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.

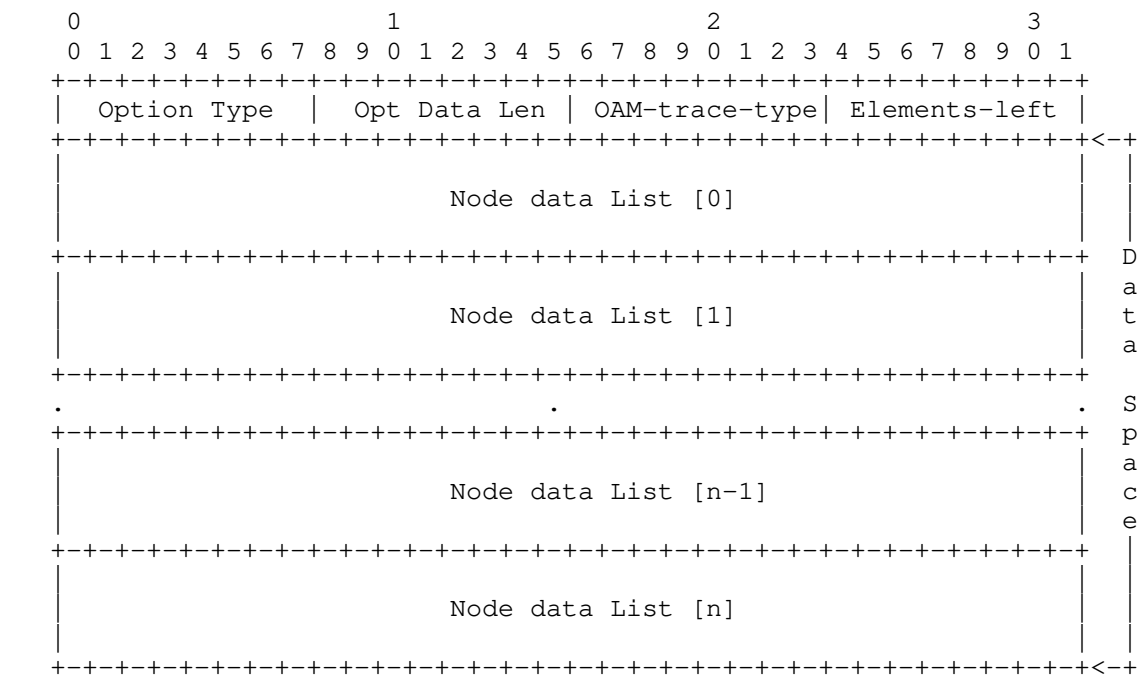
- Time of day when the packet was processed by the node. Different definitions of processing time are feasible and expected, though it is important that all devices of an in-band OAM domain follow the same definition.

- 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.

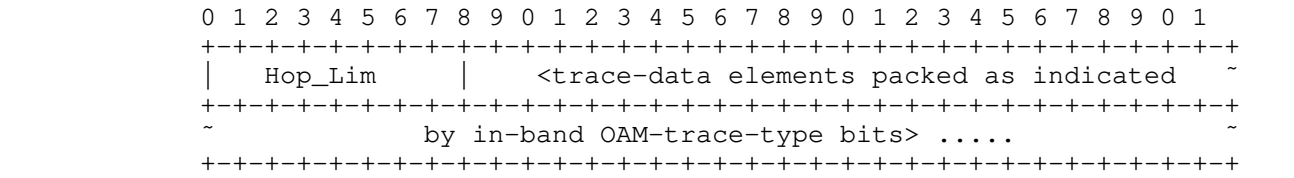
Elements-left: 8-bit unsigned integer. A pointer that indicates the next data recording point in the data space of the packet in octets. It is the index into the "Node data List" array shown above.

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.

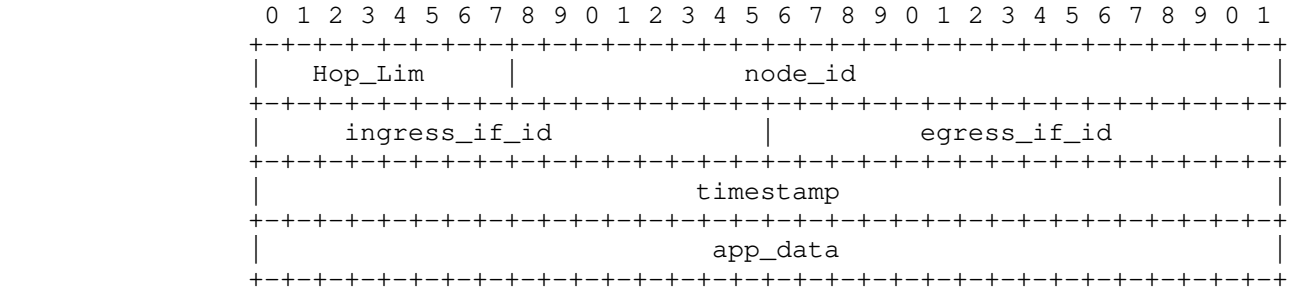
3.1.1. In-band OAM Trace Type and Node Data Element

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.

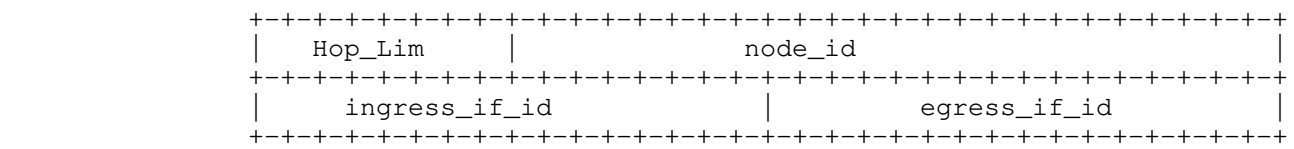
Node data has the following format:



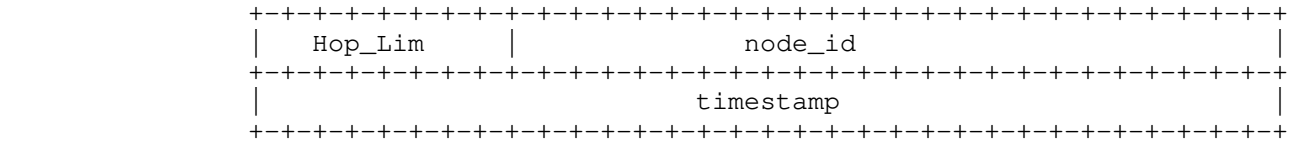
0x00011111: In-band OAM-trace-type is 0x00011111 then the format of node data is:



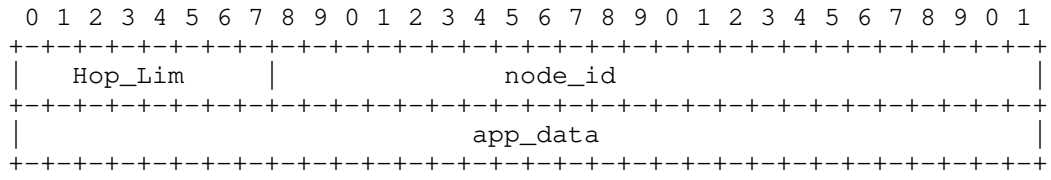
0x00000111: In-band OAM-trace-type is 0x00000111 then the format is:



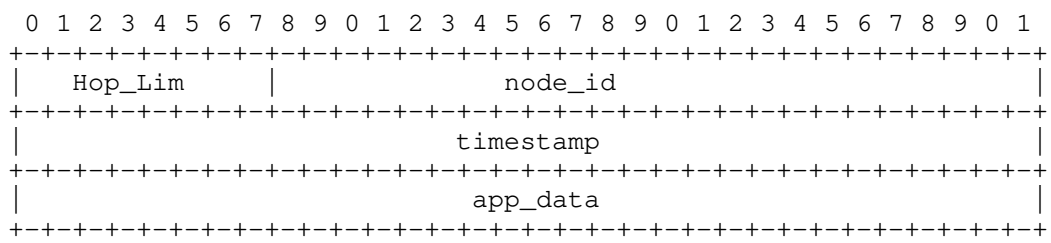
0x00001001: In-band OAM-trace-type is 0x00001001 then the format is:



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.

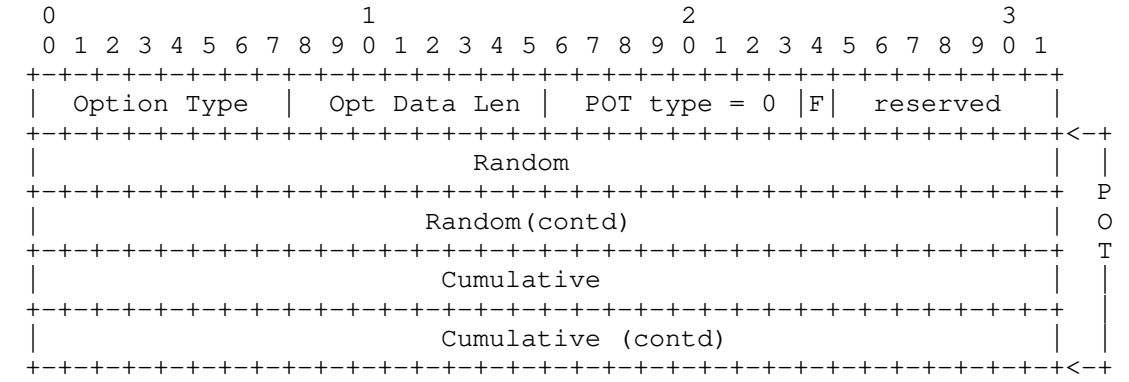
3.2. In-band OAM Proof of Transit Option

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.

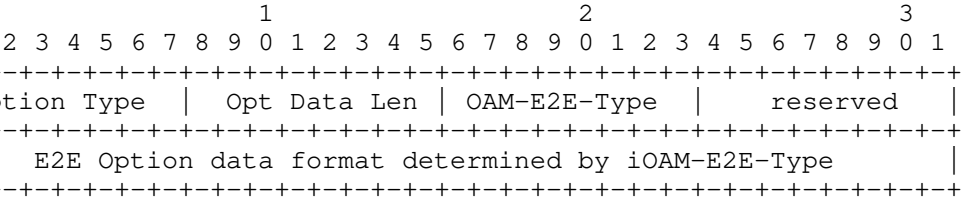
Cumulative: 64-bit Cumulative that is updated at specific nodes by processing per packet Random number field and configured parameters.

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.

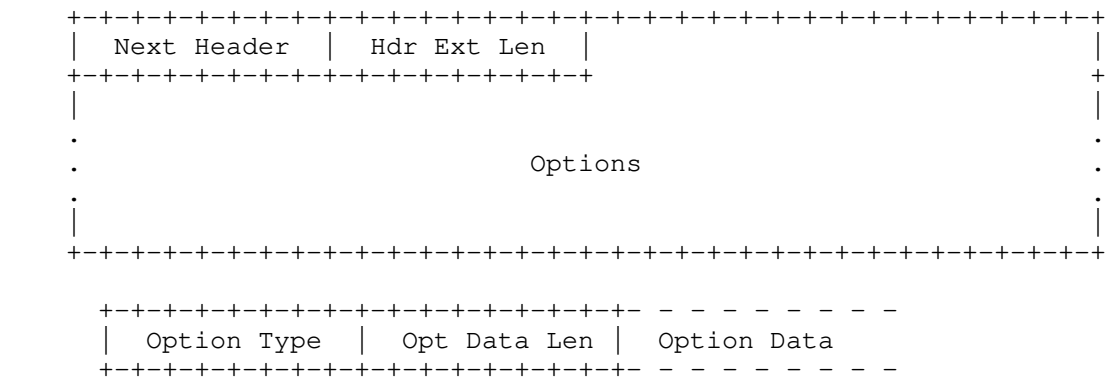
The in-band OAM options translate into options for an IPv6 extension header. The extension header would be inserted by either a host source of the packet, or by a transit/domain-edge node.

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
    }
}
```

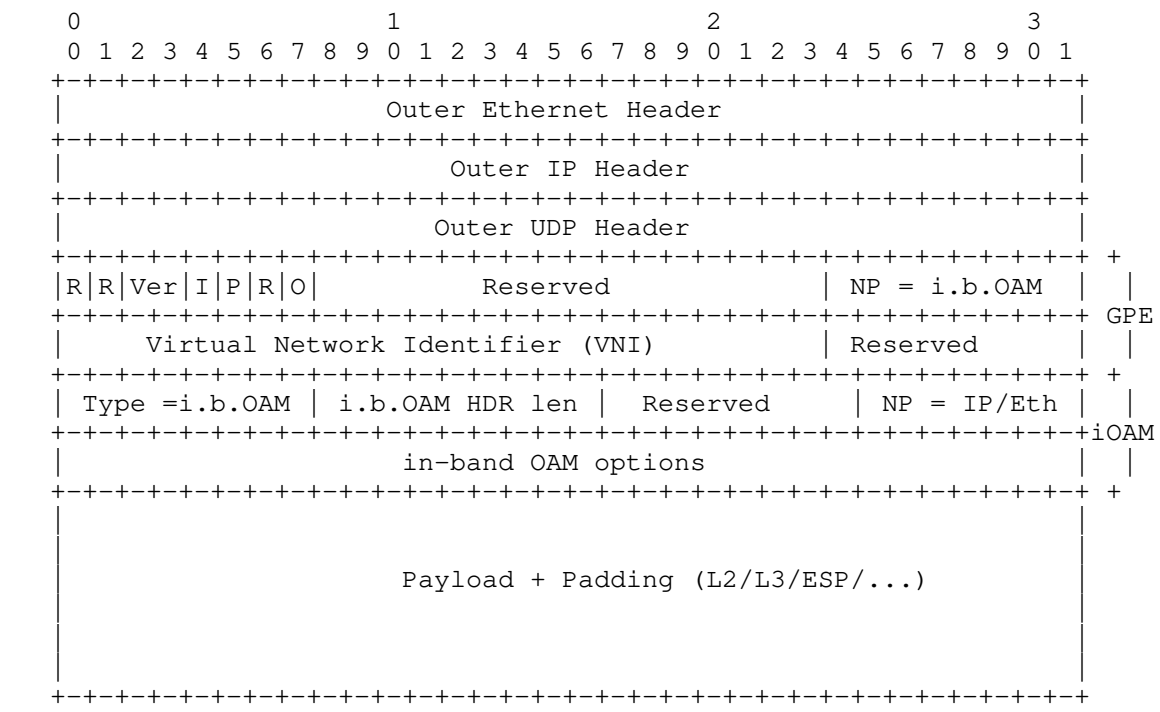
3.1.4. Procedure at the Egress Edge to Remove the In-band OAM Header

```
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:

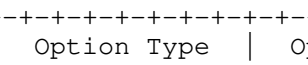


The VXLAN-GPE header and fields are defined in [I-D.ietf-nvo3-vxlan-gpe]. in-band OAM specific fields and header are defined here:

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:



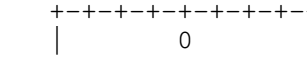
Option Type 8-bit identifier of the type of option.

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.

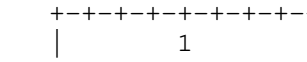
Pad1 option (alignment requirement: none)



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)

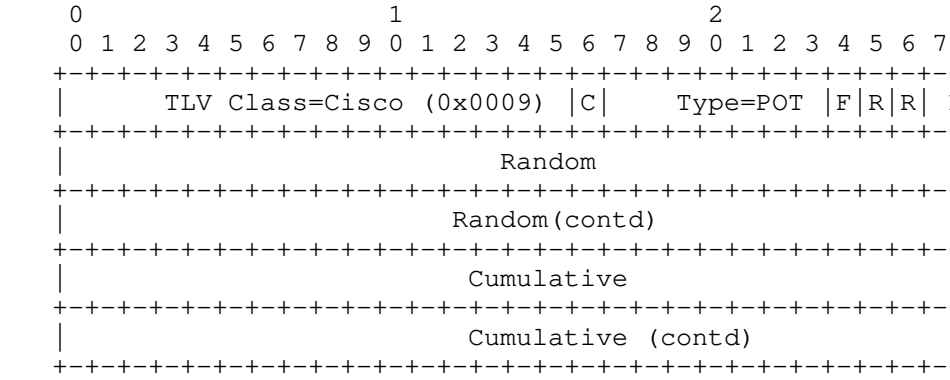


The PadN option is used to insert two or more octets of padding into the Options area of a header. For N octets of padding, the Opt Data Len field contains the value N-2, and the Option Data consists of N-2 zero-valued octets.

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.

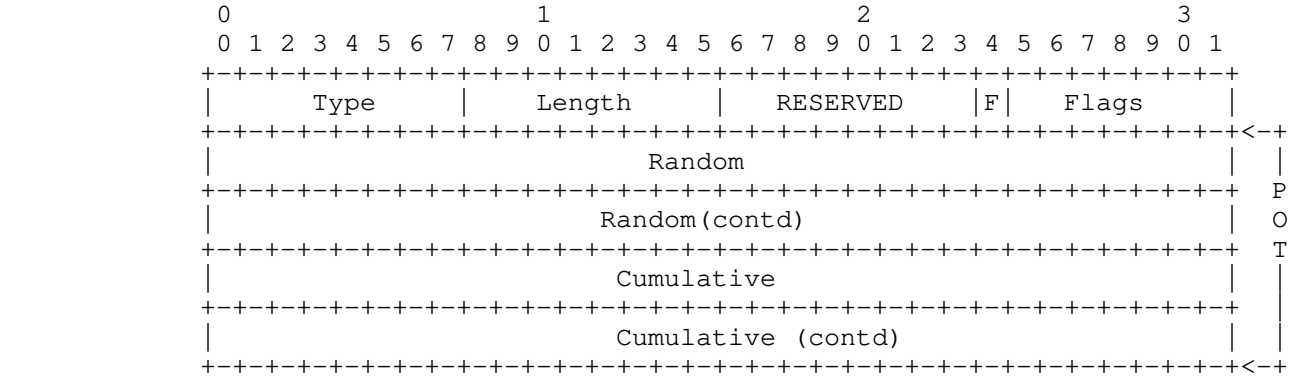
Random: 64-bit Per packet Random number.

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

INTERNET-DRAFT
Intended Status: Informational
Expires: September 14, 2017

Tom Herbert
Quantonium
Petr Lapukhov
Facebook
March 13, 2017

Identifier-locator addressing for IPv6
draft-herbert-nvo3-ila-04

Abstract

This specification describes identifier-locator addressing (ILA) for IPv6. Identifier-locator addressing differentiates between location and identity of a network node. Part of an address expresses the immutable identity of the node, and another part indicates the location of the node which can be dynamic. Identifier-locator addressing can be used to efficiently implement overlay networks for network virtualization as well as solutions for use cases in mobility.

Table of Contents

1 Introduction

This specification describes the address formats, protocol operation, and communication scenarios of identifier-locator addressing (ILA). In identifier-locator addressing, an IPv6 address is split into a locator and an identifier component. The locator indicates the topological location in the network for a node, and the identifier indicates the node's identity which refers to the logical or virtual node in communications. Locators are routable within a network, but identifiers typically are not. An application addresses a peer destination by identifier. Identifiers are mapped to locators for transit in the network. The on-the-wire address is composed of a locator and an identifier: the locator is sufficient to route the packet to a physical host, and the identifier allows the receiving host to translate and forward the packet to the addressed application.

With identifier-locator addressing network virtualization and addressing for mobility can be implemented in an IPv6 network without any additional encapsulation headers. Packets sent with identifier-locator addresses look like plain unencapsulated packets (e.g. TCP/IP packets). This method is transparent to the network, so protocol specific mechanisms in network hardware work seamlessly. These mechanisms include hash calculation for ECMP, NIC large segment offload, checksum offload, etc.

Many of the concepts for ILA are adapted from Identifier-Locator Network Protocol (ILNP) ([RFC6740], [RFC6741]) which defines a protocol and operations model for identifier-locator addressing in IPv6.

Section 5 provides a motivation for ILA and comparison of ILA with alternative methods that achieve similar functionality.

1.1 Terminology

ILA	Identifier-locator addressing.
ILA router	A network node that performs ILA translation and forwarding of translated packets.
ILA host	An end host that is capable of performing ILA translations on transmit or receive.
ILA node	A network node capable of performing ILA translations. This can be an ILA router or ILA host.
Locator	A network prefix that routes to a physical host.

Locators provide the topological location of an addressed node. In ILA locators are a sixty-four bit prefixes.

Identifier A number that identifies an addressable node in the network independent of its location. ILA identifiers are sixty-four bit values.

ILA address
An IPv6 address composed of a locator (upper sixty-four bits) and an identifier (low order sixty-four bits).

SIR Standard identifier representation.

SIR prefix A sixty-four bit network prefix used to identify a SIR address.

SIR address
An IPv6 address composed of a SIR prefix (upper sixty-four bits) and an identifier (lower sixty-four bits). SIR addresses are visible to applications and provide a means to address nodes independent of their location.

SIR domain A unique identifier namespace defined by a SIR prefix. Each SIR prefix defines a SIR domain.

ILA translation
The process of translating the upper sixty-four bits of an IPv6 address. Translations may be from a SIR prefix to a locator or a locator to a SIR prefix.

Virtual address
An IPv6 or IPv4 address that resides in the address space of a virtual network. Such addresses may be translated to SIR addresses as an external representation of the address outside of the virtual network, or they may be translated to ILA addresses for transit over an underlay network.

Topological address
An address that refers to a non-virtual node in a network topology. These address physical hosts in a network.

2 Architectural overview

Identifier-locator addressing allows a data plane method to implement network virtualization without encapsulation and its related overheads. The service ILA provides is effectively layer 3 over layer 3 network virtualization (IPv4 or IPv6 over IPv6).

2.1 Addressing

ILA performs translations on IPv6 address. There are two types of addresses introduced for ILA: ILA addresses and SIR addresses.

ILA addresses are IPv6 addresses that are composed of a locator (upper sixty-four bits) and an identifier (low order sixty-four bits). The identifier serves as the logical addresses of a node, and the locator indicates the location of the node on the network.

A SIR address (standard identifier representation) is an IPv6 address that contains an identifier and an application visible SIR prefix. SIR addresses are visible to the application and can be used as connection endpoints. When a packet is sent to a SIR address, an ILA router or host overwrites the SIR prefix with a locator corresponding to the identifier. When a peer ILA node receives the packet, the locator is overwritten with the original SIR prefix before delivery to the application. In this manner applications only see SIR addresses, they do not have visibility into ILA addresses.

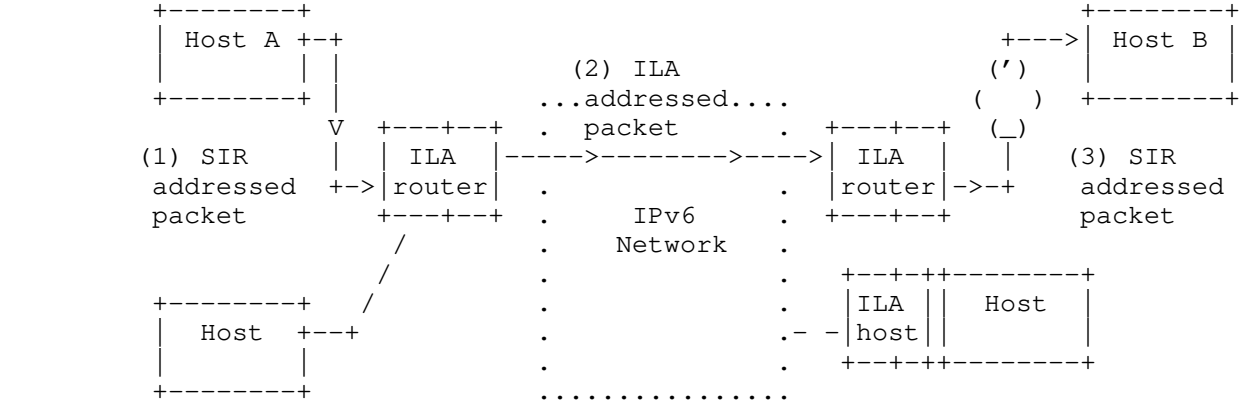
ILA translations can transform addresses from one type to another. In network virtualization virtual addresses can be translated into ILA and SIR addresses, and conversely ILA and SIR addresses can be translated to virtual addresses.

2.2 Network topology

ILA nodes are nodes in the network that perform ILA translations. An ILA router is a node that performs ILA address translation and packet forwarding to implement overlay network functionality. ILA routers perform translations on packets sent by end nodes for transport across an underlay network. Packets received by ILA routers on the underlay network have their addresses reversed translated for reception at an end node. An ILA host is an end node that implements ILA functionality for transmitting or receiving packets.

ILA nodes are responsible for transit of packets over an underlay network. On ingress to an ILA node (host or router) the virtual or SIR address of a destination is translated to an ILA address. At the a peer ILA node, the reverse translation is performed before handing packets to an application.

The figure below provides an example topology using ILA. ILA translations performed in one direction between Host A and Host B are denoted. Host A sends a packet with a destination SIR address (step (1)). An ILA router in the path translates the SIR address to an ILA address with a locator set to Host B, referring to the location of the node indicated by the identifier in the SIR address. The packet is forwarded over the network and delivered to a peer ILA node (step 2). The peer ILA node, in this case another ILA router, translates the destination address back to a SIR address and forwards to the final destination (step 3).



2.3 Translations and mappings

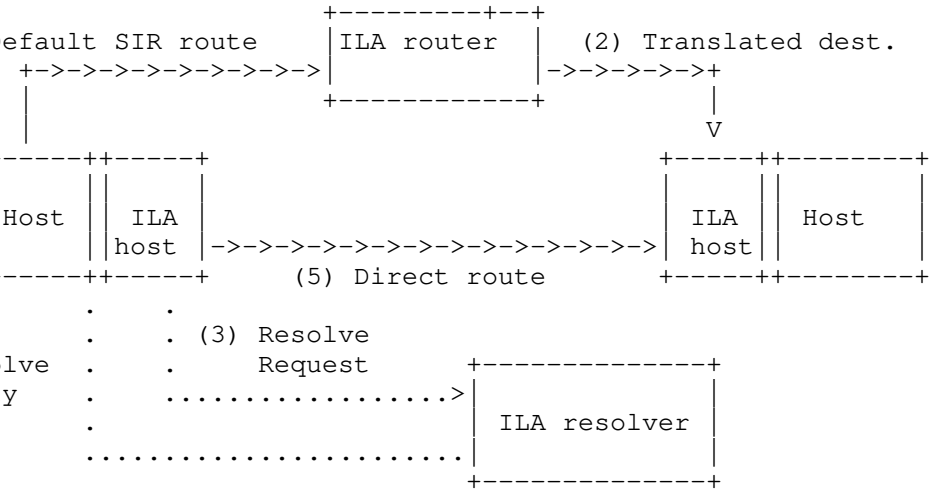
Address translation is the mechanism employed by ILA. Logical or virtual addresses are translated to topological IPv6 addresses for transport to the proper destination. Translation occurs in the upper sixty-four bits of an address, the low order sixty-four bits contains an identifier that is immutable and is not used to route a packet.

Each ILA node maintains a mapping table. This table maps identifiers to locators. The mappings are dynamic as nodes with identifiers can be created, destroyed, or migrated between physical hosts. Mappings are propagated amongst ILA routers or hosts in a network using mapping propagation protocols (mapping propagation protocols will be described in other specifications).

Identifiers are not statically bound to a host on the network, and in fact their binding (or location) may change. This is the basis for network virtualization and address migration. An identifier is mapped to a locator at any given time, and a set of identifier to locator mappings is propagated throughout a network to allow communications. The mappings are kept synchronized so that if an identifier migrates to a new physical host, its identifier to locator mapping is updated.

2.4 ILA routing

ILA is intended to be sufficiently lightweight so that all the hosts in a network could potentially send and receive ILA addressed packets. In order to scale this model and allow for hosts that do not participate in ILA, a routing topology may be applied. A simple routing topology is illustrated below.



An ILA router can be addressed by an "anycast" SIR prefix so that it receives packets sent on the network with SIR addresses. When an ILA router receives a SIR addressed packet (step (1) in the diagram) it will perform the ILA translation and send the ILA addressed packet to the destination ILA node (step (2)).

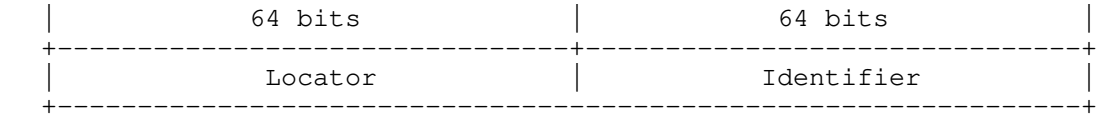
If a sending host is ILA capable the triangular routing can be eliminated by performing an ILA resolution protocol. This entails the host sending an ILA resolve request that specifies the SIR address to resolve (step (3) in the figure). An ILA resolver can respond to a resolver request with the identifier to locator mapping (step (4)). Subsequently, the ILA host can perform ILA translation and send directly to the destination specified in the locator (step (5) in the figure). The ILA resolution protocol will be specified in a companion document.

In this model an ILA host maintains a cache of identifier mappings for identifiers that it is currently communicating with. ILA routers are expected to maintain a complete list of identifier to locator mappings within the SIR domains that they service.

3 Address formats

3.1 ILA address format

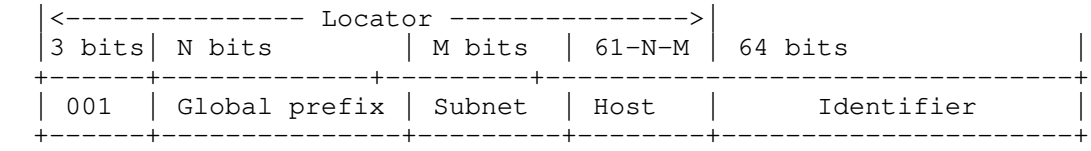
An ILA address is composed of a locator and an identifier where each occupies sixty-four bits (similar to the encoding in ILNP [RFC6741]).



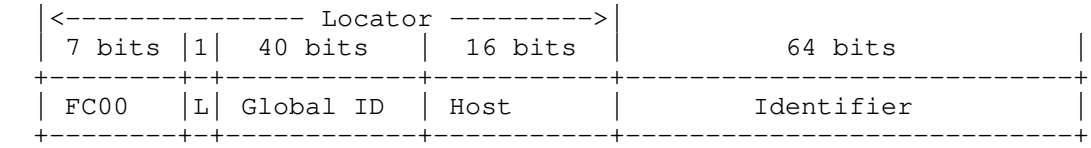
3.2 Locators

Locators are routable network address prefixes that create topological addresses for physical hosts within the network. They may be assigned from a global address block [RFC3587], or be based on unique local IPv6 unicast addresses as described in [RFC4193].

The format of an ILA address with a global unicast locator is:

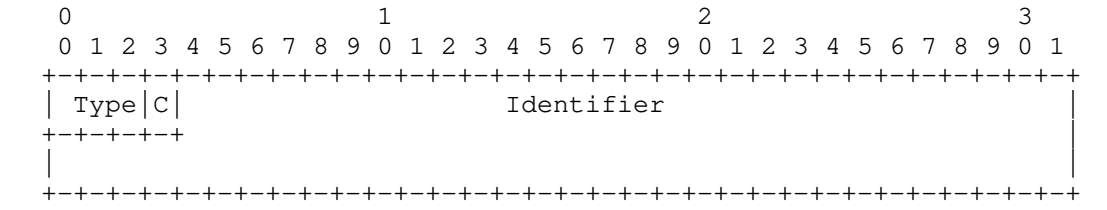


The format of an ILA address with a unique local IPv6 unicast locator is:



3.3 Identifiers

The format of an ILA identifier is:

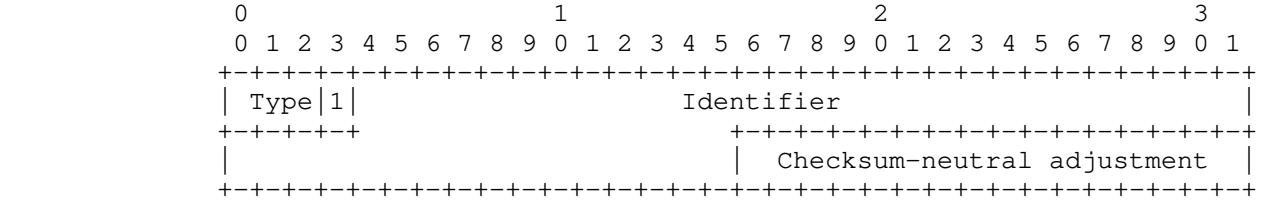


Fields are:

- Type: Type of the identifier (see section 3.3.2).
- C: The C-bit. This indicates that checksum-neutral mapping applied (see section 3.3.1).
- Identifier: Identifier value.

3.3.1 Checksum neutral-mapping format

If the C-bit is set the low order sixteen bits of an identifier contain the adjustment for checksum-neutral mapping (see section 4.4.1 for description of checksum-neutral mapping). The format of an identifier with checksum neutral mapping is:



3.3.2 Identifier types

Identifier types allow standard encodings for common uses of identifiers. Defined identifier types are:

0: interface identifier

1: locally unique identifier

2: virtual networking identifier for IPv4 address

3: virtual networking identifier for IPv6 unicast address

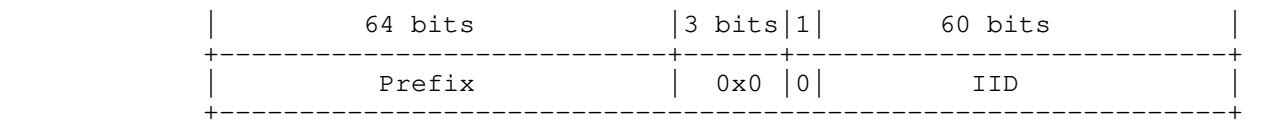
4: virtual networking identifier for IPv6 multicast address

5-7: Reserved

3.3.2.1 Interface identifiers

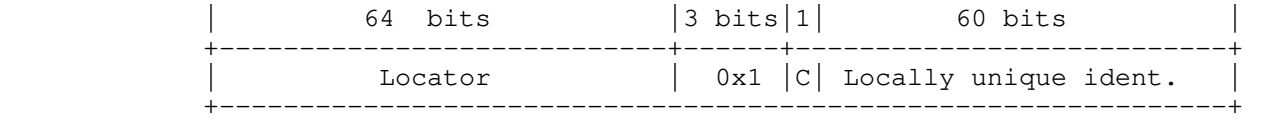
The interface identifier type indicates a plain local scope interface identifier. When this type is used the address is a normal IPv6 address without identifier-locator semantics. The purpose of this type is to allow normal IPv6 addresses to be defined within the same networking prefix as ILA addresses. Type bits and C-bit MUST be zero.

The format of an ILA interface identifier address is:

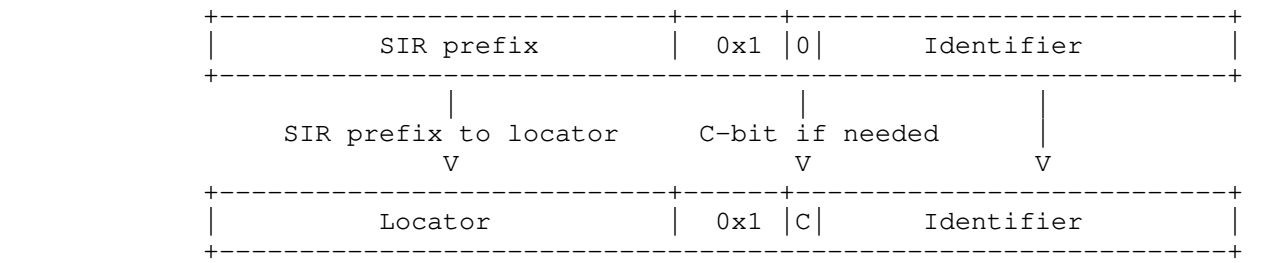


3.3.2.2 Locally unique identifiers

Locally unique identifiers (LUI) can be created for various addressable objects within a network. These identifiers are in a flat sixty bit space and must be unique within a SIR domain (unique within a site for instance). To simplify administration, hierarchical allocation of locally unique identifiers may be performed. The format of an ILA address with locally unique identifiers is:

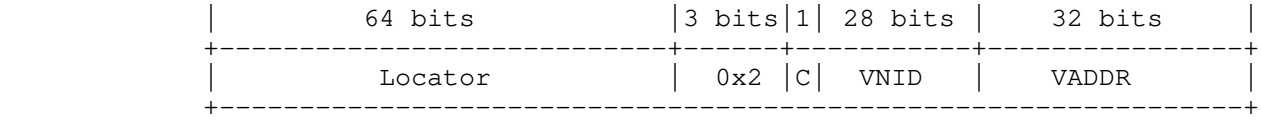


The figure below illustrates the translation from SIR address to an ILA address as would be performed when a node sends to a SIR address. Note the low order 16 bites of the identifier may be modified as the checksum-neutral adjustment. The reverse translation of ILA address to SIR address is symmetric.



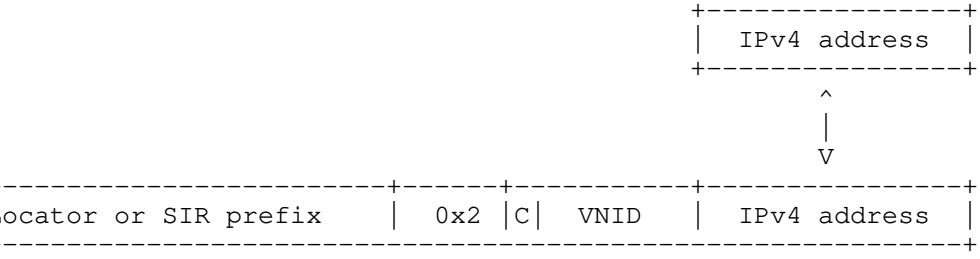
3.3.2.3 Virtual networking identifiers for IPv4

This type defines a format for encoding an IPv4 virtual address and virtual network identifier within an identifier. The format of an ILA address for IPv4 virtual networking is:



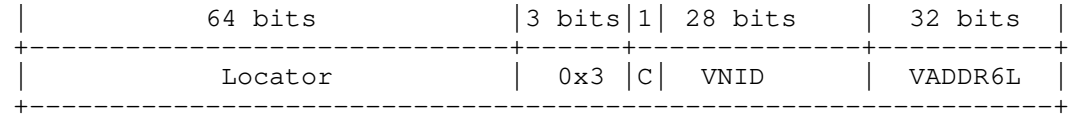
VNID is a virtual network identifier and VADDR is a virtual address within the virtual network indicated by the VNID. The VADDR can be an IPv4 unicast or multicast address, and may often be in a private address space (i.e. [RFC1918]) used in the virtual network.

Translating a virtual IPv4 address into an ILA or SIR address and the reverse translation are straight forward. Note that the low order 16 bits of the IPv6 address may be modified as the checksum-neutral adjustment and that this translation implies protocol translation when sending IPv4 packets over an ILA IPv6 network.



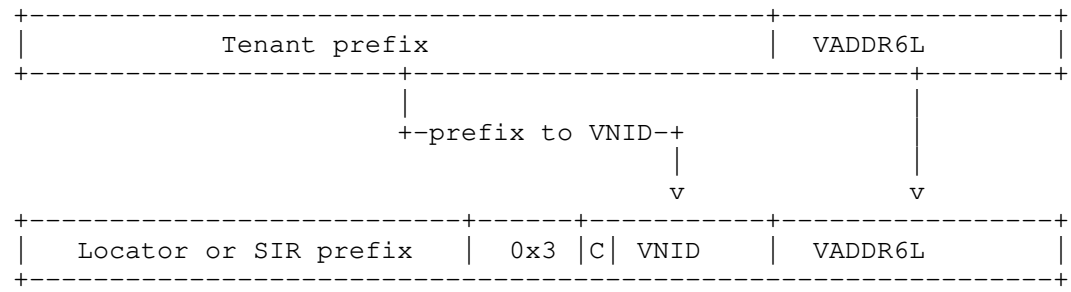
3.3.2.4 Virtual networking identifiers for IPv6 unicast

In this format, a virtual network identifier and virtual IPv6 unicast address are encoded within an identifier. To facilitate encoding of virtual addresses, there is a unique mapping between a VNID and a ninety-six bit prefix of the virtual address. The format an IPv6 unicast encoding with VNID in an ILA address is:

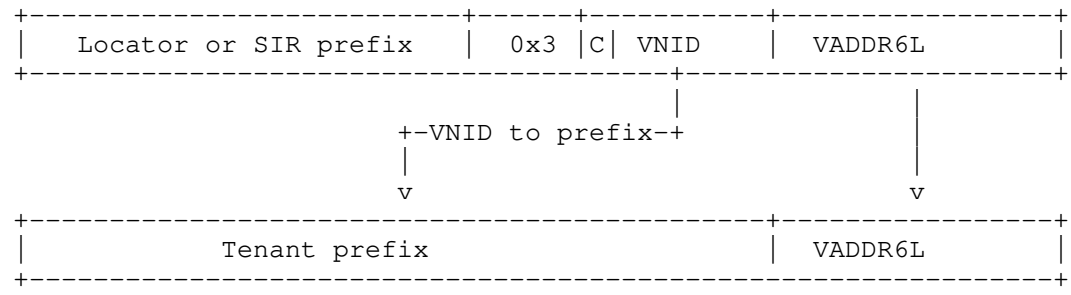


VADDR6L contains the low order 32 bits of the IPv6 virtual address. The upper 96 bits of the virtual address are inferred from the VNID to prefix mapping. Note that for ILA translations the low order sixteen of the VADDR6L may be modified for checksum-neutral adjustment.

The figure below illustrates encoding a tenant IPv6 virtual unicast address into a ILA or SIR address.

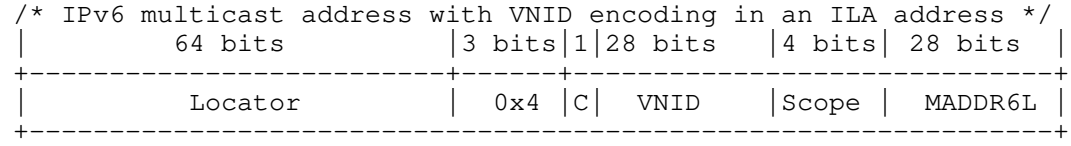


This encoding is reversible, given an ILA address, the virtual address visible to the tenant can be deduced:



3.3.2.5 Virtual networking identifiers for IPv6 multicast

In this format, a virtual network identifier and virtual IPv6 multicast address are encoded within an identifier.



This format encodes an IPv6 multicast address in an identifier. The scope indicates multicast address scope as defined in [RFC7346]. MADDR6L is the low order 28 bits of the multicast address. The full multicast address is thus:

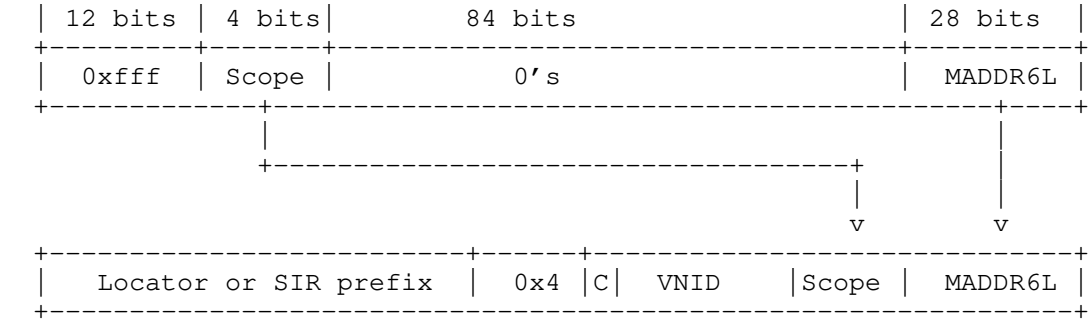
ff0<Scope>::<MADDRL6 high 12 bits>:<MADDRL6 low 16 bits>

And so can encode multicast addresses of the form:

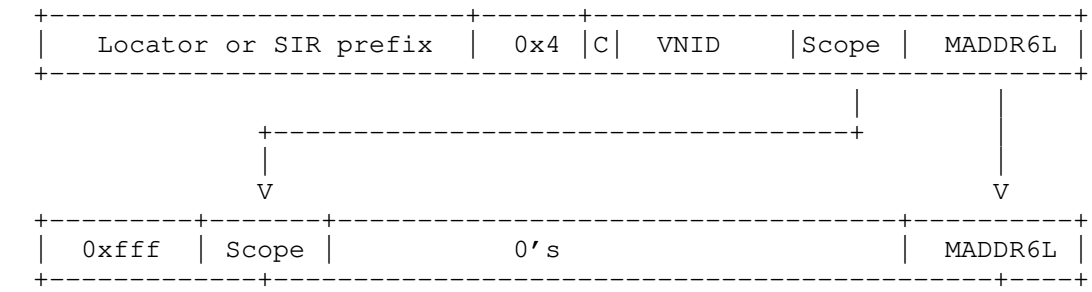
ff0X::0 to ff0X::0fff:ffff

The figure below illustrates encoding a tenant IPv6 virtual multicast

address in an ILA or SIR address. Note that low order sixteen bits of MADDR6L may be modified to be the checksum-neutral adjustment.



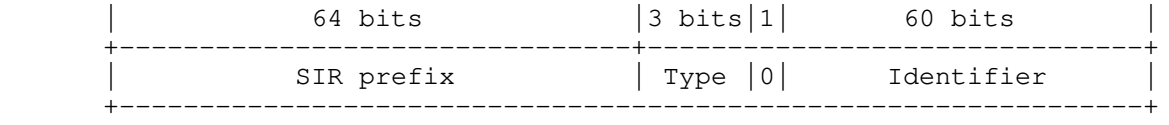
This translation is reversible:



3.4 Standard identifier representation addresses

An identifier identifies objects or nodes in a network. For instance, an identifier may refer to a specific host, virtual machine, or tenant system. When a host initiates a connection or sends a packet, it uses the identifier to indicate the peer endpoint of the communication. The endpoints of an established connection context also referenced by identifiers. It is only when the packet is actually being sent over a network that the locator for the identifier needs to be resolved.

In order to maintain compatibility with existing networking stacks and applications, identifiers are encoded in IPv6 addresses using a standard identifier representation (SIR) address. A SIR address is a combination of a prefix which occupies what would be the locator portion of an ILA address, and the identifier in its usual location. The format of a SIR address is:



The C-bit (checksum-neutral mapping) MUST be zero for a SIR address. Type may be any identifier type except zero (interface identifiers)

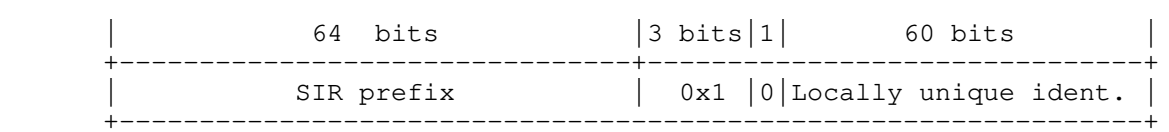
A SIR prefix may be site-local, or globally routable. A globally routable SIR prefix facilitates connectivity between hosts on the Internet and ILA nodes. A gateway between a site's network and the Internet can translate between SIR prefix and locator for an identifier. A network may have multiple SIR prefixes where each prefix defines a unique identifier space.

Locators MUST only be associated with one SIR prefix. This ensures that if a translation from a SIR address to an ILA address is performed when sending a packet, the reverse translation at the receiver yields the same SIR address that was seen at the transmitter. This also ensures that a reverse checksum-neutral mapping can be performed at a receiver to restore the addresses that were included in a pseudo header for setting a transport checksum.

A standard identifier representation address can be used as the externally visible address for a node. This can used throughout the network, returned in DNS AAAA records [RFC3363], used in logging, etc. An application can use a SIR address without knowledge that it encodes an identifier.

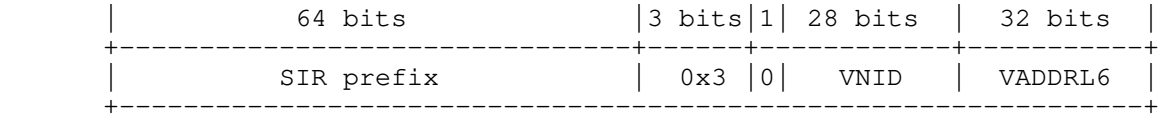
3.4.1 SIR for locally unique identifiers

The SIR address for a locally unique identifier has format:



3.4.2 SIR for virtual addresses

A virtual address can be encoded using the standard identifier representation. For example, the SIR address for an IPv6 virtual address may be:



Note that this allows three representations of the same address in he network: as a virtual address, a SIR address, and an ILA address.

3.4.3 SIR domains

Each SIR prefix defines a SIR domain. A SIR domain is a unique name space for identifiers within a domain. The full identity of a node is thus determined by an identifier and SIR domain (SIR prefix). Locators MUST map to only one SIR domain in order to ensure that translation from a locator to SIR prefix is unambiguous.

4 Operation

This section describes operation methods for using identifier-locator addressing.

4.1 Identifier to locator mapping

An application initiates a communication or flow using a SIR address or virtual address for a destination. In order to send a packet on the network, the destination address is translated by an ILA router or an ILA host in the path. An ILA node maintains a list of mappings from identifier to locator to perform this translation.

The mechanisms of propagating and maintaining identifier to locator mappings are outside the scope of this document.

4.2 Address translations

With ILA, address translation is performed to convert SIR addresses to ILA addresses, and ILA addresses to SIR addresses. Translation is usually done on a destination address as a form of source routing, however translation on source virtual addresses to SIR addresses can also be done to support some network virtualization scenarios (see appendix A.7 for example).

4.2.1 SIR to ILA address translation

When translating a SIR address to an ILA address the SIR prefix in the address is overridden with a locator, and checksum neutral mapping may be performed. Since this operation is potentially done for every packet the process should be very efficient (particularly the lookup and checksum processing operations).

The typical steps to transmit a packet using ILA are:

1) Host stack creates a packet with source address set to a local address (possibly a SIR address) for the local identity, and

the destination address is set to the SIR address or virtual address for the peer. The peer address may have been discovered through DNS or other means.

2) An ILA router or host translates the packet to use the locator. If the original destination address is a SIR address then the SIR prefix is overwritten with the locator. If the original packet is a virtually addressed tenant packet then the virtual address is translated per section 3.3.2. The locator is discovered by a lookup in the locator to identifier mappings.

3) The ILA node performs checksum-neutral mapping if configured for that (section 4.4.1).

4) Packet is forwarded on the wire. The network routes the packet to the host indicated by the locator.

4.2.2 ILA to SIR address translation

When a destination node (ILA router or end host) receives an ILA addressed packet, the ILA address MUST be translated back to a SIR address (or tenant address) before upper layer processing.

The steps of receive processing are:

1) Packet is received. The destination locator is verified to match a locator assigned to the host.

2) A lookup is performed on the destination identifier to find if it addresses a local identifier. If match is found, either the locator is overwritten with SIR prefix (for locally unique identifier type) or the address is translated back to a tenant virtual address as shown in appendix A.7.

3) Perform reverse checksum-neutral mapping if C-bit is set (section 4.4.1).

4) Perform any optional policy checks; for instance that the source may send a packet to the destination address, that packet is not illegitimately crossing virtual networks, etc.

5) Forward packet to application processing.

4.3 Virtual networking operation

When using ILA with virtual networking identifiers, address translation is performed to convert tenant virtual network and virtual addresses to ILA addresses, and ILA addresses back to a

virtual network and tenant's virtual addresses. Translation may occur on either source address, destination address, or both (see scenarios for virtual networking in Appendix A). Address translation is performed similar to the SIR translation cases described above.

4.3.1 Crossing virtual networks

With explicit configuration, virtual network hosts may communicate directly with virtual hosts in another virtual network by using SIR addresses for virtualization in both the source and destination addresses. This might be done to allow services in one virtual network to be accessed from another (by prior agreement between tenants). See appendix A.13 for example of ILA addressing for such a scenario.

4.3.2 IPv4/IPv6 protocol translation

An IPv4 tenant may send a packet that is converted to an IPv6 packet with ILA addresses. Similarly, an IPv6 packet with ILA addresses may be converted to an IPv4 packet to be received by an IPv4-only tenant. These are IPv4/IPv6 stateless protocol translations as described in [RFC6144] and [RFC6145]. See appendix A.12 for a description of these scenarios.

4.4 Transport layer checksums

Packets undergoing ILA translation may encapsulate transport layer checksums (e.g. TCP or UDP) that include a pseudo header that is affected by the translation.

ILA provides two alternatives do deal with this:

- Perform a checksum-neutral mapping to ensure that an encapsulated transport layer checksum is kept correct on the wire.

- Send the checksum as-is, that is send the checksum value based on the pseudo header before translation.

Some intermediate devices that are not the actual end point of a transport protocol may attempt to validate transport layer checksums. In particular, many Network Interface Cards (NICs) have offload capabilities to validate transport layer checksums (including any pseudo header) and return a result of validation to the host. Typically, these devices will not drop packets with bad checksums, they just pass a result to the host. Checksum offload is a performance benefit, so if packets have incorrect checksums on the wire this benefit is lost. With this incentive, applying a checksum-

neutral mapping is the recommended alternative. If it is known that the addresses of a packet are not included in a transport checksum, for instance a GRE packet is being encapsulated, then a source may choose not to perform checksum-neutral mapping.

4.4.1 Checksum-neutral mapping

When a change is made to one of the IP header fields in the IPv6 pseudo-header checksum (such as one of the IP addresses), the checksum field in the transport layer header may become invalid. Fortunately, an incremental change in the area covered by the Internet standard checksum [RFC1071] will result in a well-defined change to the checksum value [RFC1624]. So, a checksum change caused by modifying part of the area covered by the checksum can be corrected by making a complementary change to a different 16-bit field covered by the same checksum.

ILA can perform a checksum-neutral mapping when a SIR prefix or virtual address is translated to a locator in an IPv6 address, and performs the reverse mapping when translating a locator back to a SIR prefix or virtual address. The low order sixteen bits of the identifier contain the checksum adjustment value for ILA.

On transmission, the translation process is:

1) Compute the one's complement difference between the SIR prefix and the locator. Fold this value to 16 bits (add-with-carry four 16-bit words of the difference).

2) Add-with-carry the bit-wise not of the 0x1000 (i.e. 0xefff) to the value from #1. This compensates the checksum for setting the C-bit.

3) Add-with-carry the value from #2 to the low order sixteen bits of the identifier.

4) Set the resultant value from #3 in the low order sixteen bits of the identifier and set the C-bit.

Note that the "adjustment" (the 16-bit value set in the identifier in set #3) is fixed for a given SIR to locator mapping, so the adjustment value can be saved in an associated data structure for a mapping to avoid computing it for each translation.

On reception of an ILA addressed packet, if the C-bit is set in an ILA address:

1) Compute the one's complement difference between the locator in

the address and the SIR prefix that the locator is being translated to. Fold this value to 16 bits (add-with-carry four 16-bit words of the difference).

2) Add-with-carry 0x1000 to the value from #1. This compensates the checksum for clearing the C-bit.

3) Add-with-carry the value from #2 to the low order sixteen bits of the identifier.

4) Set the resultant value from #3 in the low order sixteen bits of the identifier and clear the C-bit. This restores the original identifier sent in the packet.

4.4.2 Sending an unmodified checksum

When sending an unmodified checksum, the checksum is incorrect as viewed in the packet on the wire. At the receiver, ILA translation of the destination ILA address back to the SIR address occurs before transport layer processing. This ensures that the checksum can be verified when processing the transport layer header containing the checksum. Intermediate devices are not expected to drop packets due to a bad transport layer checksum.

4.5 Address selection

There may be multiple possibilities for creating either a source or destination address. A node may be associated with more than one identifier, and there may be multiple locators for a particular identifier. The choice of locator or identifier is implementation or configuration specific. The selection of an identifier occurs at flow creation and must be invariant for the duration of the flow. Locator selection must be done at least once per flow, and the locator associated with the destination of a flow may change during the lifetime of the flow (for instance in the case of a migrating connection it will change). ILA address selection should follow specifications in Default Address Selection for Internet Protocol Version 6 (IPv6) [RFC6724].

4.6 Duplicate identifier detection

As part of implementing the locator to identifier mapping, duplicate identifier detection should be implemented in a centralized control plane. A registry of identifiers could be maintained (possibly in association the identifier to locator mapping database). When a node creates an identifier it registers the identifier, and when the identifier is no longer in use (e.g. task completes) the identifier is unregistered. The control plane should able to detect a

registration attempt for an existing identifier and deny the request.

4.7 ICMP error handling

A packet that contains an ILA address may cause ICMP errors within the network. In this case the ICMP data contains an IP header with an ILA address. ICMP messages are sent back to the source address in the packet. Upon receiving an ICMP error the host will process it differently depending on whether it is ILA capable.

4.7.1 Handling ICMP errors by ILA capable hosts

If a host is ILA capable it can attempt to reverse translate the ILA address in the destination of a header in the ICMP data back to a SIR address that was originally used to transmit the packet. The steps are:

1) Assume that the upper sixty-four bits of the destination address in the ICMP data is a locator. Try match these bits back to a SIR address. If the host is only in one SIR domain, then the mapping to SIR address is implicit. If the host is in multiple domains then a locator to SIR addresses table can be maintained for this lookup.

2) If the identifier is marked with checksum-neutral mapping, undo the checksum-neutral using the SIR address found in #1. The resulting identifier address is potentially the original address used to send the packet.

3) Lookup the identifier in the identifier to locator mapping table. If an entry is found compare the locator in the entry to the locator (upper sixty-four bits) of the destination address in the IP header of the ICMP data. If these match then proceed to next step.

4) Overwrite the upper sixty-four bits of the destination address in the ICMP data with the found SIR address and overwrite the low order sixty-four bits with the found identifier (the result of undoing checksum-neutral mapping). The resulting address should be the original SIR address used in sending. The ICMP error packet can then be received by the stack for further processing.

4.7.2 Handling ICMP errors by non-ILA capable hosts

A non-ILA capable host may receive an ICMP error generated by the network that contains an ILA address in an IP header contained in the ICMP data. This would happen in the case that an ILA router performed

translation on a packet the host sent and that packet subsequently generated an ICMP error. In this case the host receiving the error message will attempt to find the connection state corresponding to the packet in headers the ICMP data. Since the host is unaware of ILA the lookup for connection state should fail. Because the host cannot recover the original addresses it used to send the packet, it won't be able any to derive any useful information about the original destination of the packet that it sent.

If packets for a flow are always routed through an ILA router in both directions, for example ILA routers are coincident with edge routes in a network, then ICMP errors could be intercepted by an intermediate node which could translate the destination addresses in ICMP data back to the original SIR addresses. A receiving host would then see the destination address in the packet of the ICMP data to be that it used to transmit the original packet.

4.8 Multicast

ILA is generally not intended for use with multicast. In the case of multicast, routing of packets is based on the source address. Neither the SIR address nor an ILA address is suitable for use as a source address in a multicast packet. A SIR address is unroutable and hence would make a multicast packet unroutable if used as a source address. Using an ILA address as the source address makes the multicast packet routable, but this exposes ILA address to applications which is especially problematic on a multicast receiver that doesn't support ILA.

If all multicast receivers are known to support ILA, a local locator address may be used in the source address of the multicast packet. In this case, each receiver will translate the source address from an ILA address to a SIR address before delivering packets to an application.

5 Motivation for ILA

5.1 Use cases

5.1.1 Multi-tenant virtualization

In multi-tenant virtualization overlay networks are established for tenants to provide virtual networks. Each tenant may have one or more virtual networks and a tenant's nodes are assigned virtual addresses within virtual networks. Identifier-locator addressing may be used as an alternative to traditional network virtualization encapsulation protocols used to create overlay networks (e.g. VXLAN [RFC7348]). Section 5.2.4 describes the advantages of using ILA in lieu of

encapsulation protocols.

Tenant systems (e.g. VMs) run on physical hosts and may migrate to different hosts. A tenant system is identified by a virtual address and virtual networking identifier of a corresponding virtual network. ILA can encode the virtual address and a virtual networking identifier in an ILA identifier. Each identifier is mapped to a locator that indicates the current host where the tenant system resides. Nodes that send to the tenant system set the locator per the mapping. When a tenant system migrates its identifier to locator mapping is updated and communicating nodes will use the new mapping.

5.1.2 Datacenter virtualization

Datacenter virtualization virtualizes networking resources. Various objects within a datacenter can be assigned addresses and serve as logical endpoints of communication. A large address space, for example that of IPv6, allows addressing to be used beyond the traditional concepts of host based addressing. Addressed objects can include tasks, virtual IP addresses (VIPs), pieces of content, disk blocks, etc. Each object has a location which is given by the host on which an object resides. Some objects may be migratable between hosts such that their location changes over time.

Objects are identified by a unique identifier within a namespace for the datacenter (appendix B discusses methods to create unique identifiers for ILA). Each identifier is mapped to a locator that indicates the current host where the object resides. Nodes that send to an object set the locator per the mapping. When an object migrates its identifier to locator mapping is updated and communicating nodes will use the new mapping.

A datacenter object of particular interest is tasks, units of execution for for applications. The goal of virtualzing tasks is to maximize resource efficiency and job scheduling. Tasks share many properties of tenant systems, however they are finer grained objects, may have a shorter lifetimes, and are likely created in greater numbers. Appendix C provides more detail and motivation for virtualizing tasks using ILA.

5.1.3 Device mobility

ILA may be applied as a solution for mobile devices. These are devices, smart phones for instance, that physically move between different networks. The goal of mobility is to provide a seamless transition when a device moves from one network to another.

Each mobile device is identified by unique identifier within some

provider domain. ILA encodes the identifier for the device in an ILA identifier. Each identifier is mapped to a locator that indicates the current network or point of attachment for the device. Nodes that send to the device set the locator per the mapping. When a mobile device moves between networks its identifier to locator mapping is updated and communicating nodes will use the new mapping.

5.2 Alternative methods

This section discusses the merits of alternative solution that have been proposed to provide network virtualization or mobility in IPv6.

5.2.1 ILNP

ILNP splits an address into a locator and identifier in the same manner as ILA. ILNP has characteristics, not present in ILA, that prevent it from being a practical solution:

- ILNP requires that transport layer protocol implementations must be modified to work over ILNP.
- ILNP can only be implemented in end hosts, not within the network. This essentially requires that all end hosts need to be modified to participate in mobility.
- ILNP employs IPv6 extension headers which are mostly considered non-deployable. ILA does not use these.
- Core support for ILA is in upstream Linux, to date there is no publicly available source code for ILNP.
- ILNP involves DNS to distribute mapping information, ILA assumes mapping information is not part of naming.

5.2.2 Flow label as virtual network identifier

The IPv6 flow label could conceptually be used as a 20-bit virtual network identifier in order to indicate a packet is sent on an overlay network. In this model the addresses may be virtual addresses within the specified virtual network. Presumably, the tuple of flow-label and addresses could be used by switches to forward virtually addressed packets.

This approach has some issues:

- Forwarding virtual packets to their physical location would require specialized switch support.

- The flow label is only twenty bits, this is too small to be a discriminator in forwarding a virtual packet to a specific destination. Conceptually, the flow label might be used in a type of label switching to solve that.

- The flow label is not considered immutable in transit, intermediate devices may change it.

- The flow label is not part of the pseudo header for transport checksum calculation, so it is not covered by any transport (or other) checksums.

5.2.3 Extension headers

To accomplish network virtualization an extension header, as a destination or routing option, could be used that contains the virtual destination address of a packet. The destination address in the IPv6 header would be the topological address for the location of the virtual node. Conceivably, segment routing could be used to implement network virtualization in this manner.

This technique has some issues:

- Intermediate devices must not insert extension headers [RFC2460bis].

- Extension headers introduce additional packet overhead which may impact performance.

- Extension headers are not covered by transport checksums (as the addresses would be) nor any other checksum.

- Extension headers are not widely supported in network hardware or devices. For instance, several NIC offloads don't work in the presence of extension headers.

5.2.4 Encapsulation techniques

Various encapsulation techniques have been proposed for implementing network virtualization and mobility. LISP is an example of an encapsulation that is based on locator identifier separation similar to ILA. The primary drawback of encapsulation is complexity and per packet overhead. For, instance when LISP is used with IPv6 the encapsulation overhead is fifty-six bytes and two IP headers are present in every packet. This adds considerable processing costs, requires considerations to handle path MTU correctly, and certain network accelerations may be lost.

6 IANA Considerations

There are no IANA considerations in this specification.

7 References

7.1 Normative References

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2460bis] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", draft-ietf-6man-rfc2460bis-03, January 2016.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, June 2011.

[RFC1071] Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum", RFC 1071, September 1988.

[RFC1624] Rijsinghani, A., "Computation of the Internet Checksum via Incremental Update", RFC 1624, May 1994.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

7.2 Informative References

[RFC6740] RJ Atkinson and SN Bhatti, "Identifier-Locator Network Protocol (ILNP) Architectural Description", RFC 6740, November 2012.

[RFC6741] RJ Atkinson and SN Bhatti, "Identifier-Locator Network Protocol (ILNP) Engineering Considerations", RFC 6741, November 2012.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC3363] Bush, R., Durand, A., Fink, B., Gudmundsson, O., and T. Hain, "Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)", RFC 3363, August 2002.

[RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global

Unicast Address Format", RFC 3587, August 2003.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, April 2011.

[NVO3ARCH] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and Narten, T., "An Architecture for Overlay Networks (NVO3)", draft-ietf-nvo3-arch-03

[GUE] Herbert, T., and Yong, L., "Generic UDP Encapsulation", draft-herbert-gue-02, work in progress.

[GUESEC] Yong, L., and Herbert, T. "Generic UDP Encapsulation (GUE) for Secure Transport", draft-hy-gue-4-secure-transport-00, work in progress

8 Acknowledgments

The author would like to thank Mark Smith, Lucy Yong, Erik Kline, Saleem Bhatti, Petr Lapukhov, Blake Matheny, Doug Porter, Pierre Pfister, and Fred Baker for their insightful comments for this draft; Roy Bryant, Lorenzo Colitti, Mahesh Bandewar, and Erik Kline for their work on defining and applying ILA.

Appendix A: Communication scenarios

This section describes the use of identifier-locator addressing in several scenarios.

A.1 Terminology for scenario descriptions

A formal notation for identifier-locator addressing with ILNP is described in [RFC6740]. We extend this to include for network virtualization cases.

Basic terms are:

A = IP Address
I = Identifier
L = Locator
LUI = Locally unique identifier
VNI = Virtual network identifier
VA = An IPv4 or IPv6 virtual address
VAX = An IPv6 networking identifier (IPv6 VA mapped to VAX)
SIR = Prefix for standard identifier representation
VNET = IPv6 prefix for a tenant (assumed to be globally routable)
Iaddr = IPv6 address of an Internet host

An ILA IPv6 address is denoted by

L:I

A SIR address with a locally unique identifier and SIR prefix is denoted by

SIR:LUI

A virtual identifier with a virtual network identifier and a virtual IPv4 address is denoted by

VNI:VA

An ILA IPv6 address with a virtual networking identifier for IPv4 would then be denoted

L:(VNI:VA)

The local and remote address pair in a packet or endpoint is denoted

A,A

An address translation sequence from SIR addresses to ILA addresses

for transmission on the network and back to SIR addresses at a receiver has notation:

A,A -> L:I,A -> A,A

A.2 Identifier objects

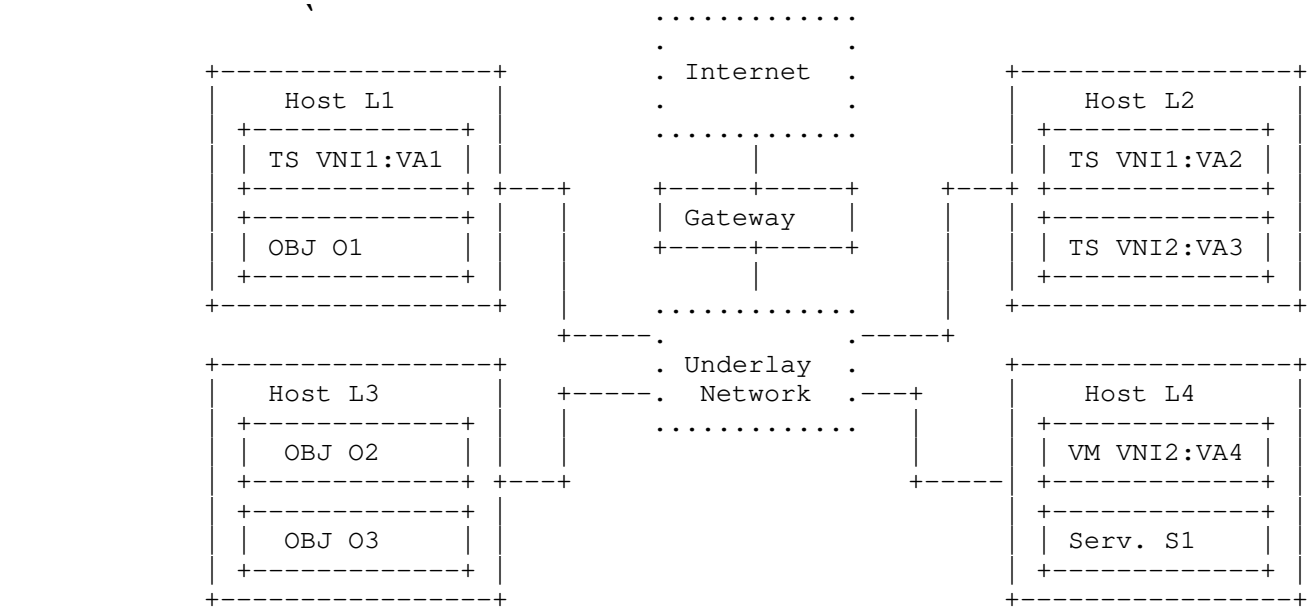
Identifier-locator addressing is broad enough in scope to address many different types of networking entities. For the purposes of this section we classify these as "objects" and "tenant systems".

Objects encompass uses where nodes are address by local unique identifiers (LUI). In the scenarios below objects are denoted by OBJ.

Tenant systems are those associated with network virtualization that have virtual addresses (that is they are addressed by VNI:VA). In the scenarios below tenant systems are denoted by TS.

A.3 Reference network for scenarios

The figure below provides an example network topology with ILA addressing in use. In this example, there are four hosts in the network with locators L1, L2, L3, and L4. There three objects with identifiers O1, O2, and O3, as well as a common networking service with identifier S1. There are two virtual networks VNI1 and VNI2, and four tenant systems addressed as: VA1 and VA2 in VNI1, VA3 and VA4 in VNI2. The network is connected to the Internet via a gateway.



Several communication scenarios can be considered:

1) Object to object
2) Object to Internet
3) Internet to object
4) Tenant system to local service
5) Object to tenant system
6) Tenant system to Internet
7) Internet to tenant system
8) IPv4 tenant system to service
9) Tenant system to tenant system same virtual network using IPv6
10) Tenant system to tenant system in same virtual network using IPv4
11) Tenant system to tenant system in different virtual network using IPv6
12) Tenant system to tenant system in different virtual network using IPv4
13) IPv4 tenant system to IPv6 tenant system in different virtual networks

A.4 Scenario 1: Object to task

The transport endpoints for object to object communication are the SIR addresses for the objects. When a packet is sent on the wire, the locator is set in the destination address of the packet. On reception the destination addresses is converted back to SIR representation for processing at the transport layer.

If object O1 is communicating with object O2, the ILA translation sequence would be:

```
SIR:O1,SIR:O2 ->                    // Transport endpoints on O1
SIR:O1,L3:O2 ->                     // ILA used on the wire
SIR:O1,SIR:O2                       // Received at O2
```

A.5 Scenario 2: Object to Internet

Communication from an object to the Internet is accomplished through use of a SIR address (globally routable) in the source address of packets. No ILA translation is needed in this path.

If object O1 is sending to an address Iaddr on the Internet, the packet addresses would be:

```
SIR:O1,Iaddr
```

A.6 Scenario 3: Internet to object

An Internet host transmits a packet to a task using an externally routable SIR address. The SIR prefix routes the packet to a gateway for the datacenter. The gateway translates the destination to an ILA address.

If a host on the Internet with address Iaddr sends a packet to object O3, the ILA translation sequence would be:

```
  Iaddr,SIR:O3 ->                    // Transport endpoint at Iaddr
  Iaddr,L1:O3 ->                     // On the wire in datacenter
  Iaddr,SIR:O3                       // Received at O3
```

A.7 Scenario 4: Tenant system to service

A tenant can communicate with a datacenter service using the SIR address of the service.

If TS VA1 is communicating with service S1, the ILA translation sequence would be:

```
  VNET:VA1,Saddr->                   // Transport endpoints in TS
  SIR:(VNET:VA1):Saddr->             // On the wire
  SIR:(VNET:VA1):Saddr               // Received at S1
```

Where VNET is the address prefix for the tenant and Saddr is the IPv6 address of the service.

The ILA translation sequence in the reverse path, service to tenant system, would be:

```
  Saddr,SIR:(VNET:VA1)               // Transport endpoints in S1
  Saddr,L1:(VNET:VA1)                // On the wire
  Saddr,VNET:VA1                     // Received at the TS
```

Note that from the point of view of the service task there is no material difference between a peer that is a tenant system versus one which is another task.

A.8 Scenario 5: Object to tenant system

An object can communicate with a tenant system through it’s externally visible address.

If object O2 is communicating with TS VA4, the ILA translation sequence would be:

```
  SIR:O2,VNET:VA4 ->                 // Transport endpoints at T2
  SIR:O2,L4:(VNI2:VAX4) ->           // On the wire
```

```
SIR:O2,VNET:VA4                    // Received at TS
```

A.9 Scenario 6: Tenant system to Internet

Communication from a TS to the Internet assumes that the VNET for the TS is globally routable, hence no ILA translation would be needed.

If TS VA4 sends a packet to the Internet, the addresses would be:

```
VNET:VA4,Iaddr
```

A.10 Scenario 7: Internet to tenant system

An Internet host transmits a packet to a tenant system using an externally routable tenant prefix and address. The prefix routes the packet to a gateway for the datacenter. The gateway translates the destination to an ILA address.

If a host on the Internet with address Iaddr is sending to TS VA4, the ILA translation sequence would be:

```
Iaddr,VNET:VA4 ->                  // Endpoint at Iaddr
Iaddr,L4:(VNI2:VAX4) ->            // On the wire in datacenter
Iaddr,VNET:VA4                     // Received at TS
```

A.11 Scenario 8: IPv4 tenant system to object

A TS that is IPv4-only may communicate with an object using protocol translation. The object would be represented as an IPv4 address in the tenant's address space, and stateless NAT64 should be usable as described in [RFC6145].

If TS VA2 communicates with object O3, the ILA translation sequence would be:

```
VA2,ADDR3 ->                       // IPv4 endpoints at TS
SIR:(VNI1:VA2),L3:O3 ->            // On the wire in datacenter
SIR:(VNI1:VA2),SIR:O3              // Received at task
```

VA2 is the IPv4 address in the tenant's virtual network, ADDR4 is an address in the tenant's address space that maps to the network service.

The reverse path, task sending to a TS with an IPv4 address, requires a similar protocol translation.

For object O3 communicate with TS VA2, the ILA translation sequence would be:

```
SIR:O3,SIR:(VNI1:VA2) ->          // Endpoints at T4
SIR:O3,L2:(VNI1:VA2) ->           // On the wire in datacenter
ADDR4,VA2                         // IPv4 endpoint at TS
```

A.12 Tenant to tenant system in the same virtual network

ILA may be used to allow tenants within a virtual network to communicate without the need for explicit encapsulation headers.

A.12.1 Scenario 9: TS to TS in the same VN using IPV6

If TS VA1 sends a packet to TS VA2, the ILA translation sequence would be:

```
VNET:VA1,VNET:VA2 ->              // Endpoints at VA1
VNET:VA1,L2:(VNI1,VAX2) ->        // On the wire
VNET:VA1,VNET:VA2 ->              // Received at VA2
```

A.12.2 Scenario 10: TS to TS in same VN using IPv4

For two tenant systems to communicate using IPv4 and ILA, IPv4/IPv6 protocol translation is done both on the transmit and receive.

If TS VA1 sends an IPv4 packet to TS VA2, the ILA translation sequence would be:

```
VA1,VA2 ->                        // Endpoints at VA1
SIR:(VNI1:VA1),L2:(VNI1,VA2) ->   // On the wire
VA1,VA2                           // Received at VA2
```

Note that the SIR is chosen by an ILA node as an appropriate SIR prefix in the underlay network. Tenant systems do not use SIR address for this communication, they only use virtual addresses.

A.13 Tenant system to tenant system in different virtual networks

A tenant system may be allowed to communicate with another tenant system in a different virtual network. This should only be allowed with explicit policy configuration.

A.13.1 Scenario 11: TS to TS in different VNs using IPV6

For TS VA4 to communicate with TS VA1 using IPv6 the translation sequence would be:

```
VNET2:VA4,VNET1:VA1->             // Endpoint at VA4
VNET2:VA4,L1:(VNI1,VAX1)->        // On the wire
VNET2:VA4,VNET1:VA1               // Received at VA1
```

Note that this assumes that VNET1 and VNET2 are globally routable between the two virtual networks.

A.13.2 Scenario 12: TS to TS in different VNs using IPv4

To allow IPv4 tenant systems in different virtual networks to communicate with each other, an address representing the peer would be mapped into each tenant's address space. IPv4/IPv6 protocol translation is done on transmit and receive.

For TS VA4 to communicate with TS VA1 using IPv4 the translation sequence may be:

```
VA4,SADDR1 ->                          // IPv4 endpoint at VA4
SIR:(VNI2:VA4),L1:(VNI1,VA1)->         // On the wire
SADDR4,VA1                             // Received at VA1
```

SADDR1 is the mapped address for VA1 in VA4's address space, and SADDR4 is the mapped address for VA4 in VA1's address space.

A.13.3 Scenario 13: IPv4 TS to IPv6 TS in different VNs

Communication may also be mixed so that an IPv4 tenant system can communicate with an IPv6 tenant system in another virtual network. IPv4/IPv6 protocol translation is done on transmit.

For TS VA4 using IPv4 to communicate with TS VA1 using IPv6 the translation sequence may be:

```
VA4,SADDR1 ->                          // IPv4 endpoint at VA4
SIR:(VNI2:VA4),L1:(VNI1,VAX1)->        // On the wire
SIR:(VNI2:VA4),VNET1:VA1               // Received at VA1
```

SADDR1 is the mapped IPv4 address for VA1 in VA4's address space.

In the reverse direction, TS VA1 using IPv6 would communicate with TS VA4 with the translation sequence:

```
VNET1:VA1,SIR:(VNI2:VA4)               // Endpoint at VA1
VNET1:VA1,L4:(VNI2:VA4)                // On the wire
SADDR1,VA4                             // Received at VA4
```

Appendix B: unique identifier generation

The unique identifier type of ILA identifiers can address 2**60 objects. This appendix describes some method to perform allocation of identifiers for objects to avoid duplicated identifiers being allocated.

B.1 Globally unique identifiers method

For small to moderate sized deployments the technique for creating locally assigned global identifiers described in [RFC4193] could be used. In this technique a SHA-1 digest of the time of day in NTP format and an EUI-64 identifier of the local host is performed. N bits of the result are used as the globally unique identifier.

The probability that two or more of these IDs will collide can be approximated using the formula:

```
P = 1 - exp(-N**2 / 2**(L+1))
```

where P is the probability of collision, N is the number of identifiers, and L is the length of an identifier.

The following table shows the probability of a collision for a range of identifiers using a 60-bit length.

Identifiers	Probability of Collision
1000	4.3368*10^-13
10000	4.3368*10^-11
100000	4.3368*10^-09
1000000	4.3368*10^-07

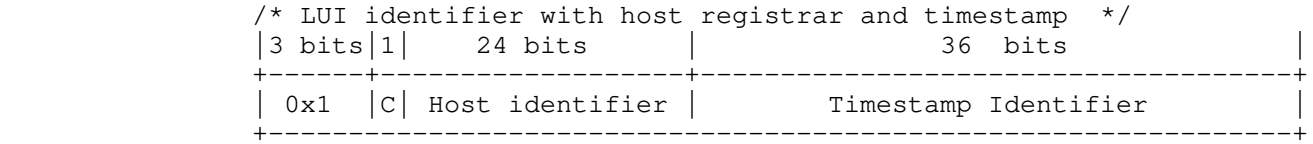
Note that locally unique identifiers may be ephemeral, for instance a task may only exist for a few seconds. This should be considered when determining the probability of identifier collision.

B.2 Universally Unique Identifiers method

For larger deployments, hierarchical allocation may be desired. The techniques in Universally Unique Identifier (UUID) URN ([RFC4122]) can be adapted for allocating unique object identifiers in sixty bits. An identifier is split into two components: a registrar prefix and sub-identifier. The registrar prefix defines an identifier block which is managed by an agent, the sub-identifier is a unique value within the registrar block.

For instance, each host in a network could be an agent so that unique identifiers for objects could be created autonomously be the host.

The identifier might be composed of a twenty-four bit host identifier followed by a thirty-six bit timestamp. Assuming that a host can allocate up to 100 identifiers per second, this allows about 21.8 years before wrap around.



Appendix C: Datacenter task virtualization

This section describes some details to apply ILA to virtualizing tasks in a datacenter.

C.1 Address per task

Managing the port number space for services within a datacenter is a nontrivial problem. When a service task is created, it may run on arbitrary hosts. The typical scenario is that the task will be started on some machine and will be assigned a port number for its service. The port number must be chosen dynamically to not conflict with any other port numbers already assigned to tasks on the same machine (possibly even other instances of the same service). A canonical name for the service is entered into a database with the host address and assigned port. When a client wishes to connect to the service, it queries the database with the service name to get both the address of an instance as well as its port number. Note that DNS is not adequate for the service lookup since it does not provide port numbers.

With ILA, each service task can be assigned its own IPv6 address and therefore will logically be assigned the full port space for that address. This a dramatic simplification since each service can now use a publicly known port number that does not need to unique between services or instances. A client can perform a lookup on the service name to get an IP address of an instance and then connect to that address using a well known port number. In this case, DNS is sufficient for directing clients to instances of a service.

C.2 Job scheduling

In the usual datacenter model, jobs are scheduled to run as tasks on some number of machines. A distributed job scheduler provides the scheduling which may entail considerable complexity since jobs will often have a variety of resource constraints. The scheduler takes these constraints into account while trying to maximize utility of

the datacenter in terms utilization, cost, latency, etc. Datacenter jobs do not typically run in virtual machines (VMs), but may run within containers. Containers are mechanisms that provide resource isolation between tasks running on the same host OS. These resources can include CPU, disk, memory, and networking.

A fundamental problem arises in that once a task for a job is scheduled on a machine, it often needs to run to completion. If the scheduler needs to schedule a higher priority job or change resource allocations, there may be little recourse but to kill tasks and restart them on a different machine. In killing a task, progress is lost which results in increased latency and wasted CPU cycles. Some tasks may checkpoint progress to minimize the amount of progress lost, but this is not a very transparent or general solution.

An alternative approach is to allow transparent job migration. The scheduler may migrate running jobs from one machine to another.

C.3 Task migration

Under the orchestration of the job scheduler, the steps to migrate a job may be:

1) Stop running tasks for the job.
2) Package the runtime state of the job. The runtime state is derived from the containers for the jobs.
3) Send the runtime state of the job to the new machine where the job is to run.
4) Instantiate the job’s state on the new machine.
5) Start the tasks for the job continuing from the point at which it was stopped.

This model similar to virtual machine (VM) migration except that the runtime state is typically much less data-- just task state as opposed to a full OS image. Task state may be compressed to reduce latency in migration.

C.3.1 Address migration

ILA facilitates address (specifically SIR address) migration between hosts as part of task migration or for other purposes. The steps in migrating an address might be:

1) Configure address on the target host.

2) Suspend use of the address on the old host. This includes handling established connections (see next section). A state may be established to drop packets or send ICMP destination

unreachable when packets to the migrated address are received.

3) Update the identifier to locator mapping database. Depending on the control plane implementation this may include pushing the new mapping to hosts.

4) Communicating hosts will learn of the new mapping via a control plane either by participation in a protocol for mapping propagation or by the ILA resolution protocol.

C.3.2 Connection migration

When a task and its addresses are migrated between machines, the disposition of existing TCP connections needs to be considered.

The simplest course of action is to drop TCP connections across a migration. Since migrations should be relatively rare events, it is conceivable that TCP connections could be automatically closed in the network stack during a migration event. If the applications running are known to handle this gracefully (i.e. reopen dropped connections) then this may be viable.

For seamless migration, open connections may be migrated between hosts. Migration of these entails pausing the connection, packaging connection state and sending to target, instantiating connection state in the peer stack, and restarting the connection. From the time the connection is paused to the time it is running again in the new stack, packets received for the connection should be silently dropped. For some period of time, the old stack will need to keep a record of the migrated connection. If it receives a packet, it should either silently drop the packet or forward it to the new location.

Author's Address

Tom Herbert
Quantonium
4701 Patrick Henry
Santa Clara, CA
EMail: tom@herbertland.com

Petr Lapukhov
1 Hacker Way
Menlo Parck, CA
EMail: petr@fb.com

NVO3 Working Group
INTERNET-DRAFT
Intended Status: Informational

Y. Li
D. Eastlake
Huawei Technologies
L. Kreeger
Arrcus, Inc
T. Narten
IBM
D. Black
Dell EMC

Expires: September 14, 2018
March 13, 2018

Split Network Virtualization Edge (Split-NVE) Control Plane Requirements
draft-ietf-nvo3-hpvr2nve-cp-req-17

Abstract

In a Split Network Virtualization Edge (Split-NVE) architecture, the functions of the NVE (Network Virtualization Edge) are split across a server and an external network equipment which is called an external NVE. The server-resident control plane functionality resides in control software, which may be part of hypervisor or container management software; for simplicity, this document refers to the hypervisor as the location of this software.

Control plane protocol(s) between a hypervisor and its associated external NVE(s) are used by the hypervisor to distribute its virtual machine networking state to the external NVE(s) for further handling. This document illustrates the functionality required by this type of control plane signaling protocol and outlines the high level requirements. Virtual machine states as well as state transitioning are summarized to help clarify the protocol requirements.

Status of this Memo

This Internet-Draft is submitted to IETF 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/1id-abstracts.html

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

Table of Contents

1. Introduction

In the Split-NVE architecture shown in Figure 1, the functionality of the NVE (Network Virtualization Edge) is split across an end device supporting virtualization and an external network device which is called an external NVE. The portion of the NVE functionality located on the end device is called the tNVE (terminal-side NVE) and the portion located on the external NVE is called the nNVE (network-side NVE) in this document. Overlay encapsulation/decapsulation functions are normally off-loaded to the nNVE on the external NVE.

The tNVE is normally implemented as a part of hypervisor or container and/or virtual switch in an virtualized end device. This document uses the term "hypervisor" throughout when describing the Split-NVE scenario where part of the NVE functionality is off-loaded to a separate device from the "hypervisor" that contains a VM (Virtual Machine) connected to a VN (Virutal Network). In this context, the term "hypervisor" is meant to cover any device type where part of the NVE functionality is off-loaded in this fashion, e.g.,a Network Service Appliance or Linux Container.

The NVO3 problem statement [RFC7364], discusses the needs for a control plane protocol (or protocols) to populate each NVE with the state needed to perform the required functions. In one scenario, an NVE provides overlay encapsulation/decapsulation packet forwarding services to Tenant Systems (TSs) that are co-resident within the NVE on the same End Device (e.g. when the NVE is embedded within a hypervisor or a Network Service Appliance). In such cases, there is no need for a standardized protocol between the hypervisor and NVE, as the interaction is implemented via software on a single device. While in the Split-NVE architecture scenarios, as shown in figure 2 to figure 4, control plane protocol(s) between a hypervisor and its associated external NVE(s) are required for the hypervisor to distribute the virtual machines networking states to the NVE(s) for further handling. The protocol is an NVE-internal protocol and runs between tNVE and nNVE logical entities. This protocol is mentioned in the NVO3 problem statement [RFC7364] and appears as the third work item.

Virtual machine states and state transitioning are summarized in this document showing events where the NVE needs to take specific actions. Such events might correspond to actions the control plane signaling protocol(s) need to take between tNVE and nNVE in the Split-NVE scenario. The high level requirements to be fulfilled are stated.

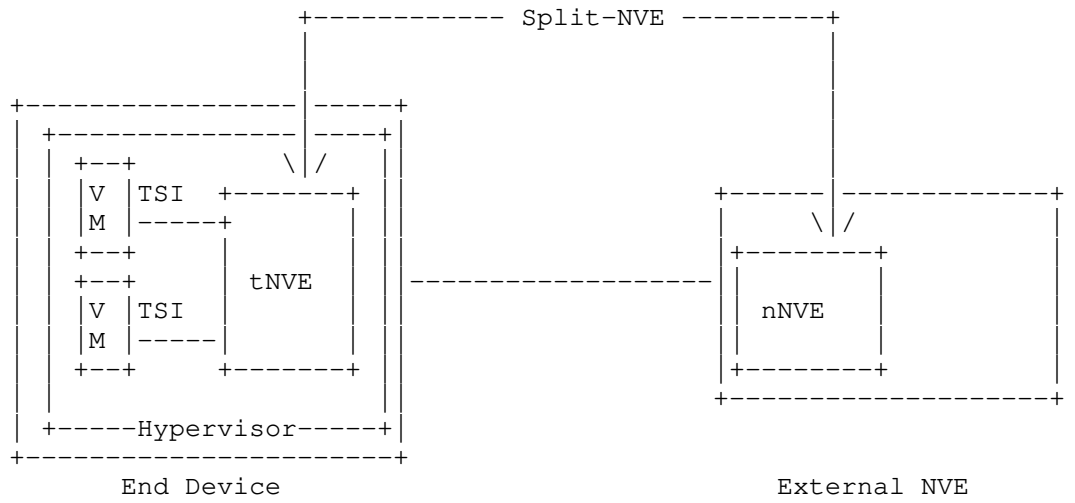


Figure 1 Split-NVE structure

This document uses VMs as an example of Tenant Systems (TSs) in order to describe the requirements, even though a VM is just one type of Tenant System that may connect to a VN. For example, a service instance within a Network Service Appliance is another type of TS, as are systems running on an OS-level virtualization technologies like containers. The fact that VMs have lifecycles (e.g., can be created and destroyed, can be moved, and can be started or stopped) results in a general set of protocol requirements, most of which are applicable to other forms of TSs although not all of the requirements are applicable to all forms of TSs.

Section 2 describes VM states and state transitioning in the VM's lifecycle. Section 3 introduces Hypervisor-to-NVE control plane protocol functionality derived from VM operations and network events. Section 4 outlines the requirements of the control plane protocol to achieve the required functionality.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the same terminology as found in [RFC7365]. This section defines additional terminology used by this document.

Split-NVE: a type of NVE (Network Virtualization Edge) where the functionalities are split across an end device supporting virtualization and an external network device.

tNVE: the portion of Split-NVE functionalities located on the end device supporting virtualization. It interacts with a tenant system through an internal interface in the end device.

nNVE: the portion of Split-NVE functionalities located on the network device that is directly or indirectly connected to the end device holding the corresponding tNVE. nNVE normally performs encapsulation to and decapsulation from the overlay network.

External NVE: the physical network device holding the nNVE

Hypervisor: the logical collection of software, firmware and/or hardware that allows the creation and running of server or service appliance virtualization. tNVE is located under a Hypervisor. Hypervisor is loosely used in this document to refer to the end device supporting the virtualization. For simplicity, we also use Hypervisor to represent both hypervisor and container.

Container: Please refer to Hypervisor. For simplicity this document use the term hypervisor to represent both hypervisor and container.

VN Profile: Meta data associated with a VN (Virtual Network) that is applied to any attachment point to the VN. That is, VAP (Virtual Access Point) properties that are applied to all VAPs associated with a given VN and used by an NVE when ingressing/egressing packets to/from a specific VN. Meta data could include such information as ACLs, QoS settings, etc. The VN Profile contains parameters that apply to the VN as a whole. Control protocols between the NVE and NVA (Network Virtualization Authority) could use the VN ID or VN Name to obtain the VN Profile.

VSI: Virtual Station Interface. [IEEE 802.1Q]

VDP: VSI Discovery and Configuration Protocol [IEEE 802.1Q]

1.2 Target Scenarios

In the Split-NVE architecture, an external NVE can provide an offload of the encapsulation / decapsulation functions and network policy enforcement as well as the VN Overlay protocol overhead. This

offloading may improve performance and/or save resources in the End Device (e.g. hypervisor) using the external NVE.

The following figures give example scenarios of a Split-NVE architecture.

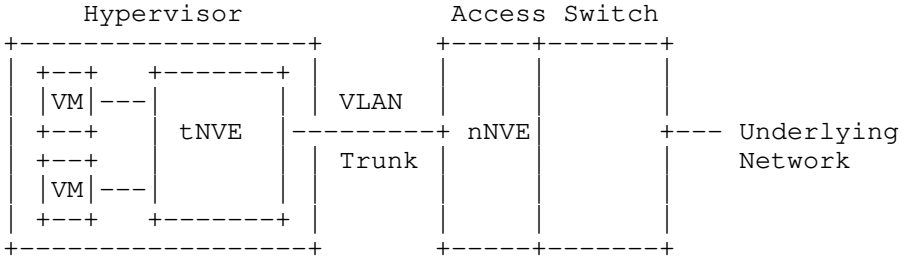


Figure 2 Hypervisor with an External NVE

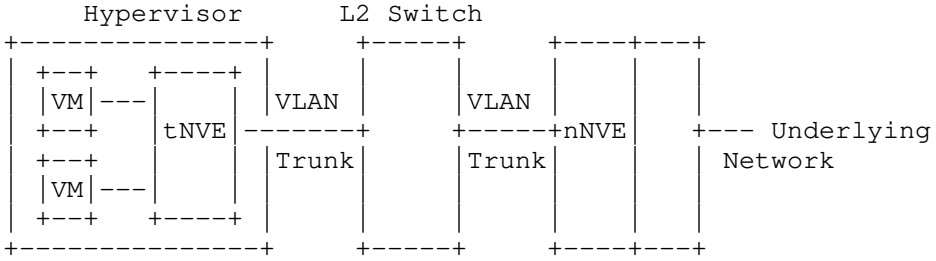


Figure 3 Hypervisor with an External NVE
connected through an Ethernet Access Switch

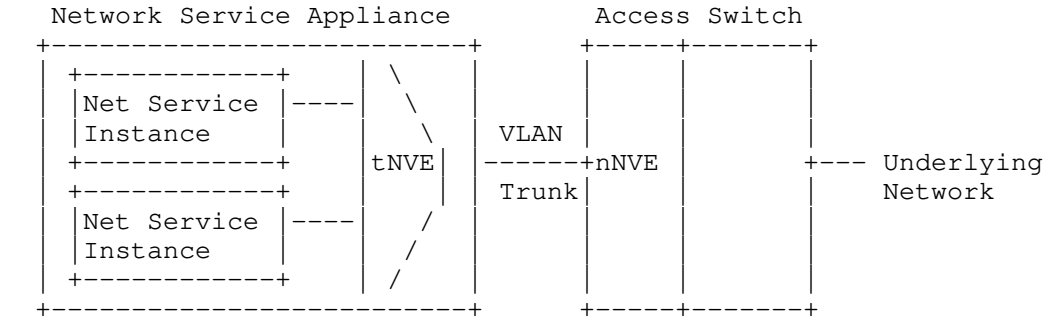


Figure 4 Physical Network Service Appliance with an External NVE

Tenant Systems connect to external NVEs via a Tenant System Interface (TSI). The TSI logically connects to the external NVE via a Virtual Access Point (VAP) [RFC8014]. The external NVE may provide Layer 2 or Layer 3 forwarding. In the Split-NVE architecture, the external NVE may be able to reach multiple MAC and IP addresses via a TSI. An IP address can be in either IPv4 or IPv6 format. For example, Tenant Systems that are providing network services (such as transparent firewall, load balancer, or VPN gateway) are likely to have a complex address hierarchy. This implies that if a given TSI disassociates from one VN, all the MAC and/or IP addresses are also disassociated. There is no need to signal the deletion of every MAC or IP when the TSI is brought down or deleted. In the majority of cases, a VM will be acting as a simple host that will have a single TSI and single MAC and IP visible to the external NVE.

Figures 2 through 4 show the use of VLANs to separate traffic for multiple VNs between the tNVE and nNVE; VLANs are not strictly necessary if only one VN is involved, but multiple VNs are expected in most cases. Hence this draft assumes the presence of VLANs.

2. VM Lifecycle

Figure 2 of [RFC7666] shows the state transition of a VM. Some of the VM states are of interest to the external NVE. This section illustrates the relevant phases and events in the VM lifecycle. Note that the following subsections do not give an exhaustive traversal of VM lifecycle state. They are intended as the illustrative examples which are relevant to Split-NVE architecture, not as prescriptive text; the goal is to capture sufficient detail to set a context for the signaling protocol functionality and requirements described in the following sections.

2.1 VM Creation Event

The VM creation event causes the VM state transition from Preparing to Shutdown and then to Running [RFC7666]. The end device allocates and initializes local virtual resources like storage in the VM Preparing state. In the Shutdown state, the VM has everything ready except that CPU execution is not scheduled by the hypervisor and VM's memory is not resident in the hypervisor. The transition from the Shutdown state to the Running state normally requires human action or a system triggered event. Running state indicates the VM is in the normal execution state. As part of transitioning the VM to the Running state, the hypervisor must also provision network connectivity for the VM's TSI(s) so that Ethernet frames can be sent and received correctly. Initially, when Running, no ongoing

migration, suspension or shutdown is in process.

In the VM creation phase, the VM's TSI has to be associated with the external NVE. Association here indicates that hypervisor and the external NVE have signaled each other and reached some agreement. Relevant networking parameters or information have been provisioned properly. The External NVE should be informed of the VM's TSI MAC address and/or IP address. In addition to external network connectivity, the hypervisor may provide local network connectivity between the VM's TSI and other VM's TSI that are co-resident on the same hypervisor. When the intra- or inter-hypervisor connectivity is extended to the external NVE, a locally significant tag, e.g. VLAN ID, should be used between the hypervisor and the external NVE to differentiate each VN's traffic. Both the hypervisor and external NVE sides must agree on that tag value for traffic identification, isolation, and forwarding.

The external NVE may need to do some preparation before it signals successful association with the TSI. Such preparation may include locally saving the states and binding information of the tenant system interface and its VN, communicating with the NVA for network provisioning, etc.

Tenant System interface association should be performed before the VM enters the Running state, preferably in the Shutdown state. If association with an external NVE fails, the VM should not go into the Running state.

2.2 VM Live Migration Event

Live migration is sometimes referred to as "hot" migration in that, from an external viewpoint, the VM appears to continue to run while being migrated to another server (e.g., TCP connections generally survive this class of migration). In contrast, "cold" migration consists of shutting down VM execution on one server and restarting it on another. For simplicity, the following abstract summary of live migration assumes shared storage, so that the VM's storage is accessible to the source and destination servers. Assume VM live migrates from hypervisor 1 to hypervisor 2. Such a migration event involves state transitions on both source hypervisor 1 and destination hypervisor 2. The VM state on source hypervisor 1 transits from Running to Migrating and then to Shutdown [RFC7666]. The VM state on destination hypervisor 2 transits from Shutdown to Migrating and then Running.

The external NVE connected to destination hypervisor 2 has to associate the migrating VM's TSI with it by discovering the TSI's MAC

and/or IP addresses, its VN, locally significant VLAN ID if any, and provisioning other network related parameters of the TSI. The external NVE may be informed about the VM's peer VMs, storage devices and other network appliances with which the VM needs to communicate or is communicating. The migrated VM on destination hypervisor 2 should not go to Running state until all the network provisioning and binding has been done.

The states of VM on the source and destination hypervisors both are Migrating during transfer of migration execution. The migrating VM should not be in Running state at the same time on the source hypervisor and destination hypervisor during migration. The VM on the source hypervisor does not transition into Shutdown state until the VM successfully enters the Running state on the destination hypervisor. It is possible that the VM on the source hypervisor stays in Migrating state for a while after the VM on the destination hypervisor enters Running state.

2.3 VM Termination Event

A VM termination event is also referred to as "powering off" a VM. A VM termination event leads to its state becoming Shutdown. There are two possible causes of VM termination [RFC7666]. One is the normal "power off" of a running VM; the other is that the VM has been migrated to another hypervisor and the VM image on the source hypervisor has to stop executing and be shutdown.

In VM termination, the external NVE connecting to that VM needs to deprovision the VM, i.e. delete the network parameters associated with that VM. In other words, the external NVE has to de-associate the VM's TSI.

2.4 VM Pause, Suspension and Resumption Events

A VM pause event leads to the VM transiting from Running state to Paused state. The Paused state indicates that the VM is resident in memory but no longer scheduled to execute by the hypervisor [RFC7666]. The VM can be easily re-activated from Paused state to Running state.

A VM suspension event leads to the VM transiting from Running state to Suspended state. A VM resumption event leads to the VM transiting state from Suspended state to Running state. Suspended state means the memory and CPU execution state of the virtual machine are saved to persistent store. During this state, the virtual machine is not scheduled to execute by the hypervisor [RFC7666].

In the Split-NVE architecture, the external NVE should not

disassociate the paused or suspended VM as the VM can return to Running state at any time.

3. Hypervisor-to-NVE Control Plane Protocol Functionality

The following subsections show illustrative examples of the state transitions of an external NVE which are relevant to Hypervisor-to-NVE Signaling protocol functionality. It should be noted this is not prescriptive text for the full state machine.

3.1 VN Connect and Disconnect

In the Split-NVE scenario, a protocol is needed between the End Device (e.g. Hypervisor) and the external NVE it is using in order to make the external NVE aware of the changing VN membership requirements of the Tenant Systems within the End Device.

A key driver for using a protocol rather than using static configuration of the external NVE is because the VN connectivity requirements can change frequently as VMs are brought up, moved, and brought down on various hypervisors throughout the data center or external cloud.

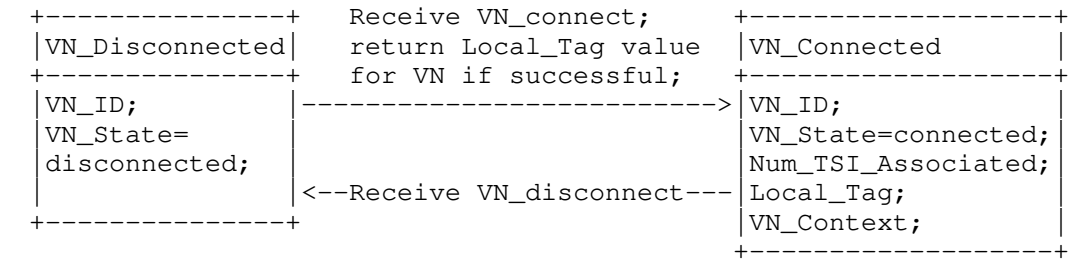


Figure 5. State Transition Example of a VAP Instance on an External NVE

Figure 5 shows the state transition for a VAP on the external NVE. An NVE that supports the hypervisor to NVE control plane protocol should support one instance of the state machine for each active VN. The state transition on the external NVE is normally triggered by the hypervisor-facing side events and behaviors. Some of the interleaved interaction between NVE and NVA will be illustrated to better explain the whole procedure; while others of them may not be shown.

The external NVE must be notified when an End Device requires connection to a particular VN and when it no longer requires connection. Connection clean up for the failed devices should be employed which is out of the scope of the protocol specified in this document.

In addition, the external NVE should provide a local tag value for each connected VN to the End Device to use for exchanging packets between the End Device and the external NVE (e.g. a locally significant [IEEE 802.1Q] tag value). How "local" the significance is depends on whether the Hypervisor has a direct physical connection to the external NVE (in which case the significance is local to the physical link), or whether there is an Ethernet switch (e.g. a blade switch) connecting the Hypervisor to the NVE (in which case the significance is local to the intervening switch and all the links connected to it).

These VLAN tags are used to differentiate between different VNs as packets cross the shared access network to the external NVE. When the external NVE receives packets, it uses the VLAN tag to identify their VN coming from a given TSI, strips the tag, adds the appropriate overlay encapsulation for that VN, and sends it towards the corresponding remote NVE across the underlying IP network.

The Identification of the VN in this protocol could either be through a VN Name or a VN ID. A globally unique VN Name facilitates portability of a Tenant’s Virtual Data Center. Once an external NVE receives a VN connect indication, the NVE needs a way to get a VN Context allocated (or receive the already allocated VN Context) for a given VN Name or ID (as well as any other information needed to transmit encapsulated packets). How this is done is the subject of the NVE-to-NVA protocol which are part of work items 1 and 2 in [RFC7364]. The external NVE needs to synchronize the mapping information of the local tag and VN Name or VN ID with NVA.

The VN_connect message can be explicit or implicit. Explicit means the hypervisor sends a request message explicitly for the connection to a VN. Implicit means the external NVE receives other messages, e.g. very first TSI associate message (see the next subsection) for a given VN, that implicitly indicate its interest in connecting to a VN.

A VN_disconnect message indicates that the NVE can release all the resources for that disconnected VN and transit to VN_disconnected state. The local tag assigned for that VN can possibly be reclaimed for use by another VN.

3.2 TSI Associate and Activate

Typically, a TSI is assigned a single MAC address and all frames transmitted and received on that TSI use that single MAC address. As mentioned earlier, it is also possible for a Tenant System to exchange frames using multiple MAC addresses or packets with multiple IP addresses.

Particularly in the case of a TS that is forwarding frames or packets from other TSs, the external NVE will need to communicate the mapping between the NVE's IP address on the underlying network and ALL the addresses the TS is forwarding on behalf of the corresponding VN to the NVA.

The NVE has two ways it can discover the tenant addresses for which frames are to be forwarded to a given End Device (and ultimately to the TS within that End Device).

1. It can glean the addresses by inspecting the source addresses in packets it receives from the End Device.

2. The hypervisor can explicitly signal the address associations of a TSI to the external NVE. An address association includes all the MAC and/or IP addresses possibly used as source addresses in a packet sent from the hypervisor to external NVE. The external NVE may further use this information to filter the future traffic from the hypervisor.

To use the second approach above, the "hypervisor-to-NVE" protocol must support End Devices communicating new tenant addresses associations for a given TSI within a given VN.

Figure 6 shows the example of a state transition for a TSI connecting to a VAP on the external NVE. An NVE that supports the hypervisor to NVE control plane protocol may support one instance of the state machine for each TSI connecting to a given VN.

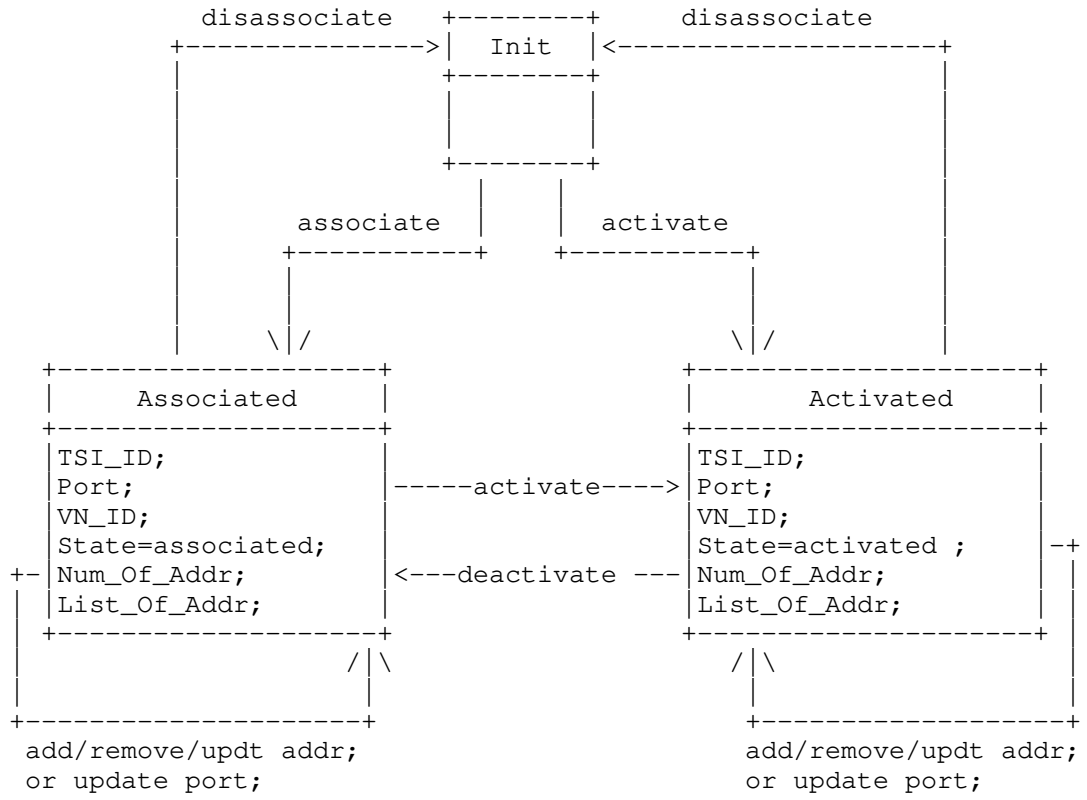


Figure 6 State Transition Example of a TSI Instance on an External NVE

The Associated state of a TSI instance on an external NVE indicates all the addresses for that TSI have already associated with the VAP of the external NVE on a given port e.g. on port p for a given VN but no real traffic to and from the TSI is expected and allowed to pass through. An NVE has reserved all the necessary resources for that TSI. An external NVE may report the mappings of its underlay IP address and the associated TSI addresses to NVA and relevant network nodes may save such information to their mapping tables but not their forwarding tables. An NVE may create ACL or filter rules based on the associated TSI addresses on that attached port p but not enable them yet. The local tag for the VN corresponding to the TSI instance should be provisioned on port p to receive packets.

The VM migration event (discussed section 2) may cause the hypervisor to send an associate message to the NVE connected to the destination hypervisor of the migration. A VM creation event may also cause to

the same practice.

The Activated state of a TSI instance on an external NVE indicates that all the addresses for that TSI are functioning correctly on a given port e.g. port p and traffic can be received from and sent to that TSI via the NVE. The mappings of the NVE's underlay IP address and the associated TSI addresses should be put into the forwarding table rather than the mapping table on relevant network nodes. ACL or filter rules based on the associated TSI addresses on the attached port p in the NVE are enabled. The local tag for the VN corresponding to the TSI instance must be provisioned on port p to receive packets.

The Activate message makes the state transit from Init or Associated to Activated. VM creation, VM migration, and VM resumption events discussed in Section 4 may trigger sending the Activate message from the hypervisor to the external NVE.

TSI information may get updated in either the Associated or Activated state. The following are considered updates to the TSI information: add or remove the associated addresses, update the current associated addresses (for example updating IP for a given MAC), and update the NVE port information based on where the NVE receives messages. Such updates do not change the state of TSI. When any address associated with a given TSI changes, the NVE should inform the NVA to update the mapping information for NVE's underlying address and the associated TSI addresses. The NVE should also change its local ACL or filter settings accordingly for the relevant addresses. Port information updates will cause the provisioning of the local tag for the VN corresponding to the TSI instance on new port and removal from the old port.

3.3 TSI Disassociate and Deactivate

Disassociate and deactivate behaviors are conceptually the reverse of associate and activate.

From Activated state to Associated state, the external NVE needs to make sure the resources are still reserved but the addresses associated to the TSI are not functioning. No traffic to or from the TSI is expected or allowed to pass through. For example, the NVE needs to tell the NVA to remove the relevant addresses mapping information from forwarding and routing tables. ACL and filtering rules regarding the relevant addresses should be disabled.

From Associated or Activated state to the Init state, the NVE releases all the resources relevant to TSI instances. The NVE should also inform the NVA to remove the relevant entries from mapping table. ACL or filtering rules regarding the relevant addresses should

be removed. Local tag provisioning on the connecting port on NVE should be cleared.

A VM suspension event (discussed in section 2) may cause the relevant TSI instance(s) on the NVE to transit from Activated to Associated state.

A VM pause event normally does not affect the state of the relevant TSI instance(s) on the NVE as the VM is expected to run again soon.

A VM shutdown event will normally cause the relevant TSI instance(s) on the NVE to transition to Init state from Activated state. All resources should be released.

A VM migration will cause the TSI instance on the source NVE to leave Activated state. When a VM migrates to another hypervisor connecting to the same NVE, i.e. source and destination NVE are the same, NVE should use TSI_ID and incoming port to differentiate two TSI instances.

Although the triggering messages for the state transition shown in Figure 6 does not indicate the difference between a VM creation/shutdown event and a VM migration arrival/departure event, the external NVE can make optimizations if it is given such information. For example, if the NVE knows the incoming activate message is caused by migration rather than VM creation, some mechanisms may be employed or triggered to make sure the dynamic configurations or provisionings on the destination NVE are the same as those on the source NVE for the migrated VM. For example an IGMP query [RFC2236] can be triggered by the destination external NVE to the migrated VM so that VM is forced to send an IGMP report to the multicast router. Then a multicast router can correctly route the multicast traffic to the new external NVE for those multicast groups the VM joined before the migration.

4. Hypervisor-to-NVE Control Plane Protocol Requirements

Req-1: The protocol MUST support a bridged network connecting End Devices to the External NVE.

Req-2: The protocol MUST support multiple End Devices sharing the same External NVE via the same physical port across a bridged network.

Req-3: The protocol MAY support an End Device using multiple external NVEs simultaneously, but only one external NVE for each VN.

Req-4: The protocol MAY support an End Device using multiple external NVEs simultaneously for the same VN.

Req-5: The protocol MUST allow the End Device to initiate a request to its associated External NVE to be connected/disconnected to a given VN.

Req-6: The protocol MUST allow an External NVE initiating a request to its connected End Devices to be disconnected from a given VN.

Req-7: When a TS attaches to a VN, the protocol MUST allow for an End Device and its external NVE to negotiate one or more locally-significant tag(s) for carrying traffic associated with a specific VN (e.g., [IEEE 802.1Q] tags).

Req-8: The protocol MUST allow an End Device initiating a request to associate/disassociate and/or activate/deactive some or all address(es) of a TSI instance to a VN on an NVE port.

Req-9: The protocol MUST allow the External NVE initiating a request to disassociate and/or deactivate some or all address(es) of a TSI instance to a VN on an NVE port.

Req-10: The protocol MUST allow an End Device initiating a request to add, remove or update address(es) associated with a TSI instance on the external NVE. Addresses can be expressed in different formats, for example, MAC, IP or pair of IP and MAC.

Req-11: The protocol MUST allow the External NVE and the connected End Device to authenticate each other.

Req-12: The protocol MUST be able to run over L2 links between the End Device and its External NVE.

Req-13: The protocol SHOULD support the End Device indicating if an associate or activate request from it is the result of a VM hot migration event.

5. VDP Applicability and Enhancement Needs

Virtual Station Interface (VSI) Discovery and Configuration Protocol (VDP) [IEEE 802.1Q] can be the control plane protocol running between the hypervisor and the external NVE. Appendix A illustrates VDP for the reader's information.

VDP facilitates the automatic discovery and configuration of Edge

Virtual Bridging (EVB) stations and Edge Virtual Bridging (EVB) bridges. An EVB station is normally an end station running multiple VMs. It is conceptually equivalent to a hypervisor in this document. An EVB bridge is conceptually equivalent to the external NVE.

VDP is able to pre-associate/associate/de-associate a VSI on an EVB station with a port on the EVB bridge. A VSI is approximately the concept of a virtual port by which a VM connects to the hypervisor in this document's context. The EVB station and the EVB bridge can reach agreement on VLAN ID(s) assigned to a VSI via VDP message exchange. Other configuration parameters can be exchanged via VDP as well. VDP is carried over the Edge Control Protocol(ECP) [IEEE 802.1Q] which provides a reliable transportation over a layer 2 network.

VDP protocol needs some extensions to fulfill the requirements listed in this document. Table 1 shows the needed extensions and/or clarifications in the NVO3 context.

<table>
<tr><th>Req</th><th>Supported by VDP?</th><th>remarks</th></tr>
<tr><td>Req-1
Req-2
Req-3
Req-4</td><td>Partially</td><td>Needs extension. Must be able to send to a specific unicast MAC and should be able to send to a non-reserved well known multicast address other than the nearest customer bridge address.</td></tr>
<tr><td>Req-5</td><td>Yes</td><td>VN is indicated by GroupID</td></tr>
<tr><td>Req-6</td><td>Yes</td><td>Bridge sends De-Associate</td></tr>
<tr><td>Req-7</td><td>Yes</td><td>VID==NULL in request and bridge returns the assigned value in response or specify GroupID in request and get VID assigned in returning response. Multiple VLANs per group are allowed.</td></tr>
<tr><td>Req-8</td><td>Partially</td><td><table><tr><th>requirements</th><th>VDP equivalence</th></tr><tr><td>associate/disassociate
activate/deactivate</td><td>pre-asso/de-associate
associate/de-associate</td></tr></table>Needs extension to allow associate->pre-assoc</td></tr>
</table>

Req-9	Yes	VDP bridge initiates de-associate
Req-10	Partially	Needs extension for IPv4/IPv6 address. Add a new "filter info format" type.
Req-11	No	Out-of-band mechanism is preferred, e.g. MACSec or 802.1X. Implicit authentication based on control of physical connectivity exists in VDP when the External NVE connects to the End Device directly and is reachable with the nearest customer bridge address.
Req-12	Yes	L2 protocol naturally
Req-13	Partially	M bit for migrated VM on destination hypervisor and S bit for that on source hypervisor. It is indistinguishable when M/S is 0 between no guidance and events not caused by migration where NVE may act differently. Needs new New bits for migration indication in new "filter info format" type.

Table 1 Compare VDP with the requirements

Simply adding the ability to carry layer 3 addresses, VDP can serve the Hypervisor-to-NVE control plane functions pretty well. Other extensions are the improvement of the protocol capabilities for better fit in an NVO3 network.

6. Security Considerations

External NVEs must ensure that only properly authorized Tenant Systems are allowed to join and become a part of any particular Virtual Network. In some cases, tNVE may want to connect to the nNVE for provisioning purposes. This may require that the tNVE authenticate the nNVE in addition to the nNVE authenticating the tNVE. If a secure channel is required between tNVE and nNVE to carry encrypted split-NVE control plane protocol, then existing mechanisms such as MACsec [IEEE 802.1AE] can be used. In some deployments, authentication may be implicit based on control of physical connectivity, e.g., if the nNVE is located in the bridge that is directly connected to the server that contains the tNVE. Use of "nearest customer bridge address" in VDP [IEEE 802.1Q] is an example where this sort of implicit authentication is possible, although explicit authentication also applies in that case.

As the control plane protocol results in configuration changes for

both the tNVE and nNVE, tNVE and nNVE implementations should log all state changes, including those described in Section 3. Implementations should also log significant protocol events, such as establishment or loss of control plane protocol connectivity between the tNVE and nNVE and authentication results.

In addition, external NVEs will need appropriate mechanisms to ensure that any hypervisor wishing to use the services of an NVE is properly authorized to do so. One design point is whether the hypervisor should supply the external NVE with necessary information (e.g., VM addresses, VN information, or other parameters) that the external NVE uses directly, or whether the hypervisor should only supply a VN ID and an identifier for the associated VM (e.g., its MAC address), with the external NVE using that information to obtain the information needed to validate the hypervisor-provided parameters or obtain related parameters in a secure manner. The former approach can be used in a trusted environment so that the external NVE can directly use all the information retrieved from the hypervisor for local configuration. It saves the effort on the external NVE side from information retrieval and/or validation. The latter approach gives more reliable information as the external NVE needs to retrieve them from some management system database. Especially some network related parameters like VLAN IDs can be passed back to hypervisor to be used as a more authoritative provisioning. However in certain cases, it is difficult or inefficient for an external NVE to have access or query on some information to those management systems. Then the external NVE has to obtain those information from hypervisor.

7. IANA Considerations

No IANA action is required.

8. Acknowledgements

This document was initiated based on the merger of the drafts draft-kreeger-nvo3-hypervisor-nve-cp, draft-gu-nvo3-tes-nve-mechanism, and draft-kompella-nvo3-server2nve. Thanks to all the co-authors and contributing members of those drafts.

The authors would like to specially thank Lucy Yong and Jon Hudson for their generous help in improving this document.

8. References

8.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for DC Network Virtualization", October 2014.

[RFC7666] Asai H., MacFaden M., Schoenwaelder J., Shima K., Tsou T., "Management Information Base for Virtual Machines Controlled by a Hypervisor", October 2015.

[RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., Narten, T., "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", December 2016.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words ", BCP 14, RFC 8174, May 2017.

[IEEE 802.1Q] IEEE, "Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks", IEEE Std 802.1Q-2014, November 2014.

8.2 Informative References

[RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, November 1997.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC7364] Narten, T., Gray, E., Black, D., Fang, L., Kreeger, L., and M. Napierala, "Problem Statement: Overlays for Network Virtualization", October 2014.

[IEEE 802.1AE] IEEE, "MAC Security (MACsec)", IEEE Std 802.1AE-2006, August 2006.

Appendix A. IEEE 802.1Q VDP Illustration (For information only)

The VDP (VSI Discovery and Discovery and Configuration Protocol, clause 41 of [IEEE 802.1Q]) can be considered as a controlling protocol running between the hypervisor and the external bridge. VDP association TLV structure are formatted as shown in Figure A.1.

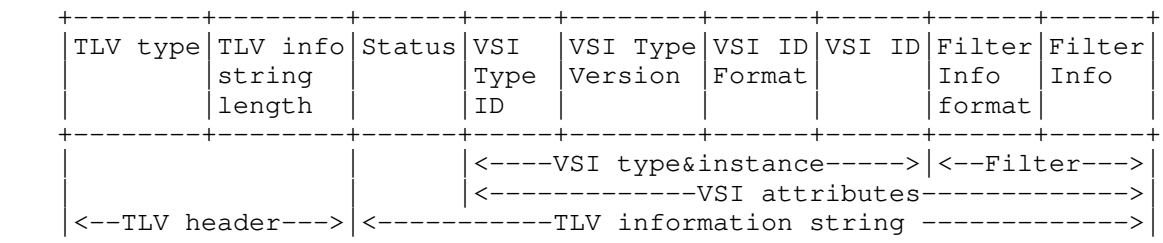


Figure A.1: VDP association TLV

There are basically four TLV types.

1. Pre-associate: Pre-associate is used to pre-associate a VSI instance with a bridge port. The bridge validates the request and returns a failure Status in case of errors. Successful pre-associate does not imply that the indicated VSI Type or provisioning will be applied to any traffic flowing through the VSI. The pre-associate enables faster response to an associate, by allowing the bridge to obtain the VSI Type prior to an association.

2. Pre-associate with resource reservation: Pre-associate with Resource Reservation involves the same steps as Pre-associate, but on success it also reserves resources in the bridge to prepare for a subsequent Associate request.

3. Associate: Associate creates and activates an association between a VSI instance and a bridge port. An bridge allocates any required bridge resources for the referenced VSI. The bridge activates the configuration for the VSI Type ID. This association is then applied to the traffic flow to/from the VSI instance.

4. De-associate: The de-associate is used to remove an association between a VSI instance and a bridge port. Pre-associated and associated VSIs can be de-associated. De-associate releases any resources that were reserved as a result of prior associate or pre-Associate operations for that VSI instance.

De-associate can be initiated by either side and the other types can only be initiated by the server side.

Some important flag values in VDP Status field:

1. M-bit (Bit 5): Indicates that the user of the VSI (e.g., the VM) is migrating (M-bit = 1) or provides no guidance on the migration of the user of the VSI (M-bit = 0). The M-bit is used as an indicator relative to the VSI that the user is migrating to.

2. S-bit (Bit 6): Indicates that the VSI user (e.g., the VM) is suspended (S-bit = 1) or provides no guidance as to whether the user of the VSI is suspended (S-bit = 0). A keep-alive Associate request with S-bit = 1 can be sent when the VSI user is suspended. The S-bit is used as an indicator relative to the VSI that the user is migrating from.

The filter information format currently defines 4 types. Each of the filter information is shown in details as follows.

1. VID Filter Info format

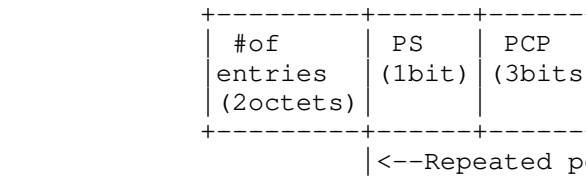


Figure A.2 VID Filter Info format

2. MAC/VID Filter Info format

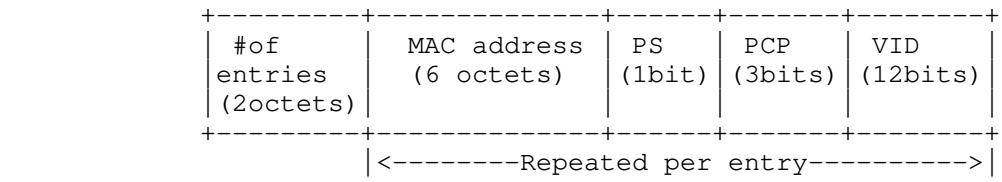


Figure A.3 MAC/VID filter format

3. GroupID/VID Filter Info format

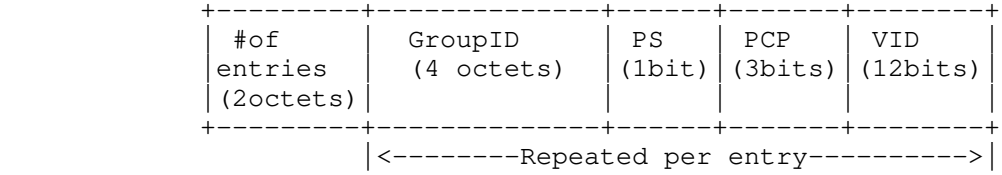


Figure A.4 GroupID/VID filter format

4. GroupID/MAC/VID Filter Info format

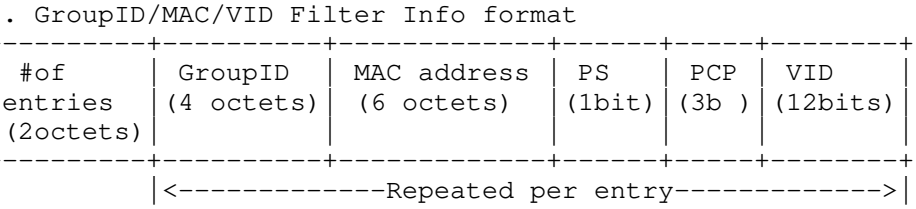


Figure A.5 GroupID/MAC/VID filter format

The null VID can be used in the VDP Request sent from the station to the external bridge. Use of the null VID indicates that the set of VID values associated with the VSI is expected to be supplied by the bridge. The set of VID values is returned to the station via the VDP Response. The returned VID value can be a locally significant value. When GroupID is used, it is equivalent to the VN ID in NVO3. GroupID will be provided by the station to the bridge. The bridge maps GroupID to a locally significant VLAN ID.

The VSI ID in VDP association TLV that identify a VM can be one of the following format: IPV4 address, IPV6 address, MAC address, UUID [RFC4122], or locally defined.

Authors' Addresses

Yizhou Li
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China

Phone: +86-25-56625409
EMail: liyizhou@huawei.com

Donald Eastlake
Huawei R&D USA
155 Beaver Street
Milford, MA 01757 USA

Phone: +1-508-333-2270
EMail: d3e3e3@gmail.com

Lawrence Kreeger
Arrcus, Inc

Email: lkreeger@gmail.com

Thomas Narten
IBM

Email: narten@us.ibm.com

David Black
Dell EMC
176 South Street,
Hopkinton, MA 01748 USA

Email: david.black@dell.com

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

X. Xu
Huawei
H. Shah
Ciena Corp
Y. Fan
China Telecom

July 1, 2016

NVo3 Control Plane Protocol Using IS-IS
draft-xu-nvo3-isis-cp-02

Abstract

This document describes the use of IS-IS as a light-weight control plane protocol for Network Virtualization over L3 (NVo3) overlay networks. This light-weight control plane protocol is intended for small and even medium sized enterprise campus networks where the NVo3 technology is to be used.

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 2, 2017.

Table of Contents

1. Introduction

[RFC7364] discusses the need of an overlay-based network virtualization approach, referred to as Network Virtualization over Layer3 (NVo3), for providing multi-tenancy capabilities in large data centers networks and outlines the needs for a control plane protocol to facilitate running these NVo3 overlay networks. [RFC7365] provides a framework for NVo3 overlay networks and meanwhile describes the needs for a control plane protocol to provide the following capabilities such as auto-provisioning/service discovery, address mapping advertisement and tunnel management.

Due to the succuss of the NVo3 technology in data center networks, more and more enterpises are considering the deployment of this technology in their campus networks so as to replace the old spanning tree protocols. Although BGP or Software Defined Network (SDN) controller could still be used as the control plane protocol in campus networks, both of them seem a bit heavyweight, especially for small and even medium sized campus networks.

IS-IS protocol [IS-IS] is a much proven and well-known routing protocol which has been widely deployed in campus networks for many years. Due to its extendibility, IS-IS protocol now is not only used for propagating IP reachability information in Layer3 networks (see [RFC1195]), but also used for propagating MAC reachability information in Layer2 networks or Layer2 overlay networks [RFC6165].

By using IS-IS as a lightweight control plane protocol for NVo3 overlay networks, the network provisiong is greatly simplified ((e.g., only a single protocol to be deployed)), which is much significant to campus networks.

This IS-IS based NVo3 control plane protocol could support any specific NVo3 data encapsulation formats such as VXLAN [RFC7348], VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe] , and NVGRE [RFC7637].

1.1. Requirements Language

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

2. Terminology

This memo makes use of the terms defined in [RFC7365] and [I-D.ietf-bier-architecture].

3. VN Membership Auto-discovery

By propagating the VN membership info among Network Virtualization Edges (NVEs), NVEs belonging to the same VN instance could discover one another automatically. The VN membership info is carried in a VN Membership Info sub-TLV (as shown in Section 3.1) of the following TLVs originated by that NVE:

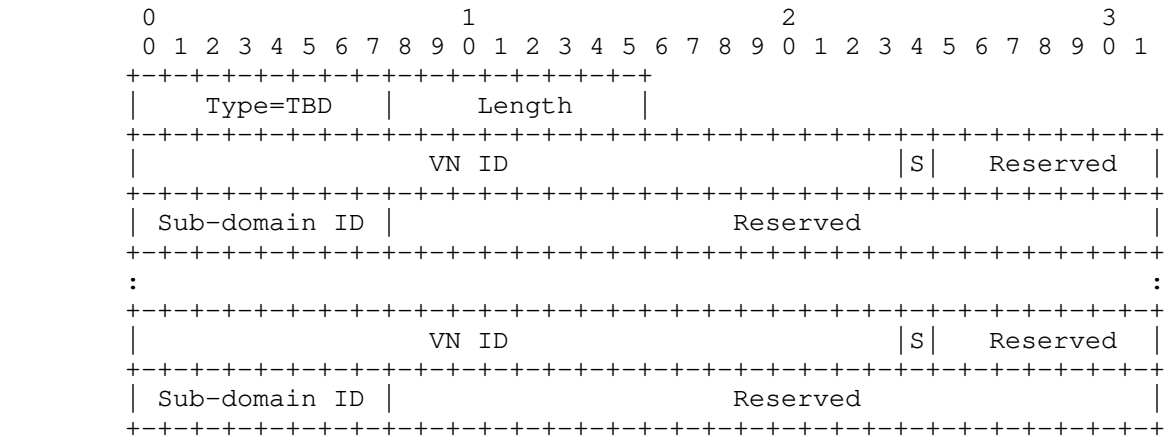
1. TLV-135 (IPv4) defined in [RFC5305].

2. TLV-236 (IPv6) defined in [RFC5308]

When the above TLV is propagated across level boundaries, the VN Membership Info sub-TLV contained in that TLV SHOULD be kept.

3.1. VN Membership Info Sub-TLV

The VN Membership Info sub-TLV has the following format:



Type: TBD;

Length: Variable;

VN ID: This field is filled with a 24-bit globally significant VN ID for a particular attached VN instance.

S-Flag: This field indicates the existence of the Sub-domain ID field. When the S-Flag is set, the Sub-domain ID field MUST be filled with a valid sub-domain ID. Otherwise, it SHOULD be set to zero.

Sub-domain ID: This field is filled with a 8-bit BIER sub-domain ID to which the VN has been associated [I-D.ietf-bier-architecture]. The field is only useful in the case where the Broadcast, Unknown-unicast and Multicast (BUM) packets within a VN are transported across the underlay by using the BIER forwarding mode.

4. Tunnel Encapsulation Capability Advertisement

To reach a consensus on what specific tunnel encapsulation format to be used between ingress and egress NVE pairs automatically, egress NVEs SHOULD advertise their own tunnel encapsulation capabilities by using the Encapsulation Capability sub-TLV as defined in [I-D.xu-isis-encapsulation-cap]

5. MAC Address Learning

MAC addresses of local CE hosts would still be learnt by NVEs as normal bridges. As for learning MAC addresses of remote CE hosts, there are two options: 1) data-plane based MAC learning and 2) control- plane based MAC learning. If unknown unicast flood suppression is strongly required even at the cost of consuming more forwarding table resources, the control-plane based MAC learning option could be considered. Otherwise, the data-plane based MAC learning option is RECOMMENDED.

5.1. Control-plane based MAC Learning for Remote CE Hosts

In the control-plane based MAC address learning mechanism, MAC reachability information of a given VN instance would be exchanged across NVEs of that VN instance via IS-IS as well. Upon learning MAC addresses of their local TES's somehow, NVEs SHOULD immediately advertise these MAC addresses to remote NVEs of the same VN instance by using the MAC-Reachability TLV as defined in [RFC6165]. One or more MAC-Reachability TLVs are carried in a LSP which in turn is encapsulated with an Ethernet header. The source MAC address is the originating NVE's MAC address whereas the destination MAC address is a to-be-defined multicast MAC address specifically identifying all NVEs. Such Ethernet frames containing IS-IS LSPs are forwarded towards remote NVEs as if they were customer multicast Ethernet frames. Egress NVEs receiving the above frames SHOULD intercept them and accordingly process them. The routable IP address of the NVE originating these MAC routes could be derived either from the "IP Interface Address" field contained in the corresponding LSPs (Note that the IP address here SHOULD be identical to the routable IP address associated with the VN membership Info) or from the tunnel source IP address of the NVo3 encapsulated packet containing such MAC routes. Since these LSPs are fully transparent to core routers of the underlying networks (i.e., non-NVE routers), there is no impact on the control plane of core routers at all.

6. IANA Considerations

The type code for VN Membership Info sub-TLV is required to be allocated by IANA.

7. Security Considerations

This document doesn't introduce additional security risk to IS-IS, nor does it provide any additional security feature for IS-IS.

8. Acknowledgements

TBD

9. References

9.1. Normative References

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

[RFC4971] Vasseur, JP., Ed., Shen, N., Ed., and R. Aggarwal, Ed., "Intermediate System to Intermediate System (IS-IS) Extensions for Advertising Router Information", RFC 4971, DOI 10.17487/RFC4971, July 2007, <http://www.rfc-editor.org/info/rfc4971>.

[RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <http://www.rfc-editor.org/info/rfc5305>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <http://www.rfc-editor.org/info/rfc5308>.

9.2. Informative References

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

[I-D.ietf-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.xu-isis-encapsulation-cap] Xu, X., Decraene, B., Raszuk, R., Chunduri, U., Contreras, L., and L. Jalil, "Advertising Tunnelling Capability in IS-IS", draft-xu-isis-encapsulation-cap-06 (work in progress), November 2015.

[IS-IS] "ISO/IEC 10589, "Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)", 2005.".

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <http://www.rfc-editor.org/info/rfc1195>.

[RFC6165] Banerjee, A. and D. Ward, "Extensions to IS-IS for Layer-2 Systems", RFC 6165, DOI 10.17487/RFC6165, April 2011, <http://www.rfc-editor.org/info/rfc6165>.

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

[RFC7364] Narten, T., Ed., Gray, E., Ed., Black, D., Fang, L., Kreeger, L., and M. Napierala, "Problem Statement: Overlays for Network Virtualization", RFC 7364, DOI 10.17487/RFC7364, October 2014, <http://www.rfc-editor.org/info/rfc7364>.

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

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

Authors' Addresses

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Himanshu Shah
Ciena Corp

Email: hshah@ciena.com

Yongbing Fan
China Telecom

Email: fanyb@gsta.com

Network Working Group L. Yong
Internet-Draft W. Hao
Intended status: Standard Track Huawei

Expires: January 2017 July 8, 2016

Tunnel Stitching for Network Virtualization Overlay
draft-yong-nvo3-tunnel-stitching-00

Abstract

This document describes a tunnel stitching method for delivering network virtualization overlay traffic that traverses multiple underlay networks.

Status of This Document

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

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

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

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

Copyright Notice

Table of Contents

1. Introduction

Network Virtualization Overlay Traffic often traverses multiple underlay networks from a source to a destination. When using a UDP based tunnel encapsulation to deliver overlay traffic, each underlay network constructs a tunnel, then multiple tunnels chain together to form an end-to-end path that the traffic. Figure 1 shows a use case that overlay traffic traverse from DC1 to WAN, to DC2 by using VXLAN encapsulation [RFC7348]. Overlay traffic from VTEP1 to VTEP3 will be carried by Tunnel1, Tunnel2, and Tunnel3 in sequence.

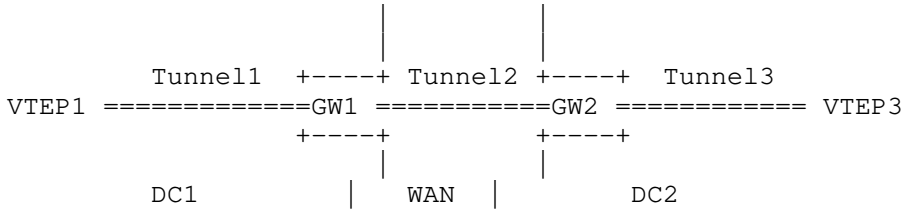


Figure 1 Tunnel Stitching Example

In this example, GW1 and GW2 is the egress for a tunnel and also the ingress for a next tunnel, where two tunnels provide a path for the traffic. Such piggybacked tunnels are referred to as tunnel stitching. A common practice for a node to perform tunnel stitching is for the node to decapsulate a received packet sent from the tunnel ingress, perform payload destination address lookup to determine the next tunnel endpoint, and then encapsulate the packet again with the next tunnel end point address and forward the packet upon.

This draft proposes a tunnel stitching method that avoids the payload lookup. The method can significantly reduce the complexity at a tunnel stitching node and shorten overlay traffic delay time from source to destination.

2. Terminology

2.1. Requirements Language

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

3. Tunnel Stitching Technique

Assumption of this technique is that tunnel encapsulation header is able to encode a next tunnel identifier.

When a tunnel ingress node encapsulates a packet, it adds encapsulation header and outer header and place next tunnel identifier in the encapsulation header and a tunnel egress address in the outer header.

When a tunnel stitching node, i.e. a tunnel egress receives the encapsulated packet, it gets the VN ID and next tunnel identifier from the encapsulation header; and performs a local lookup to get next tunnel egress address and next-next tunnel identifier; it replace the next-next tunnel identifier in the encapsulation header and next tunnel egress address in the outer header on the packet, and forward upon. This process continues until reaching a last tunnel node, where the packet is decapsulated and forwarded upon the destination address on the packet.

This technique avoids tunnel stitching node to perform payload destination lookup that can be an extreme large table due to the address space and unable to aggregate. Comparing the payload address space, the number of next tunnels for overlay traffic is much, much smaller; thus if a tunnel stitching node assigns a next tunnel identifier to identify each next tunnel results a small table lookup at the node.

This solution only requires the first tunnel ingress node to perform original packet destination lookup to get a tunnel egress address and a next tunnel identifier.

4. Next Tunnel Identifier Field

This draft proposes having an optional field for Next Tunnel Identifier in NVO3 encapsulation protocol header and allocate one bit in the header to indicate the field present. When the bit is clear, the optional field does not present or the value in the field is invalid; when the bit is set, the optional field is present and the field contains a next tunnel identifier.

The draft proposes that a optional field for Next Tunnel Identifier contains 24 bits. 24 bits give enough space for next tunnel identifier. How a tunnel stitching node to assign a value to identify a next tunnel is outside scope of this draft.

The proposal applies to an NV03 encapsulation protocol such as VXLAN [RFC7348], GUE[GUE], VXLAN-GPE [GPE], Geneve[GNV], etc.

Note MPLS technology naturally has the capability for an egress node to assign a label that identifies a next tunnel and inform the label to an ingress node.

5. SDN Controller

When using a NVA [RFC7365] to push overlay to underlay mappings to the first tunnel ingress node, the NVA sends <NVID/NVO address, a tunnel end point address, a next tunnel identifier> mappings; the node forms a table per a VN (see Figure 2 as an example); the NVA also sends <NVID/tunnel identifier, tunnel end point address, next tunnel identifier> mappings to each tunnel stitching node. A stitching node forms a lookup table based on tunnel identifier per VN base. (See Figure 3 as an example)

MAC Address	Egress Tunnel IP Address	Next Tunnel ID
00-1B-63-84-45-E6	192.10.10.30	7654321
00-BB-78-48-45-E6	192.10.20.200	2222222
00-AA-11-34-23-D4	192.10.20.200	6666666

Figure 2 VN 100 Table at first tunnel node

Tunnel ID	Egress Tunnel IP Address	Next Tunnel ID
7654321	192.168.20.300	1234567
2222222	192.168.20.200	1111111
6666666	192.168.100.2	6666666

Figure 3 VN 100 Table at the first tunnel stitching node

Note: It is possible that two different overlay address has the same tunnel egress address but different next tunnel identifier.

6. Distributed Control Plane

When a distributed control protocol such as MP-BGP is used to distribute overlay prefix to tunnel endpoint mappings [ROSEN]; it also carries next tunnel identifier. To achieve that, a new extended community is required for the next tunnel identifier.

When a tunnel stitching node receives BGP update; it allocates the tunnel identifier and add an entry in local lookup; to redistribute the route, it inserts its IP address in remote extended community and tunnel identifier in the next tunnel ID extended community.

Each tunnel stitching node can form a local lookup table as shown in Figure 2.

7. IANA Considerations

8. Security Considerations

9. References

9.1. Normative References

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

9.2. Informative Reference

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014,.

[RFC7365] Lasserre, M., et al, "Framework for Data Center (DC) Network Virtualization", RFC7365, October 2014.

[GUE] Herbert, T., Yong, L., Zia, O., "Generic UDP Encapsulation", draft-ietf-nvo3-gue-03, work in progress.

[GPE] Kreeger, L., Elzir, U., "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-02, work in progress.

[GNV] Gross, J., Ganga, I., "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-01, work in progress.

[ROSEN] Ronsen, E., Patel, K., Van de velde, G., "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-01, work in progress.

10. Authors' Addresses

Lucy Yong
Huawei Technologies

Email: lucy.yong@huawei.com

Weiguo Hao
Huawei Technologies

Email: Haoweiguo@huawei.com