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

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

July 8, 2016

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

Abstract

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

Status of This Memo

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

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

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

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

Table of Contents

1. Introduction

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

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

2. Conventions

Abbreviations used in this document:

MTU: Maximum Transmit Unit

OAM: Operations, Administration, and Maintenance

SR: Segment Routing

SID: Segment Identifier

NSH: Network Service Header

SFC: Service Function Chain

TLV: Type-Length-Value

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

3. In-band OAM Data Types and Data Format

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

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

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

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

3.1. In-band OAM Tracing Option

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

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

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

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

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

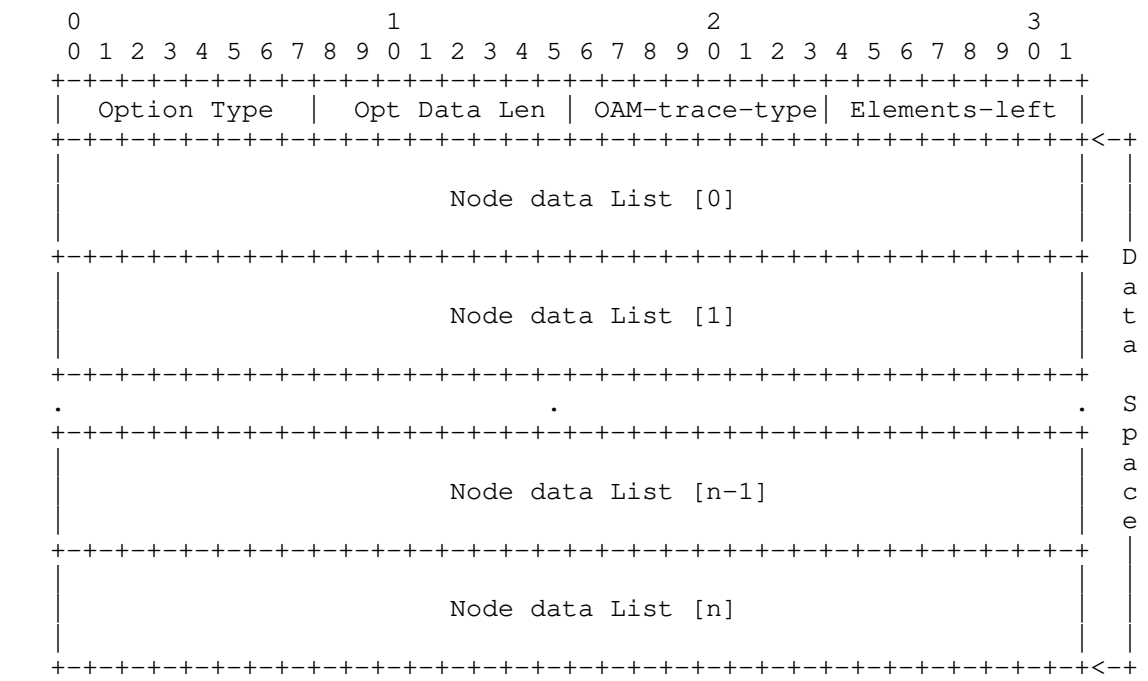
- Time of day when the packet was processed by the node. Different definitions of processing time are feasible and expected, though it is important that all devices of an in-band OAM domain follow the same definition.

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

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

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

In-band OAM Tracing Option:



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

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

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

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

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

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

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

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

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

Bit 5-7 Undefined in this document.

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

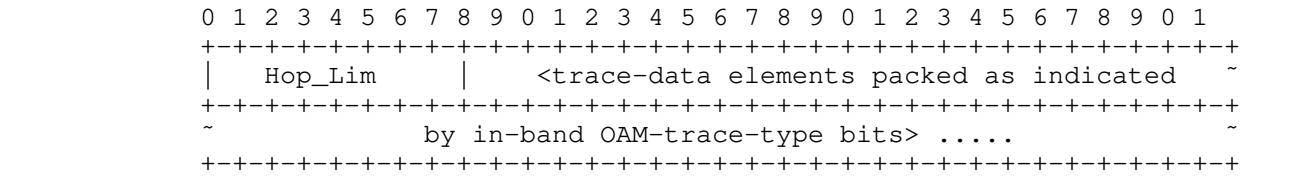
Elements-left: 8-bit unsigned integer. A pointer that indicates the next data recording point in the data space of the packet in octets. It is the index into the "Node data List" array shown above.

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

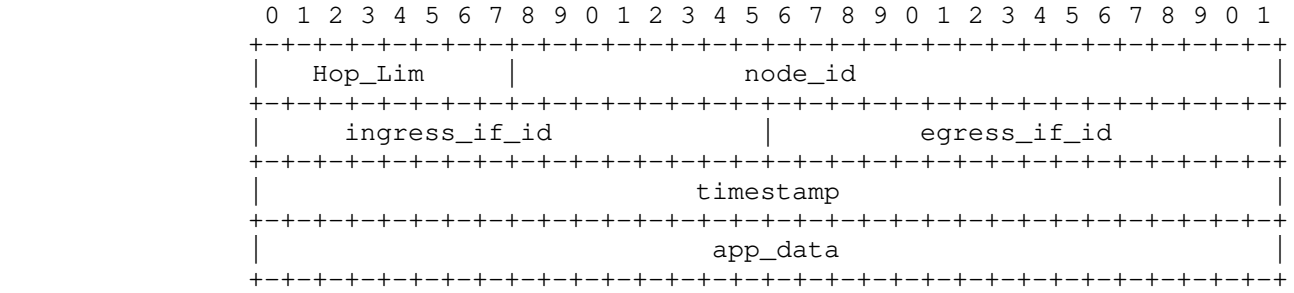
3.1.1. In-band OAM Trace Type and Node Data Element

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

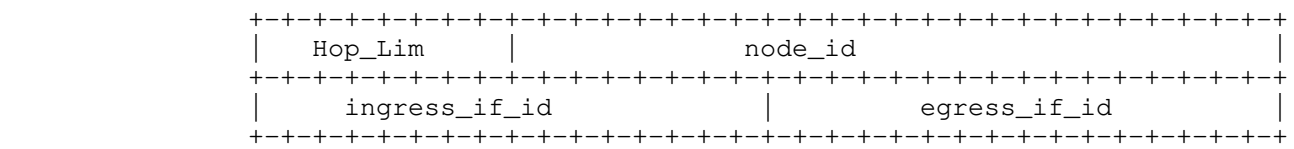
Node data has the following format:



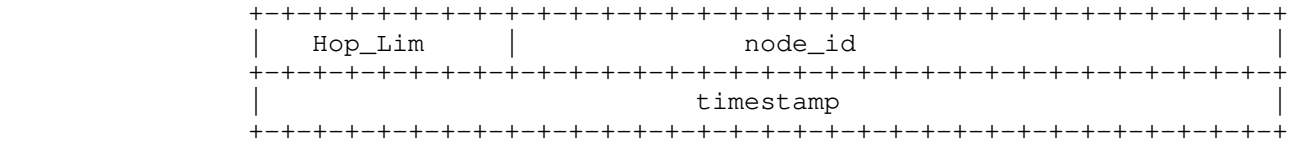
0x00011111: In-band OAM-trace-type is 0x00011111 then the format of node data is:



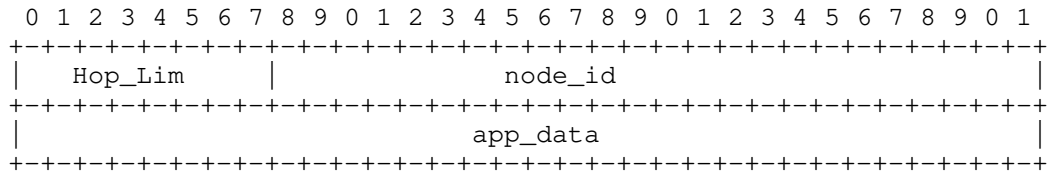
0x00000111: In-band OAM-trace-type is 0x00000111 then the format is:



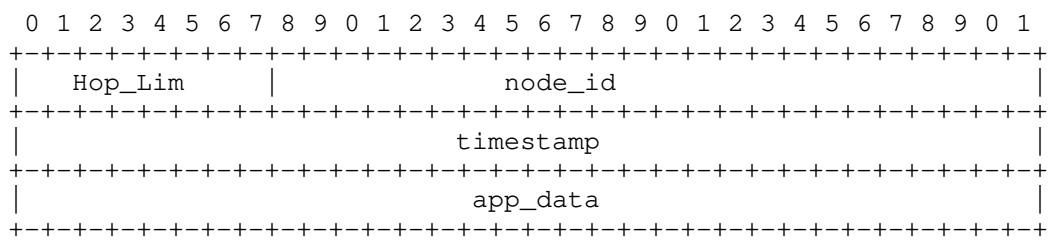
0x00001001: In-band OAM-trace-type is 0x00001001 then the format is:



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



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



Trace data elements in Node data are defined as follows:

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

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

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

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

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

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

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

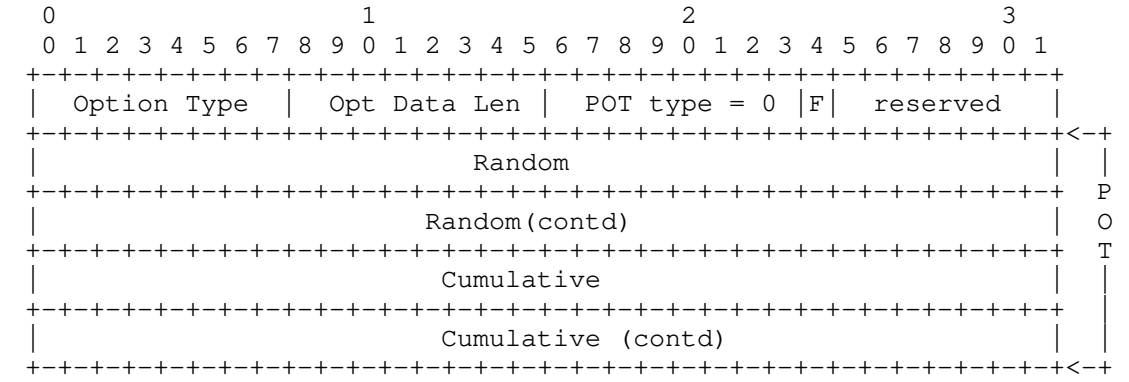
3.2. In-band OAM Proof of Transit Option

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

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

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

In-band OAM Proof of Transit option:



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

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

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

* 16 Octet field as described below

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

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

Random: 64-bit Per packet Random number.

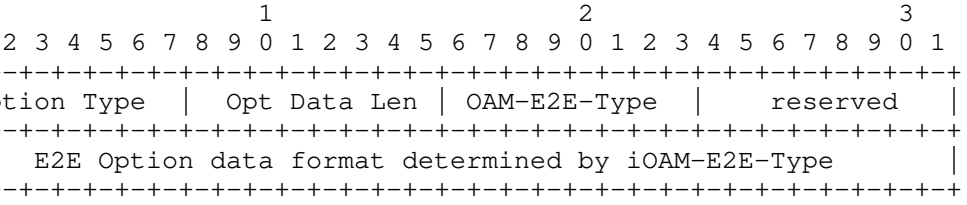
Cumulative: 64-bit Cumulative that is updated at specific nodes by processing per packet Random number field and configured parameters.

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

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

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

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



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

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

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

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

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

4. In-band OAM Data Export

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

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

5. IANA Considerations

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

6. Manageability Considerations

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

7. Security Considerations

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

8. Acknowledgements

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

9. References

9.1. Normative References

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

9.2. Informative References

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

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

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

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

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

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

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

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

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

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

July 8, 2016

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

Abstract

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

Table of Contents

1. Introduction

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

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

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

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

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

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

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

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

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

2. Conventions

Abbreviations used in this document:

ECMP: Equal Cost Multi-Path

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

OAM: Operations, Administration, and Maintenance

PMTU: Path MTU

SLA: Service Level Agreement

SFC: Service Function Chain

SR: Segment Routing

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

3. Motivation for In-band OAM

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

3.1. Path Congruency Issues with Dedicated OAM Packets

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

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

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

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

3.2. Results Sent to a System Other Than the Sender

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

3.3. Overlay and Underlay Correlation

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

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

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

3.4. SLA Verification

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

3.5. Analytics and Diagnostics

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

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

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

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

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

3.7. Proof of Transit

Several deployments use traffic engineering, policy routing, segment routing or Service Function Chaining (SFC) [RFC7665] to steer packets through a specific set of nodes. In certain cases regulatory obligations or a compliance policy require to prove that all packets that are supposed to follow a specific path are indeed being forwarded across the exact set of nodes specified. If a packet flow is supposed to go through a series of service functions or network nodes, it has to be proven that all packets of the flow actually went through the service chain or collection of nodes specified by the policy. In case the packets of a flow weren't appropriately processed, a verification device would be required to identify the policy violation and take corresponding actions (e.g., drop or redirect the packet, send an alert etc.) corresponding to the policy. In today's deployments, the proof that a packet traversed a particular service chain is typically delivered in an indirect way: Service appliances and network forwarding are in different trust domains. Physical hand-off-points are defined between these trust domains (i.e., physical interfaces). Or in other terms, in the "network forwarding domain" things are wired up in a way that traffic is delivered to the ingress interface of a service appliance and received back from an egress interface of a service appliance. This "wiring" is verified and trusted. The evolution to Network Function Virtualization (NFV) and modern service chaining concepts (using

technologies such as LISP, NSH, Segment Routing, etc.) blurs the line between the different trust domains, because the hand-off-points are no longer clearly defined physical interfaces, but are virtual interfaces. Because of that very reason, networks operators require that different trust layers not to be mixed in the same device. For an NFV scenario a different proof is required. Offering a proof that a packet traversed a specific set of service functions would allow network operators to move away from the above described indirect methods of proving that a service chain is in place for a particular application.

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

3.8. Use Cases

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

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

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

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

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

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

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

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

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

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

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

4. Considerations for In-band OAM

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

4.1. Type of Information to Be Recorded

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

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

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

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

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

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

4.2. MTU and Packet Size

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

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

4.3. Administrative Boundaries

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

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

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

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

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

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

4.4. Selective Enablement

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

4.5. Optimization of Node and Interface Identifiers

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

4.6. Loop Communication Path (IPv6-specifics)

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

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

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

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

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

5. Requirements for In-band OAM Data Types

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

5.1. Generic Requirements

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

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

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

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

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

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

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

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

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

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

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

REQ-H5: Time-related requirements

REQ-H5.1: Delay: Data shall be present that allows to retrieve the delay between two or more points of interest within the system. Those points can be within the same device or on different devices.

REQ-H5.2: Jitter: Data shall be present that allows to retrieve the jitter between two or more points of interest within the system. Those points can be within the same device or on different devices.

REQ-H5.3: Wall-clock time: Data shall be present that allows to retrieve the wall-clock time visited a particular point of interest in the system.

REQ-H5.4: Time precision: The precision of the time related data should be configurable. Use-case dependent, the required precision could e.g., be nano-seconds, micro-seconds, milli-seconds, or seconds.

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

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

5.3. In-band OAM with Selected Hop Scope

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

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

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

REQ-E1: Sequence numbering:

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

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

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

6. Security Considerations and Requirements

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

6.1. Proof of Transit

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

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

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

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

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

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

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

7. IANA Considerations

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

This document has no IANA actions.

8. Acknowledgements

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

9. Informative References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: sadara@cisco.com

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

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

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

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

July 8, 2016

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

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

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

2. Conventions

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

Abbreviations used in this document:

MTU: Maximum Transmit Unit

OAM: Operations, Administration, and Maintenance

SR: Segment Routing

SID: Segment Identifier

NSH: Network Service Header

POT: Proof of Transit

SFC: Service Function Chain

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

3. In-Band OAM Metadata Transport in IPv6

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

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

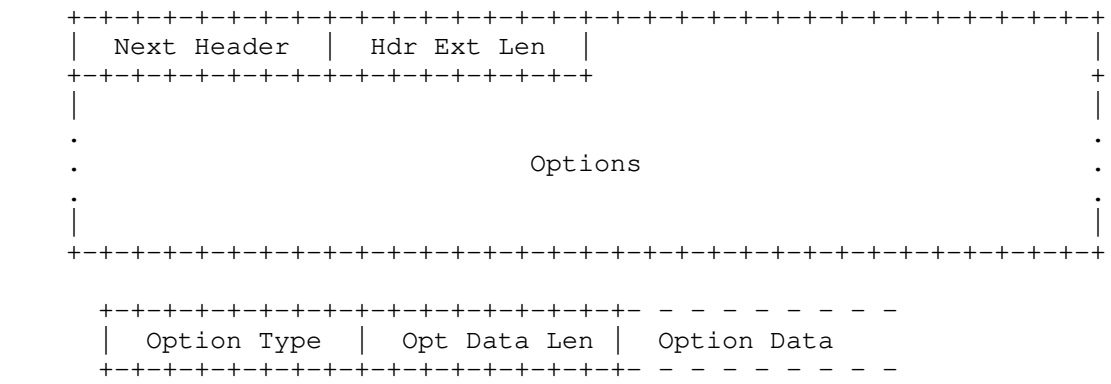
The in-band OAM options translate into options for an IPv6 extension header. The extension header would be inserted by either a host source of the packet, or by a transit/domain-edge node.

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

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

3.1.1. In-band OAM Hop by Hop Options

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



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

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

01 - discard the packet.

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

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

3rd highest bit:

0 - Option Data does not change en-route

1 - Option Data may change en-route

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

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

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

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

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

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

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

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

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

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

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

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

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

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

 }
```

3.1.3. Procedure at Intermediate Nodes

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

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

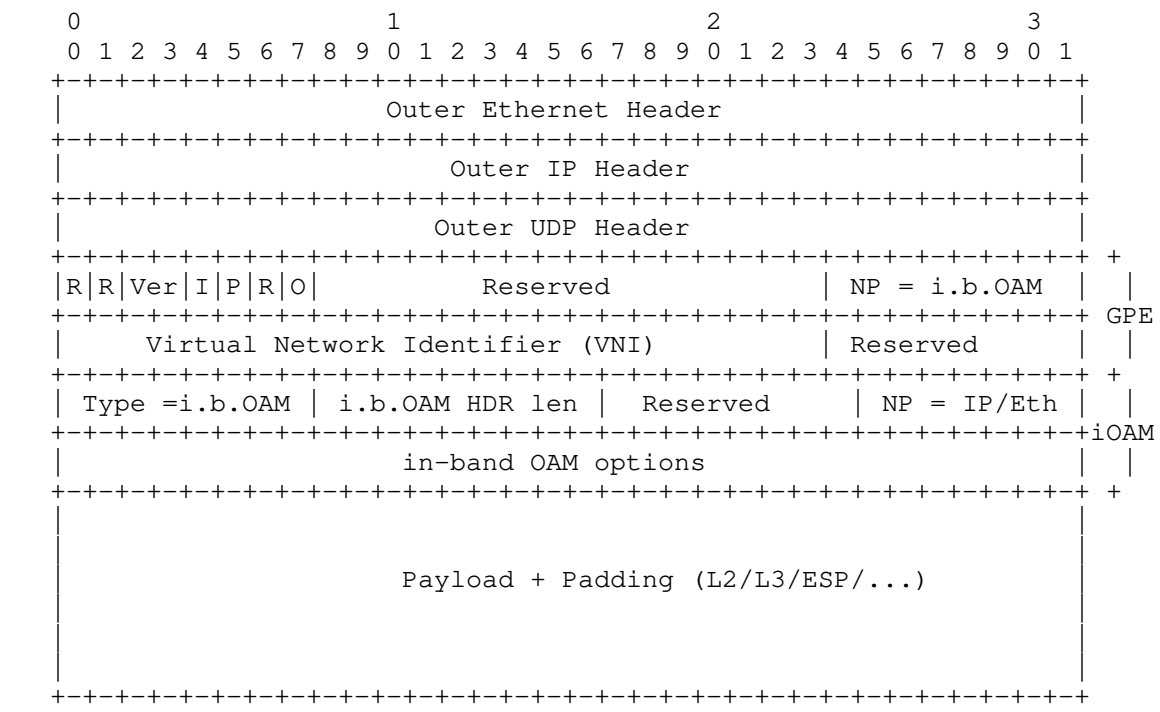
3.1.4. Procedure at the Egress Edge to Remove the In-band OAM Header

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

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

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

In-band OAM header in VXLAN GPE header:

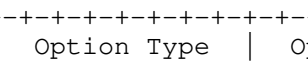


The VXLAN-GPE header and fields are defined in [I-D.ietf-nvo3-vxlan-gpe]. in-band OAM specific fields and header are defined here:

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

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

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



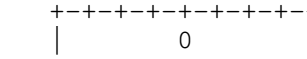
Option Type 8-bit identifier of the type of option.

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

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

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

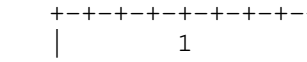
Pad1 option (alignment requirement: none)



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

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

PadN option (alignment requirement: none)

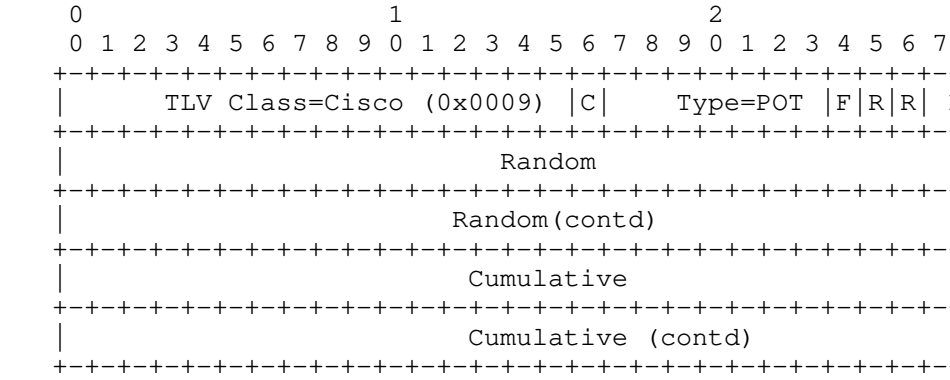


The PadN option is used to insert two or more octets of padding into the Options area of a header. For N octets of padding, the Opt Data Len field contains the value N-2, and the Option Data consists of N-2 zero-valued octets.

5. In-band OAM Metadata Transport in NSH

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

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



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

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

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

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

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

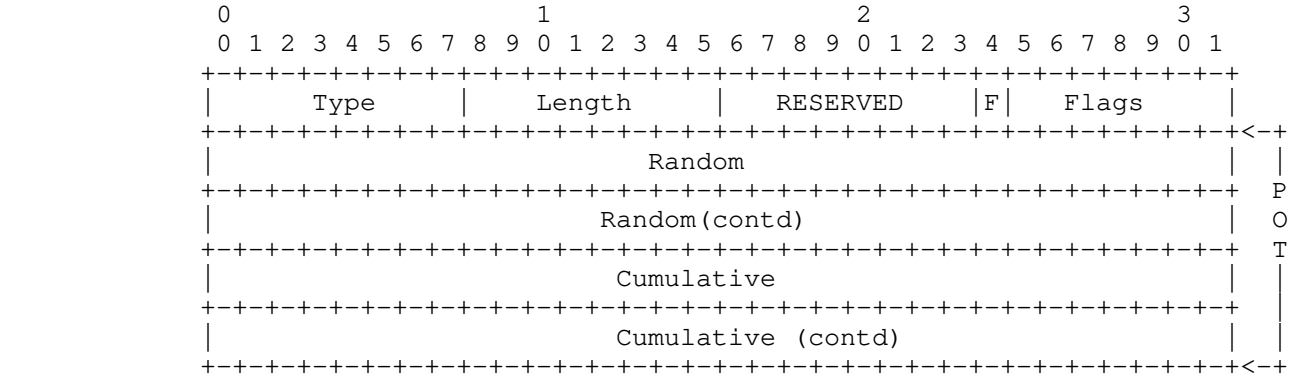
Random: 64-bit Per packet Random number.

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

6. In-band OAM Metadata Transport in Segment Routing

6.1. In-band OAM in SR with IPv6 Transport

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



Type: To be assigned by IANA.

Length: 18.

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

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

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

Random: 64-bit per packet random number.

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

6.2. In-band OAM in SR with MPLS Transport

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

7. IANA Considerations

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

8. Manageability Considerations

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

9. Security Considerations

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

10. Acknowledgements

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

11. References

11.1. Normative References

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

11.2. Informative References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

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

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

July 8, 2016

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

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

2. Conventions

Abbreviations used in this document:

MTU: Maximum Transmit Unit

SR: Segment Routing

NSH: Network Service Header

SFC: Service Function Chain

POT: Proof of Transit

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

3. Proof of Transit

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

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

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

3.1. Basic Idea

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

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

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

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

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

3.2. Solution Approach

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

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

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

3.2.1. Setup

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

3.2.2. In Transit

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

3.2.3. Verification

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

3.3. Example for Illustration

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

3.3.1. Basic Version

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

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

3.3.1.1. Secret Shares

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

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

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

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

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

3.3.1.2. Lagrange Polynomials

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

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

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

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

3.3.1.3. LPC Computation

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

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

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

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

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

3.3.1.4. Reconstruction

Reconstruction of the polynomial is well defined as

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

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

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

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

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

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

3.3.1.5. Verification

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

3.3.2. Enhanced Version

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

3.3.2.1. Random Polynomial

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

3.3.2.2. Reconstruction

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

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

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

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

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

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

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

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

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

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

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

The verification using CML is discussed in next section.

3.3.2.3. Verification

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

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

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

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

3.4. Operational Aspects

To operationalize this scheme, a central controller is used to generate the necessary polynomials, the secret share per node, the prime number, etc. and distributing the data to the nodes participating in proof of transit. The identified node that performs the verification is provided with the verification key. The information provided from the Controller to each of the nodes participating in proof of transit is referred to as a proof of transit profile (POT-profile).

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

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

2. A pre-evaluated portion of the public polynomial for each of the nodes is calculated and added to the POT-profile. Without this all the coefficients of the public polynomial had to be added to the POT profile and each node had to evaluate them.

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

4. Sizing the Data for Proof of Transit

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

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

2. CML: Cumulative

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

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

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

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

5. Node Configuration

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

5.1. Procedure

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

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

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

5.2. YANG Model

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

```
module ietf-pot-profile {

  yang-version 1;

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

  prefix ietf-pot-profile;

  organization "IETF xxx Working Group";

  contact "";

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

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

```
    }

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


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

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

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

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

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

        leaf lpc {
```

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

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

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

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

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

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

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

        leaf active-profile-index {
```

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

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

6. IANA Considerations

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

7. Manageability Considerations

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

8. Security Considerations

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

8.1. Proof of Transit

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

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

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

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

8.2. Anti Replay

A passive attacker observing CML values across nodes (i.e., as the packets entering and leaving), cannot perform differential analysis to construct the points on POLY-1 as the operations are done modulo prime. The solution approach is flexible, one could use different points on POLY-1 or different polynomials as POLY-1 across different paths, traffic profiles or service chains.

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

8.3. Anti Tampering

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

8.4. Recycling

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

8.5. Redundant Nodes and Failover

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

8.6. Controller Operation

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

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

8.7. Verification Scope

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

8.7.1. Node Ordering

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

8.7.2. Stealth Nodes

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

9. Acknowledgements

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

10. Normative References

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

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: sadara@cisco.com

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

Email: cpignata@cisco.com

OPSA Working Group
Internet-Draft
Intended status: Informational
Expires: January 19, 2017

H. Deng
China Mobile

July 18, 2016

Requirements of Composed VPN Service Model
draft-deng-opsawg-composed-vpn-sm-requirements-01

Abstract

The operator facing data model is valuable to reduce the operation and management. This document describes requirements of the composed VPN service model for operators to deploy end to end PE-based VPN services across multiple autonomous systems.

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 January 19, 2017.

Copyright Notice

Table of Contents

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 across multiple autonomous systems (AS) with an orchestrator.

The model 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 know.

However, the network operators have a different view of the managed network. An operator facing model is required to reduce the operation and management while still having reasonable control on the network. So that the operators can verify and optimize the VPN deployment based on the existing network.

This document describes requirements of the generic VPN 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 model is limited to PE-Based VPNs as described in RFC 4110 [RFC4110] with the combination of layer 2 and layer 3 VPN services in multiple ASes.

2. Definitions

- Segment VPN service: The VPN service deployed for one segment which is usually an AS.

- Composed 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 composed VPN service model design. The composed VPN service data model described in this document covers the following scenarios:

- Multi-AS VPN Service: Customer sites are located in different autonomous systems(AS). ISP need to deploy the VPN service across multiple ASes.

- Composed L2 and 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.

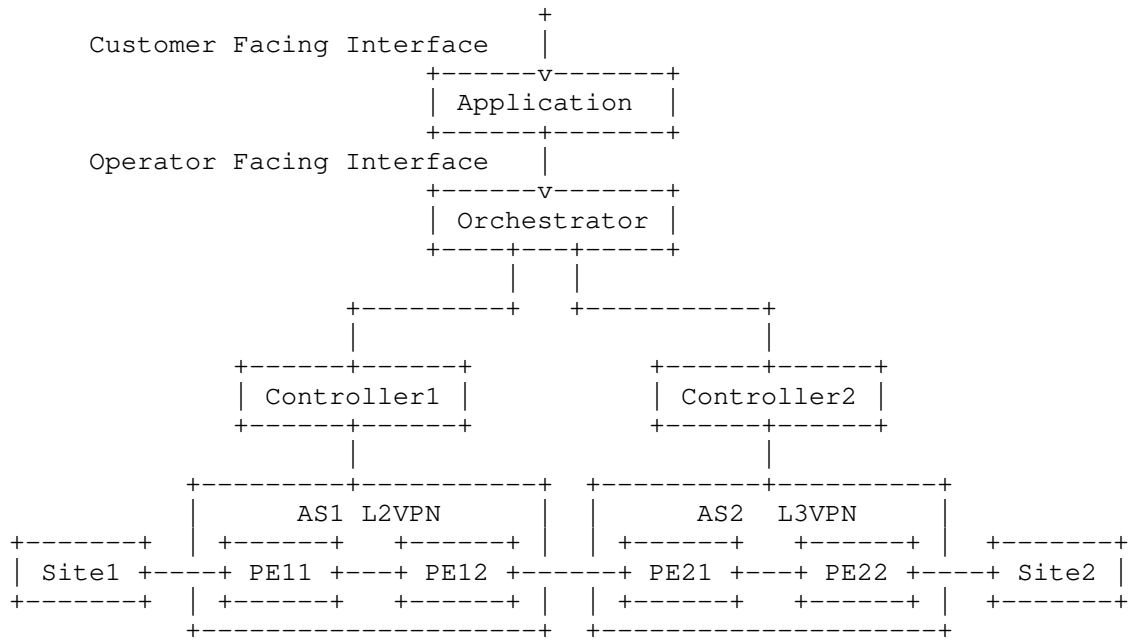
- Dynamic Site Insertion: The customer site that is not in the previously provisioned VPN can be quickly included.

A typical usage of this operator facing model is as an input for an orchestration layer which will be responsible to translate it to segment VPN information for the configutation of domain controllers. As shown in the following figure, while, for example, users may send highly abstracted layer 3 VPN service requests to the application (e.g. BSS), it's not enough for operators to deploy an end to end VPN service. The operator facing interface enables configuration of VPN deployment by introducing more network knowlegde and garvenance policies. For example :

- 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 (IPRAN, PTN, IPCore) in the end to end environment.

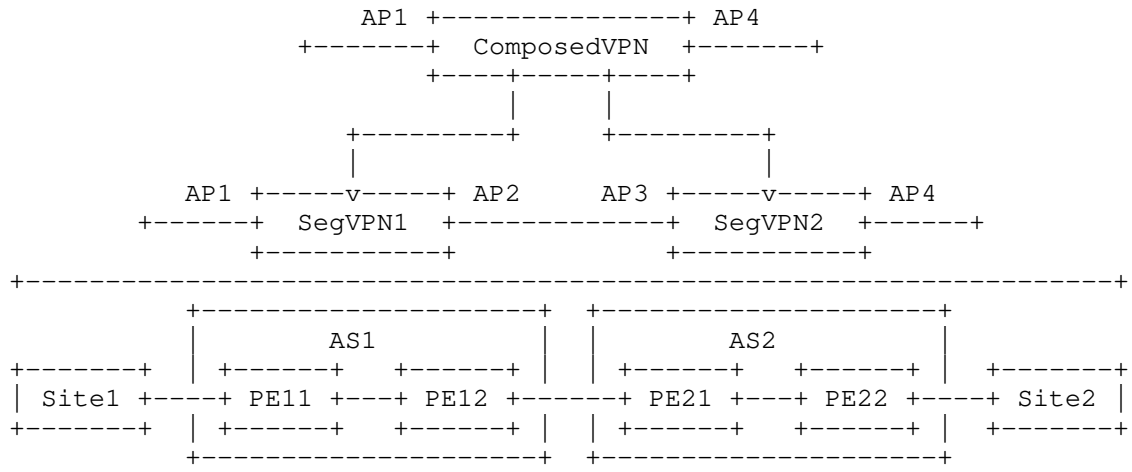
- Add the operation requirements, e.g. operation visualization, monitoring, diagnosis.

- Manage various policies for different customers.



4. Design Requirements

The PE-based VPN service is modeled with a recursive pattern as shown in the following figure. The VPN service deployed within each AS is modeled as a Segment 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 Composed VPN object with the overall VPN information and the APs that are used to connect to the peered customer sites.



Generic PE-based VPN Modeling

The composed VPN model can be structured as in the following figure. The Composed VPN top container contains VPN basic information, a list of segment VPN information, and a list of access point information.

The Basic Information here includes overall description for this composed VPN service. I.e., all the properties (e.g., topology, service type) in this object describe the overview that the customer want, no matter with any segment VPN information.

The Access Point List in the Composed VPN container describes a list of APs that are used to connect to the peered customer sites. However, the AP is modeled with generic Access Point Information provided by the PE either in the composed VPN view or in the segment VPN view. The AP contains:

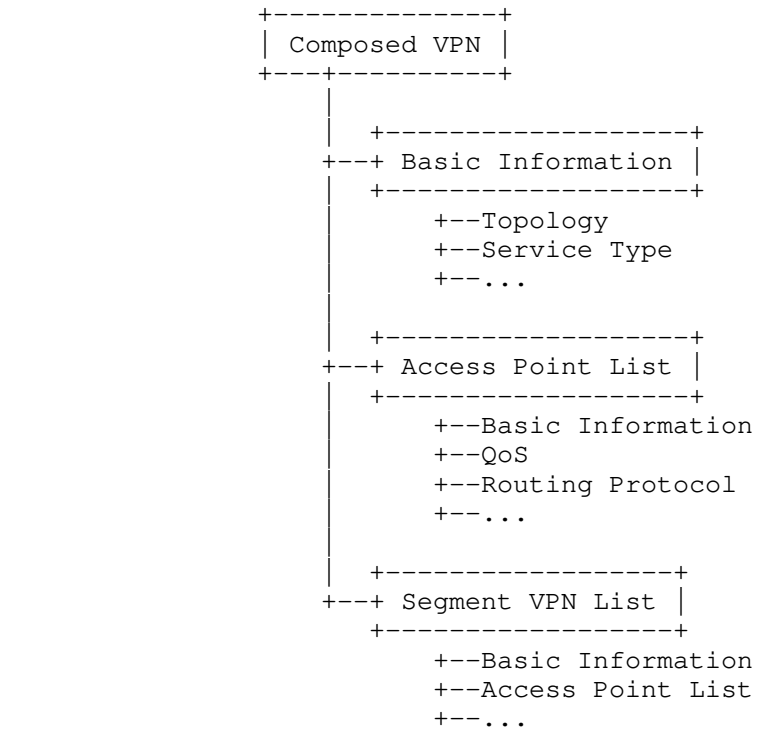
- o the basic information that is relatively static, no matter which exact peer AP is going to connect.

- o the information about the routing protocol that is used to exchange the routing information with the remote peer. This object is extensible with any posible routing protocols. The BGP and static routing listed are examples to show how these two widely used solutions are described.

- o the QoS description. 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.

A composed VPN includes one or more segment VPN desribed by the Segment VPN List. Each Segment VPN Information is only described from the segment point of view. I.e., the description here takes care about how the segment VPN looks like and how it can communicate with peered devices outside this segment VPN. The segment information is composed of the basic 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 segment VPN could be a layer 2 VPN, or layer 3 VPN.



Composed VPN Model Structure

5. IANA Considerations

TBD

6. Security Considerations

TBD

7. Acknowledgements

TBD

8. References

8.1. 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>.

8.2. Informative References

[I-D.ietf-l3sm-l3vpn-service-model] Litkowski, S., Shakir, R., Tomotaki, L., Ogaki, K., and K. D'Souza, "YANG Data Model for L3VPN service delivery", draft-ietf-l3sm-l3vpn-service-model-12 (work in progress), July 2016.

Authors' Addresses

Hui Deng
China Mobile
No.32 Xuanwumen West Street
Beijing 100053
China

Email: denghui02@hotmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 2, 2017

E. Lear
R. Droms
Cisco Systems
D. Romascanu
Avaya

August 01, 2016

Manufacturer Usage Description Specification
draft-lear-ietf-netmod-mud-04

Abstract

This memo specifies the necessary components to implement manufacturer usage descriptions (MUD). This includes two YANG modules, IPv4 and IPv6 DHCP options, an LLDP TLV, a URL suffix specification, 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 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 February 2, 2017.

Copyright Notice

Table of Contents

1. Introduction

Manufacturer Usage Descriptions (MUDs) provide advice to end networks on how to treat specific classes of devices. The MUD architecture is explained in [I-D.lear-mud-framework]. 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 [RFC6020] because of the time and effort spent to develop accurate and adequate models for use by network devices. JSON is used as a serialization for compactness and readability.

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 second 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.

In this memo three means are defined to emit the MUD URL. One is a DHCP option[RFC2131],[RFC3315] that the DHCP client uses to inform the DHCP server. The DHCP server may take further actions, such as retrieve the URL or otherwise pass it along to network management system or controller. Finally, an LLDP frame is defined.

The other method defined is 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.

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 therefore, mature. Therefore, only a limited subset of NETCONF-like content is permitted.

1.1. Terminology

MUD: manufacturer usage description.

MUD file: a file containing YANG-based JSON that describes a recommended behavior.

MUD file server: a web server that hosts a MUD file.

MUD controller: 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 URL: a URL that can be used by the MUD controller to receive the MUD file.

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. The MUD Model and Semantic Meaning

A MUD file consists of JSON based on a YANG model. For purposes of MUD, the elements that can be modified are access lists as augmented by this model. The MUD file is limited to the serialization of a small number of YANG schema, including the models specified in the following documents:

- [I-D.ietf-netmod-acl-model]
- [RFC6991]

Publishers of MUD files MUST NOT include other elements except as described in Section 13, and MUST only contain information relevant to the device being described. Devices parsing MUD files MUST cease processing if they find other elements.

This module is structured into three parts. The first container holds information that is relevant to retrieval and validity of the MUD file itself. The second container augments the access list to indicate direction the ACL is to be applied. The final container augments the matching container of the ACL model to add several elements that are relevant to the MUD URL, or other otherwise abstracted for use within a local environment.

```
module: ietf-mud
  +--rw meta-information
      +--rw last-update?         yang:date-and-time
      +--rw previous-mud-file?   yang:uri
      +--rw cache-validity?      uint32
      +--rw masa-server?         inet:uri
      +--rw is-supported?        boolean
augment /acl:access-lists/acl:acl:
   +--rw packet-direction?   direction
augment /acl:access-lists/acl:acl
        /acl:access-list-entries/acl:ace/acl:matches:
   +--rw manufacturer?          inet:host
   +--rw same-manufacturer?     empty
   +--rw model?                 string
   +--rw local-networks?        empty
   +--rw controller?            inet:uri
   +--rw direction-initiated?   direction
```

3. Element Definitions

The following elements are defined.

3.1. last-update

This is a date-and-time value of the last time the MUD file was updated. This is akin to a version number. Its form is taken from [RFC6991] which, for those keeping score, turn was taken from Section 5.6 of [RFC3339], which was taken from [ISO.8601.1988].

3.2. previous-mud-file

This is a URL that should point to the previous MUD URL for auditing purposes. Because it should not be necessary to resign a MUD file when a new one is released, the archival location of a current MUD file should be identified prior to its release. Note the signature file MUST also be available. For example, if previous-mud-file is set to "https://example.com/.mud/v1/xxx", the corresponding signature would be found at "https://example.com/.mud/v1/xxx.p7s".

3.3. cache-validity

This uint32 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 no more than 1440 for any device that is supported.

3.4. masa-server

This optional element refers to the URL that should be used to resolve the location any MASA service, as specified in [I-D.ietf-anima-bootstrapping-keyinfra].

3.5. is-supported

This boolean is an indication from the manufacturer to the network administrator as to whether or not the device is supported. In this context a device is said to be supported if the manufacturer might issue an update to the device or if the manufacturer might update the MUD file.

3.6. packet-direction

[I-D.ietf-netmod-acl-model] describes access-lists but does not attempt to indicate where they are applied as that is handled elsewhere in a configuration. However, in this case, a MUD file must be explicit in describing the communcation pattern of a device, and that includes indicating what is to be permitted or denied in either direction of communication. This element takes a single value of either "to-device" or "from-device", based on a typedef "direction".

3.7. manufacturer

This element consists of a hostname that would be matched against the authority section of another device's MUD URL.

3.8. same-manufacturer

This is an equivalent for when the manufacturer element is used to indicate the authority that is found in another device's MUD URL matches that of the authority found in this device's MUD URL.

3.9. model

This string matches the one and only segment following the authority section of the MUD URL. It refers to a model that is unique within the context of the authority. It may also include product version information. Thus how this field is constructed is entirely a local matter for the manufacturer.

3.10. local-networks

This null-valued element expands to include local networks. Its default expansion is that packets must not traverse toward a default route that is received from the router.

3.11. controller

This URI specifies a value that a controller will register with the network management station. The element then is expanded to the set of hosts that are so registered.

In addition, some meta information is defined in order to determine when a usage description should be refreshed.

3.12. direction-initiated

When applied this matches packets when the flow was initiated in the corresponding direction. [RFC6092] provides guidance for IPv6 guidance best practices. While that document is scoped specifically to IPv6, its contents are applicable for IPv4 as well. When this flag is set, and the system has no reason to believe a flow has been initiated it MUST drop the packet. This match SHOULD be applied with specific transport parameters, such as protocol.

4. Processing of the MUD file

To keep things relatively simple in addition to whatever definitions exist, we also apply two additional default behaviors:

- Anything not explicitly permitted is denied.
- Local DNS, DHCP, and NTP are, by default, permitted to and from the device.

5. What does a MUD URL look like?

To begin with, MUD takes full advantage of both the https: scheme and the use of .well-known. HTTPS is important in this case because a man in the middle attack could otherwise harm the operation of a class of devices. .well-known is used because we wish to add additional structure to the URL. And so the URL appears as follows:

```
mud-url   = "https://" authority  "/.well-known/mud/" mud-rev
            "/" model ( "?" extras )
            ; authority is from RFC3986
mud-rev   = "v1"
model     = segment  ; from RFC3986
extras    = query    ; from RFC3986
```

mud-rev signifies the version of the manufacturer usage description file. This memo specifies "v1" of that file. Later versions may permit additional schemas or modify the format.

"model" represents a device model as the manufacturer wishes to represent it. It could be a brand name or something more specific. It also may provide a means to indicate what version the product is. Specifically if it has been updated in the field, this is the place where evidence of that update would appear. The field should be changed when the intended communication patterns of a device change. While from a controller standpoint, only comparison and matching operations are safe, it is envisioned that updates will require some administrative review. Processing of this URL occurs as specified in [RFC2818] and [RFC3986].

6. The MUD YANG Model

```
<CODE BEGINS>file "ietf-mud@2016-07-20.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
       WG Chair: Warren Kumari
       warren@kumari.net
       WG Chair: Zhou Tianran
       zhoutianran@huawei.com
       Editor: Eliot Lear
       lear@cisco.com
       Editor: Ralph Droms
       rdroms@cisco.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.

    Copyright (c) 2016 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 "2016-07-20"  {
    description "Base version of MUD extensions to ACL model";
    reference "RFC XXXX: Manufacturer Usage Description Specification";
  }

  typedef direction {
     type enumeration {
         enum to-device {
           description "packets or flows destined to the target device";
         }
         enum from-device {
           description "packets or flows destined from
                        the target device";
         }
         }
     description "Which way are we talking about?";
  }

  container meta-information {

    description "Information about when support end(ed), and
                 when to refresh";

    leaf last-update {
      type yang:date-and-time;
      description "This is intended to be the time and date that
                   the MUD file was generated.";
    }

    leaf previous-mud-file {
       type inet:uri;
```

```
       description "Use to find the previous MUD file location
                    for auditing purposes.";
    }

    leaf cache-validity {
      type uint32;
      description "The information retrieved from the MUD server is
                  valid for these many hours, after which it should
                  be refreshed.";
    }

    leaf masa-server {
      type inet:uri;
      description "The URI of the MASA server that network
      elements should forward requests to for this device.";
    }

    leaf is-supported {
      type boolean;
      description "The element is currently supported
                   by the manufacturer.";
    }
  }

  augment "/acl:access-lists/acl:acl"   {
    description "add inbound or outbound.  Normally access lists
                 are applied in an inbound or outbound direction
                 separately from their definition.  This is not
                 possible with MUD.";
    leaf packet-direction
    {
      type direction;
      description "inbound or outbound ACL.";
    }
  }


  augment "/acl:access-lists/acl:acl/" +
     "acl:access-list-entries/acl:ace/" +
     "acl:matches" {
    description "adding abstractions to avoid need of IP addresses";

    leaf manufacturer {
      type inet:host;
      description "authority component of the manufacturer URI";
    }

    leaf same-manufacturer {
```

```
      type empty;
      description "expand to ACEs for each device
                   with the same origin";
    }

    leaf model {
        type string;
        description "specific model (including version) for a
                     given manufacturer";
    }

    leaf local-networks {
      type empty;
      description "this string is used to indicate networks
                   considered local in a given environment.";
    }
    leaf controller {
      type inet:uri;
      description "expands to one or more controllers for a
                   given service that is codified by inet:uri.";
    }
    leaf direction-initiated {
      type direction;
      description "which direction a flow was initiated";
    }
  }
}
```

<CODE ENDS>

7. 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" element. 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:

```
   augment
   /acl:access-lists/acl:acl/acl:access-list-entries
      /acl:ace/acl:matches/acl:ace-type/acl:ace-ip:
      +--rw src-dnsname?        inet:host
      +--rw dst-dnsname?        inet:host
```

The choice of this particular point 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 following elements are defined.

7.1. source-dnsname

The argument corresponds to a domain name of a source as specified by inet:host. Depending on how the model is used, it may or may not be resolved, as required by the implementation and circumstances.

7.2. destination-dnsname

The argument corresponds to a domain name of a destination as specified by inet:host. Depending on how the model is used, it may or may not be resolved, as required by the implementation and circumstances.

7.3. The ietf-acldns Model

```
 <CODE BEGINS>file "ietf-acldns@2007-07020.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
        WG Chair: Warren Kumari
        warren@kumari.net
        WG Chair: Zhou Tianran
        zhoutianran@huawei.com
        Editor: Eliot Lear
```

```
        lear@cisco.com
        Editor: Ralph Droms
        rdroms@cisco.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 "2016-07-20" {
     description "Base version of dnsname extension of ACL model";
     reference "RFC XXXX: Manufacturer Usage Description Specification";
   }

   augment "/acl:access-lists/acl:acl/" +
      "acl:access-list-entries/acl:ace/" +
      "acl:matches/acl:ace-type/acl:ace-ip" {
     description "adding domain names to 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";
     }
   }
 }
```

<CODE ENDS>

8. MUD File Example

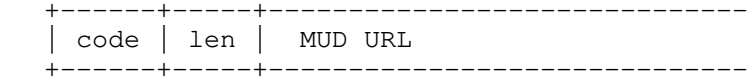
This example contains two access lists that are intended to provide outbound access to a cloud service on TCP port 443.

```
{
   "ietf-mud:support-information": {
      "last-update": "2016-05-18T20:00:50Z",
      "cache-validity": 1440
      },
   "ietf-access-control-list:access-lists":  {
     "acl": [ {
      "acl-name": "inbound-stuff",
      "acl-type" : "ipv4-acl",
```

```
        "ietf-mud:direction" : "to-device",
        "access-list-entries": {
           "ace": [
              {
                 "rule-name": "access-cloud",
                 "matches": {
                   "ietf-acldns:src-dnsname":
                       "lighting-system.example.com",
                    "protocol" : 6,
                    "source-port-range" : {
                       "lower-port" : 443,
                       "upper-port" : 443
                       }
                    },
                 "actions" : {
                   "permit" : [null]
                   }
                }
              ]
             }
           },
           {
        "acl-name": "outbound-stuff",
        "acl-type" : "ipv4-acl",
        "ietf-mud:direction" : "from-device",
        "access-list-entries": {
           "ace": [
              {
                 "rule-name": "access-cloud",
                 "matches": {
                    "ietf-acldns:dst-dnsname":
                         "lighting-system.example.com",
                     "protocol" : 6,
                     "destination-port-range" : {
                       "lower-port" : 443,
                       "upper-port" : 443
                       }
                    },
                 "actions" : {
                     "permit" : [null]
                   }
                }
              ]
             }
           }
          ]
      }
   }
```

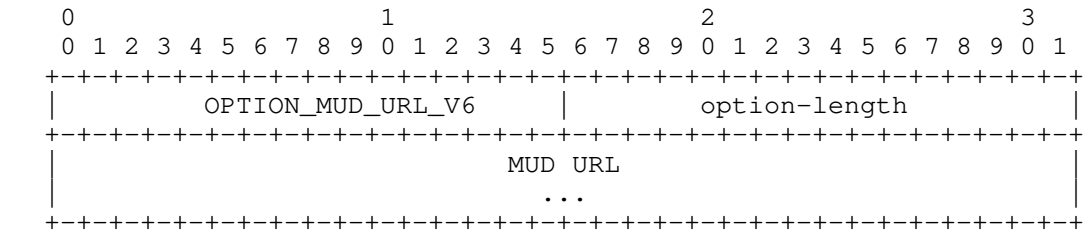
9. The MUD URL DHCP Option

The IPv4 MUD URL client option has the following format:



Code OPTION_MUD_URL_V4 (TBD) is assigned by IANA. len is a single octet that indicates the length of the URL in octets. MUD URL is a URL. The length of a MUD URL does not exceed 255 bytes.

The IPv6 MUD URL client option has the following format:



OPTION_MUD_URL_V6 (TBD; assigned by IANA).

option-length contains the length of the URL in octets. The length MUST NOT exceed 255 octets.

The intent of this option is to provide both a new device 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 device to the network in a structured way such that the policy can be easily found with existing toolsets.

9.1. Client Behavior

A DHCP client MAY emit either a DHCPv4 or DHCPv6 option or both. These options 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.

Clients SHOULD log or otherwise report improper acknowledgments from servers, but they MUST NOT modify their MUD URL configuration based on a server's response. The server's response is only an acknowledgment that the server has processed the option, and promises no specific network behavior to the client. In particular, it may not be possible for the server to retrieve the file associated with the MUD URL, or the local network administration may not wish to use the usage description. Neither of these situations should be considered in any way exceptional.

9.2. Server Behavior

A DHCP server may ignore these options or take action based on receipt of these options. For purposes of debugging, if a server successfully parses the option and the URL, it MUST return the option with the same URL as an acknowledgment. Even in this circumstance, no specific network behavior is guaranteed. When a server consumes this option, it will either forward the URL and relevant client information to a network management system (such as the giaddr), or it will retrieve the usage description by resolving the URL.

DHCP servers may implement MUD functionality themselves or they may pass along appropriate information to a network management system or controller. 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 network management system MUST notify the network management of any corresponding change to the DHCP state of the client (such as expiration or explicit release of a network address lease).

9.3. Relay Requirements

There are no additional requirements for relays.

10. The Manufacturer Usage Description (MUD) URL X.509 Extension

[RFC7299] provides a procedure and means to specify extensions to X.509 certificates. The MUD URL is a non-critical Certificate extension that points to an on-line Manufacturer Usage Description concerning the certificate subject. This extension contains a single Uniform Resource Identifier (URI). Internationalized Resource Identifiers must be represented as URI's in the way described in RFC 5280, section 7.4.

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.

The new extension is identified as follows:

- The MUD URI extension id-pe-mud-url OBJECT IDENTIFER ::= { id-pe TBD }

The extension returns a single value:

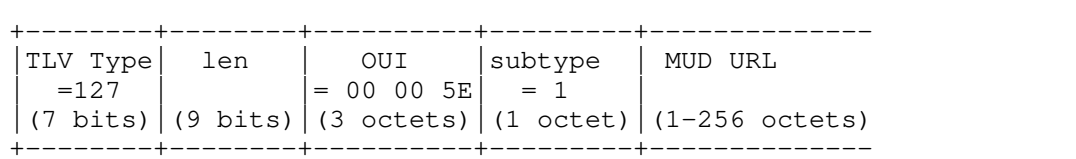
mudURLSyntax ::= IA5String - for use with MUD architecture.

The semantics of the URI are defined Section 5.

11. The Manufacturer Usage Description LLDP extension

The IEEE802.1AB Link Layer Discovery Protocol (LLDP) [IEEE8021AB] is a one hop vendor-neutral link layer protocols used by end hosts network devices 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:



where:

- TLV Type = 127 indicates a vendor-specific TLV
- len - indicates the TLV string length
- OUI = 00 00 5E is the organizationally unique identifier of IANA

- subtype = 1 (to be assigned by IANA for the MUD URL)

- MUD URL - the length MUST NOT exceed 256 octets (consistent with the DHCP option defined in Section 9)

The intent of this extension is to provide both a new device 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 device to the network in a structured way such that the policy can be easily found with existing toolsets.

Hosts, routers, or other network devices 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 devices 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 device and response sent by another device. The other devices do not acknowledge LLDP information received from a device. No specific network behavior is guaranteed. When a device consumes this extension, it may either forward the URL and relevant remote device information to a network management system, or it will retrieve the usage description by resolving the URL.

12. 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 insure 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.

12.1. 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. If the device that is being described supports IEEE 802.1AR, its manufacturer certificate and the certificate in the MUD file MUST share a common trust anchor in order to insure that manufacturer of the device is also the provider of the MUD file. The signature is stored at the same location as the MUD

URL but with the suffix of ".p7s". Signatures are transferred using content-type "Application/pkcs7-signature".

For example:

```
% openssl cms -sign -signer mancertfile -inkey mankey \
              -in mudfile -binary -outform DER - \
              -certfile intermediatecert -out mudfile.p7s
```

Note: A MUD file may need to be resigned if the signature expires.

12.2. Verifying a MUD file signature

Prior to retrieving a MUD file the MUD controller SHOULD retrieve the MUD signature file using the MUD URL with a suffix of ".p7s". For example, if the MUD URL is "https://example.com/.well-known/v1/modela", the MUD signature URL will be "https://example.com/.well-known/v1/modela.p7s".

Upon retrieving a MUD file, a MUD controller MUST validate the signature of the file before continuing with further processing. A MUD controller SHOULD produce an error and it MUST cease all processing of that file if the signature cannot be validated. If the MUD controller has received the MUD URL via IEEE 802.1AR containing an IDevID (a manufacturer certificate), it MUST further confirm that the manufacturer certificate and that of the MUD file share a common trust anchor.

For Example:

```
% openssl cms -verify -in mudfile.p7s -inform DER -content mudfile
```

Note the additional step of verifying the common trust root.

13. 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 device are permitted. Specifically, augmenting of the meta-information container is permitted with the understanding that such additions may be ignored. In addition, augmentation of the ACL model is permitted so long as it remains safe for a given ACE to be ignored by the MUD Controller or the network elements it configures. Most specifically, is is not permitted to include as an augmentation that modifies "deny" behavior without bumping the version. Furthermore, implementations that are not able to parse a component of the ACE array MUST ignore the entire array entry (e.g., not the entire array) and MAY ignore the entire MUD file.

14. Security Considerations

Based on the means a URL is procured, a device may be able to lie about what it is, thus gaining additional network access. There are several means to limit risk in this case. The most obvious is to only believe devices that make use of certificate-based authentication such as IEEE 802.1AR certificates. When those certificates are not present, devices claiming to be of a certain manufacturer SHOULD NOT be included in that manufacturer grouping without additional validation of some form. This will occur when it makes use of primitives such as "manufacturer" for the purpose of accessing devices of a particular type.

Network management systems SHOULD NOT deploy a usage description for a device with the same MAC address that has indicated a change of authority without some additional validation (such as review of the class). New devices that present some form of unauthenticated MUD URL SHOULD be validated by some external means when they would be otherwise 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 the signature of the MUD file. 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 device should be relatively small as well. Second, it may be useful to limit retrieval of MUD URLs to only those sites that are known to have decent web 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 controllers would consider valid. There are a few approaches that can mitigate this attack. First, MUD file

servers SHOULD cache certificates used by the MUD file server. When a new certificate is retrieved for whatever reason, the MUD controller 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 controller MAY check the WHOIS database to see if registration ownership of a domain has changed. If a change has occured, 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 device that was produced long ago and only recently fielded, or the case where a new MUD controller has been installed.

15. IANA Considerations

15.1. DHCPv4 and DHCPv6 Options

IANA is requested to allocated the DHCPv4 and v6 options as specified in Section 9.

15.2. PKIX Extensions

The IANA is requested to assign a value for id-pe-mud-uri in the "SMI Security for PKIX Certificate Extension" Registry. Its use is specified in Section 10.

15.3. Well Known URI Suffix

The IANA is requested to register the URL suffix of "mud" as follows:

o URI Suffix: "mud" o Specification documents: this document o Related information: n/a

15.4. MIME Media-type Registration for MUD files

The following media-type is defined for transfer of MUD file:

- Type name: application
- Subtype name: mud+json
- Required parameters: n/a
- Optional parameters: n/a
- Encoding considerations: 8bit; application/mud+json values are represented as a JSON object; UTF-8 encoding SHOULD be employed.
- Security considerations: See {{secon}} of this document.
- Interoperability considerations: n/a
- Published specification: this document
- Applications that use this media type: MUD controllers as specified by this document.
- Fragment identifier considerations: n/a
- Additional information:

    Magic number(s): n/a
    File extension(s): n/a
    Macintosh file type code(s): n/a

- Person & email address to contact for further information: Eliot Lear <lear@cisco.com>, Ralph Droms <rdroms@cisco.com>
- Intended usage: COMMON
- Restrictions on usage: none

- Author: Eliot Lear <lear@cisco.com>, Ralph Droms <rdroms@cisco.com>
- Change controller: IESG
- Provisional registration? (standards tree only): No.

15.5. 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

Description = the Manufacturer Usage Description (MUD) Uniform Resource Locator (URL)

Reference = < this document >

16. Acknowledgments

The authors would like to thank Einar Nilsen-Nygaard, Bernie Volz, Tom Gindin, Brian Weis, Sandeep Kumar, Thorsten Dahm, John Bashinski, Steve Rich, Jim Bieda, and Dan Wing for their valuable advice and reviews. The remaining errors in this work are entirely the responsibility of the author.

17. References

17.1. Normative References

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., and S. Bjarnason, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-03 (work in progress), June 2016.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-08 (work in progress), July 2016.

[IEEE8021AB]
Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks-- Station and Media Access Control Connectivity Discovery", n.d..

[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>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <http://www.rfc-editor.org/info/rfc2131>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://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, <http://www.rfc-editor.org/info/rfc3315>.

[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, <http://www.rfc-editor.org/info/rfc3986>.

[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, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

[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>.

[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, <http://www.rfc-editor.org/info/rfc6092>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[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, <http://www.rfc-editor.org/info/rfc7227>.

[RFC7299] Housley, R., "Object Identifier Registry for the PKIX Working Group", RFC 7299, DOI 10.17487/RFC7299, July 2014, <http://www.rfc-editor.org/info/rfc7299>.

17.2. Informative References

[I-D.lear-mud-framework]
Lear, E., "Manufacturer Usage Description Framework", draft-lear-mud-framework-00 (work in progress), January 2016.

[IEEE8021AR]
Institute for Electrical and Electronics Engineers, "Secure Device Identity", 1998.

[ISO.8601.1988]
International Organization for Standardization, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, June 1988.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[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, <http://www.rfc-editor.org/info/rfc7042>.

Appendix A. Changes from Earlier Versions

RFC Editor to remove this section prior to publication.

Draft -03 to -04: * add LLDP extension. * add Dan Romascanu as co-author.

Draft -02 to -03: * incorporate domain name model. * discuss extensibility. * leave placeholder for LLDP TLV.

Draft -01 to -02:

- XML->JSON
- Remove device versioning information from URL
- Add PKIX and DHCP options
- Add Content-type information
- Clean up IANA considerations to match registration templates
- Ralph Droms carried over as author from DHCP option.
- Signing information
- Expanded Security Considerations
- Add directionality for both packets and flows.

- add previous-mud-file

Draft -00 to -01:

- Add MASA server element

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Ralph Droms
Cisco Systems
55 Cambridge Parkway
Cambridge 1057
United States

Phone: +1 617 621 1904
Email: rdroms@cisco.com

Dan Romascanu
Avaya
26, HaRokhmim Str., Bldg. D
Holon 5885849
Israel

Phone: +972-3-645-8414
Email: dromasca@avaya.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

Table of Contents

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 optimazation 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 mediculous. 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 preferablely 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 feture 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	bgpCommunity
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 specifed 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 specifed 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

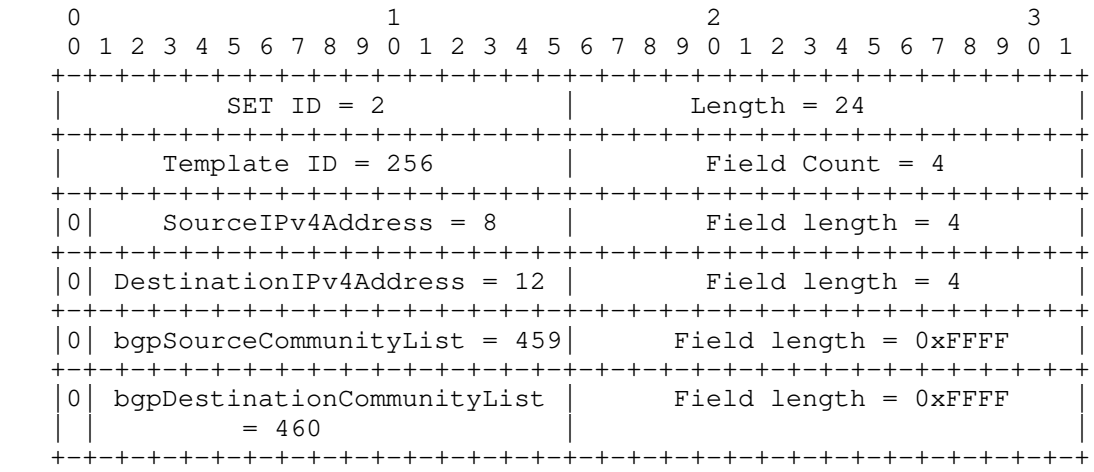
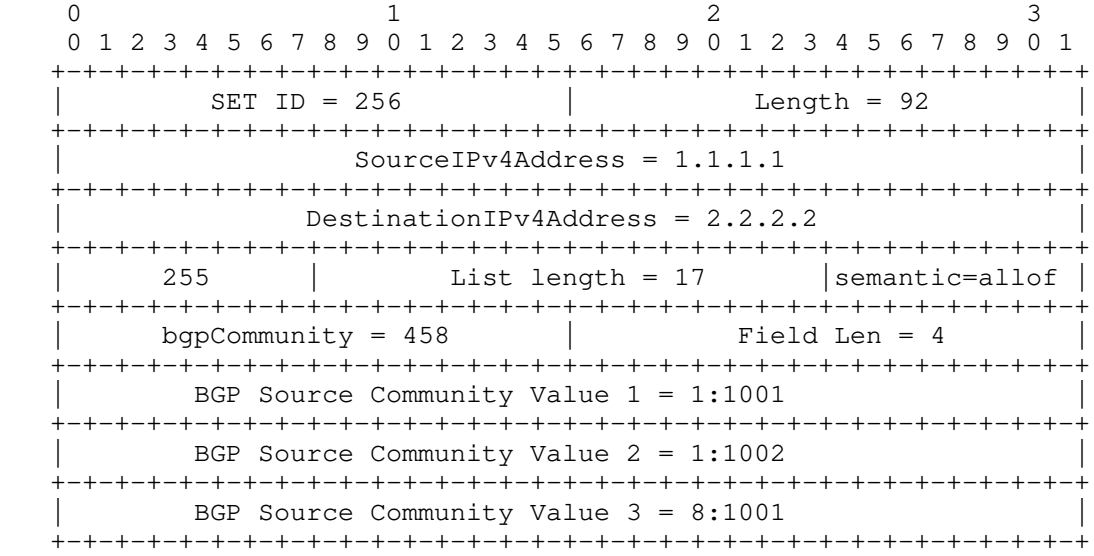


Figure 5: Template Record Encoding Format

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:



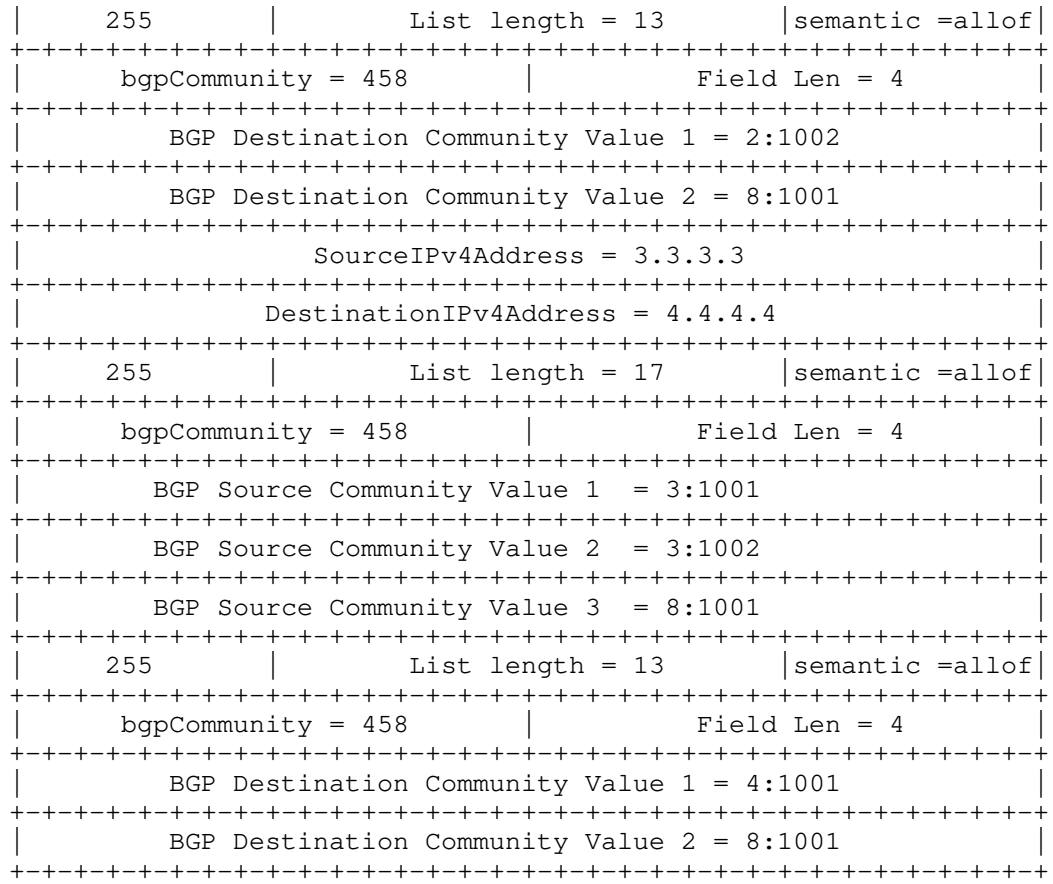


Figure 6: Data Set Encoding Format

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: gurong_cmcc@outlook.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Internet Engineering Task Force
Internet Draft
Intended status: Informational
Expires: February 2017

P. Unbehagen, Ed.
D. Romascanu
J. Seligson
C. Keene
Avaya

August 2016

A Framework for Automatic Attachment of Network Objects to Core Networks

draft-romascanu-opsawg-auto-attach-framework-01.txt

Abstract

This informational document describes a method that allows for the automatic attachment of network objects (e.g. end stations, network devices, sensors, automation elements) to a core network based on the individual services that are run or configured on the objects, and the mapping of the services to the managed paths in the network. The framework proposed by this document describes the operations that need to happen in order to have the network objects connected to the network ('attached') in an automatic manner and start providing their functionality and services without any requirement or dependency between the protocol stack on the network objects and the method used to build the bridging or routing paths in the network core.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on February 18, 2016.

Table of Contents

1. Introduction

Large networking deployments are often faced with the problem of connecting 'legacy' end stations to an upgraded network infrastructure, or with the demand to interconnect large numbers of network objects that perform different tasks, transmit information, or and controlled remotely across this network infrastructure. This informational document describes a method that allows for the automatic attachment of network objects (e.g. end stations, network devices, sensors, automation elements) to a core network based on the individual services that are run or configured on the objects, and the mapping of the services to the managed paths in the network. The framework proposed by this document describes the operations that need to happen in order to have the network objects connected to the network ('attached') in an automatic manner and start providing their functionality and services without any requirement or dependency between the protocol stack on the network objects and the method used to build the bridging or routing paths in the network core.

2. Terminology and Abbreviations

AAC – Auto Attach Client agent that resides on a non-SPB/PBB capable element that uses LLDPDUs to request I-SID assignment for the VLANs which have been configured on its network port.

AAS – Auto Attach Server agent that processes VLAN to I-SID requests from AAC elements that are connected to a SPB BEB

Element – Any end device or network node that may implement the auto attach functionality

LAN – Local Area Network

LLDP – Link Layer Discovery Protocol

MUD – Manufacturer User Description

VLAN – Virtual Local Area Network

3. 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].

4. Auto Attachment Framework, Model and Components

This section provides an overview of the behavior of auto attachment functionality. The scheme proposed in this document is applicable at the link layer or at the network layer. Is the currently subject of proposed standardization work in the IEEE 802.1 Working Group [AA], and it is consistent with work proposed in the IETF in the Network Virtualization Overlays (nvo3) and Interface to the Routing System (i2rs) Working Groups.

The purpose of Auto Attach is to allow an end-device to connect to a networking device at the edge of a bridged or routed network in the core without any further knowledge or assumption of the protocol(s) being run in the core. The end-device is called an AA client (AAC) and the networking device at the edge of the core network is called the AA server (AAS). An AA Client is a device that does not support the bridging or routing protocol in the core but supports some form of binding definition between the applications or services that it is running and the format of the data packets that it sends to the network (for example VLAN tags, or tunnel identifiers), if connectivity permits, has the ability to advertise this data to a directly connected AA Server. An AA Server is network device that potentially accepts externally generated service to tags or tunnels assignments that can be used for automated configuration purposes. The client identifies itself to the server and then requests service to tags binding(s). The server will either accept or reject each binding request. If accepted, any traffic on the (locally significant) VLAN or tunnel is forwarded through the routed network at the parameters required by the service.

The simplification brought by the auto-attachment scheme consists of the use of widely deployed protocols on the first hop connection between the AAC and the AAS which allow for the interaction between the two entities to happen in an automatic manner, without requiring manual configuration of attachment information at multiple locations. AACs that utilize this automated method for service assignment pass the assignment information to the AASs where the mappings are processed and approved or rejected. Specific actions are taken on both entities based on the outcome of the mapping request.

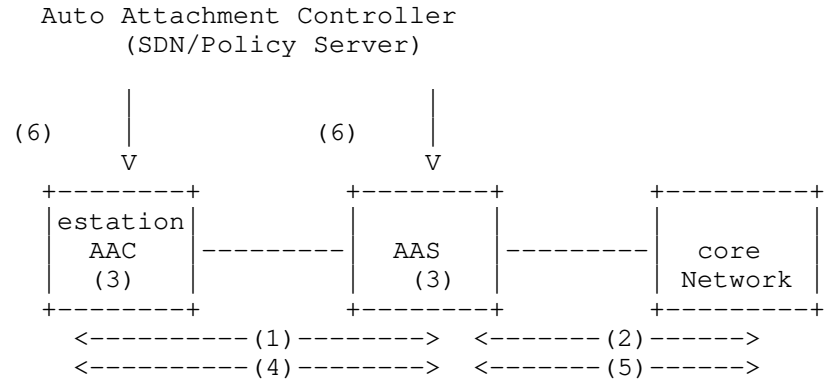


Figure 1: Conceptual Auto Attach Model

(1) Auto-Attachment Primitives
(2) Routing Control Plane
(3) 'horizontal' data model
(4) Service Tag
(5) Service Route/Tunnel ID
(6) 'vertical' data model

Figure 1 depicts a conceptual example of the process where an AAC can use a one-hop protocol (1) e.g. LLDP that includes the Auto-Attachment Primitives to communicate the need to connect a service to the appropriate route or tunnel in the bridged or routed network which runs the core bridging or routing protocol (2). A 'horizontal' data model (3) is shared by the AAC and AAS and synchronized after the initial exchange of information. If the binding requests are being accepted, the AAC will start sending traffic using the service tag (4), traffic which will be routed or tunneled appropriately in the bridged or routed paths (5). The policy data model (6) allows for the interaction between the network devices and the end-stations to a policy server, allowing for the integration of AA scheme in a policy-based service management environment.

An Auto Attach Client (AAC) can run in any device (end station, estation) that connects to a core network. One field of applications can be for example Internet of Things (IoT) devices that connect to

a network. The service association is automated with relative low resources, allowing connection of the devices to the appropriate network services and applications.

The different classes of devices that run AACs may be configured by means that are specific for the respective classes of applications or services. A YANG module may instantiate the 'vertical' data model (6) of the Auto Attachment framework allowing for standard-based interaction with the Auto Attachment Controllers, which are instantiated as policy or Software Define Networks (SDN) servers.

4.1. Discussion

At least one proprietary implementation introduces the concept of an Auto Attach Proxy. Such an optional block placed between an AAC and AAS would allow for multiplexing and/or virtualizing the access to servers. At this phase we decided to leave this optional block for further study.

5. Auto Attachment Process

The auto-attachment process is composed of a one-hop two steps protocol that performs the following functions:

- Element Discovery
- Services Configuration

5.1. Element Discovery

The first stage of establishing AA connectivity involves element discovery. An Auto Attach agent resides on all capable elements. Server agents control the Auto Attach (AA) of service tags to routes or tunnels when enabled to accept and process such requests from AAC elements. Typically this is done through a global service setting and through per-port settings that control the transmission of information in the one-hop protocol (1) on the appropriate links that interface AAC's and AAS's.

Once the required AA settings are enabled on the elements (e.g., the AA service and the per-port AA settings) the AA agent on each element type, both AAC and AAS, advertises its capabilities (i.e., server/client) through protocol (1) primitives to each other.

Following discovery of AA capabilities by both the AAC and the AAS, the AA agent on each element is aware of all AA services currently provided by the network elements to which it is directly connected. Based on this information, an AAC agent can determine whether Auto

Attach data, namely locally administered assignments, should be exported to the AAS that is associated with an edge networking device to which it is attached to on its network uplink ports.

5.2. Services Configuration

Service mappings can be established when these two criteria are met:

1. AA Server found during discovery

   AAC - - AAS peering was established during the discovery process

2. Service Tags / Routes mapping are defined locally

   Assuming that an administrator has defined one or more tags / routes mappings, or AAC bindings have been received for processing

Each mapping assignment in an AA request received by the AAS is processed individually and can be accepted or rejected. An assignment may be rejected for a number of reasons, such as server resource limitations or, for example, restrictions related only to the source AAC. Rejected assignments are passed back to the originating AAC with a rejected state and, if appropriate, an indication as to why the rejection occurred. Limited state information may be maintained on the server related to rejected assignments.

Each route (or tunnel, or VLAN) that is associated with an accepted assignment is instantiated on the AAS bridge if it does not already exist.

The AAS agent is responsible for tracking which, if any, of these actions are performed so that settings can be cleared when they are no longer needed. This can occur, for example, when configuration changes on an AAC updates the received assignment list when an AAC associated with a downlink port changes or an AAC connection disappears entirely.

Each station tag (e.g. VLAN, subnet) that is associated with a service assignment must be defined on the client’s device. The port associated with the uplink connecting the AAC to the AAS must be a member of the VLAN or Subnet assignment lists that are sent to and accepted by the AAS. This allows tagged traffic on to pass through the edge networking device into the core routed network when required. To ensure that markings are maintained between devices,

traffic on the uplink port MUST be tagged. If a tag has not been created before the assignment itself, it is automatically created by the AAC agent when a proposed assignment is accepted. Port tagging and the port VLAN or subnet membership update are also performed by the AAC automatically based on assignment acceptance. To ensure consistency, tags SHOULD NOT be deleted as long as they are referenced in any I-SID/VLAN assignments on the device.

An AAC must handle primary AAS loss and this requires maintenance of a server's inactivity timer. In order to make this possible a time-alive mechanism needs to be implemented between any AAC and its primary AAS. If an interruption of communication is detected, the service is considered interrupted, the service tags / routes assignments accepted by the server are considered rejected. Assignment data is then defaulted (reverts to the 'pending' state) and the AA agent, which resides on the AAC, removes related settings. If a back-up mechanism (alternate routes to the primary AAS, secondary AAS) exists it will be activated.

A "last updated" timestamp is associated with all active assignments on the AAS. When this value is not updated for a pre-determined amount of time, the service tags / routes assignment is considered obsolete. Obsolete assignment data and related settings are removed by the AAS.

The current routes / tags assignment list is advertised by an AAC at regular intervals. During processing of this data, an AAS must handle list updates and delete assignments from previous advertisements that are no longer present. Though these entries would be processed appropriately when they timeout, the AAS attempts to update the data in real-time and SHOULD initiate deletion immediately upon detection of this condition.

6. Security Considerations

It is important to provide an option to ensure that the aforementioned Auto Attach communication is secure in terms of data integrity (i.e., the data has not been altered in transit) and authenticity (i.e., the data source is valid).

There are several ways this can be ensured:

- Assume that the one-hop link between the end device and the network device makes use of certificate based authentication like IEEE 802.1AR [AR] certificates
- Check data integrity and perform source validation by using an optional keyed-hash message authentication code (HMAC) to protect

the Discovery and Configuration message exchanges. This type of message authentication allows communicating parties to verify that the contents of the message have not been altered and that the source is authentic. Use of this mechanism is optional and is controlled through a user-configurable attribute.

7. IANA Considerations

This memo includes no request to IANA.

Note: the section will be removed during conversion into an RFC by

the RFC Editor.

8. Further and Related Work

- Standard extensions to the IEEE 802.1AB (LLDP) [LLDP] protocol are developed by the IEEE 802.1 Working Group. The relevant project is IEEE 802.1Qcj [AA] for 'Automatic Attachment to Provider Backbone Bridges (PBB) services'.
- Element Discovery could be implemented by using one of the protocols that implement the Manufacturer User Description Specification [MUD-SPEC]. The MUD LLDP Extension, the MUD URL DHCP Option and the MUD URC X.509 Extension defined in [MUD-SPEC] can be used for the purpose of instantiating the Discovery process.

9. Acknowledgments

The first version of this document that introduced the auto attach concept was co-authored by Nigel Bragg and Ludovic Beliveau.

We would like to thank the following people (in no particular order) for their contributions:

Zenon Kuc

Cristian Mema

Roger Lapuh

Craig Griffin

Chris Buerger

Keith Krajewski

Eliot Lear

This document was prepared using 2-Word-v2.0.template.dot.

10. References

10.1. Normative References

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

[LLDP] IEEE STD 802.1AB, "IEEE Standard for Local and Metropolitan Area Networks-- Station and Media Access Control Connectivity Discovery", 2005.

[AR] IEEE STD 802.1AR, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", 2009.

10.2. Informative References

[AA] IEEE 802.1Qcj, "Standard for Local and Metropolitan Area Networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment: Automatic Attachment to Provider Backbone Bridging (PBB) services"

[MUD-SPEC] Lear E. and R. Droms, "Manufacturer Usage Description Specification",draft-lear-ietf-netmod-mud-04 (work in progress), August 2016.

Authors' Addresses

Paul Unbehagen Jr., editor
Avaya
1300 W. 120th Avenue
Westminster, CO 80234
US

Email: unbehagen@avava.com

Dan Romascanu,
Avaya
Azrieli Center Holon
26, HaRokhmim Str., Bldg. D
Holon, 5885849

Israel

Phone: +972-3-645-8414
Email: dromasca@avaya.com

John Seligson
Avaya
4655 Great America Parkway
Santa Clara, CA 95054
US

Email: jseligso@avaya.com

Carl Keene
Avaya
600 Technology Park Dr
Boston, MA 01821
US

Email: ckeene@avava.com

Internet Engineering Task Force
Internet Draft
Intended status: Informational
Expires: February 2017

P. Unbehagen, Ed.
D. Romascanu
J. Seligson
C. Keene
Avaya
August 2016

Auto-attach using LLDP with IEEE 802.1aq SPBM networks
draft-romascanu-opsawg-auto-attach-lldp-01.txt

Abstract

Automatic attachment or auto-attach is a procedure that allows for the automatic connection of network objects (e.g. end stations, network devices, sensors, automation elements) to a core network based on the individual services that are run or configured on the objects, and the mapping of the services to the managed paths in the network.

This document describes an implementation of the auto-attach concept based on the IEEE802.1AB Link Layer Discovery Protocol (LLDP) which is used to automatically attach network devices not supporting the IEEE 802.1ah Provider Backbone Bridges (PBB) to individual services in an IEEE 802.1aq Shortest Path Bridging (SPB) network.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on February 18, 2017.

Table of Contents

1. Introduction

The Auto-Attachment Framework described in [AA-FRWK] describes a method that allows for the automatic attachment of network objects (e.g. end stations, network devices, sensors, automation elements) to a core network based on the individual services that are run or configured on the objects, and the mapping of the services to the managed paths in the network. The framework proposed by that document describes the operations that need to happen in order to have the network objects connected to the network ('attached') in an automatic manner and start providing their functionality and services without any requirement or dependency between the protocol stack on the network objects and the method used to build the bridging or routing paths in the network core.

This informational document describes a compact method of implementing the Auto-Attachment Framework by using standards IEEE 802.1 protocols. Specifically this implementation uses IEEE802.1AB Link Layer Discovery Protocol (LLDP)[LLDP] to automatically attach network devices not supporting IEEE 802.1ah [PBB] to individual services in a with IEEE 802.1aq Shortest Path Bridging (SPB) [SPB] network. These network devices typically do not support SPBM, MAC-in-MAC (802.1ah), nor I-SID usage and therefore cannot easily take advantage of the SPB infrastructure without manual configuration of attachment of VLANs to I-SIDs in multiple locations. A motivation for this draft is to suggest a useful means to simplify and automate connections to PBB L2VPN based service networks such as those defined in SPBM-EVPN.

2. Terminology

802.1aq - defines a technology for providing a link state protocol for the control of a common Ethernet switching layer.

802.1ah - Provider Backbone Bridges (PBBs), MAC-IN-MAC encapsulation

AAC - Auto Attach Client agent that resides on a non-SPB/PBB capable element that uses LLDPDUs to request I-SID assignment for the VLANs which have been configured on its network port.

AAS - Auto Attach Server agent that processes VLAN to I-SID requests from AAC elements that are connected to a SPB BEB

BCB - Backbone Core Bridge

BEB - Backbone Edge Bridge

B-TAG - Backbone VLAN Tag

C-TAG - Customer VLAN Tag

Element - Any end device or network node that may implement the auto attach functionality

I-SID - Backbone Service Instance Identifier

IS-IS - Intermediate System to Intermediate System Protocol

L2VPN - - Layer 2 Virtual Private Networks

LAN - Local Area Network

LLDP - IEEE 802.1AB Link Layer Discovery Protocol

SPB - IEEE 802.1aq Shortest Path Bridging

SPBM - Shortest Path Bridging, MAC mode

VLAN - Virtual Local Area Network

3. 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].

4. Relation to the Auto Attachment Framework

Section 4 in [AA-FRWK] defines the framework, model and components of the auto-attachment functionality. Figure 1 in that document depicts the conceptual Auto Attach Model.

In the implementation described by this document, the role of the discovery protocol is played by IEEE 802.1AB (LLDP). The LLDP exchanges trigger the IS-IS SPBM announcements.

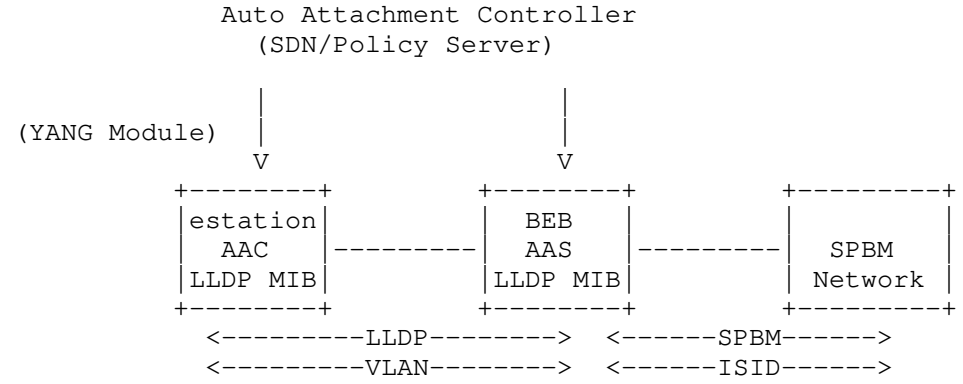


Figure 1: Conceptual LLDP-SPB Auto Attach model

Figure 1 depicts a conceptual example of the process where an AAC can use LLDP to communicate the need to connect a VLAN to the appropriate I-SID on the SPB BEB it is attached to on its network uplink port. The IEEE 802.1AB Link Layer Discovery Protocol (LLDP) and the LLDP MIB are part of the Auto Attachment Framework and will be implemented in the Backbone Edge Bridges (BEB) and the Backbone Core Bridges (BCB).

The mapping to the framework architecture is as follows:

(1) Auto-Attachment Primitives - - LLDP announcements
(2) Routing Control Plane - - IS-IS SPBM announcements
(3) 'horizontal' data model - - LLDP MIB
(4) Service Tag - - VLAN tag
(5) Service Route/Tunnel ID - - IS-ID
(6) 'vertical' data model - - Auto Attach YANG Model

The purpose of Auto Attach is to allow a non-SPB device to connect to an SPB capable networking device. The non-SPB device is called an AA client (AAC) and the SPB capable networking device is called the AA server (AAS). An AA Client is a non-SPBM device that supports some form of I-SID/VLAN binding definition and, if connectivity permits, has the ability to advertise this data to a directly connected AA Server. An AA Server is a SPBM device that potentially accepts externally generated I-SID/VLAN assignments that can be used for automated configuration purposes. The client identifies itself to the server and then requests VLAN ID to SPB ISID binding(s). The server will either accept or reject each binding request. If

accepted, any traffic on the (locally significant) VLAN is forwarded through the SPB cloud on the specified ISID.

A prototype of the extension proposed in the memo was successfully implemented and tested with Open vSwitch. IEEE 802.1aq SPB software is available from multiple vendors of Ethernet switches to connect end devices and non-SPB compliant switches to the SPB enabled backbone network. Edge switches in SPBM that utilize the 802.1ah PBB encapsulation are referred to as Backbone Edge Bridges (BEB). In support of SPBM, these bridges map a VLAN ID on the UNI to an I-SID (Individual Service ID), as defined in IEEE 802.1ah. In order to facilitate an automatic way in which a AAC can request individual service connectivity from an SPBM Backbone Edge Bridge BEB acting as a AAS, this method of using IEEE 802.1AB Link Layer Discovery Protocol (LLDP) with IEEE 802.1aq Shortest Path Bridging network can be used. These widely deployed client devices typically do not support SPBM, IEEE 802.1ah and therefore cannot easily take advantage of the SPB infrastructure without manual configuration of attachment of VLANs to I-SIDs in multiple locations.

Elements that utilize this automated method for service assignment pass this data to attached SPBM capable BEB nodes where the mappings are processed and approved or rejected. Specific actions are taken on the non-SPBM devices, referred to as Auto Attach Clients (AAC), as well as the SPBM device, referred to as Auto-Attach Server (AAS), based on the outcome of the mapping request.

5. Auto Attachment Layer 2 Functionality

5.1. Element Discovery

The first stage of establishing AA connectivity involves element discovery. An Auto Attach agent resides on all capable elements. Server agents control the Auto Attach (AA) of VLANs to I-SIDs on themselves when enabled to accept and process such requests from AAC elements. Typically this is done through a global service setting and through per-port settings that control the transmission of information in LLDPDUs on the appropriate links that interface AAC's and AAS's.

Once the required AA settings are enabled on the elements (e.g., the AA service and the per-port AA settings) the AA agent on each element type, both AAC and AAS, advertises its capabilities (i.e., server/client) through LLDPDU packets to each other.

Following discovery of AA capabilities by both the AAC and the AAS, the AA agent on each element is aware of all AA services currently

provided by the network elements to which it is directly connected. Based on this information, an AAC agent can determine whether Auto Attach data, namely locally administered I-SID/VLAN assignments, should be exported to the AAS that is associated with an SPBM BEB to which it is attached to on its network uplink ports.

Initial Auto Attach functionality, when enabled, can be used to extract management VLAN data from the primary AA server advertisements and can use this data to update the in-band management VLAN and initiate IP address acquisition using techniques such as DHCP.

5.2. Service Requests

Service mappings can be established when these two criteria are met:

1. AA Server found during discovery

   Assuming that an administrator has defined one or more ports for auto attach mode a discovery message is sent out each port defined using LLDP. Element information is forwarded using LLDP TLV extensions defined in section 6.1.

2. I-SID/VLAN bindings are defined locally

   Assuming that an administrator has defined one or more I-SID/VLAN assignments (or AAC bindings have been received for processing), an AAC sends the I-SID/VLAN assignment list to the discovered AAS. I-SID/VLAN data is exported using LLDP TLV extensions defined in section 6.2.

5.2.1. Element Inactivity Timeout

An AAC must handle primary AAS loss and this requires maintenance of a server's inactivity timer. If primary AAS advertisements are not received for a pre-determined amount of time, the I-SID/VLAN assignments accepted by the server are considered rejected. I-SID/VLAN assignment data is then defaulted (reverts to the 'pending' state) and the AA agent, which resides on the AAC, removes related settings.

5.3. Server Mapping Request Processing

Each I-SID/VLAN assignment in an AA request received by the AAS is processed individually and can be accepted or rejected. An assignment may be rejected for a number of reasons, such as server resource limitations or, for example, restrictions related only to

the source AAC. Rejected assignments are passed back to the originating AAC with a rejected state and, if appropriate, an indication as to why the rejection occurred. Limited state information may be maintained on the server related to rejected I-SID/VLAN assignments.

Each VLAN that is associated with an accepted I-SID/VLAN assignment is instantiated on the AAS bridge if it does not already exist. These VLANs are designated SPBM UNI VLANs on a BEB. The port through which the AA I-SID/VLAN assignment list was received (i.e., the AAS downlink) must be a member of the VLAN(s) in the I-SID/VLAN assignment list that are accepted by the AAS. Port membership is automatically updated when the UNI service (I-SID/VLAN/port) is created. To ensure that VLAN markings are maintained between switches, traffic on the downlink port MUST be tagged. The AA agent on the serving BEB handles all of these tasks automatically. No administrator intervention is required.

The AAS agent is responsible for tracking which, if any, of these actions are performed so that settings can be cleared when they are no longer needed. This can occur, for example, when configuration changes on an AAC updates the received I-SID/VLAN assignment list when an AAC associated with a downlink port changes or an AAC connection disappears entirely. Specifically, when an SPBM switched UNI-based VLAN and a switched UNI have been created on a downlink port because of an accepted AA I-SID/VLAN assignment (and not because of an explicit administrator port action), then the UNI and associated VLAN SHOULD be deleted when the related I-SID/VLAN assignment is cleared by the AAS.

5.4. Server Mapping Response Processing

Each VLAN that is associated with an AAC I-SID/VLAN assignment must be defined on the client’s device. The port associated with the uplink connecting the AAC to the AAS must be a member of the VLAN(s) in the I-SID/VLAN assignment list that are sent to and accepted by the AAS. This allows traffic on these VLANs to pass through the switch into the SPB fabric when required. To ensure that VLAN markings are maintained between devices, traffic on the uplink port MUST be tagged. If a VLAN has not been created before the I-SID/VLAN assignment itself, it is automatically created by the AAC agent when a proposed assignment is accepted. Port tagging and the port VLAN membership update are also performed by the AAC automatically based on assignment acceptance. To ensure consistency, VLANs SHOULD NOT be deleted while they are referenced in any I-SID/VLAN assignments on the device.

5.5. Service Mapping Timeout

A "last updated" timestamp is associated with all active assignments on the AAS. When this value is not updated for a pre-determined amount of time, the I-SID/VLAN assignment is considered obsolete. Obsolete assignment data and related settings are removed by the AAS, subject to the constraints imposed by section 4.3.

The current I-SID/VLAN assignment list is advertised by an AAC at regular intervals (dictated by LLDP operation). During processing of this data, an AAS must handle list updates and delete assignments from previous advertisements that are no longer present. Though these entries would be processed appropriately when they timeout, the AAS attempts to update the data in real-time and SHOULD initiate deletion immediately upon detection of this condition.

6. Auto Attach LLDP Extensions

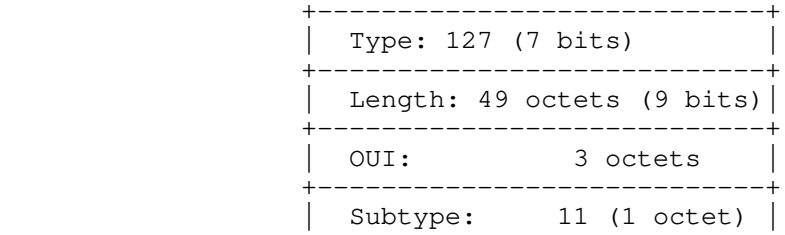
The text in this section is not normative. The complete definition of the Auto Attach TLVs is provided in the IEEE 802.1Qcj [AA] Amendment of the IEEE 802.1Q standard.

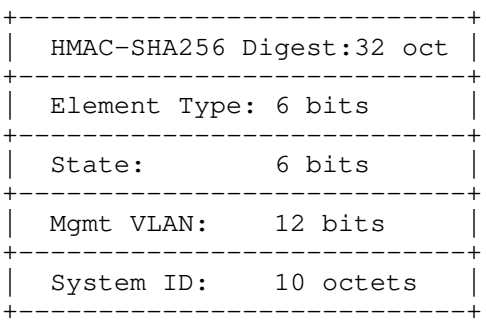
The Auto Attach TLVs are implemented as extensions to the LLPD standard, using its flexible extension mechanism. They SHOULD be implemented as vendor-specific TLVs using TLV type 127 as described in the 802.1AB (LLDP) standard. TLVs supporting the exchange of AA element data and I-SID/VLAN assignment data have been defined below.

6.1. AA Element TLV

The Element TLV is used by an AA device to announce its capabilities to its LLDP peer on a given interface. Use of the Auto Attach functionality is encoded in to the 802.1AB LLDP Custom Element TLV as follows:

AA Element TLV





Subtype = 11 for AA Element TLV

HMAC-256 Digest:

The Element TLV data integrity and source validation is supported through the use of the HMAC-SHA256 message authentication algorithm. The HMAC-SHA256 generated digest size is 32 octets and the Element TLV includes a field to support the digest exchange between source and destination parties. Symmetric private keys are used for digest generation.

The HMAC-SHA256 data digest computation starts at [0-based] byte 38 of the TLV. The digest is then placed in the HMAC-SHA256 Digest field in the TLV prior to transmission. Upon receipt, the digest is again computed and the resulting digest is compared against the received digest. If the received digest is the same as the newly computed digest, the TLV is considered valid and processing can commence. If the comparison fails, the TLV is discarded and processing is terminated.

Element Type:

The element type identifies the capability of the advertising AA node. The AA Server describes an AAS capable device that can map incoming VLAN to I-SID and announce I-SID connectivity to the SPB network. AA Clients may operate in either tagged or untagged modes. If an AA client announces untagged, then the entire port MUST be mapped to the I-SID on the BEB.

The AA Element TLV can only exist once in a LLDPDU. It is included in all LLDPDUs when the Auto Attach service is enabled and when the

per-port transmission flags associated with this TLV, as required by the 802.1AB standard, are enabled.

A number of AA Element Type values, including the AA Server and several AA Client element types, are currently defined. The list of supported element types will expand as additional devices incorporate AA signaling.

Currently supported Auto Attach Element Type values:

AA Element Type - Other (1)

AA Server (2)

AA Server No Authentication (3)

AA Client - Wireless Access Point Type 1 (4) [wireless clients get direct network attachment]

AA Client - Wireless Access Point Type 2 (5) [wireless clients get tunneled to a controller]

AA Client - Switch (6)

AA Client - Router (7)

AA Client - IP Phone (8)

AA Client - IP Camera (9)

AA Client - IP Video (10)

AA Client - Security Device (11) [FW, IPS/IDS, etc.]

AA Client - Virtual Switch (12)

AA Client - Server/Endpoint (13)

AA Client - SDN Controller (14)

AA Client - SPB-over-IP Network Device (15)

State:

The AA Element TLV State field settings indicate AA Client link tagging requirements in AA Client-sourced frames and current provisioning mode information (bits are numbered left to right):

Link VLAN Tagging Requirements (bit 1)

0 - All traffic tagged on link

1 - Tagged and untagged traffic on link

Automatic Provisioning Mode (bits 2/3)

0 - Automatic provisioning disabled

1 - SPB provisioning

2 - VLAN provisioning

System ID: conveys information that the TLV recipient can use to enforce connectivity restrictions. It includes System MAC Address, connection type and identifiers. Detailed specification of the System ID sub-fields is TBD.

6.2. I-SID/VLAN Assignment TLV

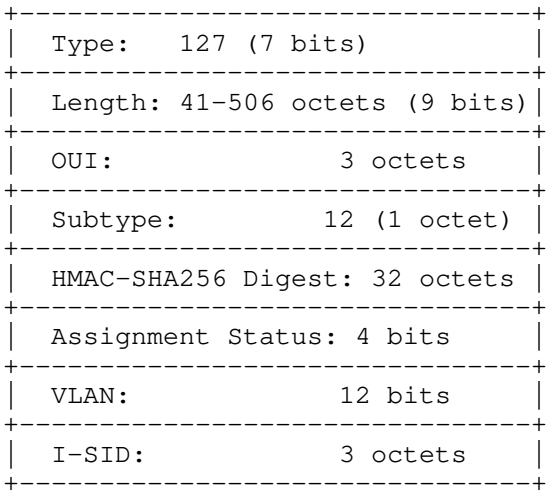
The AA I-SID/VLAN Assignment TLV is used by the AAC to announce I-SID/VLAN assignments that it would like supported by a directly connected AAS. It is also used by the AAS to announce that that I-SID/VLAN bindings processed by the AAS are active or rejected.

The AA I-SID/VLAN Assignment TLV can only exist once in a LLDPDU. It is only included in a LLDPDU when complementary AA element (i.e., AA server/ client) devices are directly connected. Data integrity and source validation is supported through the use of the HMAC-SHA256 message authentication algorithm. The HMAC-SHA256 generated digest size is 32 octets and the AA I-SID/VLAN Assignment TLV includes a field to support the digest exchange between source and destination parties.

Per-port TLV transmission flags must be enabled on the communicating devices as well. The AA Element TLV must also be present in the LLDPDU for the AA I-SID/VLAN Assignment TLV to be processed. The TLV cannot exceed the LLDP 512 byte TLV size limit, which implies a maximum of 94 I-SID/VLAN assignments in a LLDPDU

The format of the TLV is as follows:

Service Assignment TLV



The HMAC-SHA256 digest is computed for the series (1-94) of I-SID/VLAN assignments (i.e. data for the digest computation starts at [0-based] byte 38 of the TLV). The digest is then placed in the HMAC-SHA256 Digest field in the TLV prior to transmission. Upon receipt, the digest is again computed for the series (1-94) of I-SID/VLAN assignments in the received TLV and the resulting digest is compared against the received digest. If the received digest is the same as the newly computed digest, the TLV is considered valid and processing can commence. If the comparison fails, the TLV is discarded and processing is terminated. Additionally the value for the I-SID in the incoming LLDP exchanges SHOULD trigger an IS-IS SPBM announcement using normal IEEE 802.1aq mechanisms if not already being announced by the BEB.

The assignment status data is returned by the AA Server for each pending I-SID/VLAN assignment request. Assignment rejections may include information to indicate the reason for the rejection. A limited number of detailed rejection error codes will initially be supported.

Assignment Pending(1)

Assignment Accepted(2)

Rejection: Generic(3)

Rejection: AA resources unavailable(4) -the resources that are required for the Auto Attach agent to support additional I-SID/VLAN assignments are currently exhausted. The maximum number of assignments that can be supported has been reached.

Rejection: Duplicate(5)

Rejection: VLAN invalid(6) - the specified VLAN can't be used to create a switched UNI at this time. The VLAN already exists and is either inactive or has an incorrect type for this application.

Rejection: VLAN unknown(7)

Rejection: VLAN resources unavailable(8) - the maximum number of VLANs that can be supported by the device has been reached.

Rejection: Application interaction issue(9) - a failure has been detected during AA interactions with the VLAN and/or the SPBM applications. The VLAN operations to create the required SPBM switched UNI VLAN or enable port tagging may have failed or the SPBM operation to create the switched UNI may have failed

Please note that the status field is only valid when generated by an AA Server. Any Assignment TLVs which are received by an AA server are assumed to be requests. It is recommended that the status field of assignments generated by AA clients be set to 0 or 1.

VLAN: A VLAN value of 0 may indicate that the AAC traffic is untagged.

7. Security Considerations

It is important to provide an option to ensure that the aforementioned Auto Attach communication is secure in terms of data integrity (i.e., the data has not been altered in transit) and authenticity (i.e., the data source is valid).

If communication is occurring between non-secure systems, the HMAC-SHA256 Digest data should always be zero and the digest data, regardless of the value, is ignored. A misconfiguration can occur with one system operating in secure mode and the other operating in

non-secure mode. In this scenario, the Element TLV or the I-SID/VLAN Assignment TLV will always be discarded prior to processing by the system operating in secure mode.

These security requirements are satisfied by using an optional keyed-hash message authentication code (HMAC) to protect the AAC/AAS Element Discovery and I-SID/VLAN assignment exchanges. This type of message authentication allows communicating parties to verify that the contents of the message have not been altered and that the source is authentic. Use of this mechanism is optional and is controlled through a user-configurable attribute.

7.1. TLV Security Considerations

A HMAC-SHA256 digest is computed for Element TLV or for the series of I-SID/VLAN assignments, where the digest computation starts [0 based] at byte 38 of the TLV. The resulting digest is then placed in the TLV prior to sending. Where upon receipt of the digest, the contents are again computed in the same manner and the digests are compared, if the comparison fails then the TLV is discarded, otherwise if both digests are the same the TLV is considered valid and processed appropriately.

8. IANA Considerations

This memo includes no request to IANA.

Note: the section will be removed during conversion into an RFC by the RFC Editor.

9. Further and Related Work

The standard extensions to the IEEE 802.1AB (LLDP) [LLDP] protocol are developed by the IEEE 802.1 Working Group. The relevant project is IEEE 802.1Qcj [AA] for 'Automatic Attachment to Provider Backbone Bridges (PBB) services'.

Current open issues:

- Define whether an AA Proxy needs to be made part of the architecture and if yes, define its role
- Further details on the two AA TLVs fields
- Define semantics of I-SID value of 0
- Alignment with the (normative) definitions in IEEE 802.1Qcj (as they progress)

The Auto Attachment YANG data model is developed as [TBA1].

10. Acknowledgments

The first version of this document that introduced the auto attach concept was co-authored by Nigel Bragg and Ludovic Beliveau.

We would like to thank the following people (in no particular order) for their contributions:

Zenon Kuc

Cristian Mema

Roger Lapuh

Craig Griffin

Chris Buerger

Keith Krajewski

This document was prepared using 2-Word-v2.0.template.dot.

11. References

11.1. Normative References

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

[AA-FRWK] Unbehagen, P., Romascanu, D., Seligson, J., and C. Keene - A Framework for Automatic Attachment of Network Objects to Core Networks", draft-romascanu-opsawg-auto-attach-framework-00.txt, June 2016.

[LLDP] IEEE STD 802.1AB, "IEEE Standard for Local and Metropolitan Area Networks-- Station and Media Access Control Connectivity Discovery", 2005.

[PBB] IEEE STD 802.1ah, "IEEE Standard for Local and Metropolitan Area Networks / Virtual Bridged Local Area Networks / Amendment 7: Provider Backbone Bridges", 2008.

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

[RFC6329] Fedyk, D., Ashwood-Smith, P., Allan, D., Bragg, A. and P. Unbehagen, "IS-IS Extensions Supporting IEEE 802.1aq Shortest Path Bridging", RFC 6329, April 2012.

[SPB] IEEE STD 802.1aq, "IEEE Standard for Local and metropolitan area networks--Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment 20: Shortest Path Bridging", 2012.

11.2. Informative References

[AA] IEEE 802.1Qcj, "Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment: Automatic Attachment to Provider Backbone Bridging (PBB) services"

Authors' Addresses

Paul Unbehagen Jr, editor
Avaya
1300 W. 120th Avenue
Westminster, CO 80234
US

Email: unbehagen@avava.com

Dan Romascanu
Avaya
Azrieli Center Holon
26, HaRokhmim Str., Bldg. D
Holon, 5885849
Israel

Phone: +972-3-645-8414
Email: dromasca@avaya.com

John Seligson
Avaya
4655 Great America Parkway
Santa Clara, CA 95054
US

Email: jseligso@avaya.com

Carl Keene
Avaya
600 Technology Park Dr
Boston, MA 01821
US

Email: ckeene@avava.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.

Table of Contents

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.

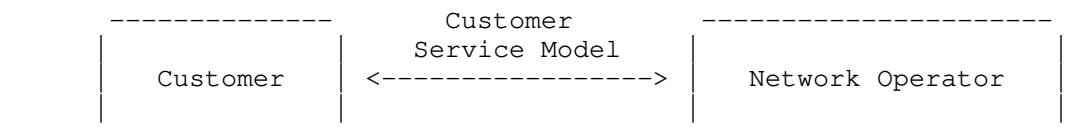


Figure 1: The Customer Service Models used on the Interface between Customers and Network Operators

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.

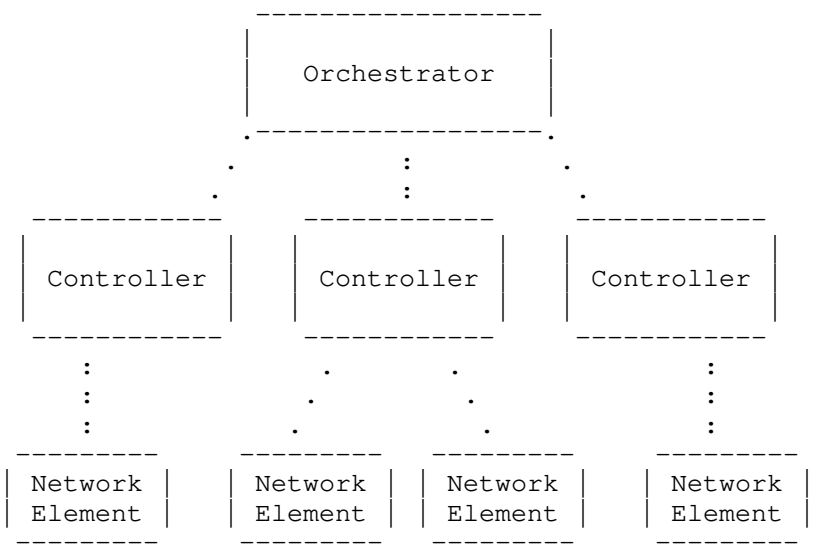


Figure 2: A Sample SDN Architecture

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.

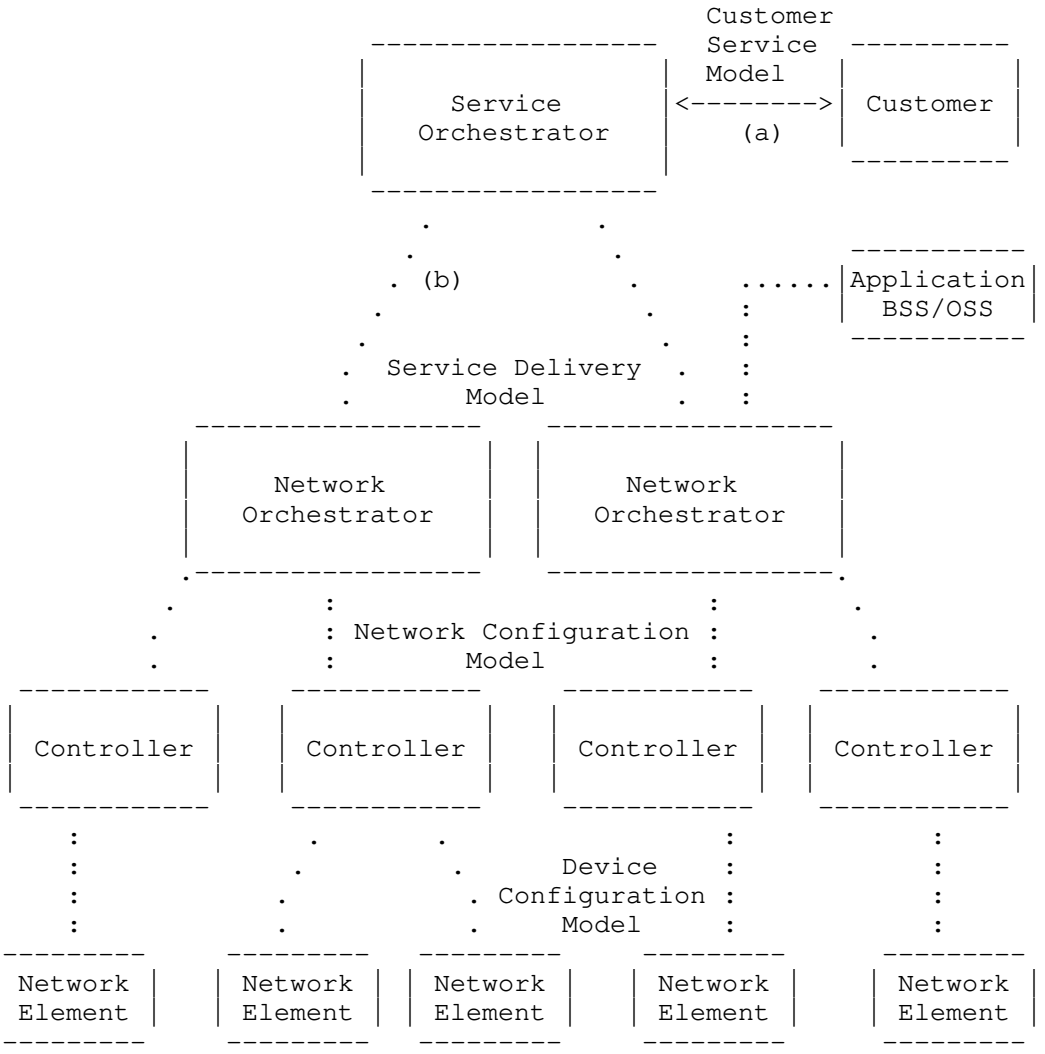


Figure 3: An Example SDN Architecture with a Service Orchestrator

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:

- 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.

- [RFC7491] describes an interface between an "Application Service Coordinator" and an "Application-Based Network Operations Controller".

- 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:

- 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.

- 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.

- 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.

- 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.

- 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.

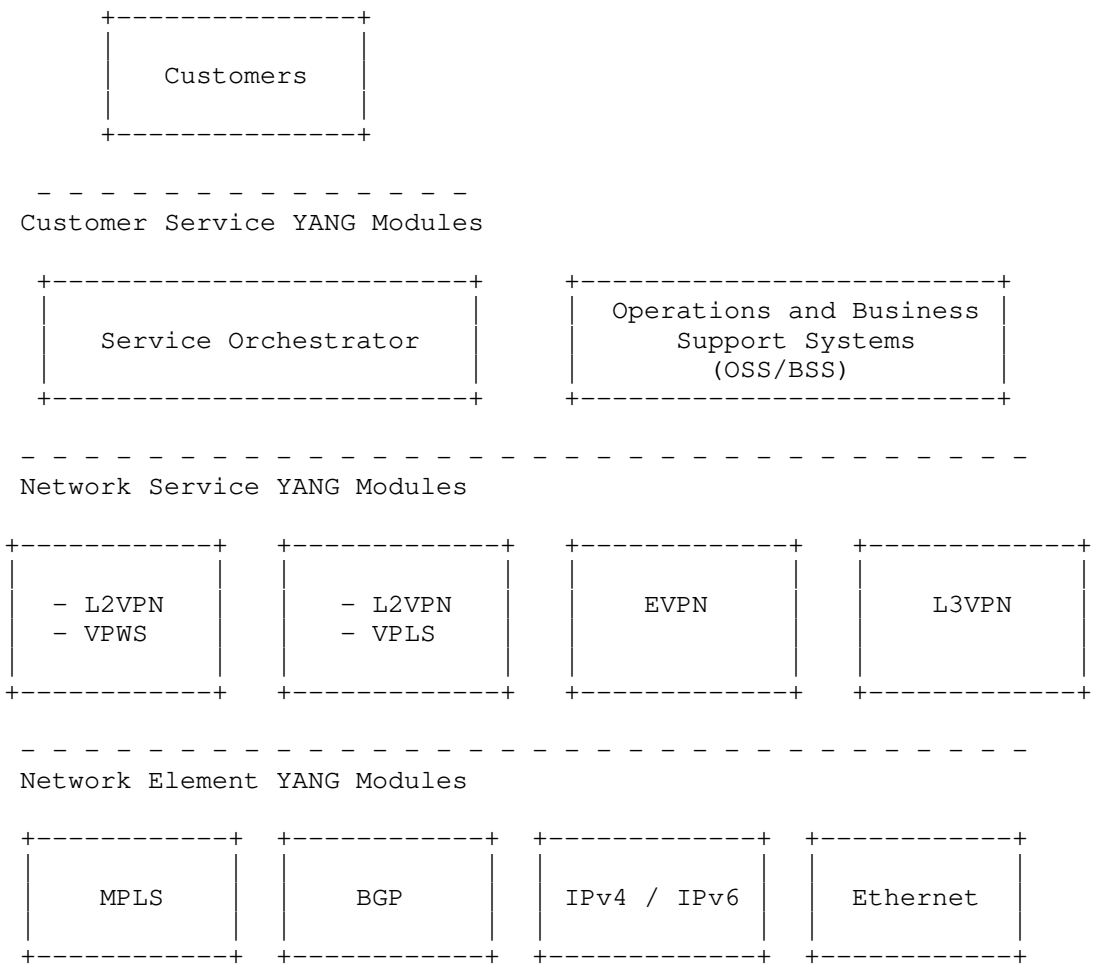
6. Comparison With Other Work

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.



L2VPN: Layer 2 Virtual Private Network
L3VPN: Layer 3 Virtual Private Network
VPWS: Virtual Private Wire Service
VPLS: Virtual Private LAN Service

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:

- [I-D.dhjain-bess-bgp-l3vpn-yang] defines a YANG model that can be used to configure and manage BGP Layer 3 VPNs.

- [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.

- [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.

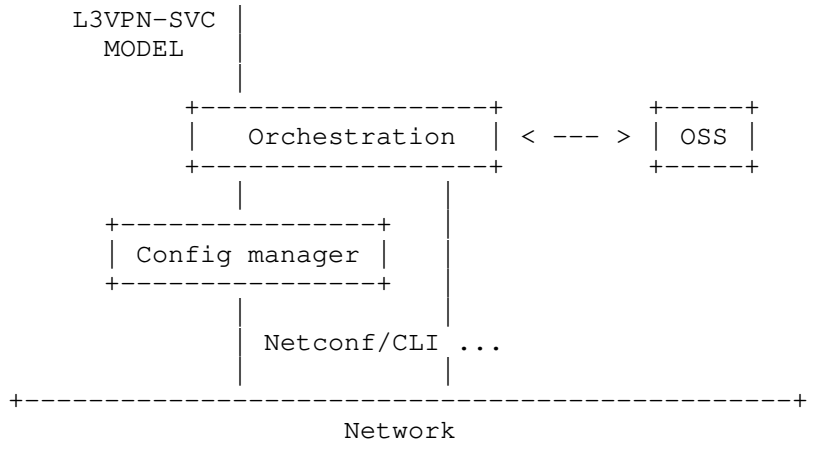


Figure 5: The L3SM Service Architecture

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.

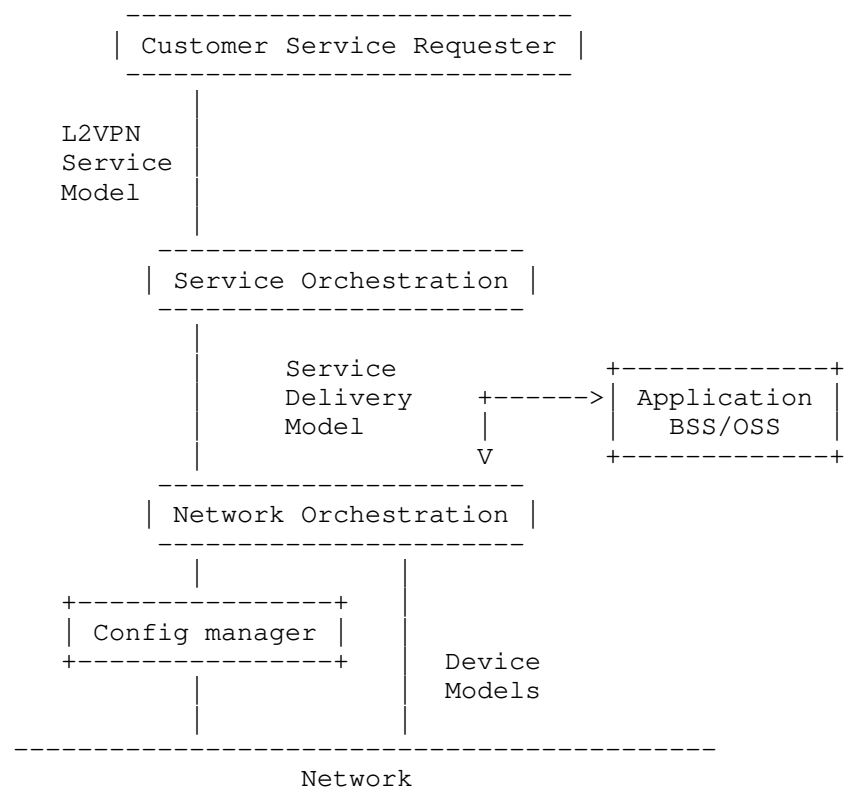


Figure 6: The L2SM Service Architecture

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:

- 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.

- 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