5.2. Network agnostic interworking.....	10
5.3. Gateway Anchor Auto-Selection.....	10
5.4. Middle boxes auto-creation and rules exchanges.....	10
5.5. Thin CPE on Third Party location.....	11
5.6. Client Defined Policies for traffic to/from client sites..	11
5.7. QoS policies.....	11
5.8. Explicit Service functions chain specified by clients....	11
5.9. Thin CPE monitoring.....	11

5.10. Alarm & Events via Thin CPE.....	11
5.11. Resource management via Thin CPE instantiated in Remote Locations.....	11
5.12. Client traffic flows management, monitoring, and reporting	11
6. Networks carried by IP tunnels in conjunction with existing L2VPN/L3VPN.....	12
7. IANA Considerations.....	12
8. Security Considerations.....	12
9. References.....	12
9.1. Normative References.....	12
9.2. Informative Reference.....	12
10. Authors' Addresses.....	12
11. Contributors Addresses.....	13

1. Introduction

This document specifies a type of private networks that interconnect thin CPEs at multiple client sites by IP tunnels, or more specifically, lay over multiple client sites' Thin CPEs via IP tunnels. Those private overlay networks not only interconnect those sites by secure IP tunnels but can also enforce the client specified policies to govern how applications or hosts within those sites communicate and how to access public internet.

Hosts or applications in those sites can be interconnected by Layer 2 networks or/and by Layer 3 networks. The network that the IP tunnels are traversing can be IPv4 or IPv6 networks. This document describes the special properties of the client defined networks over Thin CPEs.

For ease of description, the "Client Defined Private Overlay Network" is also called the client's "Overlay Private Network" or "Overlay Virtual Private Network (Overlay-VPN)" throughout this document.

A separate draft will describes the special features that those IP tunnels need to have in order to interconnect multiple sites as if those sites are directly connected by wires and how communication policies are enforced.

2. Terminology

2.1. 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 [RFC2119].

2.2. Terms defined in this document

Internet Gateway: a network function, which can be a physical device in the provider site or a virtual function instantiated to connect client site traffic to the public internet, and can enforce client specified policies.

Overlay Private Network: private network over a set of thin CPEs at multiple sites created by clients or users, who don't need to worry

about how thin CPEs are connected nor the protocol setting at network side. The "Overlay Private Network" not only interconnects multiple sites by (secure) IP tunnels but can also enforce the client specified policies to govern how applications or hosts within those sites communicate and how to access public internet.

Overlay-VPN: Overlay Private Network.

Provider site: the location where the provider have access to the devices or equipment.

Site: A place that contains switches, routers, services, appliances and these devices are configured to form L2 domain (s) or L3 domain. For example an Enterprise company data center, a college campus network center. For L3 subnets, either private IPv4 or IPv6 address or public IPv4 or Ipv6 address can be used.

SITE: Site Interconnection Tunnel Encapsulation Protocol

Thin CPE: a simple device at a customer premise that maps the site local traffic to either the IP tunnels connected to the Internet Gateway, or the IP tunnels connected to the VPN Gateway.

Overlay-VPN Gateway: the function (which can be virtual) that establish private (secure) connections to other sites belonging to the same client.

3. Brief Description of the Private networks laid over Thin CPEs

The following figure depicts multiple overlay private networks that interconnect the client's various sites. Note, the Overlay Private Network is marked as "Overlay" in the figure. The client can create multiple overlay private networks and then assign each site to specific overlay private networks. The client also specify the policies on what traffic to/from the clients can be exchanged with external network, which are enforced by the "Internet gateways" created by the provider.

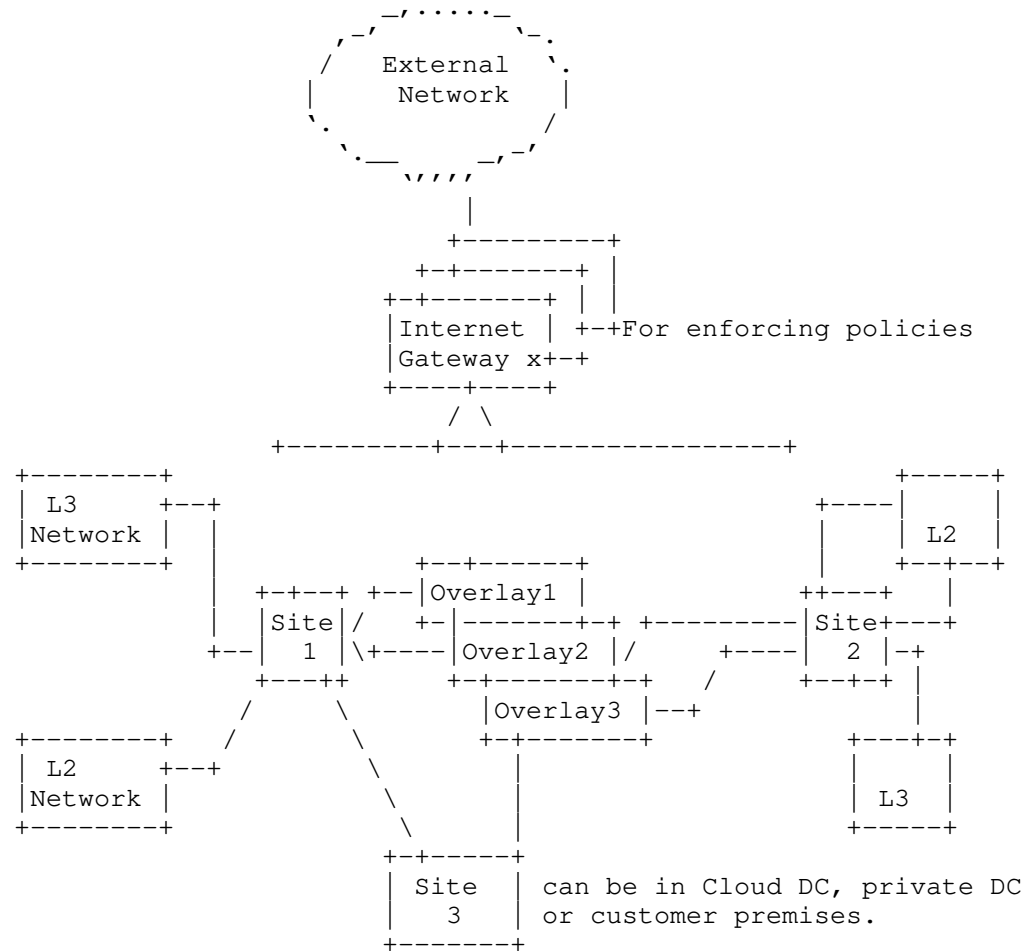


Figure 1 Overlay Private Networks interconnecting sites

Here are some key properties of Client defined Overlay Private Networks:

- Each client "Site" has a Thin CPE that is connected to a VPN gateway which is hosted in the provider site via IP Tunnel (which can be secured per customer request). The Thin CPE can be software image instantiated on virtual machines, physical CPE, or other form factors.

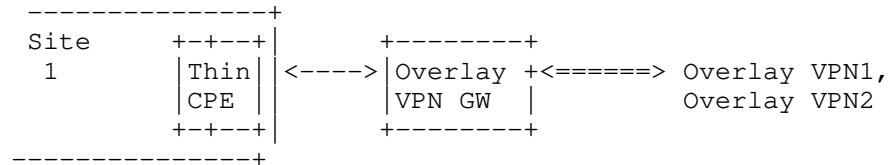


Figure 2 site Thin CPE connect to Overlay GW via IP Tunnel

- Each Thin CPE is connected to an "Internet Gateway" via IP Tunnel (that is automatically created by provider). The "Internet Gateway", virtual or physical, can be located anywhere. An IP Tunnel is created automatically between the Thin CPE and the "Internet Gateway".
- When the provider don't own the infrastructure to interconnect multiple sites, (secure) IP Tunnels are created among each site's VPN Gateway, so that each site's local networks (L2 or L3) attached to the Thin CPEs are interconnected as if those networks are directly connected by physical wire.
- Some traffic between Thin CPE have to go through secure tunnel, e.g. IPSec. Clients can specify what traffic to go through secure tunnels without specifically worrying about how to establish or maintain the secure tunnels. The client traffic can be carried by VxLAN (for interconnecting layer 2 traffic) or GRE (for L3 traffic) over the IPSEC tunnel.
- Client specifies the policies on how/what/when hosts from the interconnected sites can communicate with external peers; E.g. Hosts in one Layer 2 domain from one site may communicate with hosts in different Layer 2 domains in different sites.

The Client Defined Overlay Networks can be viewed by client as their own private networks. For ease of description, the terminology "Overlay Private Network" or "Overlay-VPN" is used throughout this document to refer to this kind of client defined overlay network over Thin CPEs.

"Overlay Private Network" is different from the IETF's L2VPN or L3VPN for the following reasons:

- Overlay-Private-Network is built upon IP network (whereas L2VPN/L3VPN is built upon MPLS network),
- Traffic originated from a client's site (where Thin CPE is instantiated) not only can communicate with hosts in other sites of the client via IP tunnels, but also can communicate with public internet (governed by the policies specified by the client),
- Client's site Thin CPE don't participate in IGP or BGP routing with provider side. Client can specify the prefixes and/or VLANs for each site so that they can be reached by external hosts,
- IP tunnel is automatically created between a Thin CPE and provider site where VPN gateway and internet gateway are instantiated and maintained.

4. Overlay Private Network Configuration from Client Perspective

4.1. Client Defined Overlay Private Networks

The client can specify multiple overlay private networks (a.k.a. Overlay-VPNs). Client can specify which sites connect to which Overlay-VPNs. Each Site can connect to multiple Overlay-VPNs.

As features on Thin CPE are very limited, each Overlay-VPN has its own Overlay VPN gateway in provider site to connect to Thin CPE via IP tunnel, as depicted in Figure 2 above.

4.2. Client's site Configuration

For each site, the client needs to specify:

- Site Identifier (include unique system Identifier, name, etc.)
- VLANs enabled on the site (i.e. the VLANs enabled on the client facing ports of the Thin CPE).
- Subnets from the site (i.e. the subnets enabled on the client facing ports of the Thin CPE)

- IP address for the Overlay-VPN Gateway that connect other sites belonging to the client
- IP address for the Internet Gateway

The configuration on the site is mainly for the Thin CPE instantiated on the site. Therefore, the client also needs to specify which VLANs/subnets are enabled on the ports of the Thin CPE facing the local network on the site.

4.3. Internet Gateway for each Site

Each site is associated with an Internet Gateway, which is automatically created by the provider. The Interconnect gateway can be a physical device on the provider site or a virtual function, to connect client site traffic to the public internet, and can enforce client specified policies.

Considering one client can have multiple sites in different geographic locations, the client can specify different policies for traffic to/from each site.

4.4. Overlay-VPN Gateway

The Overlay-VPN Gateway is on the provider site, connected to Thin CPE via IP tunnel. The purpose of the Overlay-VPN Gateway is to connect a site to its specified Overlay VPNs. Each site can be connected to multiple Overlay VPNs.

For each Overlay-VPN gateway, the client needs to specify:

- Identifier
- Which VPN is the Gateway connected to
- Upstream bandwidth from Thin CPE to the Overlay VPN GW
- Downstream bandwidth from the Overlay VPN GW to the Thin CPE

4.5. Interconnection among Sites

For each Overlay VPN, the Client can choose which sites are connected by specifying the VPN Gateway associated with each site.

5. Protocols needed for the Client Defined Overlay Private Networks

5.1. Thin CPE Auto Instantiation

Thin CPE is a simple device that maps the site local traffic to either the IP tunnels connected to the Internet Gateway, or the IP tunnels connected to the VPN Gateway.

5.2. Network agnostic interworking

IP tunnels are automatically created between Thin CPE and (Internet/VPN) gateways based on the traffic to the access network.

For Layer 2 traffic from the client local site, VxLAN is used to build the IP Tunnels to the site's Internet gateway or VPN gateway respectively.

For Layer 3 traffic from the client local site, GRE is used to build the IP Tunnels to the site's Internet gateway or VPN gateway respectively.

If the client specifies secure connection to other sites, IPsec is added to the tunnels between the Thin CPE and the VPN Gateway.

5.3. Gateway Anchor Auto-Selection

For each client site, internet gateway and VPN gateway will be automatically instantiated.

There will be protocol extension needed for the creation/deletion process and how NAT is used for client traffic from each site.

5.4. Middle boxes auto-creation and rules exchanges

To be added

5.5. Thin CPE on Third Party location

Thin CPEs can also be instantiated third party premises, such as cloud data centers. The instantiated Thin CPE can establish IP tunnels with the client's Internet Gateway or VPN Gateway.

5.6. Client Defined Policies for traffic to/from client sites

Depending on the policies specified by the clients, the Thin CPE jointly with the virtual GW will select the appropriate network security functions, i.e. (virtual) FW, IPS, IDS, or others to enforce the policies specified by the clients.

The policies specified by the clients will be more expressed in clients' oriented language, e.g. using client Identifier or virtual addresses (instead of IP addresses of the actual packets traverse the FW). Those policies will be translated to the implementable rules to the chosen network security functions, such as FW.

5.7. QoS policies

To be added

5.8. Explicit Service functions chain specified by clients

Clients can query network service functions available to them and the capabilities of those functions. Then, the client can choose a set of them, either in strict sequence or simply as a set to apply to their traffic.

The policies to service functions can follow the guideline specified by [I2NSF-framework].

5.9. Thin CPE monitoring

5.10. Alarm & Events via Thin CPE

To be added

5.11. Resource management via Thin CPE instantiated in Remote Locations

To be added

5.12. Client traffic flows management, monitoring, and reporting

To be added

6. Networks carried by IP tunnels in conjunction with existing
L2VPN/L3VPN

7. IANA Considerations

To be added

8. Security Considerations

To be added.

9. References

9.1. Normative References

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

9.2. Informative Reference

[I2NSF-Framework] Lopez, D, et al, "Framework for Interface to
Network security functions", draft-ietf-i2nsf-framework-04,
Oct 2016

10. Authors' Addresses

Linda Dunbar
Huawei Technologies
Email: linda.dunbar@huawei.com

Lucy Yong
Huawei Technologies
Email: lucy.yong@huawei.com

Song Xiao Li
Huawei Technologies
Email: sxlin@huawei.com

11. Contributors Addresses

Xuan Ming fu
Huawei Technologies
xuanmingfu@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 17, 2018

E. Lear
Cisco Systems
R. Droms
Google
D. Romascanu
June 15, 2018

Manufacturer Usage Description Specification
draft-ietf-opsawg-mud-25

Abstract

This memo specifies a component-based architecture for manufacturer usage descriptions (MUD). The goal of MUD is to provide a means for end devices to signal to the network what sort of access and network functionality they require to properly function. The initial focus is on access control. Later work can delve into other aspects.

This memo specifies two YANG modules, IPv4 and IPv6 DHCP options, an LLDP TLV, a URL, an X.509 certificate extension and a means to sign and verify the descriptions.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 17, 2018.

Copyright Notice

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

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

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. What MUD Doesn't Do	5
1.2. A Simple Example	5
1.3. Terminology	5
1.4. Determining Intended Use	6
1.5. Finding A Policy: The MUD URL	6
1.6. Processing of the MUD URL	7
1.7. Types of Policies	8
1.8. The Manufacturer Usage Description Architecture	10
1.9. Order of operations	11
2. The MUD Model and Semantic Meaning	12
2.1. The IETF-MUD YANG Module	13
3. MUD model definitions for the root mud container	14
3.1. mud-version	14
3.2. mud-url	15
3.3. to-device-policy and from-device-policy containers	15
3.4. last-update	15
3.5. cache-validity	15
3.6. is-supported	15
3.7. systeminfo	15
3.8. mfg-name, software-rev, model-name firmware-rev	16
3.9. extensions	16
4. Augmentation to the ACL Model	16
4.1. manufacturer	16
4.2. same-manufacturer	16
4.3. documentation	17
4.4. model	17
4.5. local-networks	17
4.6. controller	17
4.7. my-controller	18
4.8. direction-initiated	18
5. Processing of the MUD file	18
6. What does a MUD URL look like?	18
7. The MUD YANG Model	19
8. The Domain Name Extension to the ACL Model	25
8.1. src-dnsname	26
8.2. dst-dnsname	26
8.3. The ietf-acldns Model	26
9. MUD File Example	28

10. The MUD URL DHCP Option	30
10.1. Client Behavior	31
10.2. Server Behavior	31
10.3. Relay Requirements	32
11. The Manufacturer Usage Description (MUD) URL X.509 Extension	32
12. The Manufacturer Usage Description LLDP extension	34
13. Creating and Processing of Signed MUD Files	35
13.1. Creating a MUD file signature	36
13.2. Verifying a MUD file signature	36
14. Extensibility	37
15. Deployment Considerations	37
16. Security Considerations	38
17. IANA Considerations	40
17.1. YANG Module Registrations	41
17.2. DHCPv4 and DHCPv6 Options	41
17.3. PKIX Extensions	41
17.4. MIME Media-type Registration for MUD files	42
17.5. LLDP IANA TLV Subtype Registry	42
17.6. The MUD Well Known Universal Resource Name (URNs)	43
17.7. Extensions Registry	43
18. Acknowledgments	43
19. References	44
19.1. Normative References	44
19.2. Informative References	47
Appendix A. Changes from Earlier Versions	49
Appendix B. Default MUD nodes	52
Appendix C. A Sample Extension: DETNET-indicator	57
Authors' Addresses	60

1. Introduction

The Internet has largely been constructed for general purpose computers, those devices that may be used for a purpose that is specified by those who own the device. [RFC1984] presumed that an end device would be most capable of protecting itself. This made sense when the typical device was a workstation or a mainframe, and it continues to make sense for general purpose computing devices today, including laptops, smart phones, and tablets.

[RFC7452] discusses design patterns for, and poses questions about, smart objects. Let us then posit a group of objects that are specifically not intended to be used for general purpose computing tasks. These devices, which this memo refers to as Things, have a specific purpose. By definition, therefore, all other uses are not intended. If a small number of communication patterns follows from those small number of uses, the combination of these two statements can be restated as a manufacturer usage description (MUD) that can be applied at various points within a network. MUD primarily addresses

threats to the device rather than the device as a threat. In some circumstances, however, MUD may offer some protection in the latter case, depending on the MUD-URL is communicated, and how devices and their communications are authenticated.

We use the notion of "manufacturer" loosely in this context to refer to the entity or organization that will state how a device is intended to be used. For example, in the context of a lightbulb, this might indeed be the lightbulb manufacturer. In the context of a smarter device that has a built in Linux stack, it might be an integrator of that device. The key points are that the device itself is assumed to serve a limited purpose, and that there exists an organization in the supply chain of that device that will take responsibility for informing the network about that purpose.

The intent of MUD is to provide the following:

- o Substantially reduce the threat surface on a device to those communications intended by the manufacturer.
- o Provide a means to scale network policies to the ever-increasing number of types of devices in the network.
- o Provide a means to address at least some vulnerabilities in a way that is faster than the time it might take to update systems. This will be particularly true for systems that are no longer supported.
- o Keep the cost of implementation of such a system to the bare minimum.
- o Provide a means of extensibility for manufacturers to express other device capabilities or requirements.

MUD consists of three architectural building blocks:

- o A URL that can be used to locate a description;
- o The description itself, including how it is interpreted, and;
- o A means for local network management systems to retrieve the description.

MUD is most effective when the network is able to identify in some way the remote endpoints that Things will talk to.

In this specification we describe each of these building blocks and how they are intended to be used together. However, they may also be

used separately, independent of this specification, by local deployments for their own purposes.

1.1. What MUD Doesn't Do

MUD is not intended to address network authorization of general purpose computers, as their manufacturers cannot envision a specific communication pattern to describe. In addition, even those devices that have a single or small number of uses might have very broad communication patterns. MUD on its own is not for them either.

Although MUD can provide network administrators with some additional protection when device vulnerabilities exist, it will never replace the need for manufacturers to patch vulnerabilities.

Finally, no matter what the manufacturer specifies in a MUD file, these are not directives, but suggestions. How they are instantiated locally will depend on many factors and will be ultimately up to the local network administrator, who must decide what is appropriate in a given circumstances.

1.2. A Simple Example

A light bulb is intended to light a room. It may be remotely controlled through the network, and it may make use of a rendezvous service of some form that an application on a smart phone. What we can say about that light bulb, then, is that all other network access is unwanted. It will not contact a news service, nor speak to the refrigerator, and it has no need of a printer or other devices. It has no social networking friends. Therefore, an access list applied to it that states that it will only connect to the single rendezvous service will not impede the light bulb in performing its function, while at the same time allowing the network to provide both it and other devices an additional layer of protection.

1.3. Terminology

MUD: manufacturer usage description.

MUD file: a file containing YANG-based JSON that describes a Thing and associated suggested specific network behavior.

MUD file server: a web server that hosts a MUD file.

MUD manager: the system that requests and receives the MUD file from the MUD server. After it has processed a MUD file, it may direct changes to relevant network elements.

MUD controller: a synonym that has been used in the past for MUD manager.

MUD URL: a URL that can be used by the MUD manager to receive the MUD file.

Thing: the device emitting a MUD URL.

Manufacturer: the entity that configures the Thing to emit the MUD URL and the one who asserts a recommendation in a MUD file. The manufacturer might not always be the entity that constructs a Thing. It could, for instance, be a systems integrator, or even a component provider.

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

1.4. Determining Intended Use

The notion of intended use is in itself not new. Network administrators apply access lists every day to allow for only such use. This notion of white listing was well described by Chapman and Zwicky in [FW95]. Profiling systems that make use of heuristics to identify types of systems have existed for years as well.

A Thing could just as easily tell the network what sort of access it requires without going into what sort of system it is. This would, in effect, be the converse of [RFC7488]. In seeking a general solution, however, we assume that a device will implement functionality necessary to fulfill its limited purpose. This is basic economic constraint. Unless the network would refuse access to such a device, its developers would have no reason to provide the network any information. To date, such an assertion has held true.

1.5. Finding A Policy: The MUD URL

Our work begins with the device emitting a Universal Resource Locator (URL) [RFC3986]. This URL serves both to classify the device type and to provide a means to locate a policy file.

MUD URLs MUST use the HTTPS scheme [RFC7230].

In this memo three means are defined to emit the MUD URL, as follows:

- o A DHCP option[RFC2131],[RFC3315] that the DHCP client uses to inform the DHCP server. The DHCP server may take further actions, such as act as the MUD manager or otherwise pass the MUD URL along to the MUD manager.
- o An X.509 constraint. The IEEE has developed [IEEE8021AR] that provides a certificate-based approach to communicate device characteristics, which itself relies on [RFC5280]. The MUD URL extension is non-critical, as required by IEEE 802.1AR. Various means may be used to communicate that certificate, including Tunnel Extensible Authentication Protocol (TEAP) [RFC7170].
- o Finally, a Link Layer Discovery Protocol (LLDP) frame is defined [IEEE8021AB].

It is possible that there may be other means for a MUD URL to be learned by a network. For instance, some devices may already be fielded or have very limited ability to communicate a MUD URL, and yet can be identified through some means, such as a serial number or a public key. In these cases, manufacturers may be able to map those identifiers to particular MUD URLs (or even the files themselves). Similarly, there may be alternative resolution mechanisms available for situations where Internet connectivity is limited or does not exist. Such mechanisms are not described in this memo, but are possible. Implementors are encouraged to allow for this sort of flexibility of how MUD URLs may be learned.

1.6. Processing of the MUD URL

MUD managers that are able to do so SHOULD retrieve MUD URLs and signature files as per [RFC7230], using the GET method [RFC7231]. They MUST validate the certificate using the rules in [RFC2818], Section 3.1.

Requests for MUD URLs SHOULD include an "Accept" header ([RFC7231], Section 5.3.2) containing "application/mud+json", an "Accept-Language" header field ([RFC7231], Section 5.3.5), and a "User-Agent" header ([RFC7231], Section 5.5.3).

MUD managers SHOULD automatically process 3xx response status codes.

If a MUD manager is not able to fetch a MUD URL, other means MAY be used to import MUD files and associated signature files. So long as the signature of the file can be validated, the file can be used. In such environments, controllers SHOULD warn administrators when cache-validity expiry is approaching so that they may check for new files.

It may not be possible for a MUD manager to retrieve a MUD file at any given time. Should a MUD manager fail to retrieve a MUD file, it SHOULD consider the existing one safe to use, at least for a time. After some period, it SHOULD log that it has been unable to retrieve the file. There may be very good reasons for such failures, including the possibility that the MUD manager is in an off-line environment, the local Internet connection has failed, or the remote Internet connection has failed. It is also possible that an attacker is attempting to interfere with the deployment of a device. It is a local decision as to how to handle such circumstances.

1.7. Types of Policies

When the MUD URL is resolved, the MUD manager retrieves a file that describes what sort of communications a device is designed to have. The manufacturer may specify either specific hosts for cloud based services or certain classes for access within an operational network. An example of a class might be "devices of a specified manufacturer type", where the manufacturer type itself is indicated simply by the authority component (e.g, the domain name) of the MUD URL. Another example might be to allow or disallow local access. Just like other policies, these may be combined. For example:

- o Allow access to devices of the same manufacturer
- o Allow access to and from controllers via Constrained Application Protocol (COAP) [RFC7252]
- o Allow access to local DNS/NTP
- o Deny all other access

A printer might have a description that states:

- o Allow access for port IPP or port LPD
- o Allow local access for port HTTP
- o Deny all other access

In this way anyone can print to the printer, but local access would be required for the management interface.

The files that are retrieved are intended to be closely aligned to existing network architectures so that they are easy to deploy. We make use of YANG [RFC7950] because it provides accurate and adequate models for use by network devices. JSON[RFC8259] is used as a

serialization format for compactness and readability, relative to XML. Other formats may be chosen with later versions of MUD.

While the policy examples given here focus on access control, this is not intended to be the sole focus. By structuring the model described in this document with clear extension points, other descriptions could be included. One that often comes to mind is quality of service.

The YANG modules specified here are extensions of [I-D.ietf-netmod-acl-model]. The extensions to this model allow for a manufacturer to express classes of systems that a manufacturer would find necessary for the proper function of the device. Two modules are specified. The first module specifies a means for domain names to be used in ACLs so that devices that have their controllers in the cloud may be appropriately authorized with domain names, where the mapping of those names to addresses may rapidly change.

The other module abstracts away IP addresses into certain classes that are instantiated into actual IP addresses through local processing. Through these classes, manufacturers can specify how the device is designed to communicate, so that network elements can be configured by local systems that have local topological knowledge. That is, the deployment populates the classes that the manufacturer specifies. The abstractions below map to zero or more hosts, as follows:

Manufacturer: A device made by a particular manufacturer, as identified by the authority component of its MUD URL

same-manufacturer: Devices that have the same authority component of their MUD URL.

controller: Devices that the local network administrator admits to the particular class.

my-controller: Devices intended to serve as controllers for the MUD-URL that the Thing emitted.

local: The class of IP addresses that are scoped within some administrative boundary. By default it is suggested that this be the local subnet.

The "manufacturer" classes can be easily specified by the manufacturer, whereas controller classes are initially envisioned to be specified by the administrator.

Because manufacturers do not know who will be using their devices, it is important for functionality referenced in usage descriptions to be relatively ubiquitous and mature. For these reasons the YANG-based configuration in a MUD file is limited to either the modules specified or referenced in this document, or those specified in documented extensions.

1.8. The Manufacturer Usage Description Architecture

With these components laid out we now have the basis for an architecture. This leads us to ASCII art.

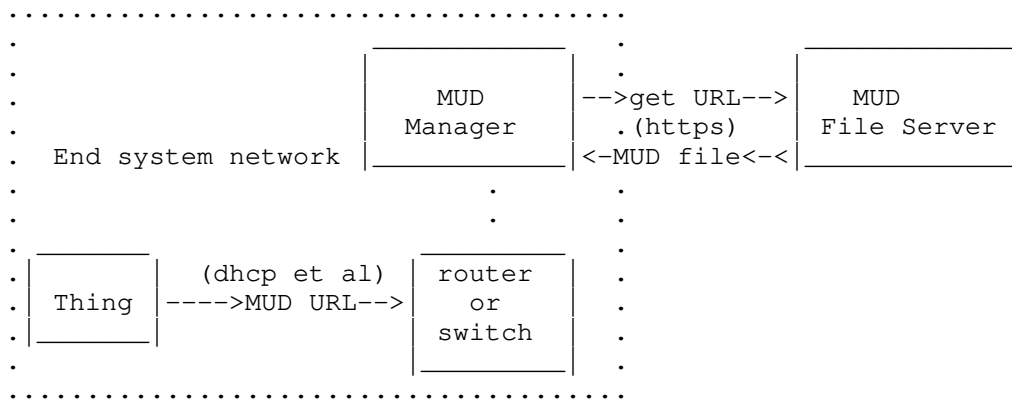


Figure 1: MUD Architecture

In the above diagram, the switch or router collects MUD URLs and forwards them to the MUD manager (a network management system) for processing. This happens in different ways, depending on how the URL is communicated. For instance, in the case of DHCP, the DHCP server might receive the URL and then process it. In the case of IEEE 802.1X [IEEE8021X], the switch would carry the URL via a certificate to the authentication server via EAP over Radius[RFC3748], which would then process it. One method to do this is TEAP, described in [RFC7170]. The certificate extension is described below.

The information returned by the MUD file server is valid for as long as the Thing is connected. There is no expiry. However, if the MUD manager has detected that the MUD file for a Thing has changed, it SHOULD update the policy expeditiously, taking into account whatever approval flow is required in a deployment. In this way, new recommendations from the manufacturer can be processed in a timely fashion.

The information returned by the MUD file server (a web server) is valid for the duration of the Thing's connection, or as specified in the description. Thus if the Thing is disconnected, any associated configuration in the switch can be removed. Similarly, from time to time the description may be refreshed, based on new capabilities or communication patterns or vulnerabilities.

The web server is typically run by or on behalf of the manufacturer. Its domain name is that of the authority found in the MUD URL. For legacy cases where Things cannot emit a URL, if the switch is able to determine the appropriate URL, it may proxy it. In the trivial case it may hardcode MUD-URL on a switch port or a map from some available identifier such as an L2 address or certificate hash to a MUD-URL.

The role of the MUD manager in this environment is to do the following:

- o receive MUD URLs,
- o fetch MUD files,
- o translate abstractions in the MUD files to specific network element configuration,
- o maintain and update any required mappings of the abstractions, and
- o update network elements with appropriate configuration.

A MUD manager may be a component of a AAA or network management system. Communication within those systems and from those systems to network elements is beyond the scope of this memo.

1.9. Order of operations

As mentioned above, MUD contains architectural building blocks, and so order of operation may vary. However, here is one clear intended example:

1. Thing emits URL.
2. That URL is forwarded to a MUD manager by the nearest switch (how this happens depends on the way in which the MUD URL is emitted).
3. The MUD manager retrieves the MUD file and signature from the MUD file server, assuming it doesn't already have copies. After validating the signature, it may test the URL against a web or domain reputation service, and it may test any hosts within the file against those reputation services, as it deems fit.

4. The MUD manager may query the administrator for permission to add the Thing and associated policy. If the Thing is known or the Thing type is known, it may skip this step.
5. The MUD manager instantiates local configuration based on the abstractions defined in this document.
6. The MUD manager configures the switch nearest the Thing. Other systems may be configured as well.
7. When the Thing disconnects, policy is removed.

2. The MUD Model and Semantic Meaning

A MUD file consists of a YANG model instance that has been serialized in JSON [RFC7951]. For purposes of MUD, the nodes that can be modified are access lists as augmented by this model. The MUD file is limited to the serialization of only the following YANG schema:

- o ietf-access-control-list [I-D.ietf-netmod-acl-model]
- o ietf-mud (this document)
- o ietf-acldns (this document)

Extensions may be used to add additional schema. This is described further on.

To provide the widest possible deployment, publishers of MUD files SHOULD make use of the abstractions in this memo and avoid the use of IP addresses. A MUD manager SHOULD NOT automatically implement any MUD file that contains IP addresses, especially those that might have local significance. The addressing of one side of an access list is implicit, based on whether it is applied as to-device-policy or from-device-policy.

With the exceptions of "name" of the ACL, "type", "name" of the ACE, and TCP and UDP source and destination port information, publishers of MUD files SHOULD limit the use of ACL model leaf nodes expressed to those found in this specification. Absent any extensions, MUD files are assumed to implement only the following ACL model features:

- o match-on-ipv4, match-on-ipv6, match-on-tcp, match-on-udp, match-on-icmp

Furthermore, only "accept" or "drop" actions SHOULD be included. A MUD manager MAY choose to interpret "reject" as "drop". A MUD manager SHOULD ignore all other actions. This is because

manufacturers do not have sufficient context within a local deployment to know whether reject is appropriate. That is a decision that should be left to a network administrator.

Given that MUD does not deal with interfaces, the support of the "ietf-interfaces" module [RFC8343] is not required. Specifically, the support of interface-related features and branches (e.g., interface-attachment and interface-stats) of the ACL YANG module is not required.

In fact, MUD managers MAY ignore any particular component of a description or MAY ignore the description in its entirety, and SHOULD carefully inspect all MUD descriptions. Publishers of MUD files MUST NOT include other nodes except as described in Section 3.9. See that section for more information.

2.1. The IETF-MUD YANG Module

This module is structured into three parts:

- o The first component, the "mud" container, holds information that is relevant to retrieval and validity of the MUD file itself, as well as policy intended to and from the Thing.
- o The second component augments the matching container of the ACL model to add several nodes that are relevant to the MUD URL, or otherwise abstracted for use within a local environment.
- o The third component augments the tcp-acl container of the ACL model to add the ability to match on the direction of initiation of a TCP connection.

A valid MUD file will contain two root objects, a "mud" container and an "acls" container. Extensions may add additional root objects as required. As a reminder, when parsing acls, elements within a "match" block are logically ANDed. In general, a single abstraction in a match statement should be used. For instance, it makes little sense to match both "my-controller" and "controller" with an argument, since they are highly unlikely to be the same value.

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is explained in [RFC8340].

```

module: ietf-mud
  +--rw mud!
    +--rw mud-version          uint8
    +--rw mud-url              inet:uri
    +--rw last-update          yang:date-and-time
    +--rw mud-signature?       inet:uri
    +--rw cache-validity?      uint8
    +--rw is-supported          boolean
    +--rw systeminfo?          string
    +--rw mfg-name?            string
    +--rw model-name?          string
    +--rw firmware-rev?        string
    +--rw software-rev?        string
    +--rw documentation?       inet:uri
    +--rw extensions*          string
    +--rw from-device-policy
      +--rw acls
        +--rw access-list* [name]
          +--rw name        -> /acl:acls/acl/name
    +--rw to-device-policy
      +--rw acls
        +--rw access-list* [name]
          +--rw name        -> /acl:acls/acl/name

augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
  +--rw mud
    +--rw manufacturer?       inet:host
    +--rw same-manufacturer?   empty
    +--rw model?              inet:uri
    +--rw local-networks?     empty
    +--rw controller?         inet:uri
    +--rw my-controller?      empty
  augment
    /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches
    /acl:l4/acl:tcp/acl:tcp:
    +--rw direction-initiated? direction

```

3. MUD model definitions for the root mud container

3.1. mud-version

This node specifies the integer version of the MUD specification.
 This memo specifies version 1.

3.2. mud-url

This URL identifies the MUD file. This is useful when the file and associated signature are manually uploaded, say, in an offline mode.

3.3. to-device-policy and from-device-policy containers

[I-D.ietf-netmod-acl-model] describes access-lists. In the case of MUD, a MUD file must be explicit in describing the communication pattern of a Thing, and that includes indicating what is to be permitted or denied in either direction of communication. Hence each of these containers indicates the appropriate direction of a flow in association with a particular Thing. They contain references to specific access-lists.

3.4. last-update

This is a date-and-time value of when the MUD file was generated. This is akin to a version number. Its form is taken from [RFC6991] which, for those keeping score, in turn was taken from Section 5.6 of [RFC3339], which was taken from [ISO.8601.1988].

3.5. cache-validity

This uint8 is the period of time in hours that a network management station MUST wait since its last retrieval before checking for an update. It is RECOMMENDED that this value be no less than 24 and MUST NOT be more than 168 for any Thing that is supported. This period SHOULD be no shorter than any period determined through HTTP caching directives (e.g., "cache-control" or "Expires"). N.B., expiring of this timer does not require the MUD manager to discard the MUD file, nor terminate access to a Thing. See Section 16 for more information.

3.6. is-supported

This boolean is an indication from the manufacturer to the network administrator as to whether or not the Thing is supported. In this context a Thing is said to not be supported if the manufacturer intends never to issue a firmware or software update to the Thing or never update the MUD file. A MUD manager MAY still periodically check for updates.

3.7. systeminfo

This is a textual UTF-8 description of the Thing to be connected. The intent is for administrators to be able to see a brief

displayable description of the Thing. It SHOULD NOT exceed 60 characters worth of display space.

3.8. mfg-name, software-rev, model-name firmware-rev

These optional fields are filled in as specified by [RFC8348]. Note that firmware-rev and software-rev MUST NOT be populated in a MUD file if the device can be upgraded but the MUD-URL cannot be. This would be the case, for instance, with MUD-URLs that are contained in 802.1AR certificates.

3.9. extensions

This optional leaf-list names MUD extensions that are used in the MUD file. Note that MUD extensions MUST NOT be used in a MUD file without the extensions being declared. Implementations MUST ignore any node in this file that they do not understand.

Note that extensions can either extend the MUD file as described in the previous paragraph, or they might reference other work. An extension example can be found in Appendix C.

4. Augmentation to the ACL Model

Note that in this section, when we use the term "match" we are referring to the ACL model "matches" node.

4.1. manufacturer

This node consists of a hostname that would be matched against the authority component of another Thing's MUD URL. In its simplest form "manufacturer" and "same-manufacturer" may be implemented as access-lists. In more complex forms, additional network capabilities may be used. For example, if one saw the line "manufacturer" : "flobbidy.example.com", then all Things that registered with a MUD URL that contained flobbity.example.com in its authority section would match.

4.2. same-manufacturer

This null-valued node is an equivalent for when the manufacturer element is used to indicate the authority that is found in another Thing's MUD URL matches that of the authority found in this Thing's MUD URL. For example, if the Thing's MUD URL were <https://b1.example.com/ThingV1>, then all devices that had MUD URL with an authority section of b1.example.com would match.

4.3. documentation

This URI consists of a URL that points to documentation relating to the device and the MUD file. This can prove particularly useful when the "controller" class is used, so that its use can be explained.

4.4. model

This string matches the entire MUD URL, thus covering the model that is unique within the context of the authority. It may contain not only model information, but versioning information as well, and any other information that the manufacturer wishes to add. The intended use is for devices of this precise class to match, to permit or deny communication between one another.

4.5. local-networks

This null-valued node expands to include local networks. Its default expansion is that packets must not traverse toward a default route that is received from the router. However, administrators may expand the expression as is appropriate in their deployments.

4.6. controller

This URI specifies a value that a controller will register with the MUD manager. The node then is expanded to the set of hosts that are so registered. This node may also be a URN. In this case, the URN describes a well known service, such as DNS or NTP, that has been standardized. Both of those URNs may be found in Section 17.6.

When "my-controller" is used, it is possible that the administrator will be prompted to populate that class for each and every model. Use of "controller" with a named class allows the user to populate that class only once for many different models that a manufacturer may produce.

Controller URIs MAY take the form of a URL (e.g. "http[s]://"). However, MUD managers MUST NOT resolve and retrieve such files, and it is RECOMMENDED that there be no such file at this time, as their form and function may be defined at a point in the future. For now, URLs should serve simply as class names and may be populated by the local deployment administrator.

Great care should be taken by MUD managers when invoking the controller class in the form of URLs. For one thing, it requires some understanding by the administrator as to when it is appropriate. Pre-registration in such classes by controllers with the MUD server

is encouraged. The mechanism to do that is beyond the scope of this work.

4.7. my-controller

This null-valued node signals to the MUD manager to use whatever mapping it has for this MUD URL to a particular group of hosts. This may require prompting the administrator for class members. Future work should seek to automate membership management.

4.8. direction-initiated

This MUST only be applied to TCP. This matches the direction in which a TCP connection is initiated. When direction initiated is "from-device", packets that are transmitted in the direction of a thing MUST be dropped unless the thing has first initiated a TCP connection. By way of example, this node may be implemented in its simplest form by looking at naked SYN bits, but may also be implemented through more stateful mechanisms.

When applied this matches packets when the flow was initiated in the corresponding direction. [RFC6092] specifies IPv6 guidance best practices. While that document is scoped specifically to IPv6, its contents are applicable for IPv4 as well.

5. Processing of the MUD file

To keep things relatively simple in addition to whatever definitions exist, we also apply two additional default behaviors:

- o Anything not explicitly permitted is denied.
- o Local DNS and NTP are, by default, permitted to and from the Thing.

An explicit description of the defaults can be found in Appendix B. These are applied AFTER all other explicit rules. Thus, a default behavior can be changed with a "drop" action.

6. What does a MUD URL look like?

MUD URLs are required to use the HTTPS scheme, in order to establish the MUD file server's identity and assure integrity of the MUD file.

Any "https://" URL can be a MUD URL. For example:

```
https://things.example.org/product_abc123/v5
https://www.example.net/mudfiles/temperature_sensor/
https://example.com/lightbulbs/colour/v1
```

A manufacturer may construct a MUD URL in any way, so long as it makes use of the "https" schema.

7. The MUD YANG Model

```
<CODE BEGINS>file "ietf-mud@2018-06-15.yang"
module ietf-mud {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud";
  prefix ietf-mud;

  import ietf-access-control-list {
    prefix acl;
  }
  import ietf-yang-types {
    prefix yang;
  }
  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF OPSAWG (Ops Area) Working Group";
  contact
    "WG Web: http://tools.ietf.org/wg/opsawg/
    WG List: opsawg@ietf.org
    Author: Eliot Lear
    lear@cisco.com
    Author: Ralph Droms
    rdroms@gmail.com
    Author: Dan Romascanu
    dromasca@gmail.com

    ";
  description
    "This YANG module defines a component that augments the
    IETF description of an access list. This specific module
    focuses on additional filters that include local, model,
    and same-manufacturer.

    This module is intended to be serialized via JSON and stored
    as a file, as described in RFC XXXX [RFC Editor to fill in with
    this document #]."
```

Copyright (c) 2016,2017 IETF Trust and the persons identified as the document authors. All rights reserved. Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>). This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

```
revision 2018-06-15 {
  description
    "Initial proposed standard.";
  reference
    "RFC XXXX: Manufacturer Usage Description
    Specification";
}

typedef direction {
  type enumeration {
    enum to-device {
      description
        "packets or flows destined to the target
        Thing";
    }
    enum from-device {
      description
        "packets or flows destined from
        the target Thing";
    }
  }
  description
    "Which way are we talking about?";
}

container mud {
  presence "Enabled for this particular MUD URL";
  description
    "MUD related information, as specified
    by RFC-XXXX [RFC Editor to fill in].";
  uses mud-grouping;
}

grouping mud-grouping {
  description
    "Information about when support end(ed), and
    when to refresh";
```



```
leaf mud-version {
  type uint8;
  mandatory true;
  description
    "This is the version of the MUD
    specification.  This memo specifies version 1.";
}
leaf mud-url {
  type inet:uri;
  mandatory true;
  description
    "This is the MUD URL associated with the entry found
    in a MUD file.";
}
leaf last-update {
  type yang:date-and-time;
  mandatory true;
  description
    "This is intended to be when the current MUD file
    was generated.  MUD Managers SHOULD NOT check
    for updates between this time plus cache validity";
}
leaf mud-signature {
  type inet:uri;
  description
    "A URI that resolves to a signature as
    described in this specification.";
}
leaf cache-validity {
  type uint8 {
    range "1..168";
  }
  units "hours";
  default "48";
  description
    "The information retrieved from the MUD server is
    valid for these many hours, after which it should
    be refreshed.  N.B. MUD manager implementations
    need not discard MUD files beyond this period.";
}
leaf is-supported {
  type boolean;
  mandatory true;
  description
    "This boolean indicates whether or not the Thing is
    currently supported by the manufacturer.";
}
leaf systeminfo {
```

```
    type string;
    description
        "A UTF-8 description of this Thing.  This
        should be a brief description that may be
        displayed to the user to determine whether
        to allow the Thing on the
        network.";
}
leaf mfg-name {
    type string;
    description
        "Manufacturer name, as described in
        the ietf-hardware YANG module.";
}
leaf model-name {
    type string;
    description
        "Model name, as described in the
        ietf-hardware YANG module.";
}
leaf firmware-rev {
    type string;
    description
        "firmware-rev, as described in the
        ietf-hardware YANG module.  Note this field MUST
        NOT be included when the device can be updated
        but the MUD-URL cannot.";
}
leaf software-rev {
    type string;
    description
        "software-rev, as described in the
        ietf-hardware YANG module.  Note this field MUST
        NOT be included when the device can be updated
        but the MUD-URL cannot.";
}
leaf documentation {
    type inet:uri;
    description
        "This URL points to documentation that
        relates to this device and any classes that it uses
        in its MUD file.  A caution: MUD managers need
        not resolve this URL on their own, but rather simply
        provide it to the administrator.  Parsing HTML is
        not an intended function of a MUD manager.";
}
leaf-list extensions {
    type string {
```

```
        length "1..40";
    }
    description
        "A list of extension names that are used in this MUD
        file. Each name is registered with the IANA and
        described in an RFC.";
    }
    container from-device-policy {
        description
            "The policies that should be enforced on traffic
            coming from the device. These policies are not
            necessarily intended to be enforced at a single
            point, but may be rendered by the controller to any
            relevant enforcement points in the network or
            elsewhere.";
        uses access-lists;
    }
    container to-device-policy {
        description
            "The policies that should be enforced on traffic
            going to the device. These policies are not
            necessarily intended to be enforced at a single
            point, but may be rendered by the controller to any
            relevant enforcement points in the network or
            elsewhere.";
        uses access-lists;
    }
}

grouping access-lists {
    description
        "A grouping for access lists in the context of device
        policy.";
    container access-lists {
        description
            "The access lists that should be applied to traffic
            to or from the device.";
        list access-list {
            key "name";
            description
                "Each entry on this list refers to an ACL that
                should be present in the overall access list
                data model. Each ACL is identified by name and
                type.";
            leaf name {
                type leafref {
                    path "/acl:acls/acl:acl/acl:name";
                }
            }
        }
    }
}
```

```
        description
          "The name of the ACL for this entry.";
      }
  }
}

augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" {
  description
    "adding abstractions to avoid need of IP addresses";
  container mud {
    description
      "MUD-specific matches.";
    leaf manufacturer {
      type inet:host;
      description
        "A domain that is intended to match the authority
        section of the MUD URL. This node is used to specify
        one or more manufacturers a device should
        be authorized to access.";
    }
    leaf same-manufacturer {
      type empty;
      description
        "This node matches the authority section of the MUD URL
        of a Thing. It is intended to grant access to all
        devices with the same authority section.";
    }
    leaf model {
      type inet:uri;
      description
        "Devices of the specified model type will match if
        they have an identical MUD URL.";
    }
    leaf local-networks {
      type empty;
      description
        "IP addresses will match this node if they are
        considered local addresses. A local address may be
        a list of locally defined prefixes and masks
        that indicate a particular administrative scope.";
    }
    leaf controller {
      type inet:uri;
      description
        "This node names a class that has associated with it
        zero or more IP addresses to match against. These
        may be scoped to a manufacturer or via a standard
```

```

        URN.";
    }
    leaf my-controller {
        type empty;
        description
            "This node matches one or more network elements that
             have been configured to be the controller for this
             Thing, based on its MUD URL.";
    }
}
}
}
augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" +
    "/acl:l4/acl:tcp/acl:tcp" {
    description
        "add direction-initiated";
    leaf direction-initiated {
        type direction;
        description
            "This node matches based on which direction a
             connection was initiated. The means by which that
             is determined is discussed in this document.";
    }
}
}
}
<CODE ENDS>

```

8. The Domain Name Extension to the ACL Model

This module specifies an extension to IETF-ACL model such that domain names may be referenced by augmenting the "matches" node. Different implementations may deploy differing methods to maintain the mapping between IP address and domain name, if indeed any are needed. However, the intent is that resources that are referred to using a name should be authorized (or not) within an access list.

The structure of the change is as follows:

```

module: ietf-acldns
    augment /acl:acls/acl:acl/acl:aces/acl:ace/
        acl:matches/acl:l3/acl:ipv4/acl:ipv4:
            +--rw src-dnsname?    inet:host
            +--rw dst-dnsname?    inet:host
    augment /acl:acls/acl:acl/acl:aces/acl:ace/
        acl:matches/acl:l3/acl:ipv6/acl:ipv6:
            +--rw src-dnsname?    inet:host
            +--rw dst-dnsname?    inet:host

```

The choice of these particular points in the access-list model is based on the assumption that we are in some way referring to IP-related resources, as that is what the DNS returns. A domain name in our context is defined in [RFC6991]. The augmentations are replicated across IPv4 and IPv6 to allow MUD file authors the ability to control the IP version that the Thing may utilize.

The following node are defined.

8.1. src-dnsname

The argument corresponds to a domain name of a source as specified by inet:host. A number of means may be used to resolve hosts. What is important is that such resolutions be consistent with ACLs required by Things to properly operate.

8.2. dst-dnsname

The argument corresponds to a domain name of a destination as specified by inet:host See the previous section relating to resolution.

Note when using either of these with a MUD file, because access is associated with a particular Thing, MUD files MUST NOT contain either a src-dnsname in an ACL associated with from-device-policy or a dst-dnsname associated with to-device-policy.

8.3. The ietf-acldns Model

```
<CODE BEGINS>file "ietf-acldns@2018-06-15.yang"
module ietf-acldns {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-acldns";
  prefix ietf-acldns;

  import ietf-access-control-list {
    prefix acl;
  }
  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF OPSAWG (Ops Area) Working Group";
  contact
    "WG Web: http://tools.ietf.org/wg/opsawg/
    WG List: opsawg@ietf.org
    Author: Eliot Lear
```

```
    lear@cisco.com
    Author: Ralph Droms
    rdroms@gmail.com
    Author: Dan Romascanu
    dromasca@gmail.com
";
description
    "This YANG module defines a component that augments the
    IETF description of an access list to allow DNS names
    as matching criteria.";

revision 2018-06-15 {
    description
        "Base version of dnsname extension of ACL model";
    reference
        "RFC XXXX: Manufacturer Usage Description
        Specification";
}

grouping dns-matches {
    description
        "Domain names for matching.";
    leaf src-dnsname {
        type inet:host;
        description
            "domain name to be matched against";
    }
    leaf dst-dnsname {
        type inet:host;
        description
            "domain name to be matched against";
    }
}

augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" +
"/acl:l3/acl:ipv4/acl:ipv4" {
    description
        "Adding domain names to matching";
    uses dns-matches;
}
augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" +
"/acl:l3/acl:ipv6/acl:ipv6" {
    description
        "Adding domain names to matching";
    uses dns-matches;
}
}
<CODE ENDS>
```

9. MUD File Example

This example contains two access lists that are intended to provide outbound access to a cloud service on TCP port 443.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://lighting.example.com/lightbulb2000",
    "last-update": "2018-03-02T11:20:51+01:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "The BMS Example Lightbulb",
    "from-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6fr"
          }
        ]
      }
    },
    "to-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6to"
          }
        ]
      }
    }
  },
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "mud-76100-v6to",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-todev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:src-dnsname": "test.example.com",
                  "protocol": 6
                },
              },
              "tcp": {
                "ietf-mud:direction-initiated": "from-device",

```



```

        "source-port": {
            "operator": "eq",
            "port": 443
        }
    },
    "actions": {
        "forwarding": "accept"
    }
}
]
}
{
    "name": "mud-76100-v6fr",
    "type": "ipv6-acl-type",
    "aces": {
        "ace": [
            {
                "name": "cl0-frdev",
                "matches": {
                    "ipv6": {
                        "ietf-acldns:dst-dnsname": "test.example.com",
                        "protocol": 6
                    },
                    "tcp": {
                        "ietf-mud:direction-initiated": "from-device",
                        "destination-port": {
                            "operator": "eq",
                            "port": 443
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            }
        ]
    }
}
]
}
}

```

In this example, two policies are declared, one from the Thing and the other to the Thing. Each policy names an access list that applies to the Thing, and one that applies from. Within each access

list, access is permitted to packets flowing to or from the Thing that can be mapped to the domain name of "service.bms.example.com". For each access list, the enforcement point should expect that the Thing initiated the connection.

10. The MUD URL DHCP Option

The IPv4 MUD URL client option has the following format:

```
+-----+-----+-----+
| code | len | MUDstring
+-----+-----+-----+
```

Code `OPTION_MUD_URL_V4` (161) is assigned by IANA. `len` is a single octet that indicates the length of MUD string in octets. The MUD string is defined as follows:

```
MUDstring = mudurl [ " " reserved ]
mudurl = URI; a URL [RFC3986] that uses the "https" schema [RFC7230]
reserved = 1*( OCTET ) ; from [RFC5234]
```

The entire option MUST NOT exceed 255 octets. If a space follows the MUD URL, a reserved string that will be defined in future specifications follows. MUD managers that do not understand this field MUST ignore it.

The IPv6 MUD URL client option has the following format:

```

0          1          2          3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|          OPTION_MUD_URL_V6          |          option-length          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     MUDstring                            |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

`OPTION_MUD_URL_V6` (112; assigned by IANA).

`option-length` contains the length of the MUDstring, as defined above, in octets.

The intent of this option is to provide both a new Thing classifier to the network as well as some recommended configuration to the routers that implement policy. However, it is entirely the purview

of the network system as managed by the network administrator to decide what to do with this information. The key function of this option is simply to identify the type of Thing to the network in a structured way such that the policy can be easily found with existing toolsets.

10.1. Client Behavior

A DHCPv4 client MAY emit a DHCPv4 option and a DHCPv6 client MAY emit DHCPv6 option. These options are singletons, as specified in [RFC7227]. Because clients are intended to have at most one MUD URL associated with them, they may emit at most one MUD URL option via DHCPv4 and one MUD URL option via DHCPv6. In the case where both v4 and v6 DHCP options are emitted, the same URL MUST be used.

10.2. Server Behavior

A DHCP server may ignore these options or take action based on receipt of these options. When a server consumes this option, it will either forward the URL and relevant client information (such as the gateway address or giaddr and requested IP address, and lease length) to a network management system, or it will retrieve the usage description itself by resolving the URL.

DHCP servers may implement MUD functionality themselves or they may pass along appropriate information to a network management system or MUD manager. A DHCP server that does process the MUD URL MUST adhere to the process specified in [RFC2818] and [RFC5280] to validate the TLS certificate of the web server hosting the MUD file. Those servers will retrieve the file, process it, create and install the necessary configuration on the relevant network element. Servers SHOULD monitor the gateway for state changes on a given interface. A DHCP server that does not provide MUD functionality and has forwarded a MUD URL to a MUD manager MUST notify the MUD manager of any corresponding change to the DHCP state of the client (such as expiration or explicit release of a network address lease).

Should the DHCP server fail, in the case when it implements the MUD manager functionality, any backup mechanisms SHOULD include the MUD state, and the server SHOULD resolve the status of clients upon its restart, similar to what it would do, absent MUD manager functionality. In the case where the DHCP server forwards information to the MUD manager, the MUD manager will either make use of redundant DHCP servers for information, or otherwise clear state based on other network information, such as monitoring port status on a switch via SNMP, Radius accounting, or similar mechanisms.

10.3. Relay Requirements

There are no additional requirements for relays.

11. The Manufacturer Usage Description (MUD) URL X.509 Extension

This section defines an X.509 non-critical certificate extension that contains a single Uniform Resource Locator (URL) that points to an on-line Manufacturer Usage Description concerning the certificate subject. URI must be represented as described in Section 7.4 of [RFC5280].

Any Internationalized Resource Identifiers (IRIs) MUST be mapped to URIs as specified in Section 3.1 of [RFC3987] before they are placed in the certificate extension.

The semantics of the URL are defined Section 6 of this document.

The choice of id-pe is based on guidance found in Section 4.2.2 of [RFC5280]:

These extensions may be used to direct applications to on-line information about the issuer or the subject.

The MUD URL is precisely that: online information about the particular subject.

In addition, a separate new extension is defined as id-pe-mudsigner. This contains the subject field of the signing certificate of the MUD file. Processing of this field is specified in Section 13.2.

The purpose of this signature is to make a claim that the MUD file found on the server is valid for a given device, independent of any other factors. There are several security considerations below in Section 16.

A new content-type id-ct-mud is also defined. While signatures are detached today, should a MUD file be transmitted as part of a CMS message, this content-type SHOULD be used.

The new extension is identified as follows:

```
<CODE BEGINS>
  MUDURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
    internet(1) security(5) mechanisms(5) pkix(7)
    id-mod(0) id-mod-mudURLExtn2016(88) }
  DEFINITIONS IMPLICIT TAGS ::= BEGIN
```

```
-- EXPORTS ALL --

IMPORTS

-- RFC 5912
EXTENSION
FROM PKIX-CommonTypes-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkixCommon-02(57) }

-- RFC 5912
id-ct
FROM PKIXCRMF-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-crmf2005-02(55) }

-- RFC 6268
CONTENT-TYPE
FROM CryptographicMessageSyntax-2010
    { iso(1) member-body(2) us(840) rsadsi(113549)
      pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

-- RFC 5912
id-pe, Name
FROM PKIX1Explicit-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-explicit-02(51) } ;

--
-- Certificate Extensions
--

MUDCertExtensions EXTENSION ::=
    { ext-MUDURL | ext-MUDsigner, ... }

ext-MUDURL EXTENSION ::=
    { SYNTAX MUDURLSyntax IDENTIFIED BY id-pe-mud-url }

id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }

MUDURLSyntax ::= IA5String

ext-MUDsigner EXTENSION ::=
    { SYNTAX MUDsignerSyntax IDENTIFIED BY id-pe-mudsigner }
```

```

id-pe-mudsigner OBJECT IDENTIFIER ::= { id-pe TBD1 }

MUDsignerSyntax ::= Name

--
-- CMS Content Types
--

MUDContentTypes CONTENT-TYPE ::=
    { ct-mud, ... }

ct-mud CONTENT-TYPE ::=
    { -- directly include the content
      IDENTIFIED BY id-ct-mudtype }
    -- The binary data that is in the form
    -- 'application/mud+json' is directly encoded as the
    -- signed data. No additional ASN.1 encoding is added.

id-ct-mudtype OBJECT IDENTIFIER ::= { id-ct TBD2 }

END
<CODE ENDS>

```

While this extension can appear in either an 802.AR manufacturer certificate (IDevID) or deployment certificate (LDevID), of course it is not guaranteed in either, nor is it guaranteed to be carried over. It is RECOMMENDED that MUD manager implementations maintain a table that maps a Thing to its MUD URL based on IDevIDs.

12. The Manufacturer Usage Description LLDP extension

The IEEE802.1AB Link Layer Discovery Protocol (LLDP) is a one hop vendor-neutral link layer protocol used by end hosts network Things for advertising their identity, capabilities, and neighbors on an IEEE 802 local area network. Its Type-Length-Value (TLV) design allows for 'vendor-specific' extensions to be defined. IANA has a registered IEEE 802 organizationally unique identifier (OUI) defined as documented in [RFC7042]. The MUD LLDP extension uses a subtype defined in this document to carry the MUD URL.

The LLDP vendor specific frame has the following format:

TLV Type	len	OUI	subtype	MUDString
=127 (7 bits)	(9 bits)	= 00 00 5E (3 octets)	= 1 (1 octet)	(1-255 octets)

where:

- o TLV Type = 127 indicates a vendor-specific TLV
- o len - indicates the TLV string length
- o OUI = 00 00 5E is the organizationally unique identifier of IANA
- o subtype = 1 (to be assigned by IANA for the MUD URL)
- o MUD URL - the length MUST NOT exceed 255 octets

The intent of this extension is to provide both a new Thing classifier to the network as well as some recommended configuration to the routers that implement policy. However, it is entirely the purview of the network system as managed by the network administrator to decide what to do with this information. The key function of this extension is simply to identify the type of Thing to the network in a structured way such that the policy can be easily found with existing toolsets.

Hosts, routers, or other network elements that implement this option are intended to have at most one MUD URL associated with them, so they may transmit at most one MUD URL value.

Hosts, routers, or other network elements that implement this option may ignore these options or take action based on receipt of these options. For example they may fill in information in the respective extensions of the LLDP Management Information Base (LLDP MIB). LLDP operates in a one-way direction. LLDPDUs are not exchanged as information requests by one Thing and response sent by another Thing. The other Things do not acknowledge LLDP information received from a Thing. No specific network behavior is guaranteed. When a Thing consumes this extension, it may either forward the URL and relevant remote Thing information to a MUD manager, or it will retrieve the usage description by resolving the URL in accordance with normal HTTP semantics.

13. Creating and Processing of Signed MUD Files

Because MUD files contain information that may be used to configure network access lists, they are sensitive. To ensure that they have not been tampered with, it is important that they be signed. We make use of DER-encoded Cryptographic Message Syntax (CMS) [RFC5652] for this purpose.

13.1. Creating a MUD file signature

A MUD file MUST be signed using CMS as an opaque binary object. In order to make successful verification more likely, intermediate certificates SHOULD be included. The signature is stored at the location specified in the MUD file. Signatures are transferred using content-type "application/pkcs7-signature".

For example:

```
% openssl cms -sign -signer mancertfile -inkey mankey \  
-in mudfile -binary -outform DER -binary \  
-certfile intermediatecert -out mudfile.p7s
```

Note: A MUD file may need to be re-signed if the signature expires.

13.2. Verifying a MUD file signature

Prior to processing the rest of a MUD file, the MUD manager MUST retrieve the MUD signature file by retrieving the value of "mud-signature" and validating the signature across the MUD file. The Key Usage Extension in the signing certificate MUST be present and have the bit digitalSignature(0) set. When the id-pe-mudsigner extension is present in a device's X.509 certificate, the MUD signature file MUST have been generated by a certificate whose subject matches the contents of that id-pe-mudsigner extension. If these conditions are not met, or if it cannot validate the chain of trust to a known trust anchor, the MUD manager MUST cease processing the MUD file until an administrator has given approval.

The purpose of the signature on the file is to assign accountability to an entity, whose reputation can be used to guide administrators on whether or not to accept a given MUD file. It is already common place to check web reputation on the location of a server on which a file resides. While it is likely that the manufacturer will be the signer of the file, this is not strictly necessary, and may not be desirable. For one thing, in some environments, integrators may install their own certificates. For another, what is more important is the accountability of the recommendation, and not just the relationship between the Thing and the file.

An example:

```
% openssl cms -verify -in mudfile.p7s -inform DER -content mudfile
```

Note the additional step of verifying the common trust root.

14. Extensibility

One of our design goals is to see that MUD files are able to be understood by as broad a cross-section of systems as is possible. Coupled with the fact that we have also chosen to leverage existing mechanisms, we are left with no ability to negotiate extensions and a limited desire for those extensions in any event. A such, a two-tier extensibility framework is employed, as follows:

1. At a coarse grain, a protocol version is included in a MUD URL. This memo specifies MUD version 1. Any and all changes are entertained when this version is bumped. Transition approaches between versions would be a matter for discussion in future versions.
2. At a finer grain, only extensions that would not incur additional risk to the Thing are permitted. Specifically, adding nodes to the mud container is permitted with the understanding that such additions will be ignored by unaware implementations. Any such extensions SHALL be standardized through the IETF process, and MUST be named in the "extensions" list. MUD managers MUST ignore YANG nodes they do not understand and SHOULD create an exception to be resolved by an administrator, so as to avoid any policy inconsistencies.

15. Deployment Considerations

Because MUD consists of a number of architectural building blocks, it is possible to assemble different deployment scenarios. One key aspect is where to place policy enforcement. In order to protect the Thing from other Things within a local deployment, policy can be enforced on the nearest switch or access point. In order to limit unwanted traffic within a network, it may also be advisable to enforce policy as close to the Internet as possible. In some circumstances, policy enforcement may not be available at the closest hop. At that point, the risk of lateral infection (infection of devices that reside near one another) is increased to the number of Things that are able to communicate without protection.

A caution about some of the classes: admission of a Thing into the "manufacturer" and "same-manufacturer" class may have impact on access of other Things. Put another way, the admission may grow the access-list on switches connected to other Things, depending on how access is managed. Some care should be given on managing that access-list growth. Alternative methods such as additional network segmentation can be used to keep that growth within reason.

Because as of this writing MUD is a new concept, one can expect a great many devices to not have implemented it. It remains a local deployment decision as to whether a device that is first connected should be allowed broad or limited access. Furthermore, as mentioned in the introduction, a deployment may choose to ignore a MUD policy in its entirety, but simply taken into account the MUD URL as a classifier to be used as part of a local policy decision.

Finally, please see directly below regarding device lifetimes and use of domain names.

16. Security Considerations

Based on how a MUD URL is emitted, a Thing may be able to lie about what it is, thus gaining additional network access. This can happen in a number of ways when a device emits a MUD URL using DHCP or LLDP, such as being inappropriately admitted to a class such as "same-manufacturer", given access to a device such as "my-controller", or being permitted access to an Internet resource, where such access would otherwise be disallowed. Whether that is the case will depend on the deployment. Implementations SHOULD be configurable to disallow additive access for devices using MUD-URLs that are not emitted in a secure fashion such as in a certificate. Similarly, implementations SHOULD NOT grant elevated permissions (beyond those of devices presenting no MUD policy) to devices which do not strongly bind their identity to their L2/L3 transmissions. When insecure methods are used by the MUD Manager, the classes SHOULD NOT contain devices that use both insecure and secure methods, in order to prevent privilege escalation attacks, and MUST NOT contain devices with the same MUD-URL that are derived from both strong and weak authentication methods.

Devices may forge source (L2/L3) information. Deployments should apply appropriate protections to bind communications to the authentication that has taken place. For 802.1X authentication, IEEE 802.1AE (MACsec) [IEEE8021AE] is one means by which this may happen. A similar approach can be used with 802.11i (WPA2) [IEEE80211i]. Other means are available with other lower layer technologies. Implementations using session-oriented access that is not cryptographically bound should take care to remove state when any form of break in the session is detected.

A rogue CA may sign a certificate that contains the same subject name as is listed in the MUDsigner field in the manufacturer certificate, thus seemingly permitting a substitute MUD file for a device. There are two mitigations available: first, if the signer changes, this may be flagged as an exception by the MUD manager. If the MUD file also changes, the MUD manager SHOULD seek administrator approval (it

should do this in any case). In all circumstances, the MUD manager MUST maintain a cache of trusted CAs for this purpose. When such a rogue is discovered, it SHOULD be removed.

Additional mitigations are described below.

When certificates are not present, Things claiming to be of a certain manufacturer SHOULD NOT be included in that manufacturer grouping without additional validation of some form. This will be relevant when the MUD manager makes use of primitives such as "manufacturer" for the purpose of accessing Things of a particular type. Similarly, network management systems may be able to fingerprint the Thing. In such cases, the MUD URL can act as a classifier that can be proven or disproven. Fingerprinting may have other advantages as well: when 802.1AR certificates are used, because they themselves cannot change, fingerprinting offers the opportunity to add artifacts to the MUD string in the form of the reserved field discussed in Section 10. The meaning of such artifacts is left as future work.

MUD managers SHOULD NOT accept a usage description for a Thing with the same MAC address that has indicated a change of the URL authority without some additional validation (such as review by a network administrator). New Things that present some form of unauthenticated MUD URL SHOULD be validated by some external means when they would be given increased network access.

It may be possible for a rogue manufacturer to inappropriately exercise the MUD file parser, in order to exploit a vulnerability. There are three recommended approaches to address this threat. The first is to validate that the signer of the MUD file is known to and trusted by the MUD manager. The second is to have a system do a primary scan of the file to ensure that it is both parseable and believable at some level. MUD files will likely be relatively small, to start with. The number of ACEs used by any given Thing should be relatively small as well. It may also be useful to limit retrieval of MUD URLs to only those sites that are known to have decent web or domain reputations.

Use of a URL necessitates the use of domain names. If a domain name changes ownership, the new owner of that domain may be able to provide MUD files that MUD managers would consider valid. There are a few approaches that can mitigate this attack. First, MUD managers SHOULD cache certificates used by the MUD file server. When a new certificate is retrieved for whatever reason, the MUD manager should check to see if ownership of the domain has changed. A fair programmatic approximation of this is when the name servers for the domain have changed. If the actual MUD file has changed, the MUD manager MAY check the WHOIS database to see if registration ownership

of a domain has changed. If a change has occurred, or if for some reason it is not possible to determine whether ownership has changed, further review may be warranted. Note, this remediation does not take into account the case of a Thing that was produced long ago and only recently fielded, or the case where a new MUD manager has been installed.

The release of a MUD URL by a Thing reveals what the Thing is, and provides an attacker with guidance on what vulnerabilities may be present.

While the MUD URL itself is not intended to be unique to a specific Thing, the release of the URL may aid an observer in identifying individuals when combined with other information. This is a privacy consideration.

In addressing both of these concerns, implementors should take into account what other information they are advertising through mechanisms such as mDNS[RFC6872], how a Thing might otherwise be identified, perhaps through how it behaves when it is connected to the network, whether a Thing is intended to be used by individuals or carry personal identifying information, and then apply appropriate data minimization techniques. One approach is to make use of TEAP [RFC7170] as the means to share information with authorized components in the network. Network elements may also assist in limiting access to the MUD URL through the use of mechanisms such as DHCPv6-Shield [RFC7610].

There is the risk of the MUD manager itself being spied on to determine what things are connected to the network. To address this risk, MUD managers may choose to make use of TLS proxies that they trust that would aggregate other information.

Please note that the security considerations mentioned in Section 4.7 of [I-D.ietf-netmod-rfc6087bis] are not applicable in this case because the YANG serialization is not intended to be accessed via NETCONF. However, for those who try to instantiate this model in a network element via NETCONF, all objects in each model in this draft exhibit similar security characteristics as [I-D.ietf-netmod-acl-model]. The basic purpose of MUD is to configure access, and so by its very nature can be disruptive if used by unauthorized parties.

17. IANA Considerations

[There was originally a registry entry for .well-known suffixes. This has been removed from the draft and may be marked as deprecated in the registry. RFC Editor: please remove this comment.]

17.1. YANG Module Registrations

The following YANG modules are requested to be registered in the "IANA Module Names" registry:

The ietf-mud module:

- o Name: ietf-mud
- o URN: urn:ietf:params:xml:ns:yang:ietf-mud
- o Prefix: ietf-mud
- o Registrant contact: The IESG
- o Reference: [RFCXXXX]

The ietf-acldns module:

- o Name: ietf-acldns
- o URI: urn:ietf:params:xml:ns:yang:ietf-acldns
- o Prefix: ietf-acldns
- o Registrant: the IESG
- o Reference: [RFCXXXX]

17.2. DHCPv4 and DHCPv6 Options

The IANA has allocated option 161 in the Dynamic Host Configuration Protocol (DHCP) and Bootstrap Protocol (BOOTP) Parameters registry for the MUD DHCPv4 option, and option 112 for DHCPv6, as described in Section 10.

17.3. PKIX Extensions

IANA is kindly requested to make the following assignments for:

- o The MUDURLExtnModule-2016 ASN.1 module in the "SMI Security for PKIX Module Identifier" registry (1.3.6.1.5.5.7.0).
- o id-pe-mud-url object identifier from the "SMI Security for PKIX Certificate Extension" registry (1.3.6.1.5.5.7.1).
- o id-pe-mudsigner object identifier from the "SMI Security for PKIX Certificate Extension" registry (TBD1).

- o id-ct-mudtype object identifier from the "SMI Security for S/MIME CMS Content Type" registry (TBD2).

The use of these values is specified in Section 11.

17.4. MIME Media-type Registration for MUD files

The following media-type is defined for transfer of MUD file:

- o Type name: application
- o Subtype name: mud+json
- o Required parameters: n/a
- o Optional parameters: n/a
- o Encoding considerations: 8bit; application/mud+json values are represented as a JSON object; UTF-8 encoding MUST be employed. [RFC3629]
- o Security considerations: See Security Considerations of RFCXXXX and [RFC8259] Section 12.
- o Interoperability considerations: n/a
- o Published specification: [RFCXXXX]
- o Applications that use this media type: MUD managers as specified by [RFCXXXX].
- o Fragment identifier considerations: n/a
- o Additional information:
 - Magic number(s): n/a
 - File extension(s): n/a
 - Macintosh file type code(s): n/a
- o Person & email address to contact for further information: Eliot Lear <lear@cisco.com>, Ralph Droms <rdroms@gmail.com>
- o Intended usage: COMMON
- o Restrictions on usage: none
- o Author:
 - Eliot Lear <lear@cisco.com>
 - Ralph Droms <rdroms@gmail.com>
- o Change controller: IESG
- o Provisional registration? (standards tree only): No.

17.5. LLDP IANA TLV Subtype Registry

IANA is requested to create a new registry for IANA Link Layer Discovery Protocol (LLDP) TLV subtype values. The recommended policy for this registry is Expert Review. The maximum number of entries in the registry is 256.

IANA is required to populate the initial registry with the value:

LLDP subtype value = 1 (All the other 255 values should be initially marked as 'Unassigned'.)

Description = the Manufacturer Usage Description (MUD) Uniform Resource Locator (URL)

Reference = < this document >

17.6. The MUD Well Known Universal Resource Name (URNs)

The following parameter registry is requested to be added in accordance with [RFC3553]

Registry name: "urn:ietf:params:mud" is requested.
Specification: this document
Repository: this document
Index value: Encoded identically to a TCP/UDP port service name, as specified in Section 5.1 of [RFC6335]

The following entries should be added to the "urn:ietf:params:mud" name space:

"urn:ietf:params:mud:dns" refers to the service specified by [RFC1123]. "urn:ietf:params:mud:ntp" refers to the service specified by [RFC5905].

17.7. Extensions Registry

The IANA is requested to establish a registry of extensions as follows:

Registry name: MUD extensions registry
Registry policy: Standards action
Standard reference: document
Extension name: UTF-8 encoded string, not to exceed 40 characters.

Each extension MUST follow the rules specified in this specification. As is usual, the IANA issues early allocations based in accordance with [RFC7120].

18. Acknowledgments

The authors would like to thank Einar Nilsen-Nygaard, who singlehandedly updated the model to match the updated ACL model, Bernie Volz, Tom Gindin, Brian Weis, Sandeep Kumar, Thorsten Dahm, John Bashinski, Steve Rich, Jim Bieda, Dan Wing, Joe Clarke, Henk Birkholz, Adam Montville, Jim Schaad, and Robert Sparks for their valuable advice and reviews. Russ Housley entirely rewrote

Section 11 to be a complete module. Adrian Farrel provided the basis for privacy considerations text. Kent Watsen provided a thorough review of the architecture and the YANG model. The remaining errors in this work are entirely the responsibility of the authors.

19. References

19.1. Normative References

- [I-D.ietf-netmod-acl-model]
Jethanandani, M., Huang, L., Agarwal, S., and D. Blair,
"Network Access Control List (ACL) YANG Data Model",
draft-ietf-netmod-acl-model-19 (work in progress), April
2018.
- [IEEE8021AB]
Institute for Electrical and Electronics Engineers, "IEEE
Standard for Local and Metropolitan Area Networks--
Station and Media Access Control Connectivity Discovery",
n.d..
- [RFC1123] Braden, R., Ed., "Requirements for Internet Hosts -
Application and Support", STD 3, RFC 1123,
DOI 10.17487/RFC1123, October 1989,
<<https://www.rfc-editor.org/info/rfc1123>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol",
RFC 2131, DOI 10.17487/RFC2131, March 1997,
<<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818,
DOI 10.17487/RFC2818, May 2000,
<<https://www.rfc-editor.org/info/rfc2818>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins,
C., and M. Carney, "Dynamic Host Configuration Protocol
for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July
2003, <<https://www.rfc-editor.org/info/rfc3315>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO
10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November
2003, <<https://www.rfc-editor.org/info/rfc3629>>.

- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <<https://www.rfc-editor.org/info/rfc3748>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <<https://www.rfc-editor.org/info/rfc3987>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, DOI 10.17487/RFC5911, June 2010, <<https://www.rfc-editor.org/info/rfc5911>>.
- [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <<https://www.rfc-editor.org/info/rfc5912>>.

- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <<https://www.rfc-editor.org/info/rfc7120>>.
- [RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014, <<https://www.rfc-editor.org/info/rfc7227>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7610] Gont, F., Liu, W., and G. Van de Velde, "DHCPv6-Shield: Protecting against Rogue DHCPv6 Servers", BCP 199, RFC 7610, DOI 10.17487/RFC7610, August 2015, <<https://www.rfc-editor.org/info/rfc7610>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/info/rfc7951>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", RFC 8348, DOI 10.17487/RFC8348, March 2018, <<https://www.rfc-editor.org/info/rfc8348>>.

19.2. Informative References

- [FW95] Chapman, D. and E. Zwicky, "Building Internet Firewalls", January 1995.
- [I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-20 (work in progress), March 2018.
- [IEEE80211i]
Institute for Electrical and Electronics Engineers, "IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11-Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications- Amendment 6- Medium Access Control (MAC) Security Enhancements", 2004.
- [IEEE8021AE]
Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks- Media Access Control (MAC) Security", 2006.
- [IEEE8021AR]
Institute for Electrical and Electronics Engineers, "Secure Device Identity", 1998.
- [IEEE8021X]
Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks--Port-Based Network Access Control", 2010.

- [ISO.8601.1988] International Organization for Standardization, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, June 1988.
- [RFC1984] IAB and IESG, "IAB and IESG Statement on Cryptographic Technology and the Internet", BCP 200, RFC 1984, DOI 10.17487/RFC1984, August 1996, <<https://www.rfc-editor.org/info/rfc1984>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <<https://www.rfc-editor.org/info/rfc3553>>.
- [RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <<https://www.rfc-editor.org/info/rfc6092>>.
- [RFC6872] Gurbani, V., Ed., Burger, E., Ed., Anjali, T., Abdelnur, H., and O. Festor, "The Common Log Format (CLF) for the Session Initiation Protocol (SIP): Framework and Information Model", RFC 6872, DOI 10.17487/RFC6872, February 2013, <<https://www.rfc-editor.org/info/rfc6872>>.
- [RFC7042] Eastlake 3rd, D. and J. Abley, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 7042, DOI 10.17487/RFC7042, October 2013, <<https://www.rfc-editor.org/info/rfc7042>>.
- [RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <<https://www.rfc-editor.org/info/rfc7170>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.

- [RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <<https://www.rfc-editor.org/info/rfc7452>>.
- [RFC7488] Boucadair, M., Penno, R., Wing, D., Patil, P., and T. Reddy, "Port Control Protocol (PCP) Server Selection", RFC 7488, DOI 10.17487/RFC7488, March 2015, <<https://www.rfc-editor.org/info/rfc7488>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.

Appendix A. Changes from Earlier Versions

RFC Editor to remove this section prior to publication.

Draft -19: * Edits after discussion with apps area to address reserved field for the future. * Correct systeminfo to be utf8. * Remove "hardware-rev" from list.

Draft -18: * Correct an error in the augment statement * Changes to the ACL model re ports.

Draft -17:

- o One editorial.

Draft -16

- o add mud-signature element based on review comments
- o redo mud-url
- o make clear that systeminfo uses UTF8

Draft -13 to -14:

- o Final WGLC comments and review comments
- o Move version from MUD-URL to Model
- o Have MUD-URL in model
- o Update based on update to draft-ietf-netmod-acl-model
- o Point to tree diagram draft instead of 6087bis.

Draft -12 to -13:

- o Additional WGLC comments

Draft -10 to -12:

These are based on WGLC comments:

- o Correct examples based on ACL model changes.
- o Change ordering nodes.
- o Additional explanatory text around systeminfo.
- o Change ordering in examples.
- o Make it VERY VERY VERY VERY clear that these are recommendations, not mandates.
- o DHCP -> NTP in some of the intro text.
- o Remove masa-server
- o "Things" to "network elements" in a few key places.
- o Reference to JSON YANG RFC added.

Draft -10 to -11:

- o Example corrections
- o Typo
- o Fix two lists.
- o Addition of 'any-acl' and 'mud-acl' in the list of allowed features.
- o Clarification of what should be in a MUD file.

Draft -09 to -10:

- o AD input.
- o Correct dates.
- o Add compliance sentence as to which ACL module features are implemented.

Draft -08 to -09:

- o Resolution of Security Area review, IoT directorate review, GenART review, YANG doctors review.
- o change of YANG structure to address mandatory nodes.
- o Terminology cleanup.
- o specify out extra portion of MUD-URL.
- o consistency changes.
- o improved YANG descriptions.
- o Remove extra revisions.
- o Track ACL model changes.
- o Additional cautions on use of ACL model; further clarifications on extensions.

Draft -07 to -08:

- o a number of editorials corrected.
- o definition of MUD file tweaked.

Draft -06 to -07:

- o Examples updated.
- o Additional clarification for direction-initiated.
- o Additional implementation guidance given.

Draft -06 to -07:

- o Update models to match new ACL model
- o extract directionality from the ACL, introducing a new device container.

Draft -05 to -06:

- o Make clear that this is a component architecture (Polk and Watson)
- o Add order of operations (Watson)

- o Add extensions leaf-list (Pritikin)
- o Remove previous-mud-file (Watson)
- o Modify text in last-update (Watson)
- o Clarify local networks (Weis, Watson)
- o Fix contact info (Watson)
- o Terminology clarification (Weis)
- o Advice on how to handle LDevIDs (Watson)
- o Add deployment considerations (Watson)
- o Add some additional text about fingerprinting (Watson)
- o Appropriate references to 6087bis (Watson)
- o Change systeminfo to a URL to be referenced (Lear)

Draft -04 to -05: * syntax error correction

Draft -03 to -04: * Re-add my-controller

Draft -02 to -03: * Additional IANA updates * Format correction in YANG. * Add reference to TEAP.

Draft -01 to -02: * Update IANA considerations * Accept Russ Housley rewrite of X.509 text * Include privacy considerations text * Redo the URL limit. Still 255 bytes, but now stated in the URL definition. * Change URI registration to be under urn:ietf:params

Draft -00 to -01: * Fix cert trust text. * change supportInformation to meta-info * Add an informational element in. * add urn registry and create first entry * add default elements

Appendix B. Default MUD nodes

What follows is the portion of a MUD file that permits DNS traffic to a controller that is registered with the URN "urn:ietf:params:mud:dns" and traffic NTP to a controller that is registered "urn:ietf:params:mud:ntp". This is considered the default behavior and the ACEs are in effect appended to whatever other "ace" entries that a MUD file contains. To block DNS or NTP one repeats the matching statement but replaces the "forwarding" action "accept" with "drop". Because ACEs are processed in the order they are

received, the defaults would not be reached. A MUD manager might further decide to optimize to simply not include the defaults when they are overridden.

Four "acl" list entries that implement default MUD nodes are listed below. Two are for IPv4 and two are for IPv6 (one in each direction for both versions of IP). Note that neither access-list name nor ace name need be retained or used in any way by local implementations, but are simply there for completeness' sake.

```
"ietf-access-control-list:acls": {
  "acl": [
    {
      "name": "mud-59776-v4to",
      "type": "ipv4-acl-type",
      "aces": {
        "ace": [
          {
            "name": "ent0-todev",
            "matches": {
              "ietf-mud:mud": {
                "controller": "urn:ietf:params:mud:dns"
              },
              "ipv4": {
                "protocol": 17
              },
              "udp": {
                "source-port": {
                  "operator": "eq",
                  "port": 53
                }
              }
            },
            "actions": {
              "forwarding": "accept"
            }
          },
          {
            "name": "ent1-todev",
            "matches": {
              "ietf-mud:mud": {
                "controller": "urn:ietf:params:mud:ntp"
              },
              "ipv4": {
                "protocol": 17
              },
              "udp": {
                "source-port": {
```

```

        "operator": "eq",
        "port": 123
    }
}
},
"actions": {
    "forwarding": "accept"
}
}
]
}
},
{
    "name": "mud-59776-v4fr",
    "type": "ipv4-acl-type",
    "aces": {
        "ace": [
            {
                "name": "ent0-frdev",
                "matches": {
                    "ietf-mud:mud": {
                        "controller": "urn:ietf:params:mud:dns"
                    },
                    "ipv4": {
                        "protocol": 17
                    },
                    "udp": {
                        "destination-port": {
                            "operator": "eq",
                            "port": 53
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            },
            {
                "name": "ent1-frdev",
                "matches": {
                    "ietf-mud:mud": {
                        "controller": "urn:ietf:params:mud:ntp"
                    },
                    "ipv4": {
                        "protocol": 17
                    },
                    "udp": {
                        "destination-port": {

```

```
        "operator": "eq",
        "port": 123
      }
    },
    "actions": {
      "forwarding": "accept"
    }
  ]
}
},
{
  "name": "mud-59776-v6to",
  "type": "ipv6-acl-type",
  "aces": {
    "ace": [
      {
        "name": "ent0-todev",
        "matches": {
          "ietf-mud:mud": {
            "controller": "urn:ietf:params:mud:dns"
          },
          "ipv6": {
            "protocol": 17
          },
          "udp": {
            "source-port": {
              "operator": "eq",
              "port": 53
            }
          }
        },
        "actions": {
          "forwarding": "accept"
        }
      },
      {
        "name": "ent1-todev",
        "matches": {
          "ietf-mud:mud": {
            "controller": "urn:ietf:params:mud:ntp"
          },
          "ipv6": {
            "protocol": 17
          },
          "udp": {
            "source-port": {
```

```
        "operator": "eq",
        "port": 123
      }
    },
    "actions": {
      "forwarding": "accept"
    }
  ]
}
},
{
  "name": "mud-59776-v6fr",
  "type": "ipv6-acl-type",
  "aces": {
    "ace": [
      {
        "name": "ent0-frdev",
        "matches": {
          "ietf-mud:mud": {
            "controller": "urn:ietf:params:mud:dns"
          },
          "ipv6": {
            "protocol": 17
          },
          "udp": {
            "destination-port": {
              "operator": "eq",
              "port": 53
            }
          }
        },
        "actions": {
          "forwarding": "accept"
        }
      },
      {
        "name": "ent1-frdev",
        "matches": {
          "ietf-mud:mud": {
            "controller": "urn:ietf:params:mud:ntp"
          },
          "ipv6": {
            "protocol": 17
          },
          "udp": {
            "destination-port": {
```

```

        "operator": "eq",
        "port": 123
      }
    },
    "actions": {
      "forwarding": "accept"
    }
  ]
}

```

Appendix C. A Sample Extension: DETNET-indicator

In this sample extension we augment the core MUD model to indicate whether the device implements DETNET. If a device claims not to use DETNET, but then later attempts to do so, a notification or exception might be generated. Note that this example is intended only for illustrative purposes.

Extension Name: "Example-Extension" (to be used in the extensions list)
 Standard: this document (but do not register the example)

This extension augments the MUD model to include a single node, using the following sample module that has the following tree structure:

```

module: ietf-mud-detext-example
  augment /ietf-mud:mud:
    +--rw is-detnet-required?  boolean

```

The model is defined as follows:

```

<CODE BEGINS>file "ietf-mud-detext-example@2018-06-15.yang"
module ietf-mud-detext-example {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-detext-example";
  prefix ietf-mud-detext-example;

  import ietf-mud {
    prefix ietf-mud;
  }
}

```

```
organization
  "IETF OPSAWG (Ops Area) Working Group";
contact
  "WG Web: http://tools.ietf.org/wg/opsawg/
  WG List: opsawg@ietf.org
  Author: Eliot Lear
  lear@cisco.com
  Author: Ralph Droms
  rdroms@gmail.com
  Author: Dan Romascanu
  dromasca@gmail.com

  ";
description
  "Sample extension to a MUD module to indicate a need
  for DETNET support.";

revision 2018-06-15 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: Manufacturer Usage Description
    Specification";
}

augment "/ietf-mud:mud" {
  description
    "This adds a simple extension for a manufacturer
    to indicate whether DETNET is required by a
    device.";
  leaf is-detnet-required {
    type boolean;
    description
      "This value will equal true if a device requires
      detnet to properly function";
  }
}
}
<CODE ENDS>
```

Using the previous example, we now show how the extension would be expressed:

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://lighting.example.com/lightbulb2000",
    "last-update": "2018-03-02T11:20:51+01:00",
```

```
"cache-validity": 48,
"extensions": [
  "ietf-mud-detext-example"
],
"ietf-mud-detext-example:is-detnet-required": "false",
"is-supported": true,
"systeminfo": "The BMS Example Lightbulb",
"from-device-policy": {
  "access-lists": {
    "access-list": [
      {
        "name": "mud-76100-v6fr"
      }
    ]
  }
},
"to-device-policy": {
  "access-lists": {
    "access-list": [
      {
        "name": "mud-76100-v6to"
      }
    ]
  }
},
"ietf-access-control-list:acls": {
  "acl": [
    {
      "name": "mud-76100-v6to",
      "type": "ipv6-acl-type",
      "aces": {
        "ace": [
          {
            "name": "cl0-todev",
            "matches": {
              "ipv6": {
                "ietf-acldns:src-dnsname": "test.example.com",
                "protocol": 6
              },
            },
            "tcp": {
              "ietf-mud:direction-initiated": "from-device",
              "source-port": {
                "operator": "eq",
                "port": 443
              }
            }
          }
        ]
      }
    }
  ],
}
```

```
        "actions": {
          "forwarding": "accept"
        }
      }
    ]
  },
  {
    "name": "mud-76100-v6fr",
    "type": "ipv6-acl-type",
    "aces": {
      "ace": [
        {
          "name": "cl0-frdev",
          "matches": {
            "ipv6": {
              "ietf-acldns:dst-dnsname": "test.example.com",
              "protocol": 6
            },
            "tcp": {
              "ietf-mud:direction-initiated": "from-device",
              "destination-port": {
                "operator": "eq",
                "port": 443
              }
            }
          },
          "actions": {
            "forwarding": "accept"
          }
        }
      ]
    }
  }
]
```

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Ralph Droms
Google
355 Main St., 5th Floor
Cambridge

Phone: +1 978 376 3731
Email: rdroms@gmail.com

Dan Romascanu

Phone: +972 54 5555347
Email: dromasca@gmail.com

opsawg
Internet-Draft
Intended status: Standards Track
Expires: August 13, 2017

Z. Li, Ed.
R. Gu, Ed.
China Mobile
J. Dong
Huawei Technologies
February 9, 2017

Export BGP community information in IP Flow Information Export (IPFIX)
draft-li-opsawg-ipfix-bgp-community-02

Abstract

This draft specifies an extension to the IPFIX information model defined in [RFC7012] to export the BGP community [RFC1997] information. Three information elements, `bgpCommunity`, `bgpSourceCommunityList` and `bgpDestinationCommunityList`, are introduced in this document to carry the BGP community information. `bgpCommunity`, containing exactly one BGP community value, is used to consist the list in `bgpSourceCommunityList` and `bgpDestinationCommunityList`, which are corresponding to a specific flow's source IP and destination IP respectively.

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 August 13, 2017.

Copyright Notice

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

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

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. BGP Community Information Elements	4
3.1. bgpCommunity	4
3.2. bgpSourceCommunityList	4
3.3. bgpDestinationCommunityList	5
4. Security Considerations	5
5. IANA Considerations	5
6. Acknowledgements	6
7. References	6
7.1. Normative References	6
7.2. Informative References	6
Appendix A. Application Example	7
A.1. Template Record	7
A.2. Data Set	8
Authors' Addresses	9

1. Introduction

IP Flow Information Export (IPFIX) [RFC7011] provides network administrators with traffic flow information using the information elements (IEs) defined in [IANA-IPFIX] registries. Based on the traffic flow information, network administrators know the amount and direction of the traffic in their network, then they can optimize their network when needed. For example, they can steer some flows from the congested links to the low utilised links.

[IANA-IPFIX] has already defined the following IEs for traffic flow information exporting in different grain: sourceIPv4Address, sourceIPv4Prefix, destinationIPv4Address, destinationIPv4Prefix, bgpSourceAsNumber, bgpDestinationAsNumber, bgpNextHopIPv4Address, etc. In some circumstances, however, especially when traffic engineering and optimization are used in the Tier 1 or Tier 2 operators' backbone networks, traffic flow information based on these IEs is not suitable. Flow information based on IP address or IP prefix is much more meticulous. On the contrary, flow information based on AS number is too coarse. BGP community [RFC1997], which describes a group of routes sharing some common properties, is preferably used for fine granularity traffic engineering

[Community-TE] [RFC4384]. Unfortunately, [IANA-IPFIX] has no IE defined for BGP community information, yet.

Flow information based on BGP community can be collected by a mediator defined in [RFC6183]. Mediator is responsible for the correlation between flow information and BGP community. However no IEs are defined in [RFC6183] for exporting BGP community information in IPFIX. Furthermore, to correlate the BGP community with the flow information, mediator needs to learn BGP routes and lookup in the BGP routing table to get the matching entry for the specific flow. Neither BGP route learning nor routing table lookup is trivial for a mediator. Mediator is mainly introduced to release the performance requirement for the exporter [RFC5982]. In fact, to obtain the information for BGP related IEs that have already been defined, such as `bgpSourceAsNumber`, `bgpDestinationAsNumber`, and `bgpNextHopIPv4Address`, etc, exporter has to hold the up-to-date BGP routing table and look up in the BGP routing table. The exporter can get the community information in the same procedure. So, getting BGP community information adds no more requirement for exporter. Some vendors have already implemented this feature in their exporters using private IEs. So, exporter is RECOMMENDED to export the BGP community information in IPFIX directly, other than the mediator.

This draft specifies an extension to the IPFIX information model defined in [RFC7012] to export the BGP community information. Three IEs, `bgpCommunity`, `bgpSourceCommunityList` and `bgpDestinationCommunityList`, are introduced to complete this task. `bgpCommunity` contains one BGP community value. `BgpSourceCommunityList` consists of a list of `bgpCommunity` corresponding with the source IP address of a specific flow, and `bgpDestinationCommunityList` consists of a list of `bgpCommunity` corresponding with the destination IP address of a specific flow.

`BgpCommunity`, `bgpSourceCommunityList` and `bgpDestinationCommunityList` IEs are applicable for both IPv4 and IPv6 traffic. Both exporter and mediator can use these three IEs to export BGP community information in IPFIX.

2. Terminology

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

3. BGP Community Information Elements

In order to export BGP community information along with other flow information defined by IPFIX, we need to introduce three new IEs. One is `bgpCommunity`, which is used to identify that the value in this IE is BGP community [RFC1997]. The other two are `bgpSourceCommunityList` and `bgpDestinationCommunityList`. They both are basicList [RFC6313] of `bgpCommunity`. `bgpSourceCommunityList` and `bgpDestinationCommunityList` are used to export BGP community information corresponding to a specific flow's source IP and destination IP respectively. Flow information based on BGP community can then be accumulated and analysed by the collector or other applications.

The details of these three new introduced IEs are illustrated below, including name, ID, type, semantics, description and units.

3.1. `bgpCommunity`

ElementID	to be assigned by IANA, 458 is suggested
Name	<code>bgpCommunity</code>
Data Type	unsigned32
Data Type Semantics	identifier
Description	BGP community as defined in [RFC1997]
Units	none

Figure 1: `bgpCommunity`

3.2. `bgpSourceCommunityList`

ElementID	to be assigned by IANA, 459 is suggested
Name	bgpSourceCommunityList
Data Type	basicList, as specified in [RFC6313]
Data Type Semantics	list
Description	zero or more BGP communities corresponding with source IP address of a specific flow
Units	none

Figure 2: bgpSourceCommunityList

3.3. bgpDestinationCommunityList

ElementID	to be assigned by IANA, 460 is suggested
Name	bgpDestinationCommunityList
Data Type	basicList, as specified in [RFC6313]
Data Type Semantics	list
Description	zero or more BGP communities corresponding with destination IP address of a specific flow
Units	none

Figure 3: bgpDestinationCommunityList

4. Security Considerations

This document only defines three new IEs for IPFIX. So, this document itself does not directly introduce security issues. The same security considerations as for the IPFIX Protocol Specification [RFC7011] and Information Model [RFC7012] apply.

5. IANA Considerations

This draft specifies three new IPFIX IEs, `bgpCommunity`, `bgpSourceCommunityList` and `bgpDestinationCommunityList`, to export BGP community information along with other flow information.

The Element IDs for these three IEs are solicited to be assigned by IANA. Number 458, 459 and 460 are suggested for `bgpCommunity`, `bgpSourceCommunityList` and `bgpDestinationCommunityList`, respectively.

6. Acknowledgements

The authors would like to thank Benoit Claise and Paul Aitken for discussion and suggestions to promote this document.

7. References

7.1. 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>>.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, DOI 10.17487/RFC6313, July 2011, <<http://www.rfc-editor.org/info/rfc6313>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<http://www.rfc-editor.org/info/rfc7011>>.
- [RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <<http://www.rfc-editor.org/info/rfc7012>>.

7.2. Informative References

- [Community-TE] Shao, W., Devienne, F., Iannone, L., and JL. Rougier, "On the use of BGP communities for fine-grained inbound traffic engineering", Computer Science 27392(1):476-487, November 2015.
- [IANA-IPFIX] "IP Flow Information Export (IPFIX) Entities", <<http://www.iana.org/assignments/ipfix/>>.

- [RFC1997] Chandra, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, DOI 10.17487/RFC1997, August 1996, <<http://www.rfc-editor.org/info/rfc1997>>.
- [RFC4384] Meyer, D., "BGP Communities for Data Collection", BCP 114, RFC 4384, DOI 10.17487/RFC4384, February 2006, <<http://www.rfc-editor.org/info/rfc4384>>.
- [RFC5982] Kobayashi, A., Ed. and B. Claise, Ed., "IP Flow Information Export (IPFIX) Mediation: Problem Statement", RFC 5982, DOI 10.17487/RFC5982, August 2010, <<http://www.rfc-editor.org/info/rfc5982>>.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC 6183, DOI 10.17487/RFC6183, April 2011, <<http://www.rfc-editor.org/info/rfc6183>>.

Appendix A. Application Example

In this section, we give an example to show the encoding format for the three new introduced IEs.

Flow information including BGP communities is shown in the below table. Suppose we want all the fields to be reported by IPFIX.

Source ip	Destination ip	Source BGP community	Destination BGP community
1.1.1.1	2.2.2.2	1:1001,1:1002,8:1001	2:1002,8:1001
3.3.3.3	4.4.4.4	3:1001,3:1002,8:1001	4:1001,8:1001

Figure 4: Flow information including BGP communities

A.1. Template Record

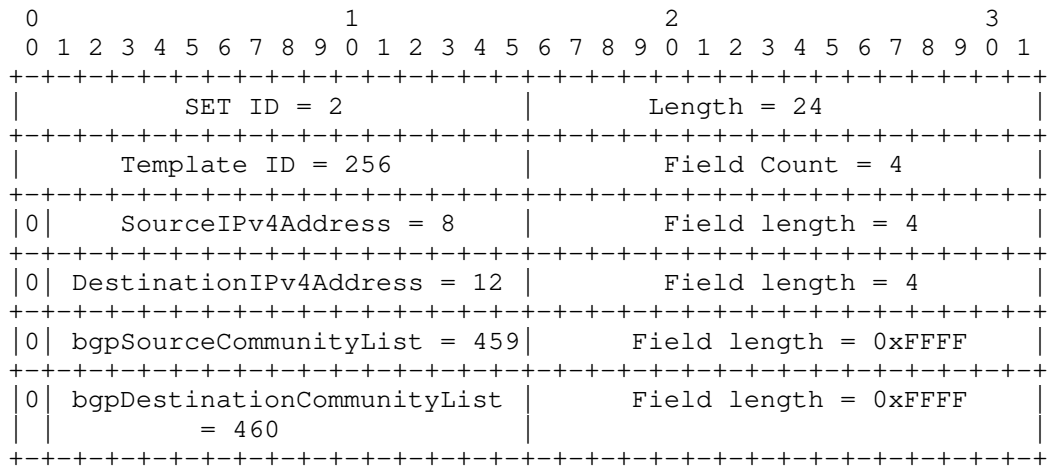
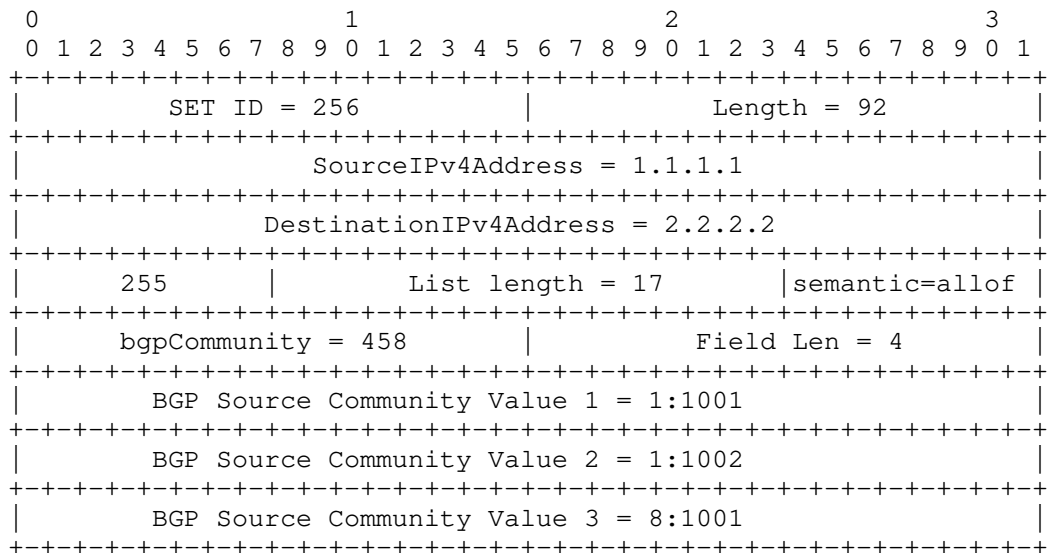


Figure 5: Template Record Encoding Format

In this example, the Template ID is 256, which will be used in the data record. The field length for `bgpSourceCommunityList` and `bgpDestinationCommunityList` is `0xFFFF`, which means the length of this IE is variable, the actual length of this IE is indicated by the list length field in the basic list format as per [RFC6313].

A.2. Data Set

The data set is represented as follows:



```

|      255      |      List length = 13      |semantic =allof|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      bgpCommunity = 458      |      Field Len = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      BGP Destination Community Value 1 = 2:1002      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      BGP Destination Community Value 2 = 8:1001      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      SourceIPv4Address = 3.3.3.3      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      DestinationIPv4Address = 4.4.4.4      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      255      |      List length = 17      |semantic =allof|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      bgpCommunity = 458      |      Field Len = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      BGP Source Community Value 1  = 3:1001      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      BGP Source Community Value 2  = 3:1002      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      BGP Source Community Value 3  = 8:1001      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      255      |      List length = 13      |semantic =allof|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      bgpCommunity = 458      |      Field Len = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      BGP Destination Community Value 1 = 4:1001      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      BGP Destination Community Value 2 = 8:1001      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 6: Data Set Encoding Format

Authors' Addresses

Zhenqiang Li (editor)
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: lizhenqiang@chinamobile.com

Rong Gu (editor)
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: gulong_cmcc@outlook.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Operations and Management Area Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2018

B. Pularikkal
Cisco Systems
T. Pauly
Apple Inc.
M. Grayson
S. Gundavelli
Cisco Systems
S. Touati
Ericsson
July 3, 2017

Carrier Wi-Fi Calling Deployment Considerations
draft-pularikkal-opsawg-wifi-calling-03

Abstract

Carrier Wi-Fi Calling is a solution that allows mobile operators to seamlessly offload mobile voice signaling and bearer traffic onto Wi-Fi access networks, which may or may not be managed by the mobile operators. Mobile data offload onto Wi-Fi access networks has already become very common, as Wi-Fi access has become more ubiquitous. However, the offload of mobile voice traffic onto Wi-Fi networks has become prevalent only in recent years. This was primarily driven by the native Wi-Fi Calling client support introduced by device vendors. The objective of this document is to provide a high level deployment reference to Mobile Operators and Wi-Fi Operators on Carrier Wi-Fi Calling.

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 4, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Terminology	4
3. Architecture Overview	6
4. Wi-Fi Calling Deployment Considerations	8
4.1. Wi-Fi to Packet Core Integration	8
4.1.1. Untrusted Model	8
4.1.1.1. IPSec Tunnel Negotiation	9
4.1.2. Hybrid Model	10
4.1.3. Trusted Model	11
4.1.4. Model Selection Criteria	13
5. Subscriber Onboarding into Wi-Fi Access Network	14
5.1. Authentication and Identity Management	14
5.2. Hotspot 2.0 for Seamless Onboarding	15
5.2.1. Hotspot 2.0 Inter-Operator Roaming for Wi-Fi Calling	17
6. Wi-Fi calling deployment in restrictive networks	17
7. RF Network Performance Optimization	18
7.1. Radio Resource Management	18
7.2. Wi-Fi Roaming Optimization	19
7.2.1. Fast BSS Transition	19
7.2.2. 802.11k based Neighbor Reports	19
7.2.3. 802.11v based Assisted Roaming and Load Balancing	20
8. QoS Deployment Considerations for Wi-Fi Calling	20
8.1. Wi-Fi Access Network QoS	20
8.2. End to End QoS	21
9. Wi-Fi Calling Client Considerations	23
9.1. Access Selection Criteria	23
9.2. Inter-RAT Handover	24
9.3. MTU Considerations	24
9.4. Congestion Management	24
9.5. NAT Traversal	25
10. Acknowledgements	25

11. Informative References	25
Authors' Addresses	26

1. Introduction

There are several SP Managed and Over the Top Voice Solutions deployed today which can leverage Wi-Fi access networks. Some of these solutions rely on standalone applications installed on the Mobile Handset and other Mobile devices such as tablets. Also there are solutions, which leverage dedicated hardware built exclusively to support Voice over Wi-Fi.e.g, in enterprise type environments. The scope of this document is VoWiFi solutions, which are deployed by Mobile Network Operators also known as Wireless Carriers. VoWiFi from the context of Mobile Voice offload is often referred to as Carrier Wi-Fi Calling. The deployment of Carrier Wi-Fi Calling requires some kind of integration between the Wi-Fi Access network and Mobile Packet Core. Carrier Wi-Fi calling solutions deployed today predominantly uses an 'untrusted Wi-Fi' model that delivers simple IP connectivity to facilitate Mobile Packet Core integration. With this 'untrusted' approach, Mobile Operators are able to make use of the existing Wi-Fi deployment footprint regardless of whether it is owned by the MNOs or by their roaming partners or Wi-Fi Operators without any kind of partnership with the MNOs. This model has definitely allowed MNOs to accelerate the adoption of Wi-Fi calling. However, this comes with some caveats, as depending on the Wi-Fi network, there may be no visibility or control over it by the MNO, impacting its ability to carry voice calls without compromising end user experience.

It is in the interest of both MNOs as well as Wi-Fi Operators to improve the quality of experience for Wi-Fi Calling delivered over a Wi-Fi access network. MNOs have the incentive to make sure that the end user experience does not get compromised while the voice service is offloaded over Wi-Fi access. Wi-Fi operators have the business incentive to enter into roaming partnerships with the MNOs and support Wi-Fi calling with certain Service Level Agreements. In some deployments, it is possible for the MNOs to own some Wi-Fi hotspot deployments. In such cases, MNO will effectively be the Wi-Fi operator as well.

Objective of this document is to provide a Carrier Wi-Fi Calling deployment reference to Wi-Fi Operators and MNOs with primary focus on the Wi-Fi Access Network and the Wi-Fi to Packet Core integration aspects.

2. Terminology

Service Provider (SP)

Refers to a provider of telecommunications services such as Broadband Operator or Mobile Operator. An SP may provide several telecommunications services.

APP

Refers to computer program typically designed to run on Mobile devices such as smartphones and tablets.

Wireless Fidelity (Wi-Fi)

Technology that allows devices to wirelessly connect using 2.4 GHz and 5.0 GHz unlicensed radio bands. Wi-Fi is defined as part of IEEE 802.11 standards

Voice over Wi-Fi (VoWiFi)

Any solution, which supports voice services over Wi-Fi.

Mobile Network Operator (MNO)

A wireless communications service provider who owns and operates licensed wireless access network and the backend infrastructure to offer mobile voice, data and multimedia services.

3rd Generation Partnership Project (3GPP)

3GPP unites seven telecommunications standards development organizations known as Organizational Partners and provides their members with a stable environment to produce the reports and specifications that define 3GPP technologies

Global System for Mobile Association (GSMA)

GSMA represents the interests of mobile operators worldwide, uniting nearly 800 operators with more than 250 companies in the broader mobile ecosystem, including handset and device makers, software companies, equipment providers and internet companies, as well as organizations in adjacent industry sectors.

User Equipment (UE)

Term represents any device used directly by an end user to communicate.

Wireless Local Area Network (WLAN)

Refers to IEEE 802.11 based Wi-Fi access networks and represents an extended service set consisting of multiple access points.

Long Term Evolution (LTE)

Is the fourth generation 3GPP standard set for wireless communication of mobile devices in end-to-end IP environment.

Evolved Packet Core (EPC)

Represents the Core Network in the 3GPP LTE system Architecture.

Packet Data Network (PDN)

PDN represents a network in the packet core a Mobile UE device wants to communicate with. PDN generally is mapped to a set of related services.

Access Point Name (APN)

APN represents a set of services available to a specific PDN. Typically UE devices will be configured to access multiple APNs corresponding various services in the packet core.

Trusted WLAN Access Gateway (TWAG)

Performs the gateway function between a trusted WLAN access network and packet core. It acts as the default gateway and DHCP Server for UE devices connected to the WLAN access network for trusted Wi-Fi to packet core integration model.

Evolved Packet Data Gateway (ePDG)

ePDG performs the gateway function between WLAN access network and Mobile Packet core in an untrusted model. Main function of ePDG is to secure the data transmission with a UE connected to the EPC.

PDN Gateway (P-GW)

P-GW is the subscriber session anchor in EPC. It enforces policy and also has a role in IP persistence in roaming scenarios. Based up on the policy, P-GW steers traffic towards various PDN networks corresponding to various APNs.

IP Multi-Media Subsystem (IMS)

An Architectural framework for delivering IP multimedia services.
And is defined in 3GPP

Policy and Charging Rule Function (PCRF)

A system in EPC, which detects service data flows, applies policies and QoS to subscriber flows to and supports flow based charging

Session Initiation Protocol (SIP)

SIP is an application layer control protocol that can establish, modify and terminate multimedia sessions or calls.

Real-time Transport Protocol (RTP)

RTP is a transport protocol, which provides end-to-end delivery services for data with real-time characteristics such as interactive audio and video.

Proxy Mobile IPv6 (PMIPv6)

PMIPv6 is a network based mobility management protocol standardized by IETF and adopted in 3GPP.

GPRS Tunneling Protocol (GTP)

Group of IP based communications protocols used in 3GPP architectures.

S2a Interface

Is the interface between TWAG and P-GW and can be either GTP or PMIPv6 based

S2b Interface

Interface between ePDG and P-GW and can be either GTP or PMIPv6

3. Architecture Overview

This section provides a very high level overview of the end-to-end Architecture for Carrier Wi-Fi Calling. It is outside the scope of this document to provide a detailed Architecture description, as all the functional entities and the protocol interfaces are well defined in the 3GPP and GSMA specifications [3GPPTS23.402, GSMAIR61, GSMAIR51]. Figure-01 below is used to describe the Architecture components at a high level.

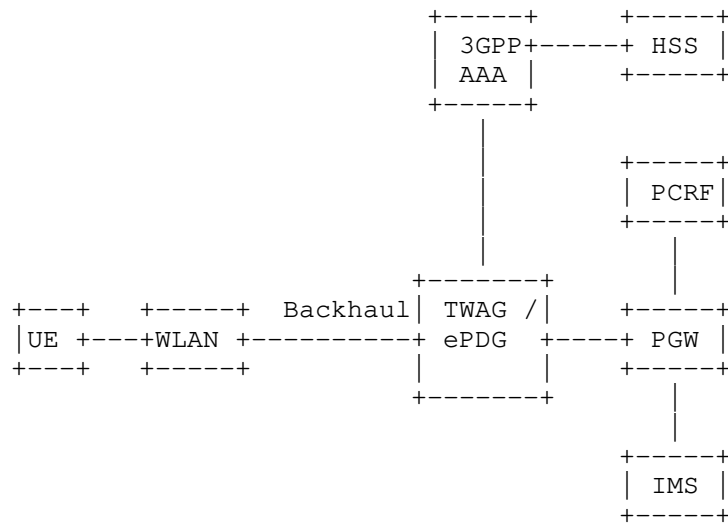


Figure 1: High Level Architecture

The UE is the end user device such as a smartphone running native Wi-Fi Calling client. The UE is connected to a Wi-Fi access network, which is represented by the block WLAN in the diagram. Depending up on the trust model, TWAG or ePDG gateway is used to integrate the WLAN access network to the MNO packet core. More details around this untrusted and trusted approaches are covered in the next section. The P-GW acts as the common anchor for the subscriber sessions regardless of whether the UE is connected to Wi-Fi or LTE (not shown), allowing the preservation of the IP Session during a handover between LTE and Wi-Fi. IMS provides several functions related to SIP based call control signaling, namely SIP authentication, basic telephony services, supplementary services, interworking with other IMS systems, and offload into circuit switched voice networks. In addition to voice, the same IMS infrastructure may be leveraged for other multi-media functions such as video calling. The IMS framework consists of several functional entities and is omitted for the sake of simplicity here. PCRF performs classical Policy and Charging Rule functions in the Mobile Packet Core. For the Wi-Fi calling solution, it will trigger the establishment of the default and dedicated bearers on the S2a or S2b interfaces for SIP and RTP traffic between the PGW and the TWAG/ePDG.

4. Wi-Fi Calling Deployment Considerations

This section covers deployment considerations for an end-to-end Wi-Fi calling Architecture that can influence the quality of experience, availability and monetization aspects of the solution offering.

4.1. Wi-Fi to Packet Core Integration

There are three different Architecture options available for Wi-Fi to Packet Core integration for the deployment of Wi-Fi calling. Each of these models are described in the sub-sections below:

4.1.1. Untrusted Model

This model is built around the assumption that the Wi-Fi access network is 'unmanaged' or untrusted from the MNOs perspective. Since this model does not rely on any security or data privacy implementations on the Wi-Fi access network, it requires the establishment of an IPSec tunnel between the UE device and the Mobile Packet Core. The ePDG gateway acts as the IPSec tunnel termination point on the packet core side. The ePDG handles the user authentication as well as the establishment of an S2b packet data network connection towards the P-GW using the GTP based S2b interface. This Architecture model is illustrated in figure-2 below.

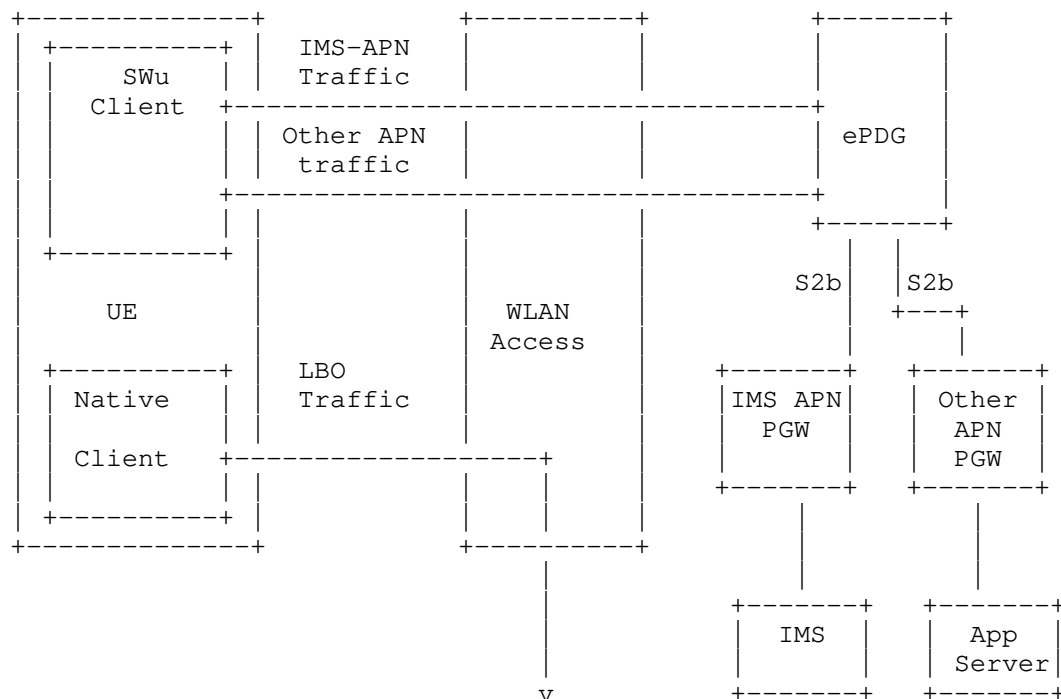


Figure 2: Untrusted Wi-Fi to Packet Core Integration Model for Wi-Fi Calling

The Wi-Fi calling client implementation uses the ePDG client for IMS APN while the default PDN or Internet APN traffic is locally offloaded (Local Breakout LBO) into the Wi-Fi access network. The "untrusted Wi-Fi" architecture supports multiple APN over SWu, allowing the MNO to also route specific applications traffic associated with one or more APN through the Packet Core, in addition to the IMS APN, if required.

4.1.1.1. IPSec Tunnel Negotiation

The IPSec tunnel from the UE to the ePDG is negotiated using IKEv2. The parameters for tunnel negotiation in Wi-Fi Calling are as follows:

- o The Initiator Identifier (IDi) will be in ID_RFC822_ADDR (email address) form, and be based on the UE's IMSI@Realm.

- o The Responder Identifier (IDr) will be in ID_FQDN form, and be the APN name that the tunnel should access through the ePDG.
- o EAP should be used for mutual authentication. When on a device with a SIM card, EAP-AKA should be used. On other devices, EAP-TLS is preferred. EAP-Only authentication (in which the server certificate is not sent in an CERT payload) may be used to reduce packet size, but only with mutually authenticating EAP types such as EAP-AKA or EAP-TLS.
- o Strong encryption and authentication algorithms should be used, such as ENCR_AES_CBC, PRF_HMAC_SHA2_256, AUTH_HMAC_SHA2_256_128, and Diffie-Hellman Group 14.
- o The Configuration Request should specify an IPv4 or IPv6 addresses used for handover. The UE may also request ePDG-specific attributes such as P_CSCF_IP4_ADDRESS and P_CSCF_IP6_ADDRESS.

4.1.2. Hybrid Model

3GPP TS 23.402 also defines the concept of "trusted Wi-Fi" architecture, providing another method to integrate with the packet core. The trustworthiness of an access network itself is left to the MNO to decide, but it generally relies on some level of control by the MNO of the Wi-Fi access network either in a direct or indirect manner. One of the key characteristics of the "Trusted Wi-Fi" architecture as defined in 3GPP Release 11, is the client-less approach to support the packet core integration. This solution lacked the support for multiple APNs signaling for the UE when over the Wi-Fi access network, therefore all Wi-Fi offloaded traffic was assumed to be part of the default PDN or Internet APN. With this limitation, Wi-Fi calling cannot be supported as it requires its own IMS APN. The hybrid architecture proposed here combines the 3GPP release 11 "trusted Wi-Fi" architecture, with the ePDG based untrusted Wi-Fi architecture. This hybrid model simultaneously supports IMS and other applications specific APNs using the untrusted Wi-Fi model, with the TWAG selectively offloading their traffic, while using the S2a interface for all other default PDN traffic toward the default PGW. This Architecture model is illustrated in figure 3 below

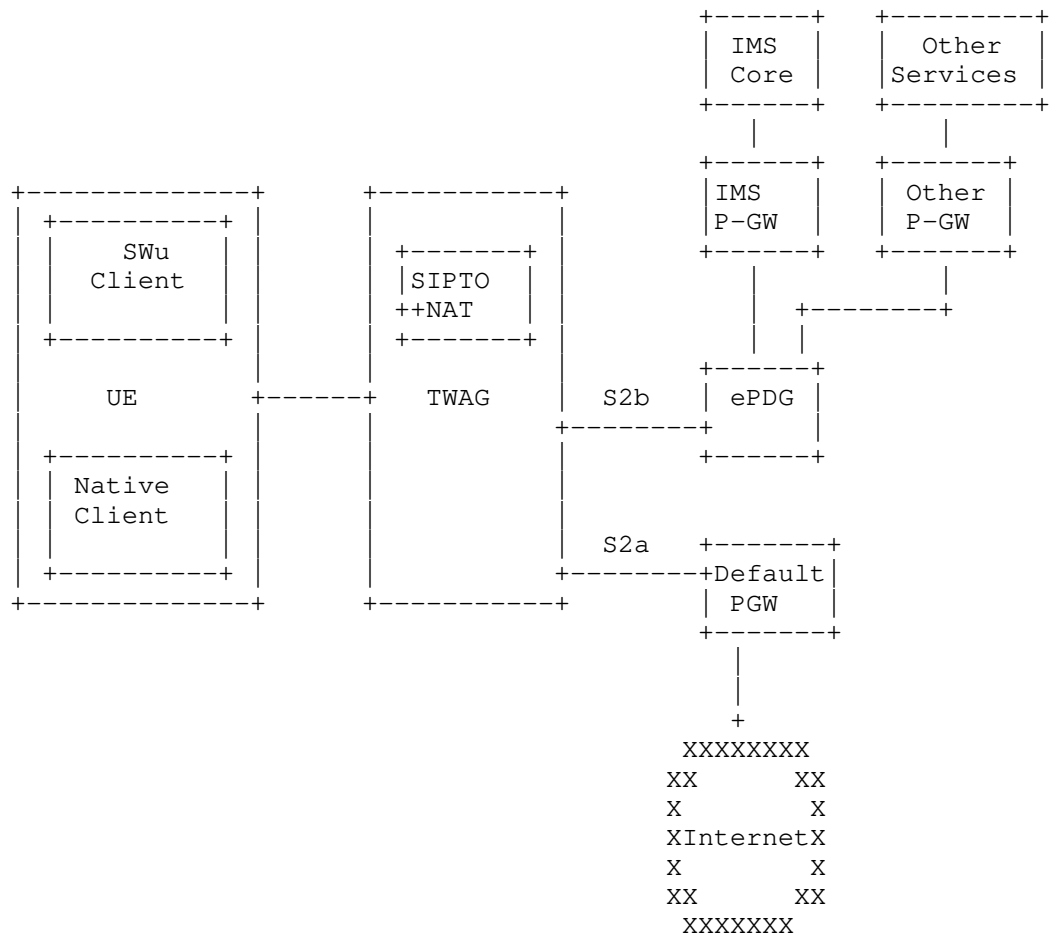


Figure 3: Hybrid Wi-Fi to Packet Core integration model for Wi-Fi calling

4.1.3. Trusted Model

Enhancements introduced in 3GPP release 12 SaMOC specifications provides the ability to support multiple APN over Wi-Fi access making the support of Wi-Fi calling, and other applications specific APNs possible without the need for IPSec connectivity between the UE and the Packet core. This Architecture model is illustrated in figure 4 below

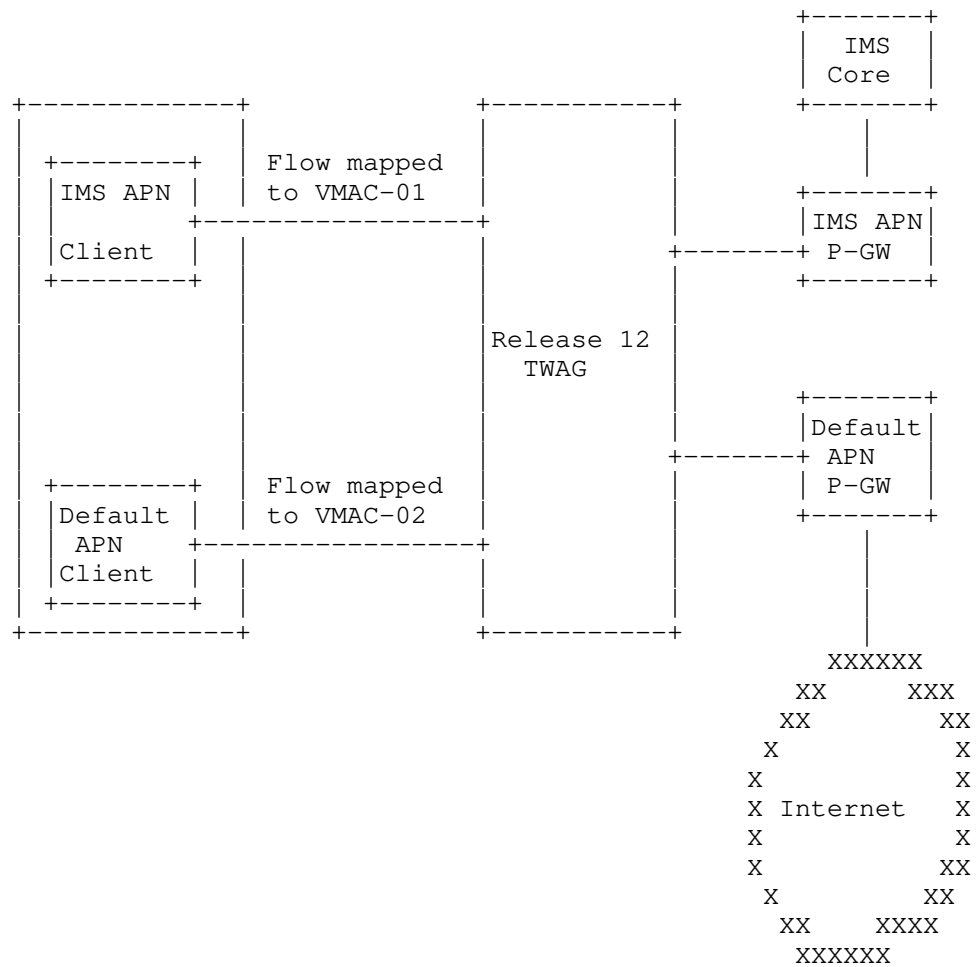


Figure 4: Trusted Wi-Fi to Packet Core integration model for Wi-Fi calling

4.1.4. Model Selection Criteria

Each of the Wi-Fi to Packet Core Architecture models described in the previous sections comes with its own pros and cons. And selection of a specific architecture model depends on several factors. Some of these factors, which can help determine the appropriate model, are listed below:

***Wi-Fi Access Network Ownership:** There are several ownership models available when it comes to Wi-Fi to packet core integration. Wi-Fi Access network may be deployed by the MNO to leverage as another RAT to complement 3G and LTE. Alternatively the Mobile Network Operator may deploy a Managed Wi-Fi network for the Enterprise and SMB customers. The MNO managed Wi-Fi footprint is only portion of the overall Wi-Fi deployment. Third parties such as broadband service providers today own a significant portion of the Wi-Fi access network. For third party owned Wi-Fi access, the Mobile Network Operator may or may not have a direct roaming partnership with the Wi-Fi operator. The ownership model influences the choice of packet core integration architecture.

***Backhaul Network Ownership:** From the context of this discussion here, the backhaul refers to the connectivity between WLAN Access network and the Packet core. It consists of a combination of wired access network of the hotspot, Broadband access last mile, Wi-Fi operator core network, Internet etc. These connectivity aspects will be deciding factor for the choice of Wi-Fi packet integration model. For example, Wi-Fi access network may be owned and or operated by the MNO, but if the backhaul involved a third party connection or Internet where MNO does not have control over security and QoS, an untrusted packet core integration may be the viable solution.

***Mobile Offload Requirements:** Choice of the Wi-Fi to packet core integration model is not only influenced by voice offload but data offload as well. The untrusted Wi-Fi and the hybrid architectures do support a flexible offload model, allowing the Mobile Network Operator to choose which traffic to backhaul to the Mobile Packet Core to provide charging and added value services, while also leveraging local breakout capabilities on the device. Using the untrusted, and when applicable, the hybrid models allow the Mobile Network Operator to leverage their deployed network architecture for Wi-Fi calling. This makes both the hybrid and the untrusted Wi-Fi architectures valid options to consider depending on the Wi-Fi network ownership requirements.

***Device Capabilities:** This greatly influences the choice of Wi-Fi to packet core integration. For example, a trusted approach with multiple PDN support requires the capability on the device to comply

with 3GPP release 12 SaMOC enhancements, while the untrusted or hybrid model can leverage existing implementations and do provide a similar level of functionality.

*Support of Non-SIM devices: The MNO can provide value-added services, including voice services on Non-SIM devices. The Untrusted Wi-Fi architecture is compatible with Non-SIM devices and provide the same capabilities to these devices as for the SIM devices.

*Network Readiness: This is another influencing factor for the choice of the trust model, as there are dependencies on the Packet Core network elements as well as Wi-Fi access network for the implementation of these models.

5. Subscriber Onboarding into Wi-Fi Access Network

Subscriber onboarding into a Wi-Fi access network is the process of getting connected to a WLAN access network and be able to offload mobile traffic successfully. In order to provide a seamless end user experience for Wi-Fi calling, the handset should be able to get connected to the WLAN with minimum or no user interaction. A seamless WLAN onboarding is critical for the smooth hand off of the voice call from LTE to Wi-Fi. There are several factors, which can influence the Wi-Fi onboarding experience. Proper choice of the available deployment options can ensure the subscriber onboarding experience is quite seamless.

5.1. Authentication and Identity Management

Before the UE device can successfully get associated with a WLAN access network it needs to get authenticated with the WLAN network. There are several types of user authentication options in use such as Web Portal based authentication, EAP-TTLS, EAP-TLS, EAP-SIM, EAP-AKA etc. Choice of the authentication mechanism depends up on the deployment preferences of the Wi-Fi operator. Web portal based authentication relies on an Open SSID configuration. Once the portal has successfully authenticated the UE device, the traffic is carried over the WLAN air interface without any encryption. EAP authentication mechanisms relies on secured SSIDs mandate the 802.11i based air encryption of the subscriber data in the WLAN access network.

In order to support Wi-Fi calling, one of the EAP based mechanisms will be preferred over the web portal based authentication. In the case of Web based authentication, the user needs to manually enter the username and password credentials or in some cases sign up for a service via Operator portal. But with any of the EAP methods, once the credentials have been established on the UE device, then

authentication happens automatically without user intervention and greatly improves the onboarding experience.

If the Wi-Fi operator decides to use a secured SSID for subscriber authentication, choice of the EAP method depends up on the business model. A Standalone Wi-Fi operator may need to rely on non-SIM based EAP authentication mechanisms such as EAP-TTLS or EAP-TLS for their home subscribers. A Wi-Fi operator who has a roaming partnership with an MNO could allow the uSIM credentials of the MNO subscriber to be used for the access. In this case, the Wi-Fi operator will act as a proxy and authenticate the customer credentials with the MNO HSS.

Identity management deals with establishing subscriber identity and associated credentials on the UE device for WLAN onboarding. Identity management and authentication goes hand in hand. Option leverages the same set of identity and credentials (unified identity) for WLAN onboarding and packet core connectivity will simplify the identity management for Wi-Fi calling. However this requires that the WLAN access network is either owned by the MNO or by their roaming partner. With unified identity, typically uSIM credentials will be leveraged for both WLAN onboarding as well as packet core connectivity for SIM devices, and an EAP method used for Non-SIM devices.

5.2. Hotspot 2.0 for Seamless Onboarding

Ability for a handset to Seamlessly get connected to WLAN access network is one of the key factors which will influence the overall subscriber experience with Wi-Fi calling. Passpoint specifications defined by the Wi-Fi alliance under the Hotspot 2.0 program supports automatic discovery, selection and onboarding of Wi-Fi clients on to a compatible Wi-Fi access network. Figure-5 below is used to illustrate the hotspot 2.0 solution at a high level:

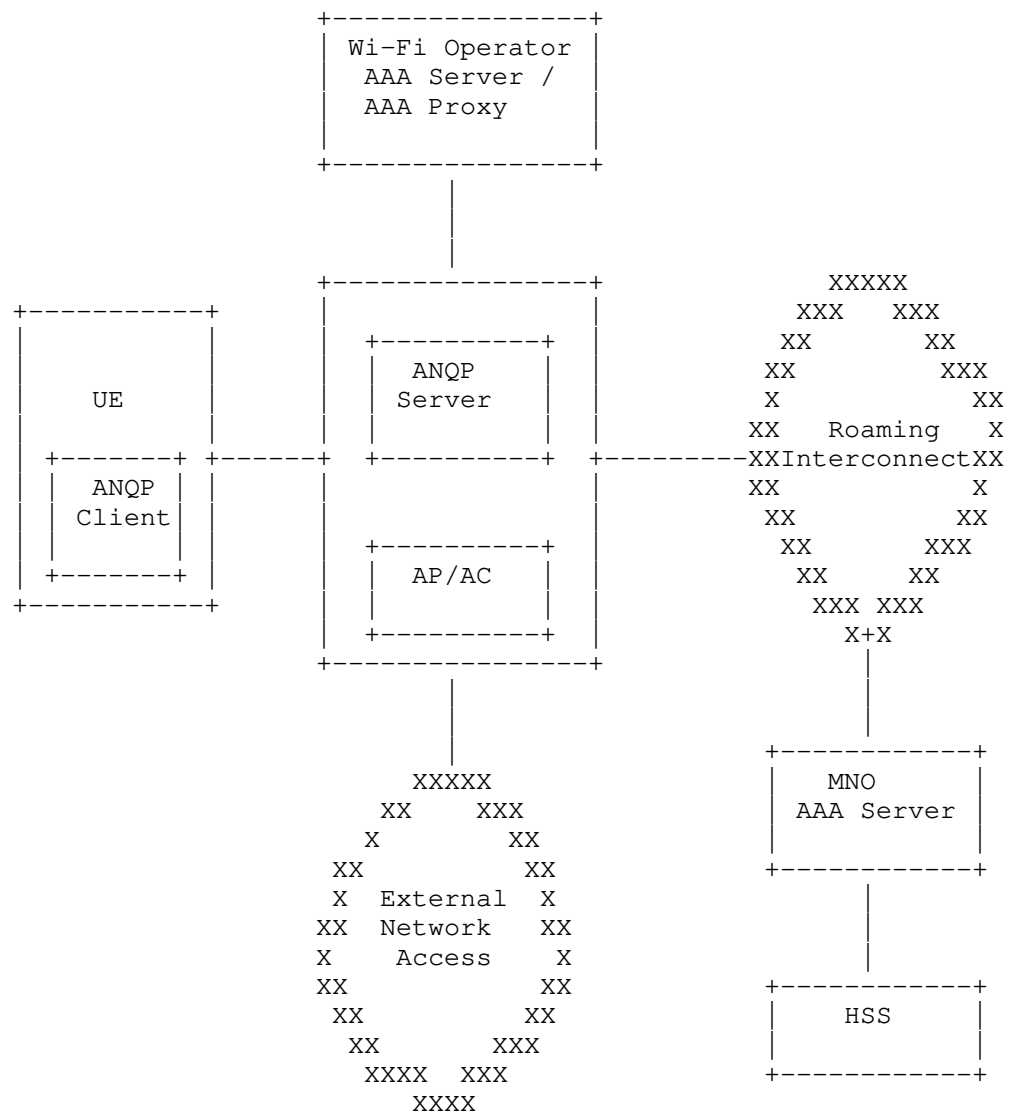


Figure 5: Hotspot 2.0 with Service Provider Roaming

ANQP server is the component, which assists with the automatic discovery of WLAN network resources by the UE device. ANQP server is

typically collocated on the Access Point (AP) or the Access Controller (AC). A Hotspot 2.0 compatible UE device will have a built in ANQP client. When a UE roams into the coverage area of a Hotspot 2.0 enabled network, it automatically learns about the network capability via Beacon or Probe Response. Then UE requests a set of network and service level information from the WLAN network. Based up on the info UE can decide which WLAN access is the most preferred and the type credentials it can use for getting connected.

5.2.1. Hotspot 2.0 Inter-Operator Roaming for Wi-Fi Calling

MNOs can enter into roaming partnership, which will allow Wi-Fi calling clients to automatically get connected to the WLAN access. This also allows the devices to leverage uSIM credentials or EAP credentials for Non-SIM devices for getting authenticated with the WLAN network. The Wi-Fi operator AAA will function as a proxy in this case and completes the authentication by interfacing with the MNO AAA Server and HSS, for EAP_SIM/EAP_AKA in the MNO packet core.

6. Wi-Fi calling deployment in restrictive networks

The use of IPSec to establish a connection to the ePDG, require that the access network allow IPSec tunnel establishment. But some networks won't allow IPSec traffic either as a security policy or as a side-effect of only allowing "web traffic". In addition, many mainly corporate environments do deploy an HTTP proxy which will also prevent the establishment of an IPSec tunnel. Performing changes to these deployments may not always be possible or cost effective for the corporation or the public venues, especially in an "Untrusted Wi-Fi" model without the MNO involvement. In such situations, the mobile device can leverage the IPSec TCP encapsulation as described in draft-paulu-ipsectcp-encaps-04 and in 3GPP TS 24302, which define the encapsulation of IPsec traffic in TCP. The Mobile device shall enable the TCP encapsulation only after failing to establish an IPSec connection to the ePDG. Figure 6 below shows the TCP encapsulation with the use for TLS to traverse a Proxy and reach the ePDG.

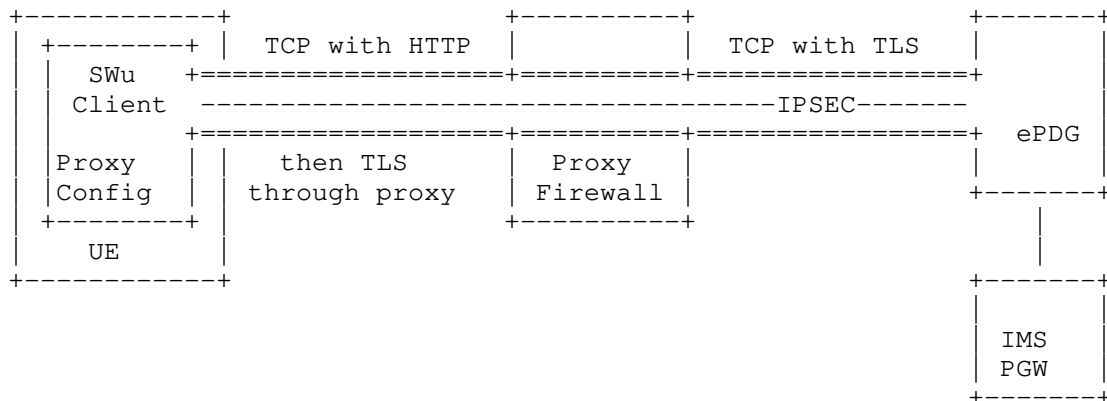


Figure 6: Use of TCP encapsulation for IPsec

When an HTTP proxy is deployed, the UE should connect to the ePDG through the proxy and then establish a TLS connection toward the ePDG. TLS is not used for securing the link, but to traverse the HTTP Proxy, and is configured with NULL-Cipher. This model allows Wi-Fi calling to operate even in restrictive networks.

7. RF Network Performance Optimization

Quality of the Wi-Fi calling experience would be as good or as bad as Radio network itself. Three network performance KPIs which impact the quality of voice are latency, jitter and packet drops. A healthy network is critical to ensure that these KPIs will meet the thresholds allowed to meet the acceptable voice quality. This section primarily talks about various performance optimization mechanisms available on the Wi-Fi Radio network.

7.1. Radio Resource Management

Radio Resource Management (RRM) aka Wi-Fi SON refers to the co-ordinated fine-tuning of the various RF network parameters among access points connected in a Wi-Fi network. It is very typical for Wi-Fi deployments from multiple operators to co-exist in the same hotspot. Scope RF fine tuning will be limited to the access points which are managed by the same operator in a specific hotspot. RRM fine-tuning will be typically performed by a centralized entity such as Access Controller. Some deployments which may not leverage AC such as Residential Gateways could leverage a cloud based RRM or SON Server. RRM controller continuously analyze the existing RF environment automatically adjust the power and channel configurations of access points to help mitigate issues such as co-channel interface and signal coverage. A proper implementation of RRM can greatly

influence the RF performance and will have a positive impact on network KPIs that influence the Wi-Fi calling experience.

7.2. Wi-Fi Roaming Optimization

Roaming from the context of the discussion here refers to the hand off of a UE device from one Access Point to another Access Point in the same Extended Services Set (ESS) or mobility domain. Unlike cellular roaming between base stations, which is initiated by the network, in Wi-Fi the roaming is initiated by the UE device. A UE typically decides to disconnect from the current access point when some of the RF measurements such as RSSI, SNR etc. drops below certain threshold. There are other APs in the range with acceptable measurements the UE will start re-association process with one of the target APs. End user experience for a Wi-Fi call, which is active at the time of the hand off, will depend up on multiple factors. One critical factor is the time taken for the UE traffic to resume during the hand off. Also it is important that UE is able to make the optimum selection of the target AP from the list of available APs in the range. Discussed below are few IEEE 802.11 based mechanisms available to optimize the roaming.

7.2.1. Fast BSS Transition

IEEE 802.11r based fast BSS transition (FT) helps reduce the handoff time for a UE when it roams from one AP to another within an ESS, which is enabled, with an EAP based authentication. Without FT, the UE will have to go through the full authentication process with the RADIUS server and device fresh set of encryption for 802.11i air encryption. When FT is enabled, the client will have an initial handshake with the target AP while still connected to the original AP. This handshake allows client and target APs to derive the encryption keys in advance to reduce the hand off time. Fast Transition can significantly improve the end user experience for the voice calls, which are active during a hand off.

7.2.2. 802.11k based Neighbor Reports

IEEE 802.11k enhancements allow a UE device to request from the current AP to which it is connected for a recommended list of neighboring APs for roaming. Upon receiving the client request, the AP responds with a list of neighbors on the same WLAN with the Wi-Fi channel numbers. Neighbor list is created by the AP based up on the Radio Resource Measurements and includes the best potential roaming targets for the UE. Neighbor list allows UE to reduce the scanning time when it is time to roam into a new AP in the same WLAN and thereby improves the roaming performance. It is recommended to enable

802.11k along with Fast BSS transmission for optimum roaming performance.

7.2.3. 802.11v based Assisted Roaming and Load Balancing

Typical WLAN deployments will have APs with overlapping coverage areas. This is done on purpose to seamless handoff and also to address capacity requirements. Load distribution of UEs in the same coverage area may be helpful to proactively manage the bandwidth requirements and thereby improve the subscriber experience. In the most rudimentary form, some of the load balancing solutions relies on the brute force method of ignoring the association requests from a UE by the APs with high load. Another more sophisticated mechanism is to leverage 802.11v based network assisted roaming. 802.11v allows unsolicited BSS transmission management messages from AP towards the client with a list of preferred APs to make roaming decisions. If the AP is experiencing high load, or bad connectivity from the client it may send an unsolicited BSS transmission management frame with the recommended list of APs to roam into. Depending up on the client implementation, it may or may not honor this info while making oaming decisions.

8. QoS Deployment Considerations for Wi-Fi Calling

This section covers the traffic prioritization mechanisms available in various segments of the overall traffic path of the Wi-Fi calling signaling and bearer sessions. Flexibility control of the QoS implementations will depend up on various factors such as ownership and management of the WLAN access network, Wi-Fi to packet core integration model etc.

8.1. Wi-Fi Access Network QoS

Traffic prioritization in the WLAN for Carrier Wi-Fi calls can be achieved by implementing Wi-Fi Multimedia (WMM). WMM consists of a subset of IEEE 802.11e enhancements for Wi-Fi. WMM defines four Access Categories, AC1, AC2, AC3 and AC4. AC1 is mapped against voice, AC2 is mapped against video, AC3 is mapped against best effort traffic and AC4 is mapped against Background traffic. Each of these Access Categories is mapped against one or more 802.11e User Priority (UP) values. UP has range from 0 to 7. Higher UP values typically gets more expedited over the air treatment EDCA mechanism for channel access defined in 802.11e is modified to make sure that traffic in higher UP queues get higher priority treatment. WMM can only leveraged if the client can do the right classification and Access points also support it.

8.2. End to End QoS

While QoS on the WLAN access network is critical, that by itself may not be sufficient to maintain the subscriber quality of experience. It is important to enable QoS prioritization across all the network segments, which form part of the end-to-end voice path. Flexibility of the QoS implementation along the network segments will depend upon the trust models, which are discussed earlier. For example, if the transit path between WLAN network and Packet Core includes Internet, no QoS prioritization can be implemented over the Internet backhaul. However, for deployment scenarios in which all network segments along the voice traffic path are managed either by the Mobile operator or their partners, then it makes much easier to implement end-to-end QoS. End-to-end QoS Classification for Wi-Fi calling is illustrated in figure 7 below.

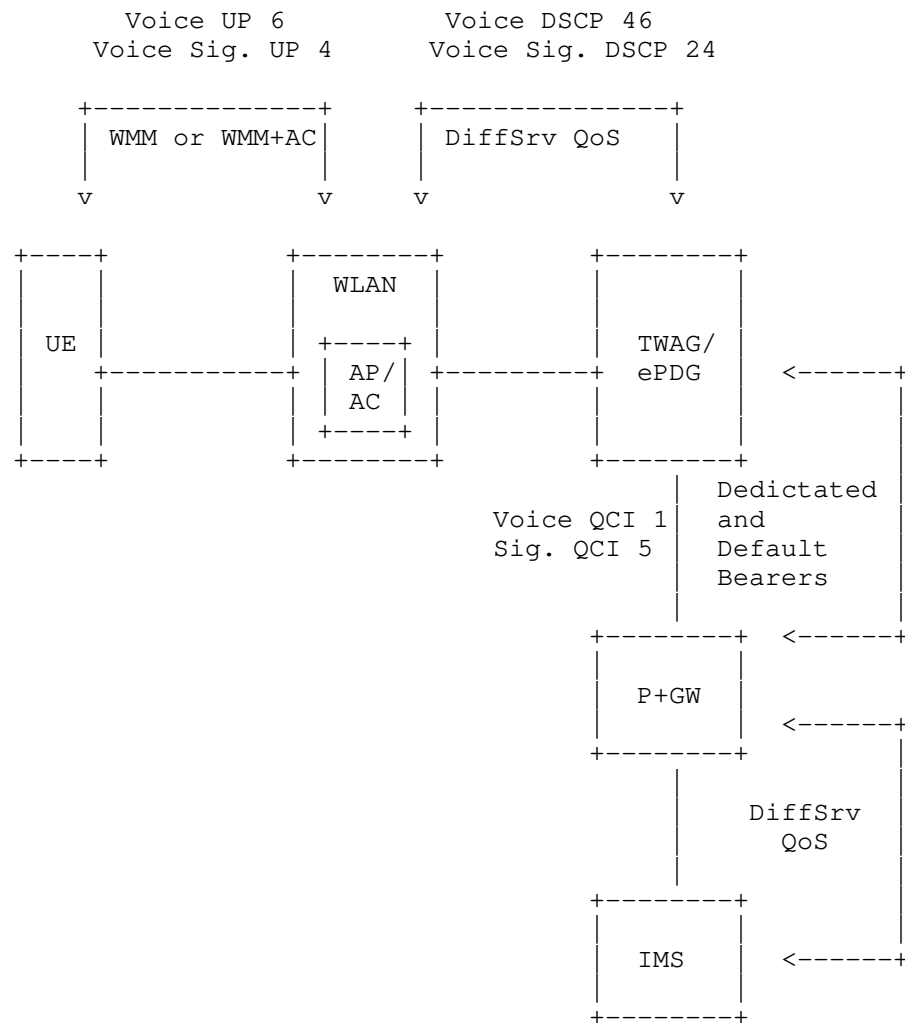


Figure 7: End-to-end QoS Reference Model

This QoS reference model assumes that, MNO or their roaming partners manage all the segments in the end-to-end path for voice signaling and voice bearer traffic. Model also assumes that transit path between WLAN and Packet core is private and secured and does not traverse Internet.

QoS reference model leverages WLAN access network leverages WMM that is described in the previous section, UP value of 6 is typically used for voice bearer traffic and UP value of 4 is used for voice signaling traffic. In order for voice to get the proper prioritization, WMM needs to be supported and enabled on both UE and the WLAN network.

In the transit IP network between WLAN and packet core, DSCP based QoS prioritization can be deployed if the connectivity is part of a managed transport. DSCP value of 46 is typically used for marking voice bearer and DSCP value of 24 is typically used for marking voice signaling. Proper traffic prioritization will depend up on whether DiffSrv QoS is enabled in the transit network.

Between P-GW and ePDG or TWAG, dedicated bearer with QCI value 1 will be established dynamically for voice calls. For signaling traffic a default bearer with QCI value of 5 will be used. These QCI values are mapped against specific QoS SLAs and allocation retention policies (ARP).

9. Wi-Fi Calling Client Considerations

Wi-Fi Calling client device functionality requirements depend on the on the models used for WLAN to packet core integration. At a minimum the clients should support IMS User Agent as defined in the 3GPP spec and be able to send and receive both IMS signaling and bearer traffic over a Wi-Fi access point. In addition, an SWu client that supports IPsec will can use ePDG-based packet core integration. This section talks about some of the client side implementation considerations for Wi-Fi calling.

9.1. Access Selection Criteria

The client device must select which RAT (cellular or Wi-Fi) it will use for communication to the cellular network. Commonly deployed access selection criteria is described below:

Device Local Policy Profile: In this case, the logic is defined by locally configured policy. Local policy may allow the end user to set preferences. It is also possible for carriers to push these profiles to the device. Some MNOs may prefer cellular instead of Wi-Fi for voice service when both RAT technologies are available. Some other carriers may have Wi-Fi preferred approach for IMS APN when both RAT technologies are available. If Passpoint is enabled on the Wi-Fi access network, the client may take into account network loading conditions learned from the ANQP server to decide whether to offload IMS traffic into the Wi-Fi network.

9.2. Inter-RAT Handover

Inter-RAT handover refers to the handover of an active voice call without service disruption when the UE switches out from one RAT technology to another. Implementations must support handovers between Wi-Fi and LTE.

Handover between LTE and Wi-Fi is achieved by maintaining IP or IPv6 addresses between the LTE interface and the IPsec tunnel over Wi-Fi. If the IPsec tunnel is negotiated while a call is already in progress, the IKEv2 Configuration Request should specify the local address of the LTE interface in order to get assigned the same address on the IPsec tunnel. Similarly, handover from an IPsec tunnel over Wi-Fi to LTE requires the LTE interface to be brought up with the same address as the tunnel. Maintaining the address allows the client to not interrupt TCP or UDP connections that are using the local address for communication. In a system that uses POSIX sockets, for example, the handover must be done in such a way that the sockets do not need to be closed and re-opened.

9.3. MTU Considerations

When handing over between LTE and IPsec tunnels over Wi-Fi, the client device should be aware of the Maximum Transmission Unit (MTU) of each interface. It is possible that the effective MTU for the IPsec tunnel (which can be calculated as the MTU of the Wi-Fi interface minus the overhead for ESP encryption) is notably smaller than the effective MTU of the LTE interface. For UDP flows, they should avoid sending large datagrams that could get fragmented when handing over between RATs. For TCP flows, the Maximum Segment Size based on the MTU SHOULD be re-calculated upon handover.

9.4. Congestion Management

Radio Network Performance management and QoS considerations described earlier can significantly contribute to the overall QoE for Wi-Fi calling. A client driven congestion management mechanism can positively augment the overall experience. The idea is to dynamically change the bandwidth requirements for the call based up on the network congestion conditions. Network resource requirements (bandwidth, packets per second etc.) per call are directly proportional to the type of codec and the packetization rate. Sometimes it may be desirable to switch out to a lower audio codec to keep the drop, delay and jitter characteristics under acceptable levels during periods of network congestion. Explicit Congestion Notification for RTP over UDP defined in RFC 6679 can be used to inform network congestion to the end clients. But this requires the

network elements to mark the ECN bits on the IP header of the packet when congestion conditions are encountered.

9.5. NAT Traversal

Since NATs are very commonly deployed primarily due to the shortage of IPv4 address space, a client side implementation should support NAT traversal for Wi-Fi calling. IPsec implementation on the client side should support the detection of NAT gateways as defined in RFC 7296 specification. If a NAT gateway is detected, client should send all subsequent IPsec traffic from port 4500. If NAT is detected ESP packets must be UDP encapsulation using port 4500. If NAT devices are not detected, SWu may use pure ESP encapsulation without UDP. It is important to understand the implications on firewall rules with and without NAT so that the Wi-Fi calling does not get blocked by the firewall. Many deployments may allow ESP with UDP encapsulation by default but may block ESP only tunnels.

10. Acknowledgements

Authors would like to acknowledge the inputs and advice provided by Eduardo Abrantes and Ajoy Singh.

11. Informative References

- [IR51] "IMS Profile for Voice, Video and SMS over untrusted Wi-Fi access Version 5.0", 2017.
- [IR92] "IMS Profile for Voice and SMS Version 10.0", 2016.
- [RFC4066] Liebsch, M., Ed., Singh, A., Ed., Chaskar, H., Funato, D., and E. Shim, "Candidate Access Router Discovery (CARD)", RFC 4066, DOI 10.17487/RFC4066, July 2005, <<http://www.rfc-editor.org/info/rfc4066>>.
- [RFC4187] Arkko, J. and H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", RFC 4187, DOI 10.17487/RFC4187, January 2006, <<http://www.rfc-editor.org/info/rfc4187>>.
- [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, DOI 10.17487/RFC4555, June 2006, <<http://www.rfc-editor.org/info/rfc4555>>.
- [RFC4881] El Malki, K., Ed., "Low-Latency Handoffs in Mobile IPv4", RFC 4881, DOI 10.17487/RFC4881, June 2007, <<http://www.rfc-editor.org/info/rfc4881>>.

- [RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, DOI 10.17487/RFC5213, August 2008, <<http://www.rfc-editor.org/info/rfc5213>>.
- [RFC5568] Koodli, R., Ed., "Mobile IPv6 Fast Handovers", RFC 5568, DOI 10.17487/RFC5568, July 2009, <<http://www.rfc-editor.org/info/rfc5568>>.
- [RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised", RFC 5944, DOI 10.17487/RFC5944, November 2010, <<http://www.rfc-editor.org/info/rfc5944>>.
- [RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <<http://www.rfc-editor.org/info/rfc6275>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<http://www.rfc-editor.org/info/rfc7296>>.
- [TS22228] "Service requirements for the Internet Protocol (IP) multimedia core network subsystem (IMS); Stage 1", 2010.
- [TS23402] "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Architecture Enhancements for non-3GPP Accesses.", 2009.
- [TS23852] "Study on S2a Mobility based on GPRS Tunneling Protocol (GTP) and Wireless Local Area Network (WLAN) access to the Enhanced Packet Core (EPC) network (SaMOG); Stage 2", 2011.
- [TS29273] "Evolved Packet System (EPS); 3GPP EPS AAA interfaces", 2011.

Authors' Addresses

Byju Pularikkal
Cisco Systems
170 West Tasman Drive
San Jose
United States

Email: byjupg@cisco.com

Tommy Pauly
Apple Inc.
1 Infinite Loop
Cupertino, California 95014
US

Email: tpauly@apple.com

Mark Grayson
Cisco Systems
10 New Square Park
Feltham
United Kingdom

Email: mgrayson@cisco.com

Sri Gundavelli
Cisco Systems
170 West Tasman Drive
San Jose
United States

Email: sgundave@cisco.com

Samy Touati
Ericsson
300 Holger Way
San Jose, California 95134
US

Email: samy.touati@ericsson.com

OPS Area Working Group
Internet-Draft
Intended status: Informational
Expires: December 1, 2017

Q. Wu
W. Liu
Huawei Technologies
A. Farrel
Juniper Networks
May 30, 2017

Service Models Explained
draft-wu-opsawg-service-model-explained-06

Abstract

The IETF has produced a considerable number of data modules in the YANG modelling language. The majority of these modules are used to construct data models to model devices or monolithic functions and they allow access for configuration and to read operational status.

A small number of YANG modules have been defined to model services (for example, the Layer Three Virtual Private Network Service Model produced by the L3SM working group and documented in RFC 8049).

This document briefly sets out the scope of and purpose of an IETF service model, and it also shows where a service model might fit into a Software Defined Networking architecture. Note that service models do not make any assumption of how a service is actually engineered and delivered for a customer; details of how network protocols and devices are engineered to deliver a service are captured in other models that are not exposed through the Customer-Provider Interface.

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 1, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Terms and Concepts	3
3. Using Service Models	6
4. Service Models in an SDN Context	8
5. Possible Causes of Confusion	10
6. Comparison With Other Work	11
6.1. Comparison With Network Service Models	12
6.2. Service Delivery and Network Element Model Work	13
6.3. Customer Service Model Work	14
6.4. The MEF Architecture	15
7. Further Concepts	16
7.1. Technology Agnostic	16
7.2. Relationship to Policy	16
7.3. Operator-Specific Features	17
7.4. Supporting Multiple Services	17
8. Security Considerations	18
9. Manageability Considerations	18
10. IANA Considerations	18
11. Acknowledgements	19
12. References	19
12.1. Normative References	19
12.2. Informative References	19
Authors' Addresses	21

1. Introduction

In recent years the number of data modules written in the YANG modelling language [RFC6020] for configuration and monitoring has blossomed. Many of these are used for device-level configuration (for example, [RFC7223]) or for control of monolithic functions or protocol instances (for example, [RFC7407]).

Within the context of Software Defined Networking (SDN) [RFC7426] YANG data models may be used on Southbound Interfaces (SBIs) between a controller and network devices, and between network orchestrators and controllers. There may also be a hierarchy of such components with super-controllers, domain controllers, and device controllers all exchanging information and instructions using YANG models.

Recently there has been interest in using YANG to define and document data models that describe services in a portable way that is independent of which network operator uses the model. For example, the Layer Three Virtual Private Network Service Model (L3SM) [RFC8049]. Such models may be used in manual and even paper-driven service request processes with a gradual transition to IT-based mechanisms. Ultimately they could be used in online, software-driven dynamic systems.

This document explains the scope and purpose of service models within the IETF and describes how a service model can be used by a network operator. Equally, this document clarifies what a service model is not, and dispels some common misconceptions.

The document also shows where a service model might fit into an SDN architecture, but it is important to note that a service model does not require or preclude the use of SDN. Note that service models do not make any assumption of how a service is actually engineered and delivered to a customer; details of how network protocols and devices are engineered to deliver a service are captured in other models that are not exposed through the Customer- Provider Interface.

Other work on classifying YANG data models has been done in [I-D.ietf-netmod-yang-model-classification]. That document provides an important reference for this document, and also uses the term "service model". Section 6.1 provides a comparison between these two uses of the same terminology.

2. Terms and Concepts

Readers should familiarize themselves with the description and classification of YANG models provided in [I-D.ietf-netmod-yang-model-classification].

The following terms are used in this document:

Network Operator: This term is used to refer to the company that owns and operates one or more networks that provide Internet connectivity services and/or other services. The term is also used to refer to an individual who performs operations and management on those networks.

Customer: This term refers to someone who purchases a service (including connectivity) from a network operator. In the context of this document, a customer is usually a company that runs their own network or computing platforms and wishes to connect to the Internet or between sites. Such a customer may operate an enterprise network or a data center. Sometimes this term may also be used to refer to the individual in such a company who contracts to buy services from a network operator. A customer as described here is a separate commercial operation from the network operator, but some companies may operate with internal customers so that, for example, an IP/MPLS packet network may be the customer of an optical transport network.

Service: A network operator delivers one or more services to a customer. A service in the context of this document (sometimes called a Network Service) is some form of connectivity between customer sites and the Internet, or between customer sites across the network operator's network and across the Internet. However, a distinction should be drawn between the parameters that describe a service as included in a customer service model (q.v.) and a Service Level Agreement (SLA) as discussed in Section 5 and Section 7.2.

A service may be limited to simple connectivity (such as IP-based Internet access), may be a tunnel (such as a virtual circuit), or may be a more complex connectivity model (such as a multi-site virtual private network). Services may be further enhanced by additional functions providing security, load-balancing, accounting, and so forth. Additionally, services usually include guarantees of quality, throughput, and fault reporting.

This document makes a distinction between a service as delivered to a customer (that is, the service as discussed on the interface between a customer and the network operator) and the service as realized within the network (as described in [I-D.ietf-netmod-yang-model-classification]). This distinction is discussed further in Section 6.

Readers may also refer to [RFC7297] for an example of how an IP connectivity service may be characterized.

Data Model: The concepts of information models and data models are described in [RFC3444]. That document defines a data model by contrasting it with the definition of an information model, so it may be helpful to quote some text to give context within this document.

The main purpose of an information model is to model managed objects at a conceptual level, independent of any specific implementations or protocols used to transport the data. The degree of specificity (or detail) of the abstractions defined in the information model depends on the modeling needs of its designers. In order to make the overall design as clear as possible, an information model should hide all protocol and implementation details. Another important characteristic of an information model is that it defines relationships between managed objects.

Data models, conversely, are defined at a lower level of abstraction and include many details. They are intended for implementors and include protocol-specific constructs.

Service Model: A service model is a specific type of data model. It describes a service and the parameters of the service in a portable way. The service model may be divided into two categories:

Customer Service Model: A customer service model is used to describe a service as offered or delivered to a customer by a network operator. It can be used by a human (via a user interface such as a GUI, web form, or CLI) or by software to configure or request a service, and may equally be consumed by a human (such as via an order fulfillment system) or by a software component. Such models are sometimes referred to simply as "service models" [RFC8049]. A customer service model is expressed as a core set of parameters that are common across network operators: additional features that are specific to the offerings of individual network operators would be defined in extensions or augmentations of the model. Except where specific technology details (such as encapsulations, or mechanisms applied on access links) are directly pertinent to the customer, customer service models are technology agnostic so that the customer does have influence over or knowledge of how the network operator engineers the service.

An example of where such details are relevant to the customer are when they describe the behavior or interactions on the interface between the equipment at the customer site (often referred to as the Customer Edge or CE equipment) and the equipment at the network operator's site (usually referred to as the Provider Edge or PE equipment).

Service Delivery Model: A service delivery model is used by a network operator to define and manage how a service is engineered in the network. It can be used by a human operator (such as via a management station) or by a software tool to instruct network components. Such models are sometimes referred to as "network service models" [I-D.ietf-netmod-yang-model-classification] and are consumed by "external systems" such as Operations Support System (OSS). A service delivery model is expressed as a core set of parameters that are common across a network type and technology: additional features that are specific to the configuration of individual vendor equipment or proprietary protocols would be defined in extensions or augmentations of the model. Service delivery models include technology-specific modules.

The distinction between a customer service model and a service delivery model needs to be repeatedly clarified. A customer service model is not a data model used to directly configure network devices, protocols, or functions: it is not something that is sent to network devices (i.e., routers or switches) for processing. Equally, a customer service model is not a data model that describes how a network operator realizes and delivers the service described by the model. This distinction is discussed further in later sections.

3. Using Service Models

As already indicated, customer service models are used on the interface between customers and network operators. This is shown simply in Figure 1

The language in which a customer service model is described is a choice for whoever specifies the model. The IETF uses the YANG data modeling language defined in [RFC6020]

The encoding and communication protocol used to exchange a customer service model between customer and network operator are deployment- and implementation-specific. The IETF has standardized the NETCONF protocol [RFC6241] and the RESTCONF protocol [RFC8040] for interactions "on the wire" between software components with data encoded in XML or JSON. However, co-located software components might use an API, while systems with more direct human interactions might use web pages or even paper forms.

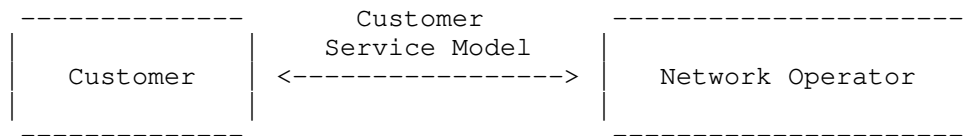


Figure 1: The Customer Service Models used on the Interface between Customers and Network Operators

How a network operator processes a customer's service request described with a customer service model depends on the commercial and operational tools, processes, and policies used by the network operator. These may vary considerably from one network operator to another.

However, the intent is that the network operator maps the service request into configuration and operational parameters that control one or more networks to deliver the requested services. That means that the network operator (or software run by the network operator) takes the information in the customer service model and determines how to deliver the service by enabling and configuring network protocols and devices. They may achieve this by constructing service delivery models and passing them to network orchestrators or controllers. The use of standard customer service models eases service delivery by means of automation.

The practicality of customer service models has been repeatedly debated. It has been suggested that network operators have such radically different business modes and such diverse commercial offerings that a common customer service model is impractical. However, the L3SM [RFC8049] results from the consensus of multiple individuals working at network operators and offers a common core of service options that can be augmented according to the needs of individual network operators.

It has also been suggested that there should be a single, base customer service module, and that details of individual services should be offered as extensions or augmentations of this. It is quite possible that a number of service parameters (such as the identity and postal address of a customer) will be common and it would be a mistake to define them multiple times, once in each customer service model. However, the distinction between a 'module' and a 'model' should be considered at this point: modules are how the data for models is logically broken out and documented especially for re-use in multiple models.

4. Service Models in an SDN Context

In an SDN system, the management of network resources and protocols is performed by software systems that determine how best to utilize the network. Figure 2 shows a sample architectural view of an SDN system where network elements are programmed by a component called an "SDN controller" (or "controller" for short), and where controllers are instructed by an orchestrator that has a wider view of the whole of, or part of, a network. The internal organization of an SDN control plane is deployment-specific.

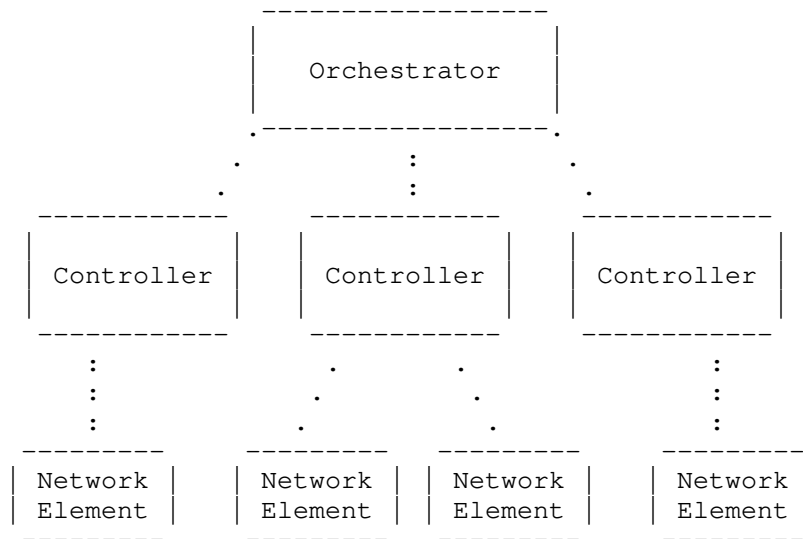


Figure 2: A Sample SDN Architecture

But a customer's service request is (or should be) technology-agnostic. That is, there should be an independence between the behavior and functions that a customer requests and the technology that the network operator has available to deliver the service. This means that the service request must be mapped to the orchestrator's view, and this mapping may include a choice of which networks and technologies to use depending on which service features have been requested.

One implementation option to achieve this mapping is to split the orchestration function between a "Service Orchestrator" and a "Network Orchestrator" as shown in Figure 3.

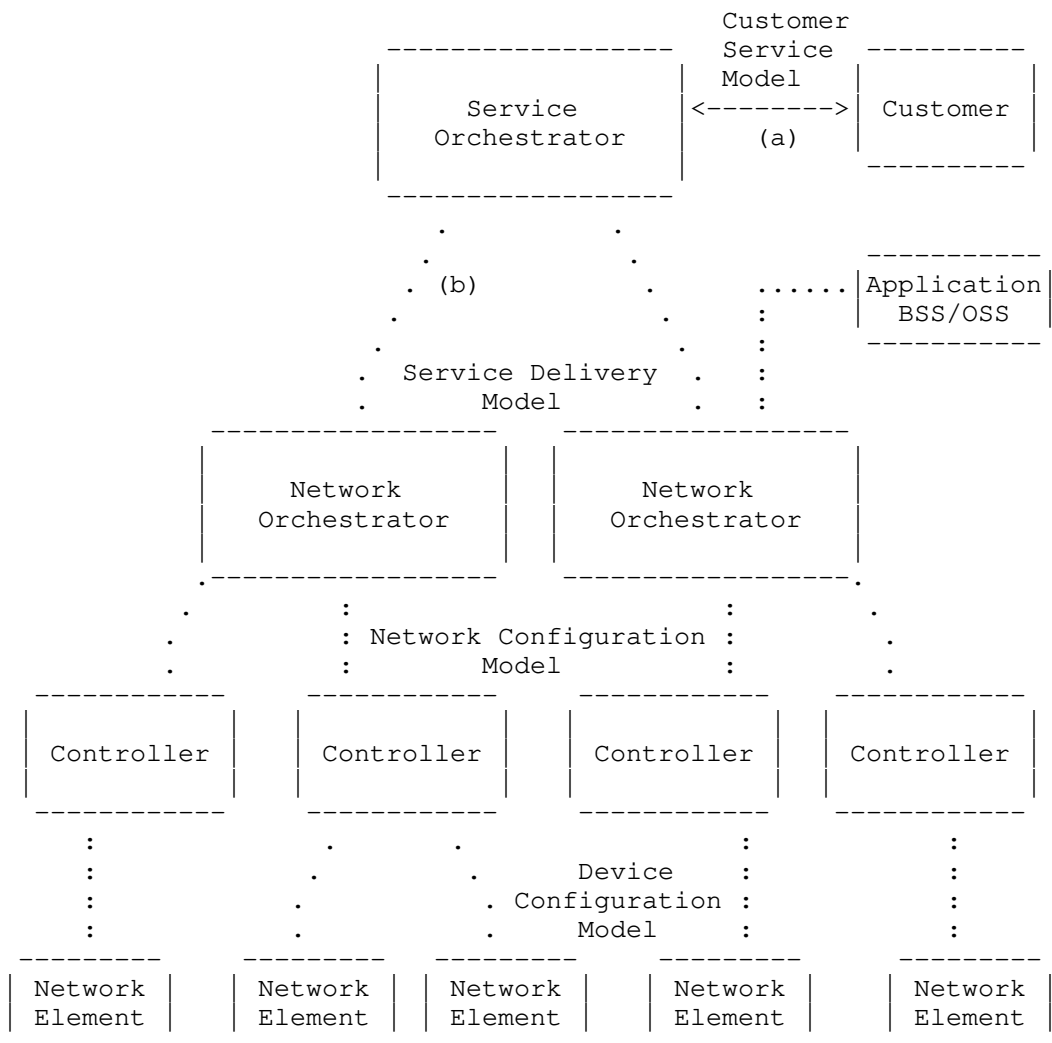


Figure 3: An Example SDN Architecture with a Service Orchestrator

Figure 3 also shows where different data models might be applied within the architecture.

The split between control components that exposes a "service interface" is present in many figures showing extended SDN architectures:

- o Figure 1 of [RFC7426] shows a separation of the "Application Plane", the "Network Services Abstraction Layer (NSAL)", and the "Control Plane". It marks the "Service Interface" as situated between the NSAL and the Control Plane.
- o [RFC7491] describes an interface between an "Application Service Coordinator" and an "Application-Based Network Operations Controller".
- o Figure 1 of [I-D.ietf-netmod-yang-model-classification] shows an interface from an OSS or a Business Support System (BSS) that is expressed in "Network Service YANG Models".

This can all lead to some confusion around the definition of a "service interface" and a "service model". Some previous literature considers the interface northbound of the Network Orchestrator (labeled "(b)" in Figure 3) to be a "service interface" used by an application, but the service described at this interface is network-centric and is aware of many features such as topology, technology, and operator policy. Thus, we make a distinction between this type of service interface and the more abstract service interface (labeled "(a)" in Figure 3) where the service is described by a service model and the interaction is between customer and network operator. Further discussion of this point is provided in Section 5.

5. Possible Causes of Confusion

In discussing service models, there are several possible causes of confusion:

- o The services we are discussing are services provided by network operators to customers. This is a completely different thing to "Foo as a Service" (for example, Infrastructure as a Service (IaaS)) where a service provider offers a service at some location that is reached across a network. The confusion arises not only because of the use of the word "service", but also because network operators may offer value-added services as well as network connection services to their customers.
- o Network operation is completely out of scope in the discussion of services between a network operator and a customer. That means that the customer service model does not reveal to the customer anything about how the network operator delivers the service. The model does not expose details of technology or network resources used to provide the service. For example, in the simple case of point-to-point virtual link connectivity provided by a network tunnel (such as an MPLS pseudowire) the network operator does not expose the path through the network that the tunnel follows. Of

course, this does not preclude the network operator from taking guidance from the customer (such as to avoid routing traffic through a particular country) or from disclosing specific details (such as might be revealed by a route trace), but these are not standard features of the service as described in the customer service model.

- o The network operator may use further data models (service delivery models) that help to describe how the service is realized in the network. These models might be used on the interface between the Service Orchestrator and the Network Orchestrator as shown in Figure 3 and might include many of the pieces of information from the customer service model alongside protocol parameters and device configuration information.

[I-D.ietf-netmod-yang-model-classification] also terms these data models as "service models" or "Network Service YANG Models" and a comparison is provided in Section 6.1. It is important that the Service Orchestrator should be able to map from a customer service model to these service delivery models, but they are not the same things.

- o Commercial terms are generally not a good subject for standardization. It is possible that some network operators will enhance standard customer service models to include commercial information, but the way this is done is likely to vary widely between network operators.
- o Service Level Agreements (SLAs) have a high degree of overlap with the definition of services present in customer service models. Requests for specific bandwidth, for example, might be present in a customer service model, and agreement to deliver a service is a commitment to the description of the service in the customer service model. However, SLAs typically include a number of fine-grained details about how services are allowed to vary, by how much, and how often. SLAs are also linked to commercial terms with penalties and so forth, and so are also not good topics for standardization.

If a network operator chooses to express an SLA using a data model, that model might be referenced as an extension or an augmentation of the customer service model.

6. Comparison With Other Work

Other work has classified YANG models, produced parallel architectures, and developed a range of YANG models. This section briefly examines that other work and shows how it fits with the description of service models introduced in this document.

6.1. Comparison With Network Service Models

As previously noted, [I-D.ietf-netmod-yang-model-classification] provides a classification of YANG data models. It introduces the term "Network Service YANG Module" to identify the type of model used to "describe the configuration, state data, operations and notifications of abstract representations of services implemented on one or multiple network elements." These are service delivery models as described in this document, that is, they are the models used on the interface between the Service Orchestrator or OSS/BSS and the Network Orchestrator as shown in Figure 3.

Figure 1 of [I-D.ietf-netmod-yang-model-classification] can be modified to make this more clear and to add an additional example of a Network Service YANG model as shown in Figure 4.

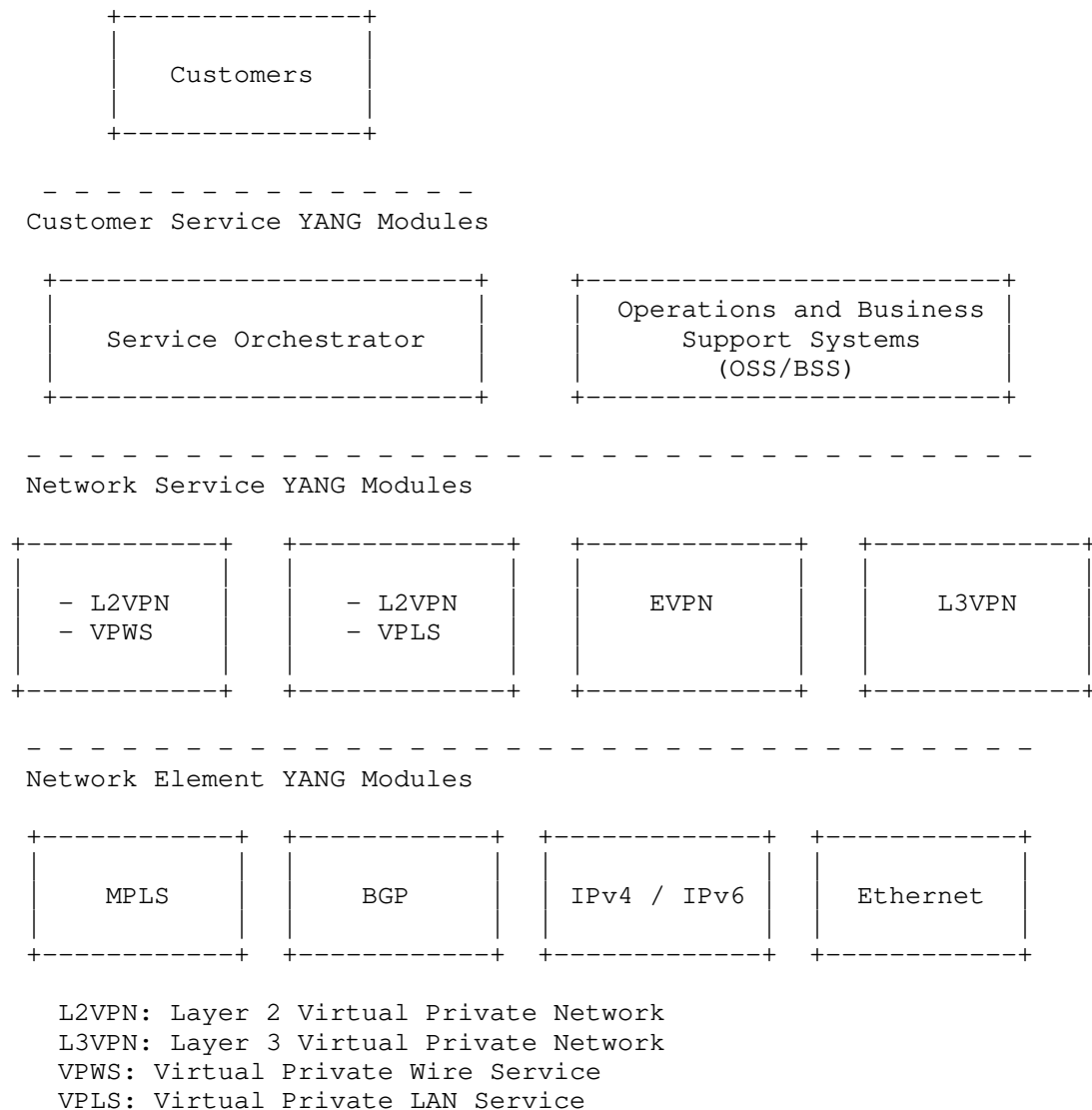


Figure 4: YANG Module Layers Showing Service Models

6.2. Service Delivery and Network Element Model Work

A number of IETF working groups are developing YANG models related to services. These models focus on how the network operator configures the network through protocols and devices to deliver a service. Some of these models are classed as service delivery models while others

have details that are related to specific element configuration and so are classed as network element models.

A sample set of these models is listed here:

- o [I-D.dhjain-bess-bgp-l3vpn-yang] defines a YANG model that can be used to configure and manage BGP Layer 3 VPNs.
- o [I-D.ietf-bess-l2vpn-yang] documents a YANG model that it is expected will be used by the management tools run by the network operators in order to manage and monitor the network resources that they use to deliver L2VPN services.
- o [I-D.ietf-bess-evpn-yang] defines YANG models for delivering an Ethernet VPN service.

6.3. Customer Service Model Work

Several initiatives within the IETF are developing customer service models. The most advanced presents the Layer Three Virtual Private Network (L3VPN) service as described by a network operator to a customer. This L3VPN service model (L3SM) is documented in [RFC8049] where its usage is described as in Figure 5 which is reproduced from that document. As can be seen, the L3SM is a customer service model as described in this document.

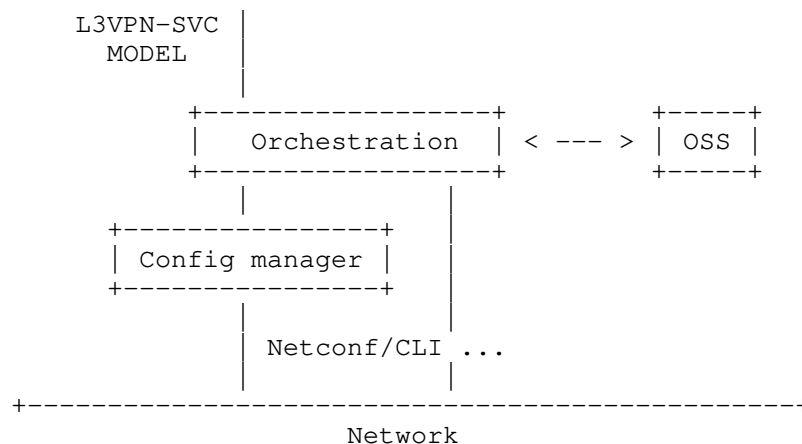


Figure 5: The L3SM Service Architecture

A Layer Two VPN service model (L2SM) is defined in [I-D.ietf-l2sm-l2vpn-service-model]. That model's usage is described

as in Figure 6 which is a reproduction of Figure 5 from that document. As can be seen, the L2SM is a customer service model as described in this document.

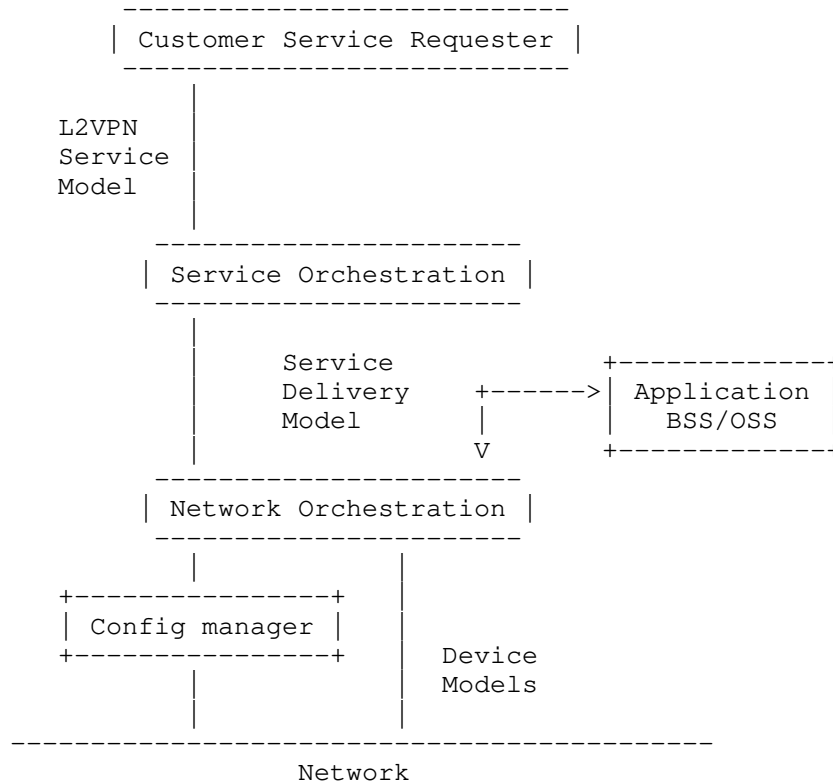


Figure 6: The L2SM Service Architecture

6.4. The MEF Architecture

The MEF Forum has developed an architecture for network management and operation. It is documented as the Lifecycle Service Orchestration (LSO) Reference Architecture and illustrated in Figure 2 of [MEF-55].

The work of the MEF Forum embraces all aspects of Lifecycle Service Orchestration including billing, SLAs, order management, and life-cycle management. The IETF's work on service models is typically smaller offering a simple, self-contained service YANG module. Thus, it may be impractical to fit IETF service models into the MEF Forum

LSO architecture. This does not invalidate either approach, but only observes that they are different.

7. Further Concepts

This section introduces a few further, more advanced concepts

7.1. Technology Agnostic

Service models should generally be technology agnostic. That is to say, the customer should not care how the service is provided so long as the service is delivered.

However, some technologies reach the customer site and make a difference to the type of service delivered. Such features do need to be described in the service model.

Two examples are:

- o The data passed between customer equipment and network operator equipment will be encapsulated in a specific way, and that data plane type forms part of the service.
- o Protocols that are run between customer equipment and network operator equipment (for example, Operations, Administration, and Maintenance protocols, protocols for discovery, or protocols for exchanging routing information) need to be selected and configured as part of the service description.

7.2. Relationship to Policy

Policy appears as a crucial function in many places during network orchestration. A Service Orchestrator will, for example, apply the network operator's policies to determine how to provide a service for a particular customer (possibly considering commercial terms). However, the policies within a service model are limited to those over which a customer has direct influence and that are acted on by the network operator.

The policies that express desired behavior of services on occurrence of specific events are close to SLA definitions: they should only be included in the base service model where they are common to all network operators' offerings. Policies that describe who at a customer may request or modify services (that is, authorization) are close to commercial terms: they, too, should only be included in the base service model where they are common to all network operators' offerings.

Nevertheless, policy is so important that all service models should be designed to be easily extensible to allow policy components to be added and associated with services as needed.

7.3. Operator-Specific Features

When work in the L3SM working group was started, there was some doubt as to whether network operators would be able to agree on a common description of the services that they offer to their customers because, in a competitive environment, each markets the services in a different way with different additional features. However, the working group was able to agree on a core set of features that multiple network operators were willing to consider as "common". They also understood that should an individual network operator want to describe additional features (operator-specific features) they could do so by extending or augmenting the L3SM model.

Thus, when a basic description of a core service is agreed and documented in a service model, it is important that that model should be easily extended or augmented by each network operator so that the standardized model can be used in a common way and only the operator-specific features varied from one environment to another.

7.4. Supporting Multiple Services

Network operators will, in general, offer many different services to their customers. Each would normally be the subject of a separate service model.

It is an implementation and deployment choice whether all service models are processed by a single Service Orchestrator that can coordinate between the different services, or whether each service model is handled by a specialized Service Orchestrator able to provide tuned behavior for a specific service.

It is expected that, over time, certain elements of the service models will be seen to repeat in each model. An example of such an element is the postal address of the customer.

It is anticipated that, while access to such information from each service model is important, the data will be described in its own module and may form part of the service model either by inclusion or by index.

8. Security Considerations

The interface between customer and service provider is a commercial interface and needs to be subject to appropriate confidentiality. Additionally, knowledge of what services are provided to a customer or delivered by a network operator may supply information that can be used in a variety of security attacks.

Clearly, the ability to modify information exchanges between customer and network operator may result in bogus requests, unwarranted billing, and false expectations. Furthermore, in an automated system, modifications to service requests or the injection of bogus requests may lead to attacks on the network and delivery of customer traffic to the wrong place.

Therefore it is important that the protocol interface used to exchange service request information between customer and network operator is subject to authorization, authentication, and encryption. This document discusses modeling that information, not how it is exchanged.

9. Manageability Considerations

This whole document discusses issues related to network management.

It is important to observe that automated service provisioning resulting from use of a customer service model may result in rapid and significant changes in traffic load within a network and that that might have an effect on other services carried in a network.

It is expected, therefore, that a Service Orchestration component has awareness of other service commitments, that the Network Orchestration component will not commit network resources to fulfill a service unless doing so is appropriate, and that a feedback loop will be provided to report on degradation of the network that will impact the service.

The operational state of a service does not form part of a customer service model. However, it is likely that a network operator may want to report some state information about various components of the service, and that could be achieved through extensions to the core service model.

10. IANA Considerations

This document makes no requests for IANA action

11. Acknowledgements

Thanks to Daniel King, Xian Zhang, and Michael Scharf for useful review and comments. Med Boucadair gave thoughtful and detailed comments on version -04 of this document. Thanks to Dean Bogdanovic and Tianran Zhou for their help coordinating with [I-D.ietf-netmod-yang-model-classification].

12. References

12.1. Normative References

- [I-D.ietf-netmod-yang-model-classification]
Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", draft-ietf-netmod-yang-model-classification-07 (work in progress), May 2017.
- [RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <<http://www.rfc-editor.org/info/rfc3444>>.
- [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <<http://www.rfc-editor.org/info/rfc7426>>.
- [RFC8049] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8049, DOI 10.17487/RFC8049, February 2017, <<http://www.rfc-editor.org/info/rfc8049>>.

12.2. Informative References

- [I-D.dhjain-bess-bgp-l3vpn-yang]
Jain, D., Patel, K., Brissette, P., Li, Z., Zhuang, S., Liu, X., Haas, J., Esale, S., and B. Wen, "Yang Data Model for BGP/MPLS L3 VPNs", draft-dhjain-bess-bgp-l3vpn-yang-02 (work in progress), August 2016.
- [I-D.ietf-bess-evpn-yang]
Brissette, P., Sajassi, A., Shah, H., Li, Z., Tiruveedhula, K., Hussain, I., and J. Rabadan, "Yang Data Model for EVPN", draft-ietf-bess-evpn-yang-02 (work in progress), March 2017.

- [I-D.ietf-bess-l2vpn-yang]
Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B.,
and K. Tiruveedhula, "YANG Data Model for MPLS-based
L2VPN", draft-ietf-bess-l2vpn-yang-05 (work in progress),
March 2017.
- [I-D.ietf-l2sm-l2vpn-service-model]
Wen, B., Fioccola, G., Xie, C., and L. Jalil, "A YANG Data
Model for L2VPN Service Delivery", draft-ietf-l2sm-l2vpn-
service-model-01 (work in progress), May 2017.
- [MEF-55] MEF Forum, "Service Operations Specification MEF 55 :
Lifecycle Service Orchestration (LSO) Reference
Architecture and Framework", March 2016.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
the Network Configuration Protocol (NETCONF)", RFC 6020,
DOI 10.17487/RFC6020, October 2010,
<<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
and A. Bierman, Ed., "Network Configuration Protocol
(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
<<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC7223] Bjorklund, M., "A YANG Data Model for Interface
Management", RFC 7223, DOI 10.17487/RFC7223, May 2014,
<<http://www.rfc-editor.org/info/rfc7223>>.
- [RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP
Connectivity Provisioning Profile (CPP)", RFC 7297,
DOI 10.17487/RFC7297, July 2014,
<<http://www.rfc-editor.org/info/rfc7297>>.
- [RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for
SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407,
December 2014, <<http://www.rfc-editor.org/info/rfc7407>>.
- [RFC7491] King, D. and A. Farrel, "A PCE-Based Architecture for
Application-Based Network Operations", RFC 7491,
DOI 10.17487/RFC7491, March 2015,
<<http://www.rfc-editor.org/info/rfc7491>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017,
<<http://www.rfc-editor.org/info/rfc8040>>.

Authors' Addresses

Qin Wu
Huawei Technologies

Email: bill.wu@huawei.com

Will Liu
Huawei Technologies

Email: liushucheng@huawei.com

Adrian Farrel
Juniper Networks

Email: afarrel@juniper.net

