ippm
Internet-Draft
Intended status: Experimental
Expires: September 30, 2017

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

March 29, 2017

Data Fields for In-situ OAM
draft-brockners-inband-oam-data-04

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, VXLAN-GPE, Geneve, native IPv6 (via extension header), or IPv4. In-situ OAM can be used to complement current out-of-band OAM mechanisms based on ICMP or other types of probe packets.

Copyright Notice

Table of Contents

1. Introduction

This document defines data fields 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 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. 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 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:

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

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

In-situ OAM deployment assumes a set of contraints, 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, VXLAN-GPE, Geneve, IPv6, or IPv4. Encapsulation 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 perimiter or edge. The operator of such a domain MUST put provisions in place to ensure that in-situ OAM data stays within the specific domain only (i.e., does not leak beyond the edge) and consider potential impact of IOAM to ECMP processing, path MTU and ICMP message handling.

In-situ OAM 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 in-situ OAM is applied to: IOAM can be deployed on all or only on subsets of the live user traffic. 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 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 in-situ OAM SHOULD be defined in a transport-independent manner. In-situ OAM 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, in-situ OAM data-records could be present at every layer. The behavior follows the ships-in-the-night model.

Combination with active OAM mechanisms: In-situ OAM SHOULD be usable for active network probing, enabling 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.

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

4. In-situ OAM Data Types and Formats

This section defines in-situ OAM data types and data fields and associated data types 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 fields 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 outside the scope of this memo, and are discussed in [I-D.brockners-inband-oam-transport]. In-situ OAM 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.

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 an 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.

4.1. 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 number of hops and the minimum path MTU 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 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 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 used 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 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 number of node data fields recorded and maximum number of node data that can be recorded are 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. In-situ OAM 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 in-situ OAM transit node that is traversed by a packet. The in-situ OAM decapsulating node removes the in-situ OAM data and processes and/or exports 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:

- 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.

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

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

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

- Generic data: Format-free information where syntax and semantic of the information is defined by the operator in a specific deployment. For a specific deployment, all in-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.

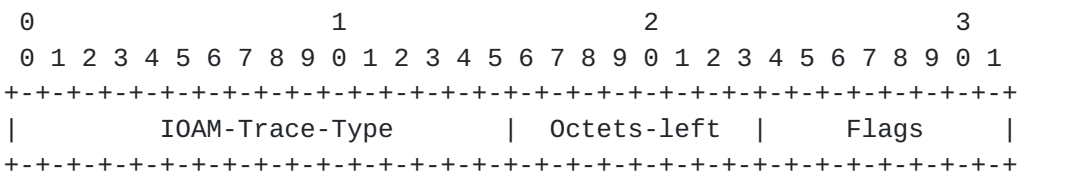
- A mechanism to detect whether in-situ OAM trace data was added at every hop or whether certain hops in the domain weren't in-situ OAM transit nodes.

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.

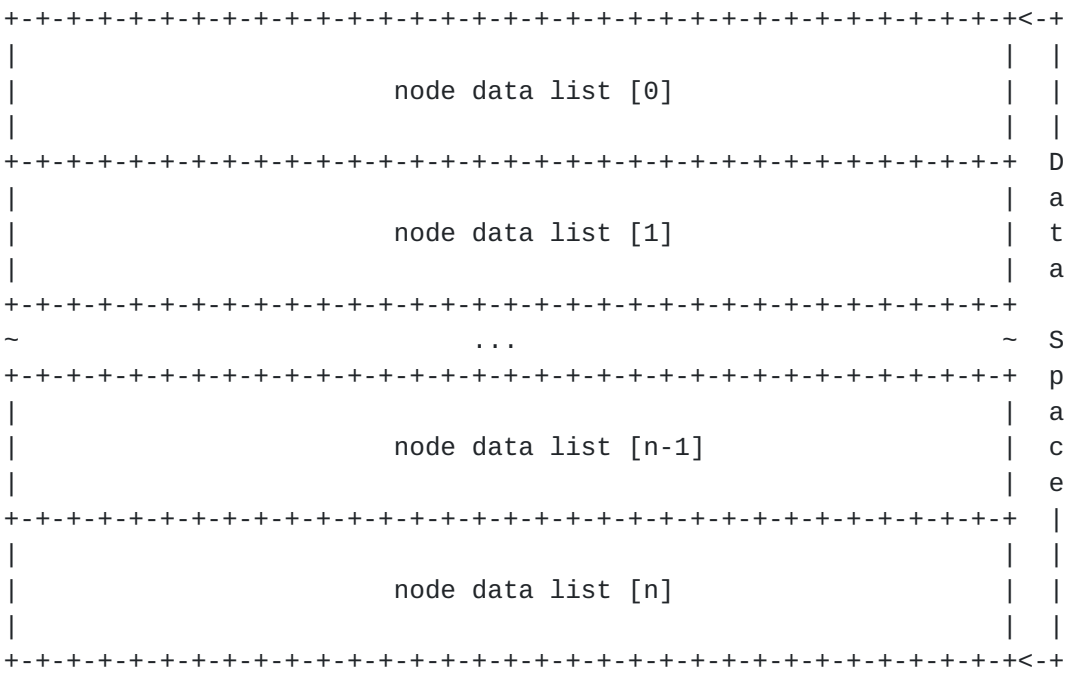
4.1.1. Pre-allocated Trace Option

In-situ OAM Pre-allocated Trace Option:

Pre-allocated Trace Option header:



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



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 (Least 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 11-15 Undefined in this draft.

Section 4.1.4 describes the in-situ OAM 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.

Octets-left: 8-bit unsigned integer. It is the data space in octets remaining for recording the node data. This is used as an offset in octets in data space to record node data element.

Flags 8-bit field. The 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 bytes 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.

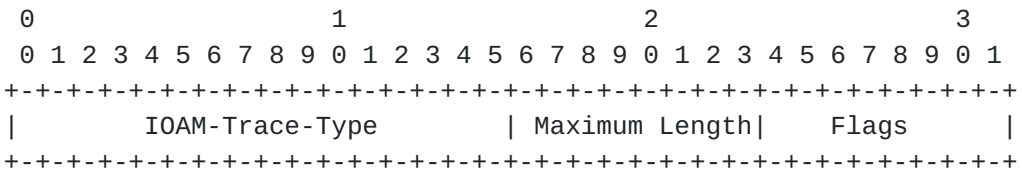
Bit 1 "Loopback" (L-bit). Loopback mode is used to send a copy of a packet back towards the source. Loopback mode assumes that a return path from transit nodes and destination nodes towards the source exists. The encapsulating node decides (e.g. using a filter) which packets loopback mode is enabled 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.

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.

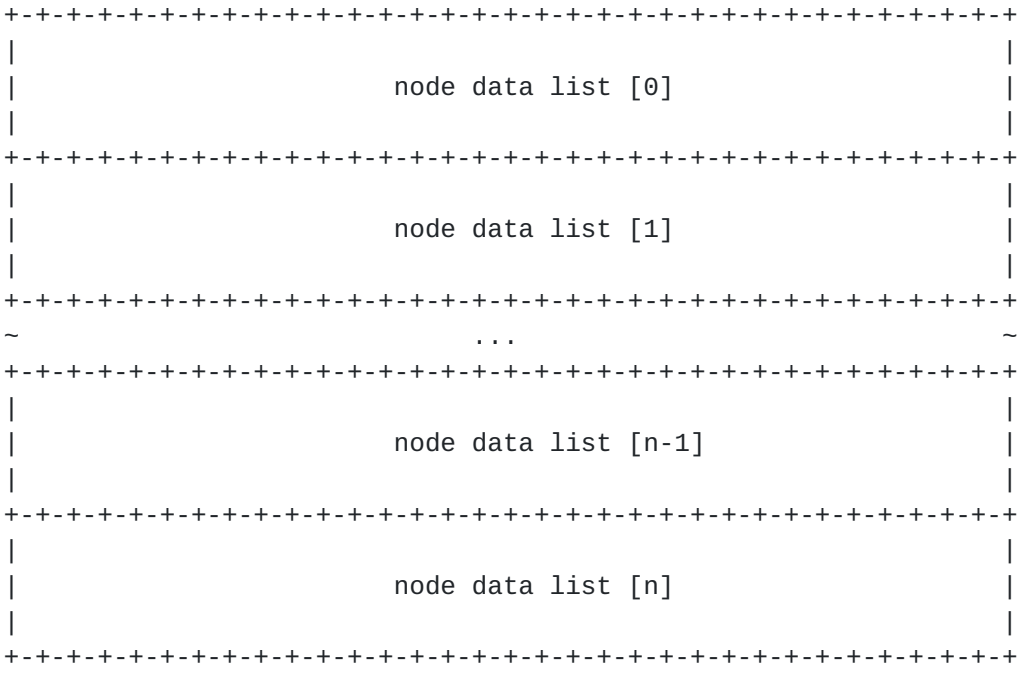
4.1.2. Incremental Trace Option

In-situ OAM Incremental Trace Option: '

In-situ OAM Incremental Trace Option Header:



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



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 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 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 wide in the node data.

Bit 10 When set indicates presence of app_data wide in the node data.

Bit 11-15 Undefined in this draft.

Section 4.1.4 describes the in-situ OAM data types and their formats.

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:

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

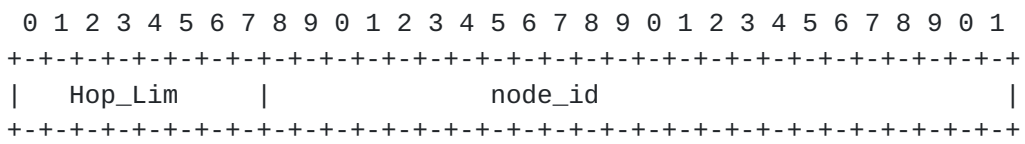
Node data List [n]: Variable-length field. The type of which is determined by the OAM Type representing the n-th node data in the node data list. The node data list is encoded starting from the last node data of the path. The first element of the node data list (node data list [0]) contains the last 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. In-situ OAM node data fields and associated formats

All the data fields MUST be 4-byte 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:

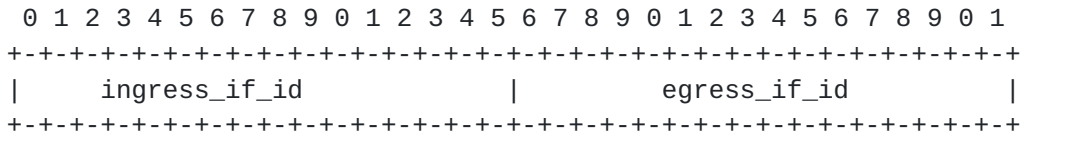
Hop_Lim and node_id: 4-octet field defined as follows:



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.

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.

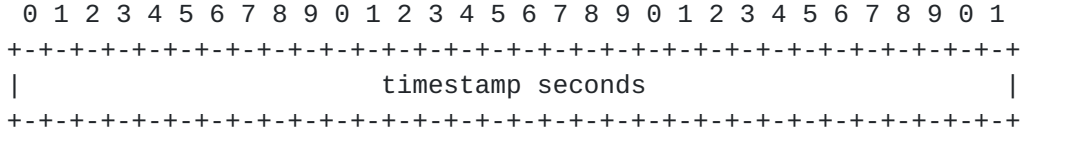
ingress_if_id and egress_if_id: 4-octet field defined as follows: When this field is part of the data field but a node populating the field is not able to fill it, the position in the field must be filled with value 0xFFFF to mean not populated.



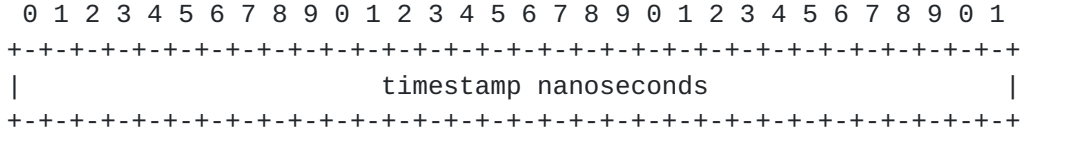
ingress_if_id: 2-octet unsigned integer. Interface identifier to record the ingress interface the packet was received on.

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).

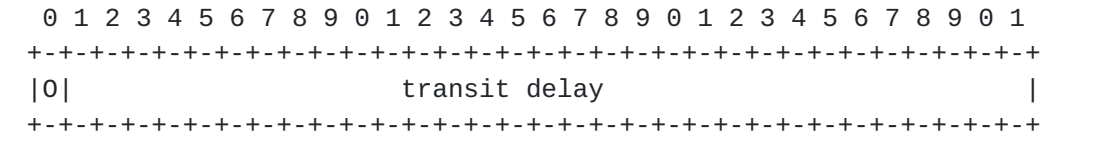


timestamp nanoseconds: 4-octet unsigned integer in the range 0 to 10^9-1. This timestamp specifies the fractional part of the wall clock time at which the packet was received by the node in units of nanoseconds. This field is identical to the 32 least significant bits of the [IEEE1588v2] timestamp. This 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 0xFFFF to mean not populated.



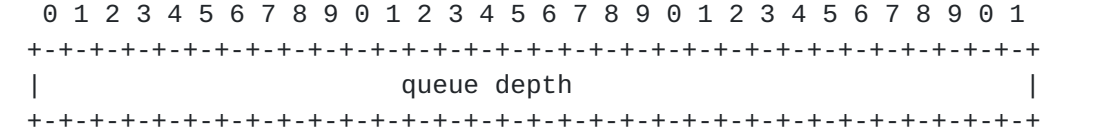
transit delay: 4-octet unsigned integer in the range 0 to 2^30-1. It is the time in nanoseconds the packet spent in the transit

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 0xFFFF to mean not populated.

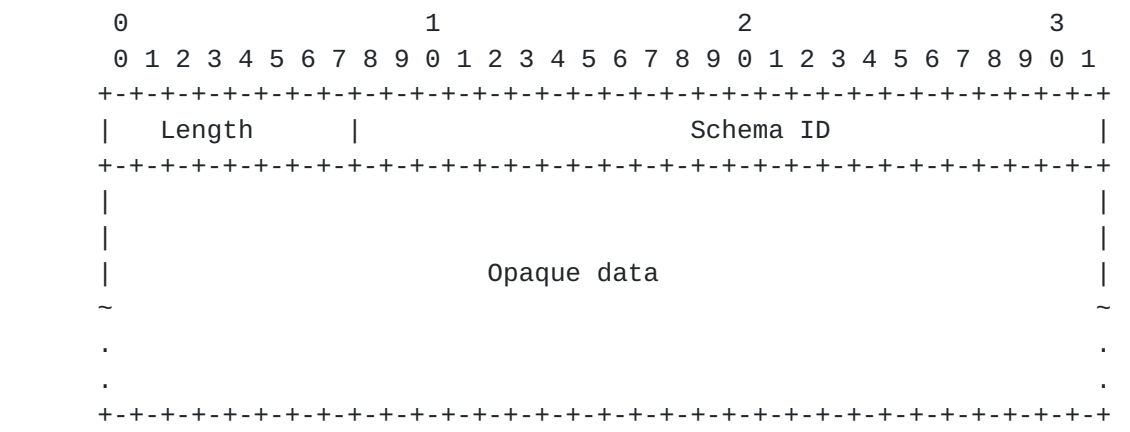


app_data: 4-octet placeholder which can be used by the node to add application specific data

queue depth: 4-octet unsigned integer field. This field indicates the 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 0xFFFF to mean not populated.



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 means. 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.

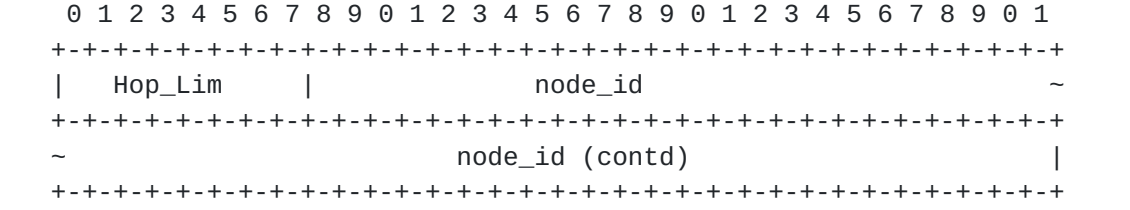


Length: 1-octet unsigned integer. It is the length in octets of the Opaque data field that follows Schema Id. It MUST always be a multiple of 4.

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:

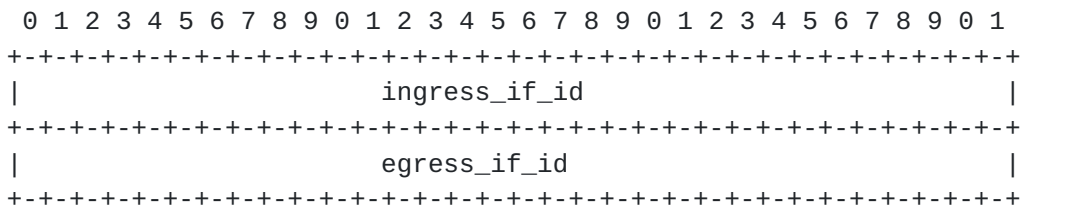


Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop Limit information is used to identify the location of the node in the communication path. This is copied from the lower layer for e.g. TTL value in IPv4 header or hop limit field 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 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 0xFFFFFFFF to mean not populated.



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

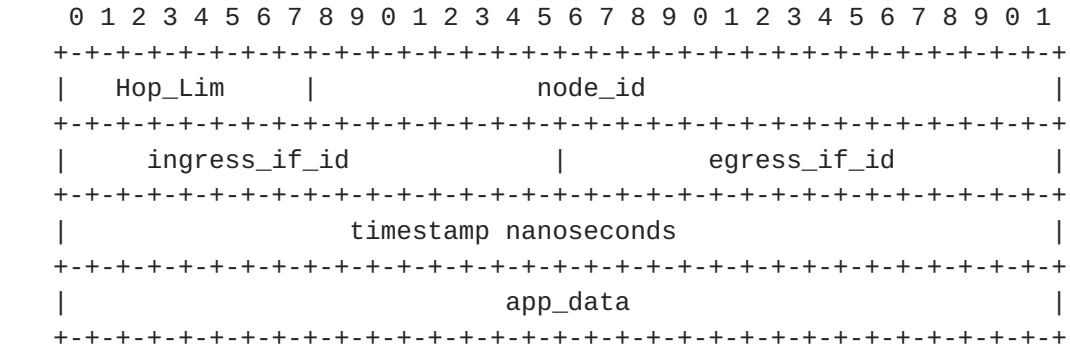
egress_if_id: 4-octet unsigned integer. Interface identifier to record the egress interface the packet is forwarded out of.

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

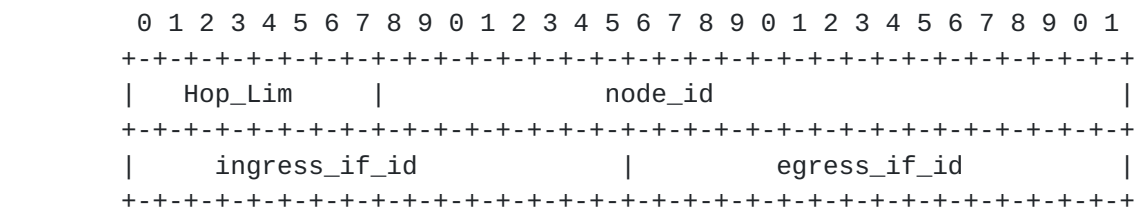
4.1.4. Examples of In-situ OAM node data

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 types that an entry in "node data list" can take.

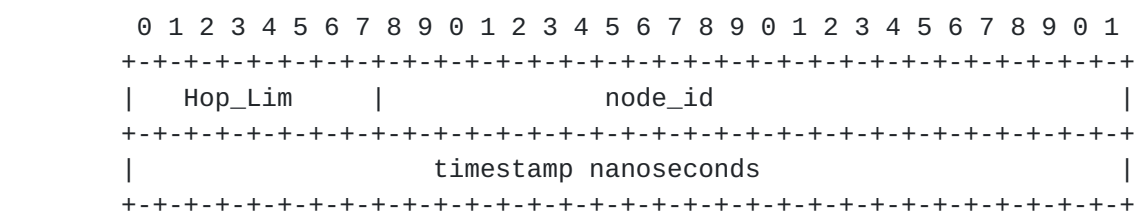
0x002B: IOAM-Trace-Type is 0x2B then the format of node data is:



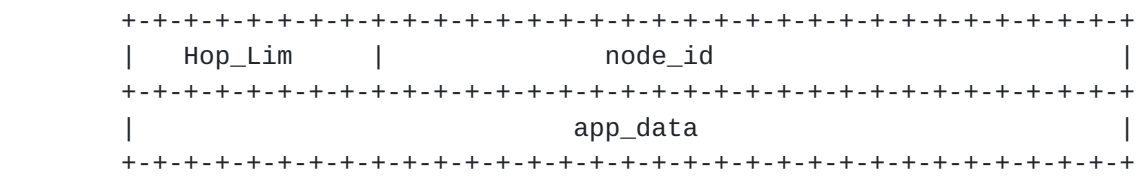
0x0003: IOAM-Trace-Type is 0x0003 then the format is:



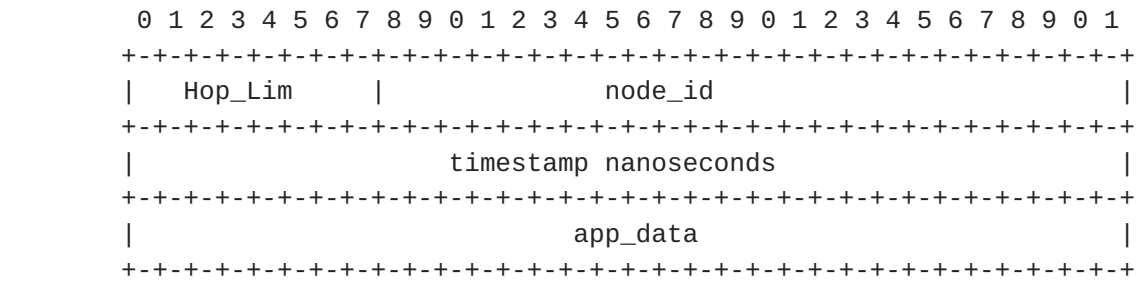
0x0009: IOAM-Trace-Type is 0x0009 then the format is:



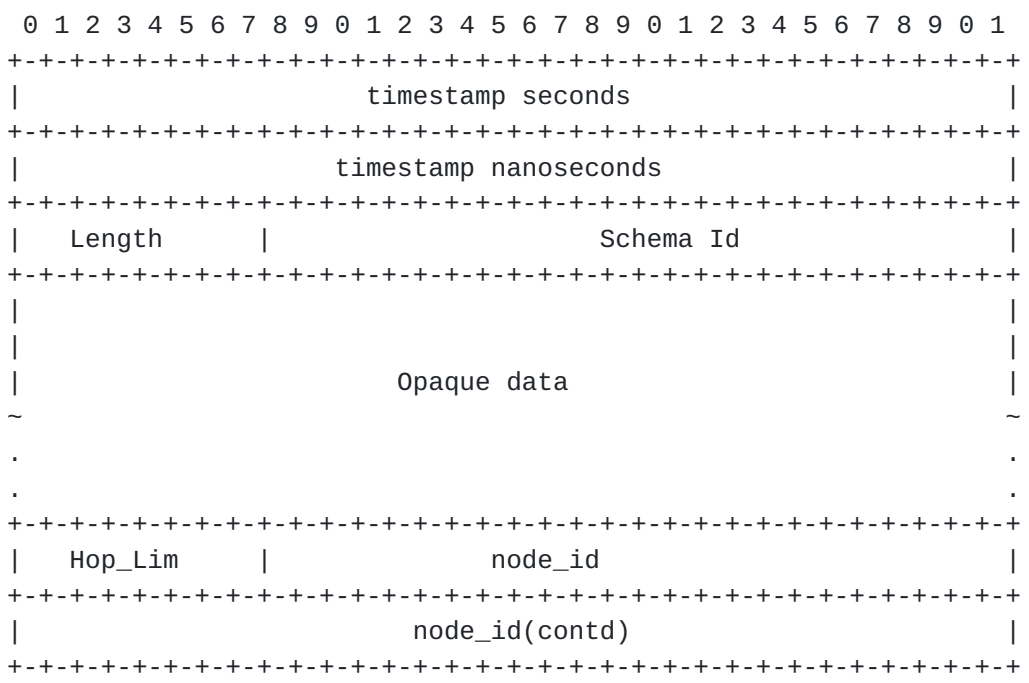
0x0021: IOAM-Trace-Type is 0x0021 then the format is:



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



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



4.2. In-situ OAM Proof of Transit Option

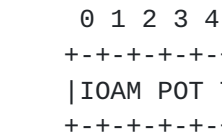
In-situ OAM Proof of Transit data is to support the path or service function chain [RFC7665] verification use cases. Proof-of-transit uses methods like nested hashing or nested encryption of the in-situ OAM data or mechanisms such as Shamir's Secret Sharing Schema (SSSS). While details on how the in-situ 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:

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

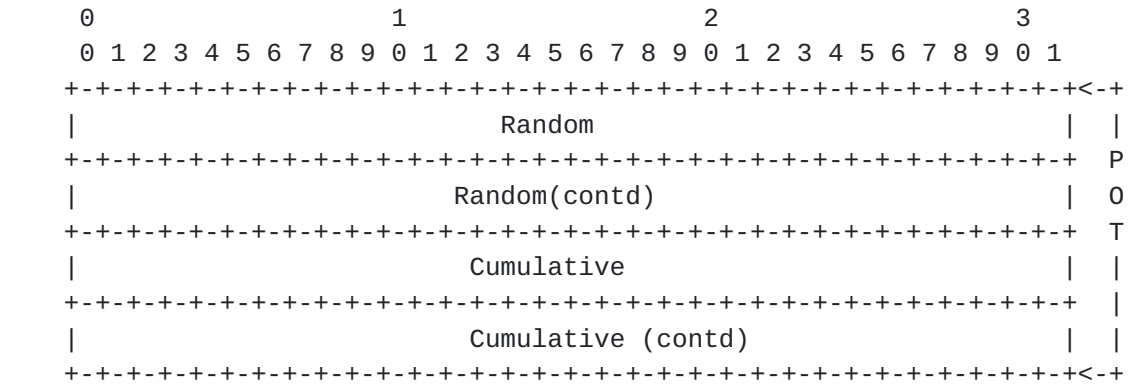
- Cumulative: Information which is handed from node to node and updated by every node according to a verification algorithm.

In-situ OAM Proof of Transit Option:

In-situ OAM Proof of Transit Option Header:



In-situ OAM Proof of Transit Option Data MUST be 4-byte aligned:



IOAM POT Type: 7-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.

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".

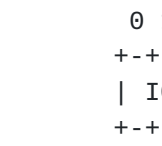
4.3. In-situ OAM Edge-to-Edge Option

The in-situ OAM Edge-to-Edge Option is to carry data that is added by the in-situ OAM encapsulating node and interpreted by in-situ OAM decapsulating node. The in-situ OAM transit nodes MAY process the data without modifying it.

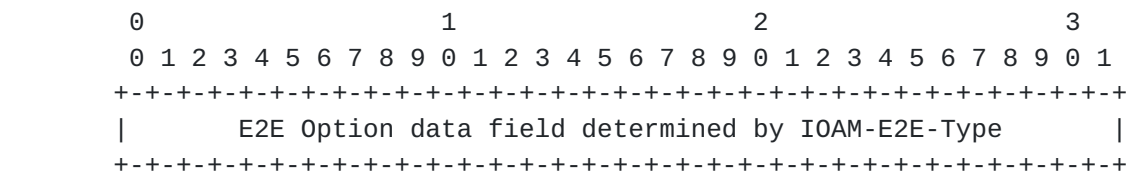
Currently only sequence numbers use the in-situ OAM Edge-to-Edge option. In order to detect packet loss, packet reordering, or packet duplication in an in-situ OAM-domain, sequence numbers can be added to packets of a particular tube (see [I-D.hildebrand-spud-prototype]). Each tube leverages a dedicated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

In-situ OAM Edge-to-Edge Option:

In-situ OAM Edge-to-Edge Option Header:



In-situ OAM Edge-to-Edge Option Data MUST be 4-byte aligned:



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. 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.

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

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.

10. References

10.1. Normative 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.

[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>.

[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>.

[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>.

10.2. Informative References

[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-03 (work in progress), October 2016.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-12 (work in progress), February 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>.

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