NVO3 WG
Internet-Draft
Intended status: Standards Track
Expires: April 14, 2016

Fangwei. Hu
Ran. Chen
ZTE Corporation
Mallik. Mahalingam
Springpath
Qiang. Zu
Ericsson

October 12, 2015

YANG Data Model for VxLAN Protocol
draft-chen-nvo3-vxlan-yang-01.txt

Abstract

This document defines a YANG data model for VxLAN protocol.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 14, 2016.

Copyright Notice

Table of Contents

1. Introduction

YANG[RFC6020]is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document defines a YANG data model for the configuration of VxLAN protocol [RFC7348].

2. Terminology

3. YANG Data Model for VxLAN Configuration

3.1. VxLAN Multicast IP Address

The vxlan-multicast-ip is used to configure the IP multicast group, which the VxLAN VNI of the VTEP is mapping to.

3.2. VxLAN Mapping Type

The VxLAN-mapping-type is used to configure the VxLAN mapping type for the interface. The value is as following:

- vlan-1:1: the vxlan mapping type is VLAN, and each VxLAN is only mapping to one VLAN.
- vlan- n:1: the vxlan mapping type is VLAN, and each VxLAN could be mapped to several VLANs.
- L3-interface: the VxLAN mapping type is layer 3 interface.
- mac: the VxLAN mapping type is MAC address.

- o vlan-l2-interface: the VxLAN mapping type is vlan + Layer 2 interface.

3.3. VTEP Mode

The VTEP-mode is used to configure the mode for the VTEP interface. There are two modes: discard-inner-vlan and no-discard-inner-vlan. If the VTEP interface works in the discard-inner-vlan mode, the VxLAN is only mapping to one VLAN. The inner VLAN tag will be stripped when encapsulating the VxLAN frame. On the decapsulation side, if VTEP receives the VxLAN frame with inner VLAN tag, it will discard the frame in this work mode. If the VTEP receives the VxLAN frame without VLAN tag, it will fill in the VLAN tag based on the VxLAN and VLAN mapping.

If the VTEP interface works in the no-discard-inner-vlan mode, the VxLAN could be mapped to several VLANs. The inner VLAN tag will not stripped when encapsulating the VxLAN frame in the VxLAN encapsulation side. On the decapsulation side, if VTEP receives the VxLAN frame, it will strip the VxLAN header, and keep the VLAN frame.

4. Design Tree of Configuration Data Model

```
module: ietf-vxlan
   +--rw vxlan
      +--rw vxlan-instance* [vxlan-id]
      |  +--rw vxlan-id                uint32
      |  +--rw multicast-ip            inet:ipv4-address
      |  +--rw (vxlan-map-types)?
      |     +--:(vxlan-map-vlan)
      |     |  +--rw map-type-vlan?          enumeration
      |     |  +--rw map-vlan* [vlan-id]
      |     |     +--rw vlan-id    uint16
      |     +--:(vxlan-map-mac)
      |     |  +--rw map-type-mac?           empty
      |     |  +--rw mac                     yang:mac-address
      |     +--:(vxlan-map-l2interface)
      |     |  +--rw map-type-l2interface?   empty
      |     |  +--rw vlan-id                 uint16
      |     |  +--rw interface-name          if:interface-ref
      |     +--:(vxlan-map-l3interface)
      |        +--rw map-type-l3interface?   empty
      |        +--rw map-l3interface* [interface-name]
      |           +--rw interface-name    if:interface-ref
      +--rw interfaces
         +--rw interface* [name]
            +--rw name              if:interface-ref
            +--rw vtep-instances* [vtep-id]
               +--rw vtep-id             uint32
               +--rw vtep-name?          string
               +--rw source-interface?   if:interface-ref
               +--rw vtep-mode?          enumeration
               +--rw bind-vxlan-id* [vxlan-id]
                  +--rw vxlan-id    uint32
```

5. VxLAN YANG Configuration Data Model

```
module ietf-vxlan {
   namespace "urn:ietf:params:xml:ns:yang:ietf-vxlan";
   prefix "vxlan";

   import ietf-interfaces {
     prefix "if";
    }

   import ietf-inet-types {
      prefix "inet";
    }
```

```
        import ietf-yang-types {
           prefix yang;
         }

        organization
          "IETF NVO3(Network Virtualization Overlays) Working Group";

          contact
           "
             WG List:  <mailto:nvo3@ietf.org>

             WG Chair: Matthew Bocci
                       <mailto:matthew.bocci@alcatel-lucent.com>

             WG Chair: Benson Schliesser
                       <mailto:bensons@queuefull.net>

             Editor:   Ran Chen
                       <mailto:chen.ran@zte.com.cn>

             Editor:   Fangwei Hu
                       <mailto:hu.fangwei@zte.com.cn>

             Editor:   Mallik Mahalingam
                       <mailto:mallik_mahalingam@yahoo.com>

             Editor:   Zu Qiang
                       <mailto:Zu.Qiang@Ericsson.com>
            ";

        description "The YANG module defines a generic configuration
           model for VxLAN protocol";

        revision 2015-10-12 {
           description
             "01 revision.";
           reference
             "draft-chen-nvo3-vxlan-yang-01";
         }

        revision 2015-05-05 {
          description "Initial revision";
          reference
             "draft-chen-nvo3-vxlan-yang-00";
        }

        /* Configuration Data */
        container vxlan{
```

```
            list vxlan-instance {
               key vxlan-id ;
               leaf vxlan-id {
                 type uint32 {
                   range 1..16777215 {
                     description 'The value of VXLAN,it must
                     between 1 to 16777215.' ;
                   }
                 }
                 description 'VXLAN ID.' ;
               }

               leaf multicast-ip {
                   type inet:ipv4-address  ;
                   mandatory true ;
                   description 'VXLAN multicast IP address.' ;
                }

               choice vxlan-map-types {

                   case vxlan-map-vlan {

                     leaf map-type-vlan {

                       type enumeration {
                         enum map-type-vlan1to1 {
                           value 1 ;
                           description 'Map type VLAN1:1.' ;
                         }
                         enum map-type-vlan1ton {
                           value 2 ;
                           description 'Map type VLAN1:n.' ;
                         }
                       }
                       default map-type-vlan1to1 ;
                       description 'VXLAN map VLAN.';
                     }

                     list map-vlan {
                       key vlan-id ;
                       leaf vlan-id {

                         type uint16 {
                           range 1..4094 {
                             description 'The value of VLAN,it must
                             between 1 to 4094.';
                           }
                         }
```

```
        description 'VLAN ID.' ;
      }
      description 'VXLAN map VLAN.' ;
    }
    description 'VXLAN map VLAN.' ;
  }

  case vxlan-map-mac {
    leaf map-type-mac {
      type empty ;
      description 'VXLAN map MAC.' ;
    }

    leaf mac {
        type yang:mac-address ;
        mandatory true ;
        description 'MAC address.' ;
      }
    description 'VXLAN map MAC.' ;
  }

  case vxlan-map-l2interface {
    leaf map-type-l2interface {
      type empty ;
      description 'VXLAN map L2 interface.' ;
    }
    leaf vlan-id {
        type uint16 {
          range 1..4094 {
            description 'The value of VLAN,it must
            between 1 to 4094.';
          }
        }
        mandatory true ;
        description 'VLAN ID.' ;
     }

    leaf interface-name {
        type if:interface-ref;
        mandatory true ;
       description 'L2 interface name.' ;
    }
    description 'VXLAN map L2 interface.' ;
  }

  case vxlan-map-l3interface {
    leaf map-type-l3interface {
      type empty ;
```

```
                    description 'VXLAN map L3 interface.' ;
                   }

                   list map-l3interface {
                     key interface-name ;
                     leaf interface-name {
                       type if:interface-ref;
                       description 'L3 interface name.' ;
                     }
                     description 'VXLAN map L3 interface.' ;
                   }
                   description 'VXLAN map L3 interface.' ;
                 }
                 description 'VXLAN map type choice.' ;
               }
             description 'VXLAN instance.' ;
           }

         container interfaces {
           list interface{
               key "name";
               leaf name {
                  type if:interface-ref;
                  description 'vtep interface name';
                }
               list vtep-instances {
                  key vtep-id ;
                  leaf vtep-id {
                    type uint32;
                    description 'VTEP id.' ;
                   }

                  leaf vtep-name{
                    type string;
                    description 'VTEP instance name.' ;
                   }

                  leaf source-interface {
                    type if:interface-ref;
                    description 'Source interface name.' ;
                   }

                  leaf vtep-mode {
                    type enumeration {
                       enum discard-inner-vlan {
                         value 1 ;
                         description 'Discard inner-VLAN.' ;
                       }
```

```
                        enum no-discard-inner-vlan {
                          value 2 ;
                          description 'No discard inner-VLAN.' ;
                        }
                      }
                      default discard-inner-vlan ;
                      description 'VTEP mode.' ;
                    }

                   list bind-vxlan-id {
                      key vxlan-id;
                      leaf vxlan-id {
                        type uint32;
                        description 'VXLAN ID.' ;
                      }
                     description 'the vxlan id list for the vtep.';
                   }
                 description 'the vtep instance.';
               }
              description 'vtep interface.';
             }
           description 'the vtep interfaces configuration.';
         }
         description 'the vxlan configure model.';
        }
```

6. Design Tree of Operational State Model

```
          +--ro vxlan-state
             +--ro vxlan-instance
             |  +--ro vxlan-id?              uint32
             |  +--ro multicast-ip?          inet:ipv4-address
             |  +--ro (vxlan-map-types)?
             |     +--:(vxlan-map-vlan)
             |     |  +--ro map-type-vlan?          enumeration
             |     |  +--ro map-vlan* [vlan-id]
             |     |     +--ro vlan-id    uint16
             |     +--:(vxlan-map-mac)
             |     |  +--ro map-type-mac?           empty
             |     |  +--ro mac?                    yang:mac-address
             |     +--:(vxlan-map-l2interface)
             |     |  +--ro map-type-l2interface?   empty
             |     |  +--ro vlan-id?                uint16
             |     |  +--ro interface-name?         if:interface-ref
             |     +--:(vxlan-map-l3interface)
             |        +--ro map-type-l3interface?   empty
             |        +--ro map-l3interface* [interface-name]
             |           +--ro interface-name    if:interface-ref
             +--ro vtep-instance
                +--ro vtep-id?            uint32
                +--ro vtep-name?          string
                +--ro source-interface?   if:interface-ref
                +--ro vtep-mode?          enumeration
                +--ro bind-vxlan-id* [vxlan-id]
                   +--ro vxlan-id    uint32
```

7. VxLAN YANG Operational State Model

```
          /* Operational  data */
         container vxlan-state{
          container vxlan-instance {
              leaf vxlan-id {
                type  uint32;
                description "show the vxlan id.";
                }
              leaf multicast-ip {
                type inet:ipv4-address  ;
                description 'show VxLAN multicast IP address.' ;
               }

              choice vxlan-map-types {

                  case vxlan-map-vlan {

                    leaf map-type-vlan {
```

```
        type enumeration {
          enum map-type-vlan1to1 {
            value 1 ;
            description 'Map type VLAN1:1.' ;
          }
          enum map-type-vlan1ton {
            value 2 ;
            description 'Map type VLAN1:n.' ;
          }
        }
        default map-type-vlan1to1 ;
        description 'VXLAN map VLAN.';
      }

      list map-vlan {
        key vlan-id ;
        leaf vlan-id {
          type uint16;
          description 'VLAN ID.' ;
        }
        description 'VXLAN map VLAN.' ;
      }
      description 'VXLAN map VLAN.' ;
    }

    case vxlan-map-mac {
      leaf map-type-mac {
        type empty ;
        description 'VXLAN map MAC.' ;
      }

      leaf mac {
          type yang:mac-address ;
          description 'MAC address.' ;
        }
      description 'VXLAN map MAC.' ;
    }

    case vxlan-map-l2interface {
      leaf map-type-l2interface {
        type empty ;
        description 'VXLAN map L2 interface.' ;
      }
      leaf vlan-id {
          type uint16;
          description 'VLAN ID.' ;
       }
```

```
                    leaf interface-name {
                        type if:interface-ref;
                       description 'L2 interface name.' ;
                    }
                    description 'VXLAN map L2 interface.' ;
                  }

                  case vxlan-map-l3interface {
                    leaf map-type-l3interface {
                      type empty ;
                      description 'VXLAN map L3 interface.' ;
                    }

                    list map-l3interface {
                      key interface-name ;
                      leaf interface-name {
                        type if:interface-ref;
                        description 'L3 interface name.' ;
                      }
                      description 'VXLAN map L3 interface.' ;
                    }
                    description 'VXLAN map L3 interface.' ;
                  }
                  description 'VXLAN map type choice.' ;
                }
              description 'show the vxlan instance information.';
            }

            container vtep-instance {
              leaf vtep-id {
                type  uint32;
                description "show the vtep id.";
                }
              leaf vtep-name {
                type  string;
                description 'show the vtep name.';
                }
              leaf source-interface{
                type if:interface-ref;
                description 'show the source interface.';
                }

              leaf vtep-mode {
                  type enumeration {
                    enum discard-inner-vlan {
                      value 1 ;
                      description 'Discard inner-VLAN.' ;
                    }
```

```
                    enum no-discard-inner-vlan {
                      value 2 ;
                      description 'No discard inner-VLAN.' ;
                    }
                  }
                  default discard-inner-vlan ;
                 description 'show VTEP mode.' ;
                }

               list bind-vxlan-id {
                  key vxlan-id;
                  leaf vxlan-id {
                    type uint32;
                    description 'show the VXLAN ID.' ;
                  }
                 description 'show the vxlan id list for the vtep.';
                }
              description 'show the vtep information';
             }
            description 'show the vxlan operational state information';
           }
```

8. Security Considerations

9. Acknowledgements

10. IANA Considerations

This document requires no IANA Actions. Please remove this section before RFC publication.

11. Normative References

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

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

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[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>.

Authors' Addresses

Fangwei Hu
ZTE Corporation
No.889 Bibo Rd
Shanghai 201203
China

Phone: +86 21 68896273
Email: hu.fangwei@zte.com.cn

Ran Chen
ZTE Corporation
No.50 Software Avenue,Yuhuatai District
Nanjing, Jiangsu Province 210012
China

Phone: +86 025 88014636
Email: chen.ran@zte.com.cn

Mallik Mahalingam
Springpath
640 W. California Ave, Suite #110
Sunnyvale, CA 94086
USA

Email: mallik_mahalingam@yahoo.com

Zu Qiang
Ericsson
8400, boul. Decarie
Ville Mont-Royal, QC
Canada

Email: Zu.Qiang@Ericsson.com

NV03 working group   L. Dunbar
Internet Draft   D. Eastlake
Category: Standards Track   Huawei
Expires: November 2016   Tom Herbert
                                                        Google

October 19, 2015

NVA Address Mapping Distribution (NAMD) Protocol

draft-dunbar-nvo3-nva-mapping-distribution-02.txt

Status of this Memo

Copyright Notice

Abstract

This draft describes the mechanism for NVA to promptly and incrementally distribute the inner (TS) to outer (NVE) mapping and VN Context to relevant NVEs in a timely manner.

Table of Contents

1. Introduction

Section 4.5 of [nvo3-problem-statement] describes the back-end Network Virtualization Authority (NVA) that is responsible for distributing the mapping information for entire overlay system. [nvo3-nve-nva-cp-req] defines the requirement for the control plane between NVA and NVE.

This draft describes a mechanism for NVA to promptly and incrementally distribute the inner (TS) to outer (NVE) mapping and VN Context to relevant NVEs in a timely manner.

For ease of description, the term "NAMD" is used to represent the NVA Address Mapping Distribution protocol.

2. Terminology

The following terms are used interchangeably in this document:

- The terms "Subnet" and "VLAN" because it is common to map one subnet to one VLAN.
- The term "Directory" and "Network Virtualization Authority (NVA)"
- The term "NVE" and "Edge"

Bridge: IEEE Std 802.1Q-2011 compliant device [802.1Q]. In this draft, Bridge is used interchangeably with Layer 2 switch.

NAMD Timeout: The time interval that an NVE can assume NVA is not reachable if the NVE hasn’t received any updates from NVA during this time. NAMD Timeout is an unsigned byte that gives the amount of time in seconds during which the NVA will send at least three update PDUs. An empty update is used as a keep alive. It defaults to 30 seconds.

DA: Destination Address

DC: Data Center

EoR: End of Row switches in data center. Also known as aggregation switches.

End Station: Guest OS running on a physical server or on a virtual machine. An end station in this document has at least one IP address and at least one MAC address, which could be in DA or SA field of a data frame.

LISP: Locator/ID Separation Protocol

NVA: Network Virtualization Authority

NVE: Network Virtualization Edge

SA: Source Address

Station: A node, or a virtual node, with IP and/or MAC addresses, which could be in the DA or SA of a data frame.

ToR: Top of Rack Switch in data center. It is also known as access switches in some data centers.

TS: Tenant System

VM: Virtual Machines

VN: Virtual Network

VNID: Virtual Network Instance Identifier

3. Overall Requirement for NVE<->NVA Control Plane

Section 3.1 of [nvo3-cp-req] describes the basic requirement of inner address to outer address mapping for NVO3. A NVE needs to know the mapping of the Tenant System destination (inner) address to the (outer) address (IP) on the Underlying Network of the egress NVE.

Section 3.1 of [nvo3-cp-req] states that a protocol is needed to provide this inner to outer mapping and VN Context to each NVE that requires it and keep the mapping updated in a timely manner. Timely updates are important for maintaining connectivity between Tenant Systems.

4. Terminologies and Assumptions

NVAs can be centralized or distributed with each NVA holding the mapping information for a subset of VNs. By saying that an NVA holds mapping information for a VN, it means that the NVA has mapping information for all the TSs in the VN.

Centralized NVA means that the NVA holds mapping information for all the VNs in the administrative domain. There could be multiple instances of centralized NVA for redundancy purpose.

A NVA could be instantiated on a server/VM attached to a NVE, very much like a TS attached to a NVE, or could be integrated within an NVE. When a NVA is a standalone server/VM attached to a NVE, it has to be reachable via the attached NVE by other NVEs. A NVA can also be instantiated on a NVE that doesn't have any TSs attached. The NVE-NVA control plane for NVA being attached to NVE (like a VM) will require additional functions on NVEs than NVA being embedded in a NVE.

NVA should have at least the following information for each TS:

- Inner Address: TS (host) Address family (IPv4/IPv6, MAC, virtual network Identifier MPLS/VLAN, etc)

- Outer Address: The list of locally attached edges (NVEs); normally one TS is attached to one edge, TS could also be attached to 2 edges for redundancy (dual homing). One TS is rarely attached to more than 2 edges, though it could be possible;

- VN Context (VN ID and/or VN Name)

- Timer for NVEs to keep the entry when pushed down to or pulled from NVEs.

- Optionally the list of interested remote edges (NVEs). This information is for NVA to promptly update relevant edges (NVEs) when there is any change to this TS' attachment to edges (NVEs). However, this information doesn't have to be kept per TS. It can be kept per VN.

By saying that a NVE is participating in a VN or the VN is active on the NVE, it means that the VN is enabled on the NVE and there is at least one TS of the VN being attached to the NVE.

5. Overview of NVA Address Mapping Distribution (NAMD) Protocol

The inner-outer address mapping could change as TSs move from NVE to another. At any given period, probably only a small set of TSs would move, resulting in a small portion of changes on the inner-outer address mapping. Therefore, it is important to have a mechanism for NVA to send incremental updates to NVEs for the changes instead of entire database of the mapping entries. This document specifies the incremental update messages (TLVs) from NVAs to NVEs, to maintain data consistency between NVAs and NVEs.

The NAMD mechanism requires messages to distribute NVA content to all the NVEs, inform the incremental changes to the relevant NVEs, and maintain the database consistency between NVA and NVEs. This document specifies the structures (a.k.a. TLVs) of those messages, which are referred to as NAMD messages throughout this document. The NAMD TLVs can be included in BGP or IGP protocol messages. How they are integrated with the BGP or IPG will be further specified in the corresponding working groups.

A NVA can offer services in a Push, Pull model, or the combination of the two.

In Push model, the NVE, upon restart or initialization, sends requests for all the interested VNs as a multicast to all the NVAs. NVAs with the requested VNs use NAMD messages to distribute the mapping entries to the requested NVEs. Whenever, there are changes in the mapping entries, NVA uses NAMD messages to only send the changed portion of the entries.

In the Pull model, an NVA periodically sends VN scoped broadcast messages to all NVEs. An NVE, upon receiving a unknown unicast or ARP/ND with unknown target NVE, sends the pull request to the NVA that supports the VN that the targets belongs to.

6. TLV for NVE reachable addresses

The Reachable Interface Addresses (IA) TLV is used to advertise a set of addresses within a VN being attached to (or reachable by) a specific NVE, and optionally the NVE Virtual Access Point.

These addresses can be in different address families. For example, it can be used to declare that a particular interface with specified IPv4, IPv6, and 48-bit MAC addresses in some particular VN is reachable from a particular NVE.

This document suggests using the Interface Addresses APPsub-TLV defined by [IA] except using NVE address subTLV in the fourth field shown below:

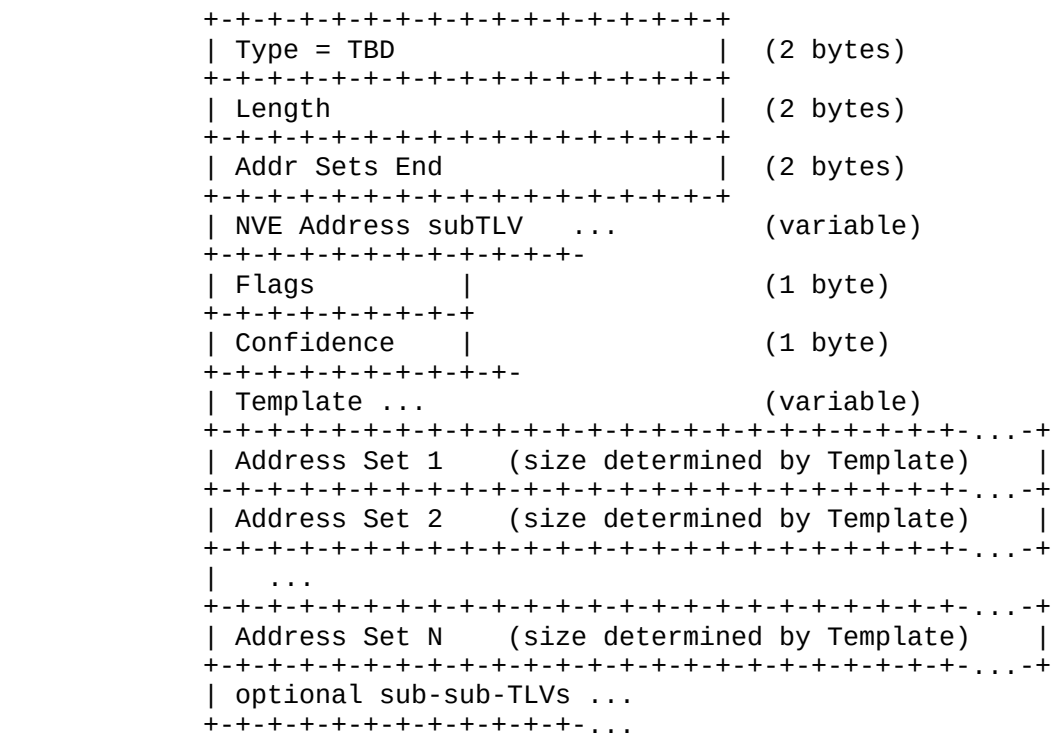


Figure 1. The Interface Addresses APPsub-TLV

Addr Sets End: The unsigned integer offset of the byte, within the IA APPsub-TLV [IA] value part, of the last byte of the last Address Set. This will be the byte just before the first sub-sub-TLV if any sub-sub-TLVs are present (see Section 3). If this is equal to Length, there are no sub-sub-TLVs. If this is greater than Length or points to before the end of the Template, the IA APPsub-TLV is corrupt and MUST be discarded. This field is always two bytes in size.

7. Push Mechanism

Under this mode, NVA pushes the inner-outer mapping for all the TSs of the VNs to relevant NVEs. This service is scoped by VN. A Push NVA also advertises whether or not it believes it has pushed complete mapping information for a VN. It might be pushing only a

subset of the mapping and/or reachability information for a VN. The Push Model uses the NAMD messages as its distribution mechanism.

With the Push model, if the destination of a data frame arriving at the Ingress NVE can't be found in its inner-outer mapping database that are pushed down from the NVA, the Ingress edge could be configured with one or more of the following policies:

- simply drop the data frame,
- flood the data frames to other NVEs that have the VN enabled, or
- start the "pull" process to get information from Pull NVA.
  When the NVE is waiting for reply from the Pull process, the NVE can either drop or queue the packet.

One drawback of the Push Mode is that it will push more mapping entries to an NVE than needed. Under the normal process of edge cache aging and unknown destination address flooding, rarely used entries would have been removed. It would be difficult for NVA to predict the communication patterns from/to TSs within one VN. Therefore, it is likely that the NVA will push down all the entries for all the VNs that are enabled on the NVE.

Another drawback with Push model: there really can't be any source-based policy. It's all or nothing.

7.1. Requesting Push Service

When a NVE is initialized or re-started, it needs to send request to the relevant NVAs to push down the mapping information for the active VNs on the NVE. NVE could use Virtual Network scoped message to announce all the Virtual Networks in which it is participating to NVAs who have the mapping information for the VNs. A new subTLV (Enabled-VN TLV) is specified here for NVE to indicate all its interested VNs in the NAMD message. The new subTLV can be included in an IGP protocol message or BGP message.

For 24-bits VN ID, there could be 16 million VNs. Multiple ways can be used to express the interested VNs:

- Starting VN & End VNs & bit map for the VNs in between.
- Starting VN & End VN (for the VNs that are contiguous)

- Individual VN listing (for a small number of VNs that are not contiguous)

Therefore 3 different types of subTLV are specified:

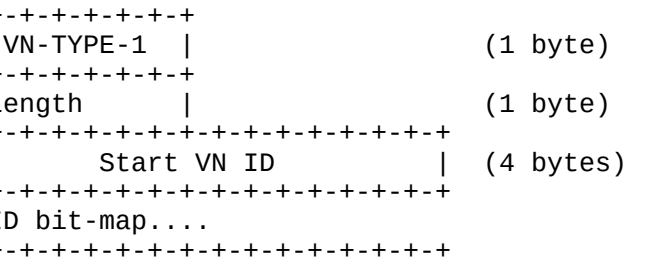


Figure 2. Enabled-VN TLV using bit map

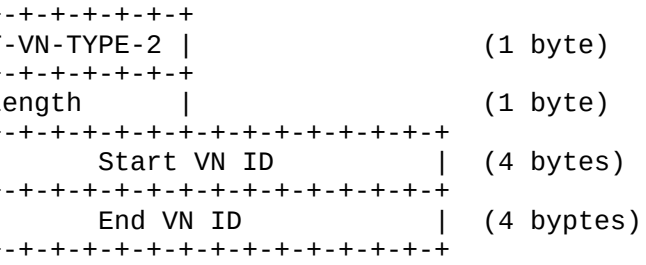


Figure 3. Enabled-VN TLV using Range

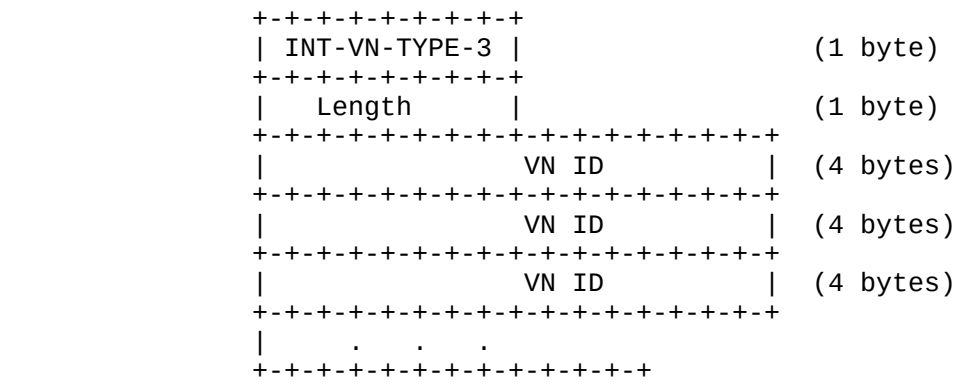


Figure 4. Enabled-VN TLV using list

- Type: indicating different ways to express the VNs that NVE is participating: INT-VN-TYPE-1 is for using bit map to express the interested VNs; INT-VN-TYPE-2 is for using range to express the interested VNs (if the interested VNs are contiguous); IT-VN-TYPE-3 is for using individual VN list to express the interested VNs.

- Length: Variable.

- RESV: 4 reserved bits that MUST be sent as zero and ignored on receipt.

- Start VN ID: The 24-bit VN-ID that is represented by the high-order bit of the first byte of the VN-ID bit-map.

VN-ID bit-map: The highest-order bit indicates the VN equal to the start VN ID, the next highest bit indicates the VN equal to start VN ID + 1, continuing to the end of the VN bit-map field.

If this sub-TLV occurs more than once in a Hello, the set of enabled VNs is the union of the sets of VNs indicated by each of the Enabled-VLAN sub-TLVs in the Hello.

When NVA is distributed, there could be multiple NVAs with each hosting mapping information for a subset of VNs.

Each NVA advertises its availability to push mapping information for a particular virtual network to all NVEs who participate in the VN. NVEs subscribe the relevant NVAs.

The subscription is VN scoped, so that a NVA doesn't need to push down the entire set of mapping entries. Each Push NVA also has a priority. For robustness, the one or two NVAs with the highest priority are considered as Active in pushing information for the VN to all NVEs who have subscribed for that VN.

7.2. Incremental Push Service

Whenever there is any change in TS' association to an NVE, which can be triggered by TS being added, removed, or de-commissioned, an incremental update has to be sent to the NVEs that are impacted by the change. Therefore, proper sequence numbers have to be maintained by NVA and edges NVEs. NAMD incremental message is used to update and maintain the database consistency between NVAs and NVEs. We assume that NVA gets notification from an authoritative source, such as VM management system when TS-NVE attachment changes occur.

A new TLV is needed for to carry NAMD timeout value and a flag for NVA to indicate it has completed all updates.

If the Push NVA is configured to believe it has complete mapping information for VN X then, after it has actually transmitted all of its messages for VN X it sets the Complete Push (CP) bit to one. It then maintains the CP bit as one as long as it is Active.

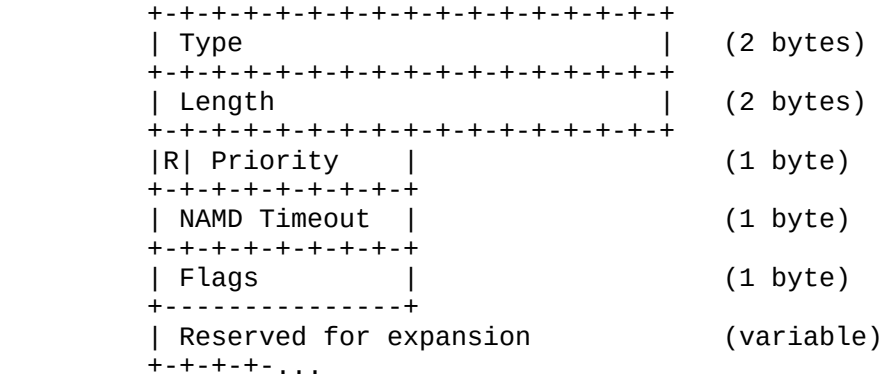
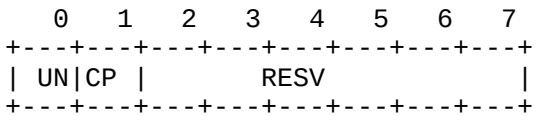


Figure 3. NAMD Complete TLV

Flags: A byte of flags defined as follows:



The UN flag indicates that the NVA will accept and properly process NVA- PDUs sent by unicast

The CP flag is to indicate that NVA has completed its update.

8. Pull Mechanism

Under this mode, an NVE pulls the mapping entries from the NVA when its cache doesn’t have the mapping entries.

The main advantage of Pull Mode is that the mapping is stored only where it needs to be stored and only when it is required. In addition, in the Pull Mode, NVEs can age out mapping entries if they haven’t been used for a certain period of time. Therefore, each NVE will only keep the entries that are frequently used, so its mapping table size will be smaller than a complete table pushed down from NVA.

The drawback of Pull Mode is that it might take some time for NVEs to pull the needed mapping from NVA. Before NVE gets the response from NVA, the NVE has to buffer the subsequent data frames with destination address to the same target. The buffer could overflow before the NVE gets the response from NVA. However, this scenario should not happen very often in data center environment because most likely the TSs are end systems which have to wait for (TCP) acknowledgement before sending subsequent data frames. Another option is forward, not flood, subsequent frames to a default location, if the NVE is configured with a default node that has the ability to forward data frames when the NVE doesn’t have the mapping information. This node can be the gateway, or a re-encapsulating NVE in NAMD context.

It worth noting that the practice of an edge waiting and dropping packets upon receiving an unknown DA is not new. Most deployed routers today drop packets while waiting for target addresses to be resolved. It is too expensive to queue subsequent packets while resolving target address. The routers send ARP/ND requests to the target upon receiving a packet with DA not in its ARP/ND cache and wait for an ARP or ND responses. This practice minimizes flooding when targets don’t exist in the subnet. When the target doesn’t exist in the subnet, routers generally re-send

an ARP/ND request a few more times before dropping the packets. The holding time by routers to wait for an ARP/ND response when the target doesn't exist in the subnet can be longer than the time taken by the Pull Mode to get mapping from NVA.

8.1. Pull Query Format

Here are some events that can trigger the pulling process:

- o An NVE receives a data frame from the attached TSs with a destination whose attached NVE is unknown, or
- o The NVE receives an ingress ARP/ND request for a target whose link address (MAC) or attached NVE is unknown.

Each Pull request can have queries for multiple inner-outer mapping entries. The message format is defined below:

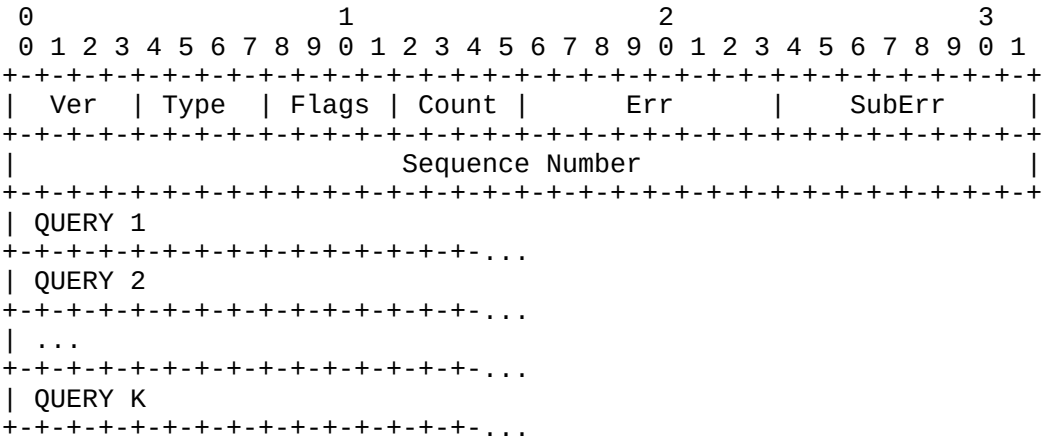


Figure 4. Pull Query TLV

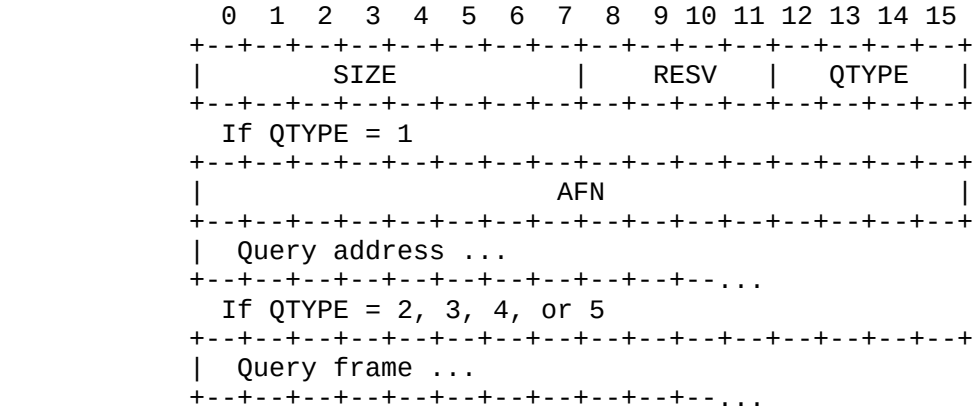
Type: 1 for Query. Queries received by an NVE that is not a Pull NVA result in an error response unless inhibited by rate limiting.

Flags, Err, and SubErr: MUST be sent as zero and ignored on receipt.

Count: Number of QUERY Records present. A Query message Count of zero is explicitly allowed, for the purpose of pinging a Pull NVA server to see if it is responding. On receipt of such

an empty Query message, a Response message that also has a Count of zero is sent unless inhibited by rate limiting.

QUERY: Each QUERY Record within a Pull Directory Query message is formatted as follows:



SIZE: Size of the QUERY record in bytes as an unsigned integer starting after the SIZE field and following byte. Thus the minimum legal value is 2. A value of SIZE less than 2 indicates a malformed QUERY record. The QUERY record with the illegal SIZE value and any subsequent QUERY records MUST be ignored and the entire Query message MAY be ignored.

RESV: A block of reserved bits. MUST be sent as zero and ignored on receipt.

QTYPE: There are several types of QUERY Records currently defined in two classes as follows: (1) a QUERY Record that provides an explicit address and asks for all addresses for the interface specified by the query address and (2) a QUERY Record that includes a frame. The fields of each are specified below. Values of QTYPE are as follows:

QTYPE	Description
0	reserved
1	address query
2	ARP query frame
3	ND query frame
4	RARP query frame
5	Unknown unicast MAC query frame
6-14	assignable by IETF Review

15 reserved

AFN: Address Family Number of the query address.

Address Query: The query is asking for any other addresses, and the address of NVE from which they are reachable, that correspond to the same interface, within the VN of the query. Typically that would be either (1) a MAC address with the querying NVE primarily interested in the NVE by which that MAC address is reachable, or (2) an IP address with the querying NVE interested in the corresponding MAC address and the NVE by which that MAC address is reachable. But it could be some other address type.

Query Frame: Where a QUERY Record is the result of an ARP, ND, RARP, or unknown unicast MAC destination address, the ingress NVE MAY send the frame to a Pull NVA if the frame is small enough that the resulting Query message not exceeding the MTU.

If no response is received to a Pull Directory Query message within a timeout configurable in milliseconds that defaults to 200, the Query message should be re-transmitted with the same Sequence Number up to a configurable number of times that defaults to three. If there are multiple QUERY Records in a Query message, responses can be received to various subsets of these QUERY Records before the timeout. In that case, the remaining unanswered QUERY Records should be re-sent in a new Query message with a new sequence number. If an NVE is not capable of handling partial responses to queries with multiple QUERY Records, it MUST NOT send a Request message with more than one QUERY Record in it.

8.2. Pull Response

There are several possibilities of the Pull Response:

1. Valid inner-outer address mapping, coupled with the valid timer indicating how long the entry can be cached by the NVE.
   The timer for cache should be short in an environment where VMs move frequently. The cache timer can also be configured.

2. The target being queried is not available. The response should include the policy if requester should forward data frame in legacy way, or drop the data frame.

3. The requestor is administratively prohibited from getting an informative response.

Pull NVA Response messages are sent as unicast to the requesting NVE. Responses are sent with the same VN. The specific data format is as follows:

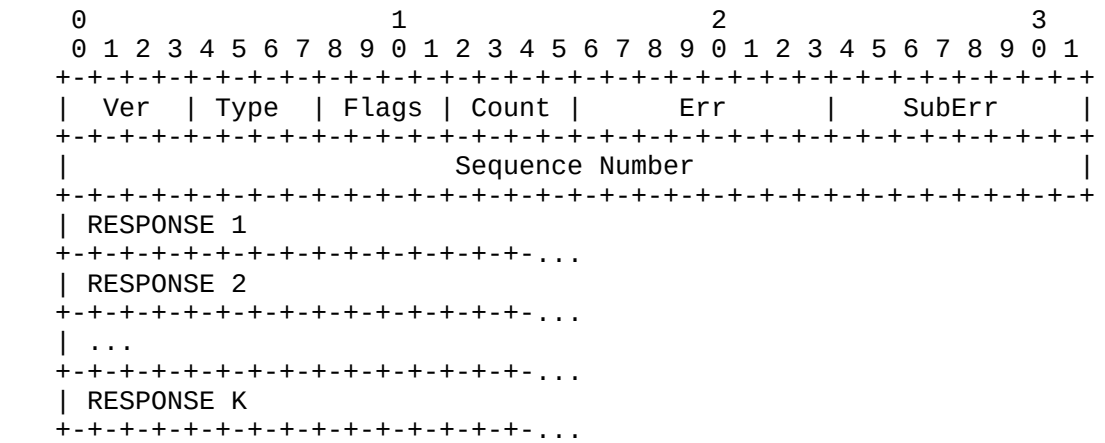


Figure 5. Pull Response TLV

Type: 2 = Response.

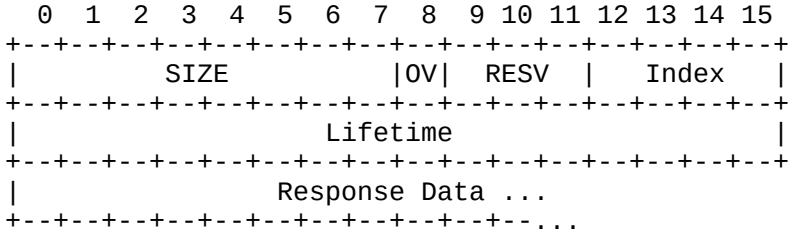
Flags: MUST be sent as zero and ignored on receipt.

Count: Count is the number of RESPONSE Records present in the Response message.

Sequence Number: There are many Pull Queries from NVEs; each Pull Query has a different sequence number. The Sequence Number in the Pull Response reflects the sequence number for the query.

Err, SubErr: A two part error code. Zero unless there was an error in the Query message, for which case see Section 3.5.

RESPONSE: Each RESPONSE record within a Pull NVA Response message is formatted as follows:



SIZE: Size of the RESPONSE Record in bytes starting after the SIZE field and following byte. Thus the minimum value of SIZE is 2. If SIZE is less than 2, that RESPONSE Record and all subsequent RESPONSE Records in the Response message MUST be ignored and the entire Response message MAY be ignored.

OV: The overflow flag. Indicates, as described below, that there was too much Response Data to include in one Response message.

RESV: Four reserved bits that MUST be sent as zero and ignored on receipt.

Index: The relative index of the QUERY Record in the Query message to which this RESPONSE Record corresponds. The index will always be one for Query messages containing a single QUERY Record. If the Index is larger than the Count that was in the corresponding Query, that RESPONSE Record MUST be ignored and subsequent RESPONSE Records or the entire Response message MAY be ignored.

Lifetime: The length of time for which the response should be considered valid in units of 200 milliseconds except that the values zero and 2**16-1 are special. If zero, the response can only be used for the particular query from which it resulted and MUST NOT be cached. If 2**16-1, the response MAY be kept indefinitely but not after the Pull NVA goes down or becomes unreachable. The maximum definite time that can be expressed is a little over 3.6 hours.

Response Data: There are various types of RESPONSE Records.

- If the Err field is non-zero, then the Response Data is a copy of the corresponding QUERY Record data, that is, either an AFN followed by an address or a query frame.

- If the Err field is zero and the corresponding QUERY Record was an address query, then the Response Data is the contents of an Interface Addresses APPsub-TLV [IA]. The maximum size of such contents is 253 bytes in the case when SIZE is 255.

- If the Err field is zero and the corresponding QUERY Record was a frame query, then the Response data consists of the response frame for ARP, ND, or RARP and a copy of the frame for unknown unicast destination MAC.

Multiple RESPONSE Records can appear in a Response message with the same index if the answer to a QUERY Record consists of multiple Interface Address APPsub-TLV contents. This would be necessary if, for example, a MAC address within a Data Label appears to be reachable by multiple NVEs. However, all RESPONSE Records to any particular QUERY Record MUST occur in the same Response message. If a Pull NVA holds more mappings for a queried address than will fit into one Response message, it selects which to include by some method outside the scope of this document and sets the overflow flag (OV) in all of the RESPONSE Records responding to that query address.

If no response is received from a Pull request within a configurable timeout, the request should be re-transmitted with the same Sequence Number up to a configurable number of times that defaults to three.

8.3. Cache Consistency

It is important that the cached information be kept consistent with the actual placement of VMs. Therefore, it is highly desirable to have a mechanism to prevent NVEs from using the staled mapping entries.

When there is any change in a Pull NVA, such as an entry being deleted or new entry added, and there may be unexpired stale information at some NVEs, the Pull NVA MUST send an unsolicited Update message to the relevant NVEs.

To achieve this goal, a Pull NVA server MUST maintain one of the following, in order of increasing specificity.

1. An overall record per VN of when the last returned query data will expire at a requestor and when the last query record specific negative response will expire.

2. For each unit of data (IA APPsub-TLV Address Set) held by the NVA and each address about which a negative response was sent, when the last expected response with that unit or negative response will expire at a requester.

Note: It is much more important to cache negative reply, because there are many invalid address queries. Study has shown that for each valid ND query, there are 100's of invalid address queries.

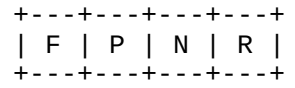
3. For each unit of data held by the NVA and each address about which a negative response was sent, a list of NVEs that were sent that unit as the response or sent a negative response to the address, with the expected time to expiration at each of them.

8.4. Update Message Format

An Update message is formatted as a Response message except that the Type field in the message header is a different value.

Update messages are initiated by a Pull NVA. The Sequence number space used is controlled by the originating Pull NVA and different from Sequence number space used in a Query and the corresponding Response that are controlled by the querying NVE.

The Flags field of the message header for an Update message is as follows:



F: The Flood bit. If zero, the response is to be unicast. If F=1, it is multicast to relevant NVEs.

P, N: Flags used to indicate positive or negative Update messages. P=1 indicates positive. N=1 indicates negative. Both may be 1 for a flooded all addresses Update.

R: Reserved. MUST be sent as zero and ignored on receipt

8.5. Acknowledge Message Format

An Acknowledge message is sent in response to an Update to confirm receipt or indicate an error unless response is inhibited by rate limiting. It is also formatted as a Response message.

If there are no errors in the processing of an Update message, the message is essentially echoed back with the Type changed to Acknowledge.

If there was an overall or header error in an Update message, it is echoed back as an Acknowledge message with the Err and SubErr fields set appropriately.

If there is a RESPONSE Record level error in an Update message, one or more Acknowledge messages may be returned.

8.6. Pull Request Errors

If errors occur at the query level, they MUST be reported in a response message separate from the results of any successful queries. If multiple queries in a request have different errors, they MUST be reported in separate response messages. If multiple queries in a request have the same error, this error response MAY be reported in one response message.

8.7. Redundant Pull NVAs

There could be multiple NVAs holding mapping information for a particular VN for reliability or scalability purposes. Pull NVAs advertise themselves by having the Pull Directory flag on in their Interested VNs sub-TLV [rfc6326bis].

A pull request can be sent to any of them that is reachable but it is RECOMMENDED that pull requests be sent to a NVA that is least cost from the requesting NVE.

9. Hybrid Mode

For some edge nodes that have great number of VNs enabled and combined number of TSs under all those VNs are large, managing the inner-outer address mapping for TSs under all those VNs can be a challenge. This is especially true for Data Center

gateway nodes, which need to communicate with a majority of VNs if not all.

For those NVE nodes, a hybrid mode should be considered. That is the Push Mode being used for some VNs, and the Pull Mode being used for other VNs. It is the network operator's decision by configuration as to which VNs' mapping entries are pushed down from NVA and which VNs' mapping entries are pulled.

In addition, NVA can inform the NVE to use legacy way to forward if it doesn't have the mapping information, or the NVE is administratively prohibited from forwarding data frame to the requested target.

10. Redundancy

For redundancy purpose, there should be multiple NVAs that hold mapping information for each VN. At any given time, only one or a small number of push NVAs is considered as active for a particular VN. All NVAs should announce its capability and priority to all the edges.

11. Inconsistency Processing

If an NVE notices that a Push NVA is no longer reachable, it MUST ignore any mapping entries from that NVA because it is no longer being updated and may be stale.

There may be transient conflicts between mapping information from different Push NVAs or conflicts between locally learned information and information received from a Push NVA. NVA may have a confidence level with address table information so, in case of such conflicts, information with a higher confidence value is preferred over information with a lower confidence. In case of equal confidence, Push NVA information is preferred to locally learned information and if information from Push NVAs conflicts, the information from the higher priority Push NVA is preferred.

12. Protocols to consider to carry NAMD messages

NAMD messages can be carried by IGP, BGP, or even OVSDB. NVO3 WG only focuses on specifying the NAMD message structure. How NAMD TLVs are integrated with BGP or IGP messages will be discussed in the corresponding WGs, e.g. BESS WG.

OVSDB (Open vSwitch Database Management protocol - RFC7047 by individual submission), is to bootstrap a vSwitch with the needed configuration (e.g. number of flow tables, the pipeline among those flow tables, path/link cost, Timer for Spanning Tree, Hello Timer, enabling Multicast snooping, etc). After OVSDB bootstrap a vSwitch, OpenFlow is used to dynamically pass down the flow entries.

Theoretically, some components of OVSDB can be potentially adopted (with update) to achieve the control plane between NVA and NVE. For example, changes to OVSDB are needed to address:

- How Edge nodes request for Push?
- How Edge nodes express the participated VNs?
- How NVA express the supported VNs ranges/list/?
- How Edge nodes feedback newly discovered attached TSs to NVA
- How Edge nodes exchange mapping among themselves.

13. Security Considerations

Incorrect information in NVA can result in a variety of security threats including the following:

Incorrect directory mappings can result in data being delivered to the wrong hosts/VMs, or set of hosts in the case of multi-destination packets, violating security policy.

Missing or incorrect data in NVA can result in denial of service due to sending data packets to black holes or discarding data on ingress due to incorrect information that their destinations are not reachable.

Push NVA data messages can be authenticated by including an Authentication TLV. See [RFC5304] and [RFC5310].

14. IANA Considerations

This section gives IANA allocation and registry considerations.

15. Acknowledgements

Special thanks to David Black, Dino Farinacci, Mingui Zhang, XiaoHu Xu for valuable suggestions and comments to this draft.

16. References

16.1. Normative References

[RFC4971] J. Vasseur et al, "Intermediate System to Intermediate System (IS-IS) Extensions for Advertising Router Information", July 2007.

[nvo3-nve-nva-cp-req] draft-ietf-nvo3-nve-nva-cp-req-00, "Network Virtualization NVE to NVA Control Protocol Requirements", Kreeger, et al. July 31, 3013.

[IA] - Eastlake, D., L. Yizhou, R. Perlman, "TRILL: Interface Addresses APPsub-TLV", draft-ietf-trill-ia-appsubtlv, work in progress.

16.2. Informative References

[802.1Q] IEEE Std 802.1Q-2011, "IEEE Standard for Local and metropolitan area networks - Virtual Bridged Local Area Networks", May 2011.

[802.1Qbg] IEEE Std 802.1Qbg-2012, "Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks-Edge Virtual Bridging", July 2012.

[RFC826] Plummer, D., "An Ethernet Address Resolution Protocol", RFC 826, November 1982.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

Authors' Addresses

Linda Dunbar
Huawei Technologies
5430 Legacy Drive, Suite #175
Plano, TX 75024, USA
Phone: (469) 277 5840
Email: linda.dunbar@huawei.com

Donald Eastlake
Huawei Technologies
155 Beaver Street
Milford, MA 01757 USA
Phone: 1-508-333-2270
Email: d3e3e3@gmail.com

Tom Herbert
Google
Email: therbert@google.com

NVO3
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

Z. Gu
Independent Reseacher
B. Khasnabish
ZTE (TX) Inc.
T. Ao
Fangwei. Hu
ZTE Corp

October 19, 2015

Virtual Network Transport Protocol (VNTP)
draft-gu-nvo3-vntp-03

Abstract

This document describes the overlay Virtual Network Transport Protocol (VNTP), which defines the interactions between NVE and NVA/ NVE and the relevant message to support virtual network implementation.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 21, 2016.

Table of Contents

1. Introduction

[RFC7364] and [RFC7365] describe the need and some characteristics of the interaction between NVE and NVA. [draft-ietf-nvo3-arch-03] has a more detailed architectural description about NVE-NVA protocol. And [draft-ietf-nvo3-nve-nva-cp-req-03] discusses the detail requirements of NVE-NVA protocol.

This draft defines a NVE-NVA protocol, Virtual Network Transport Protocol (VNTP). It belongs to the second model mentioned in [draft-ietf-nvo3-arch-03], e.g. NVE interacts with NVA directly. It defines the interactions between NVE and NVA/NVE and the relevant message formats to support virtual network implementation and fulfill the requirements described in the related documents mentioned above.

VNTP can be based on a broad transport mechanism such as TCP or UDP, or even IP. A new TCP/UDP port or protocol number allocation is needed if the transport mechanism is decided by NVO3 WG.

2. Conventions Used in This Document

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].

3. VNTP Overview

VNTP based on some basic assumptions and the main points include:

1), the first-hand mapping information provided by NVE, either configured by administrator or automatically created. Architecturally, VNTP also support other mapping resources such as downloaded from NVA.

2), NVE registers to NVA per VN and NVA may store two lists of NVE, the first one is all NVEs in a VN and another one is about all the VNs NVE resides.

3), when mapping change occurs in NVE, the NVE send update message to NVA to initiate the synchronization procedures and NVA then forward the update message to all other NVEs in the same VN. Optionally, NVA can store all the update information for latter use.

4), when a NVE register to a VN and some update messages received by NVA, the NVA may use the messages stored or request the related NVEs to send the update again to synchronize.

5), if NVA obtains the mapping information from other resources different from NVE, for example configured by administrator or from VM Orchestration, it sends the mapping/update information to all NVEs in the same VN.

The VNTP procedures can be simplified to implement by point-to-point communication between NVE and NVA. So the NVE-NVA interaction can be based on a broad transport mechanism such as TCP, UDP or even IP. A new TCP/UDP port or protocol number allocation is needed if the transport mechanism is decided.

4. VNTP Message Format

Figure 1 shows the VNTP message format. VNTP message format definition is based on some transport mechanism, such as TCP or UDP transport protocol, or even based on IP, and further using its data/ payload field.

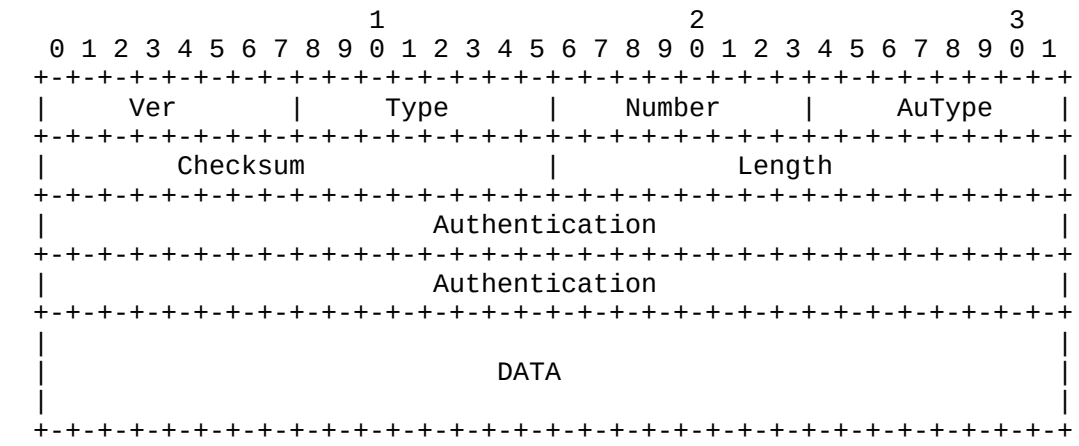


Figure 1 VNTP message format

Figure 1

4.1. VNTP Header format

The following are the Header fields definition.

Ver (8bit): for VNTP version.

Type (8bit): for VNTP Type Command or result/response definition.

Number (8bit): item/entry number of data field.

AuType and Authentication(length TBD): for authentication type and packet authentication.(Refer to RFC2328, especially section A and D; and further to RFC5709 for authentication update discussion.)

Checksum (16bit): checksum of the whole VNTP packet except authentication field.

Length (16bit): total packet octets including header.

Figure 2 shows detail Type definition.

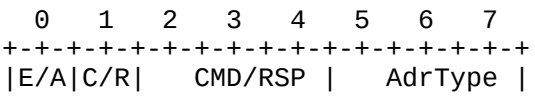


Figure 2 Type format

Figure 2

E/A (1bit): set to 1, NVE->NVA;

set to 0, NVA->NVE.

C/R (1bit): 1, CMD/RSP represents Command;

0, CMD/RSP represents Response/Result.

CMD/RSP: A command from NVE or NVA, or a response to the command

Detailed definition for the Command from NVE (C/R = 1, E/A = 1) .

CMD Description

000 NVE registration: The NVE registers to VNs.

001 NVE deregistration: The NVE de-registers from VNs.

010 NVE Update: NVE’s mapping information has been update.

011-111: Reserved for future use

Detailed definition for the Command from NVA (C/R = 1, E/A = 0)

CMD Description

000 Request for NVE Mapping information

001 Nullify NVE Mapping information/NVE deregistration

010 (NVE registered) Update NVE mapping information

011-111: Reserved for future use

Detailed definition for NVA/NVE Response/Result (C/R = 0, E/A = 0/1)

RSP Description

000 command executed successfully

001-011 Reserved for future definition

100 command execution failed

101 command execution partially successful (Optional reasons)

110-111 Reserved for future definition

AdrType (3bit): NVE address type. Detailed definition as following.

AdrType Description

000 IPv4

001 IPv6

010-111 Reserved

4.2. VNTP Data Format

VNTP Data field varies according to different command.

For Register/Deregister Command, the field contains the NVE's address and a VN-ID set. In the VN-ID set, each entry for one VN-ID. See Figure 3.

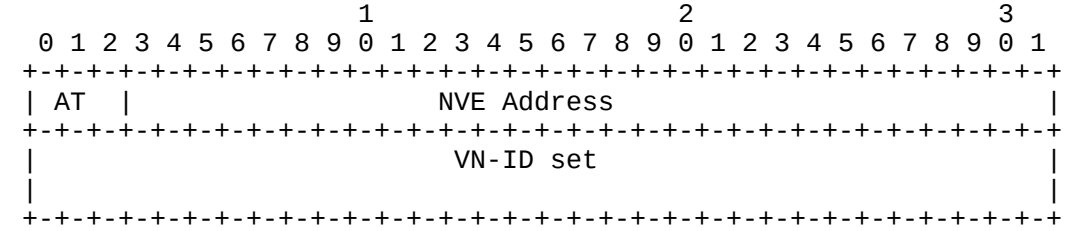


Figure 3 Data format for Register/Deregister command

Figure 3

The VN-ID set field would be Null, that means the NVE should be un-reachable anymore.

For Request mapping info command, the field contains the Inner address set, each entry for one inner address. See figure 4.

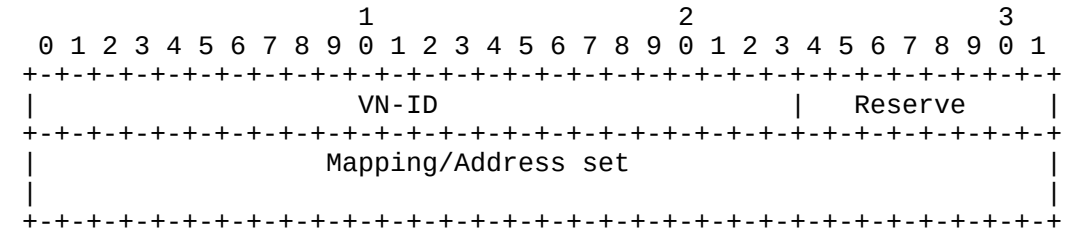


Figure 4 Data format for Request command

Figure 4

The mapping/Address set field would be Null, that means NVA is requesting all the Inner address in this VN.

For Nullify Command, the field contains the NVE address. The command is used by NVA to notify all the NVEs in the same VN that the NVE is not reachable, all the mapping information relate to the NVE should be removed.

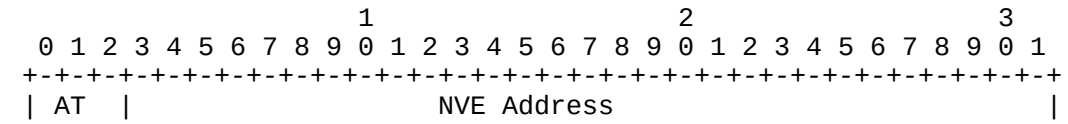


Figure 5 Data format for Nullify command

Figure 5

For Update Command, it also may include some entries, each entry has the detailed definition refers to Figure 3.

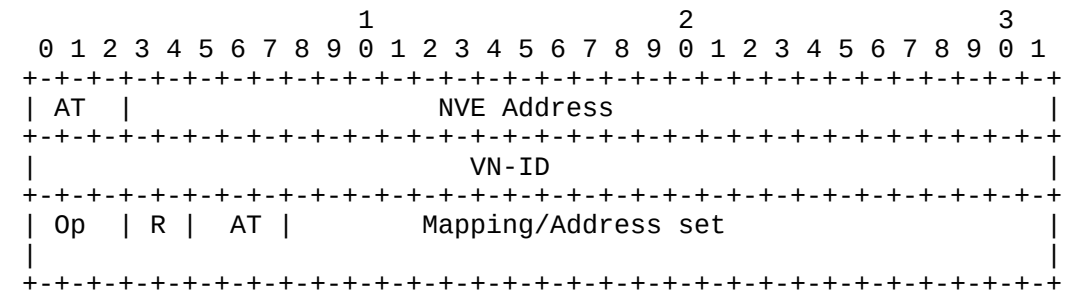


Figure 6 Data Format for Update command

Figure 6

The field means as following.

Op (3bit): for Operation Code.

Detailed definition for the Op field.

Op Description

000 Update add

001 Update delete

010 Update set to migration status

011 Update set to normal/non-migration status

100-111 Reserved for future use

AT (3bit): for Address Type.

Detailed definition for the AT field .

AT Description

000 IPv4

001 IPv6

010 MAC

011-111 Reserved

R (2bit): for Reserved.

NVE Address (length variable according AT): the outer address of mapping. If the update command is from NVE, the field is the local NVE address. If the update command is from NVA, the field is the remote NVE.

NV-ID: The Virtual Network ID that relate to the mapping information.

Mapping/Address (length variable according AT): each/inner Address needs updating. For the Update delete operation, if the field is Null, that means all the mapping info in this VN should be deleted.

5. The Operations of NVE

In the context of VNTP, the NVE works include:

1), If a VNI is created, the NVE will send Register command to NVA to register the VNI/NVE in the VN.

2), If a VNI is being deleted, the NVE will send update information to NVA to inform all the NVE related VN entry will be invalid. Or the NVA gets this information through the keep alive message, then nullify the all entries from this NVE_s VN.

3), If entries in the NVE have changed, for example, a new entry added or an existing entry deleted or become invalid, then the NVE will send update information to the NVA. Individual or batch update are supported.

4), And further, NVE also support tenant system migration.

5), The NVE accepts the updates from NVA and update the VRF table. The commands may be individual update or updates resulted from NVE failure.

6), Keep alive. Monitor the connection between NVE and NVA.

7), If the command not properly executed retransfer the command again for pre-setting times.

8), When NVA is unavailable or the NVA connection lost, optionally the NVE can connect other NVEs in the VN directly to keep the VN synchronized.

9), Security functions. TBD.

6. The operations of NVA

1), VNI creation

2), Form list of NVEs in the VN based on NVE Registration.

3), Accept updates from NVE and forward these updates to all other NVEs in the VN. Optionally, NVA store the update information for late use.

4), If NVE not register but update accepted, NVA may register it and forward the update to other NVEs.

5), if NVE registering after some updates then NVA will forward the stored updates to this NVE. Or NVA send request message to all other registered NVE for update if the previous updates not stored in NVA. And the NVA controls the updates only to this NVE other than all registered NVEs in the VN.

6), Keep alive. Monitor the connection between NVE and NVA.

7), if the command not properly executed NVA can retransfer the command again for pre-setting times.

8), When NVE in the VN is unavailable or the NVE connection lost, optionally the NVA can flood this NVE unreachable information to all other NVEs in the VN to keep the VN synchronized.

9), VNI delete. If there are not any VM or NVE in the VN, or the customer does not need the VN anymore then the NVA delete the VNI and release all the resources occupied by this VN.

10), Security functions. TBD.

7. Interaction with TS/Hypervisor-NVE protocol

Generally, VNTP can run independent of TS/Hypervisor-NVE protocol, but the interaction triggered by VRF changes because of the operation of TS/Hypervisor-NVE protocol . If the direct interaction is needed for further study.

8. Security Considerations

VNTP should support NVE and NVA mutual authentication and other security functions. The authentication has been covered by this draft, and the further security functions can be support through VNTP’s command reservations.

9. IANA/IEEE Considerations

VNTP needs a specific IP protocol value, or TCP/UDP port allocation if the transport mechanism is chosen.

10. References

10.1. Normative references

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

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, DOI 10.17487/RFC2234, November 1997, <http://www.rfc-editor.org/info/rfc2234>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <http://www.rfc-editor.org/info/rfc2328>.

[RFC5709] Bhatia, M., Manral, V., Fanto, M., White, R., Barnes, M., Li, T., and R. Atkinson, "OSPFv2 HMAC-SHA Cryptographic Authentication", RFC 5709, DOI 10.17487/RFC5709, October 2009, <http://www.rfc-editor.org/info/rfc5709>.

10.2. Informative References

[I-D.ietf-nvo3-arch] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Overlay Networks (NVO3)", draft-ietf-nvo3-arch-03 (work in progress), March 2015.

[I-D.ietf-nvo3-hpvr2nve-cp-req] Yizhou, L., Yong, L., Kreeger, L., Narten, T., and D. Black, "Split-NVE Control Plane Requirements", draft-ietf-nvo3-hpvr2nve-cp-req-03 (work in progress), August 2015.

[I-D.ietf-nvo3-nve-nva-cp-req] Kreeger, L., Dutt, D., Narten, T., and D. Black, "Network Virtualization NVE to NVA Control Protocol Requirements", draft-ietf-nvo3-nve-nva-cp-req-04 (work in progress), July 2015.

[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>.

Authors' Addresses

Zhongyu Gu
Independent Reseacher

Email: guzhongyu2015@sina.com

Bhumip Khasnabish
ZTE (TX) Inc.
55 Madison Ave, Suite 302
Morristown, New Jersey 07960
USA

Phone: +001-781-752-8003
Email: vumip1@gmail.com, bhumip.khasnabish@ztetx.com
URI: http://tinyurl.com/bhumip/

Ting Ao
ZTE Corp
No.889 Bibo Rd
Shanghai
China

Phone: +86 21 68897642
Email: ao.ting@zte.com.cn

Fangwei Hu
ZTE Corp
No.889 Bibo Rd
Shanghai 201203
China

Phone: +86 21 68896273
Email: hu.fangwei@zte.com.cn

INTERNET-DRAFT
Intended Status: Informational
Expires: April 11, 2015

Tom Herbert
Facebook

October 9, 2015

Identifier-locator addressing for network virtualization
draft-herbert-nvo3-ila-01

Abstract

This specification describes identifier-locator addressing (ILA) in IPv6 for network virtualization. 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

Table of Contents

1 Introduction

This specification describes the data path, address formats, and expected use cases of identifier-locator addressing in IPv6 ([RFC2460]). The Identifier-Locator Network Protocol (ILNP) ([RFC6740], [RFC6741]) defines a protocol and operations model for identifier-locator addressing in IPv6. Many concepts here are taken from ILNP, however there are some differences in the context of network virtualization-- for instance in ILA a method to encode a virtual network identifier and virtual address within an identifier is defined.

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 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 forward the packet to the addressed application.

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.

In network virtualization, an identifier may further be split into a virtual network identifier and virtual host address. With identifier-locator addressing network virtualization 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 "encapsulation" 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.

ILA exhibits properties of different networking techniques:

- Network Address Translation
- Source routing
- Encapsulation

2 Motivation

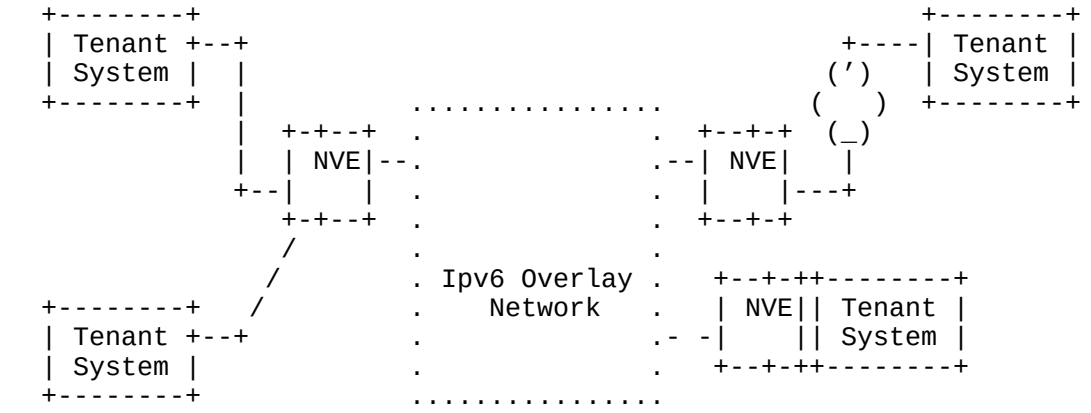
This section highlights the motivation for identifier-locator addressing.

2.1 Network virtualization

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

2.1.1 Architecture

The architecture for Network Virtualization over Layer 3 ([NVO3ARCH]) can be applied to network virtualization with ILA.



A Network Virtualization Edge (NVE) [RFC7365] is the entity that implements the overlay functionality using ILA. An NVE resides at the boundary between a Tenant System and the IPV6 overlay network as shown above. An NVE creates and maintains local state about each Virtual Network for which it is providing service on behalf of a Tenant System.

As in traditional network virtualization, NVEs are responsible for transit of tenant's packets through the overlay network. With ILA, the NVEs perform address translation on packets as opposed to encapsulation. The ingress NVE will translate the virtual address of a destination to an ILA address. At the egress NVE, the reverse translation is performed.

2.1.2 Multi-tenant virtualization

Identifier-locator addressing may be used as an alternative to nvo3 encapsulation protocols (such as GUE [GUE]). In multi-tenant virtualization, overlay networks are established for various tenants to create virtual networks and a tenant's nodes are assigned virtual addresses. Virtual networking identifiers are used to encode a virtual network identifier and a virtual address in an ILA address.

An advantage of identifier-locator addressing is that the overhead of encapsulation is reduced and use of virtualization can be transparent to the underlying network. A downside is that some features that use additional data in an encapsulation aren't available (security option in GUE for instance [GUESEC]).

Identifier-locator addressing may be appropriate in network virtualization where the users are trusted, for instance if virtual networks were assigned to different departments within an enterprise. Network virtualization in this context provides a means of isolation of traffic belonging to different departments of a single tenant. In this scenario, if the isolation breaks and packets uintentionally crosses between virtual networks, it would not be considered a security risk.

2.2 Data center virtualization

A primary motivation for identifier-locator addressing is data center virtualization. Virtualization within a data center permits malleability and flexibility in using data center resources. In particular, identifier-locator addressing virtualizes networking to allow flexible job scheduling and possibility of live task migration.

2.2.1 Address per task

Managing the port number space for services within a data center 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.

Algorithms for the creation of unique address per task are described in Appendix A.

2.2.2 Job scheduling

In the usual data center 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 data center in terms utilization, cost, latency, etc. Data center 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.

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 run time state of the job. The run time state is derived from the containers for the jobs.
3) Send the run time 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 run time 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.

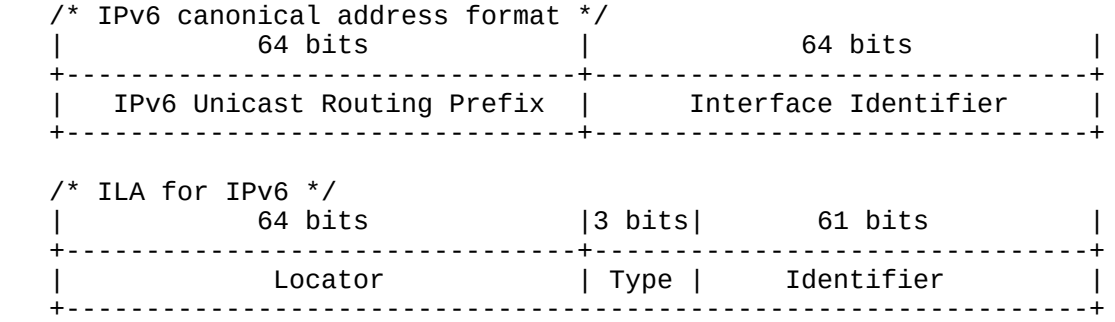
The networking state of interest to migrate are the addresses used by the task and open transport connections. The handling of these at task migration is discussed in Appendix B.

3 Address formats

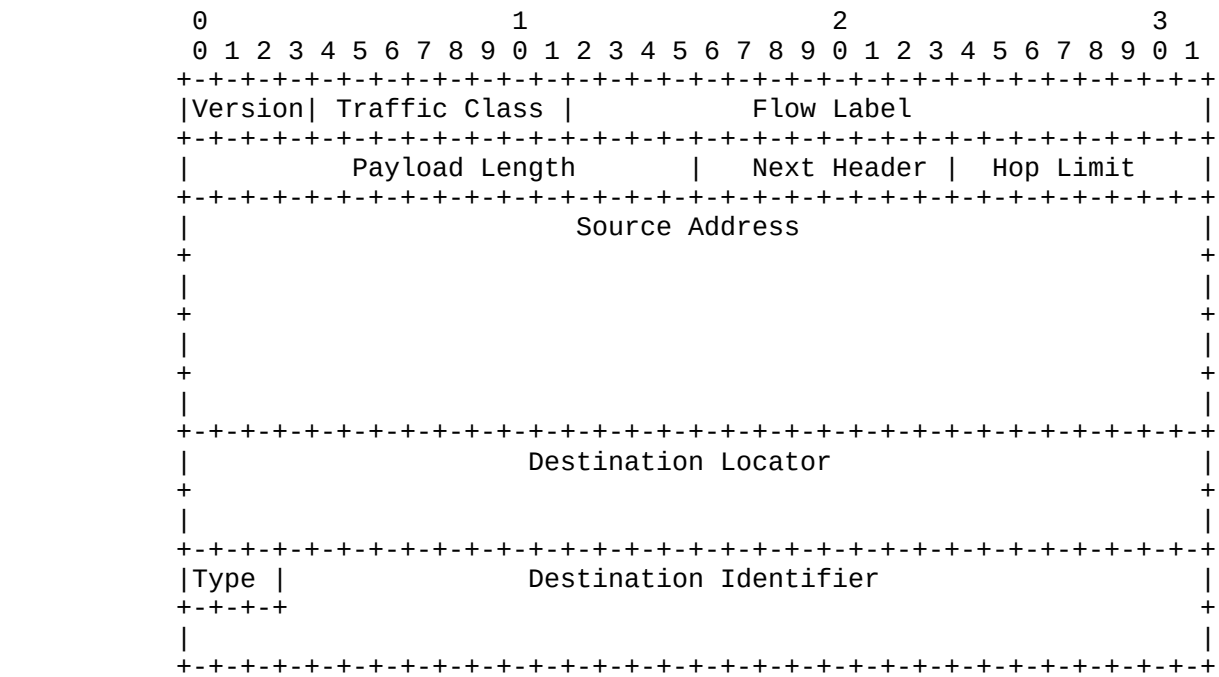
This section describes the address formats associated with identifier-locator addressing in network virtualization.

3.1 ILA format

As described in ILNP ([RFC6741]) an IPv6 address may be encoded to hold a locator and identifier where each occupies sixty-four bits. In ILA, the upper three bits of the identifier indicate an identifier type.

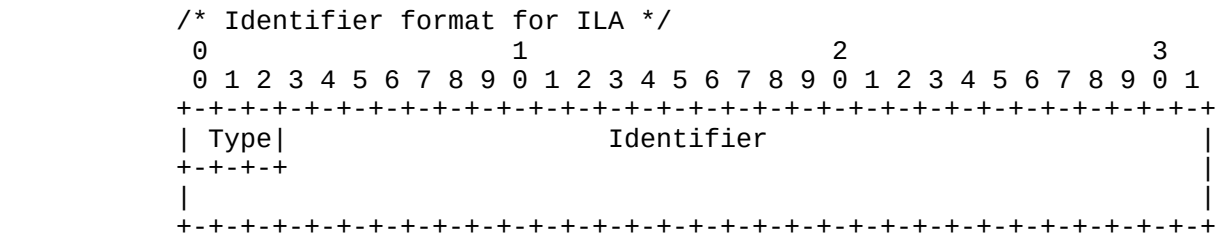


An IPv6 header with an ILA address would then have the format:



3.2 Identifier format

An ILA identifier includes a three bit type field and sixty-one bits for an identifier value.



- Type: Type of the identifier (see below).

- Identifier: Identifier value.

3.3 Identifier types

Defined identifier types are:

0: interface identifier

1: locally unique identifier

2: virtual networking identifier for IPv4 address

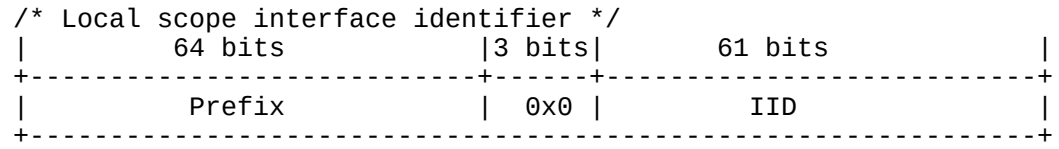
3: virtual networking identifier for IPv6 unicast address

4: virtual networking identifier for IPv6 multicast address

5-7: Reserved

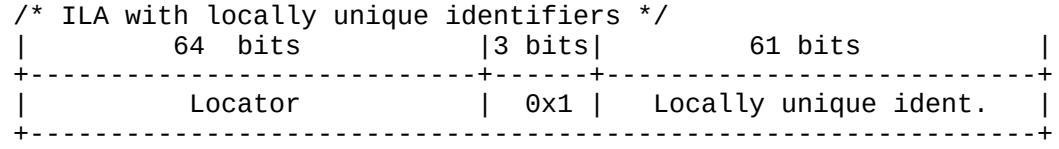
3.4 Interface identifiers

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 pupose of this type is to allow normal IPv6 addresses to be defined within the same networking prefix as ILA addresses. The type bits must be zero, and the format of the other bits (subnetting) would be site-defined. For example, the format of an interface identifer might be:



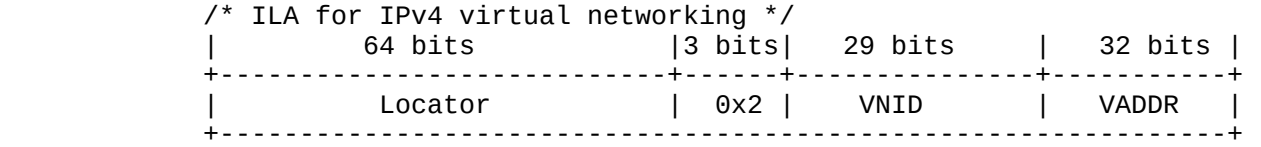
3.5 Locally unique identifiers

Locally unique identifiers (LUI) can be created for various addressable nodes within a network. These identifiers are in a flat sixty-one bit space and must be unique within a domain (unique within a site for instance). To simplify administration, hierarchical allocation of locally unique identifiers may be performed.



3.6 Virtual networking identifiers for IPv4

This type defines a format for encoding an IPv4 virtual address and virtual network identifier within an identifier.



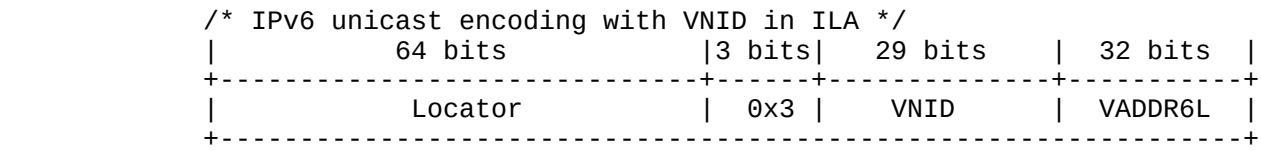
VNID is a virtual network identifier and VADDR is a virtual address within the virtual network indicated by the VNID. The VADDR can be an IPv4 unicast or multicast address, and may often be in a private address space (i.e. [RFC1918]) used in the virtual network.

3.7 Virtual networking identifiers for IPv6

A virtual network identifier and an IPv6 virtual host address (tenant visible address) can be encoded within an identifier. Encoding the virtual host address involves mapping the 128 bit address into a sixy-one bit identifier. Different encodings are used for unicast and multicast addresses.

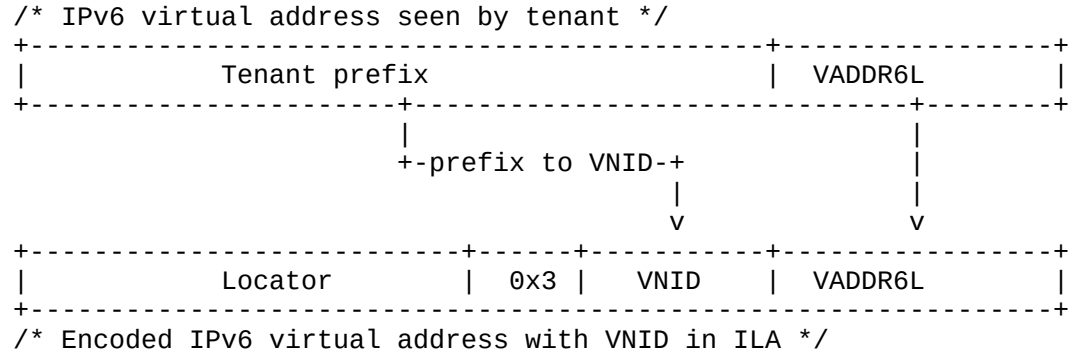
3.7.1 Virtual networking identifiers for IPv6 unicast

In this format, the 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.

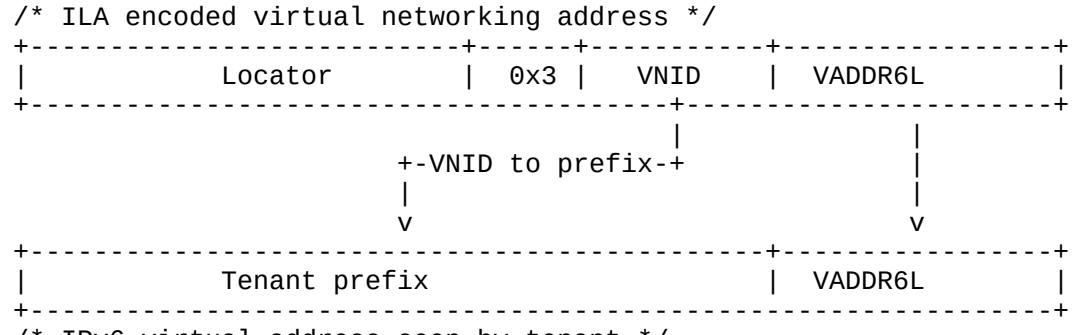


VADDR6L contains the low order 32 bits of the IPv6 virtual address. The upper 96 bits of the virtual address inferred from the VNID to prefix mapping.

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

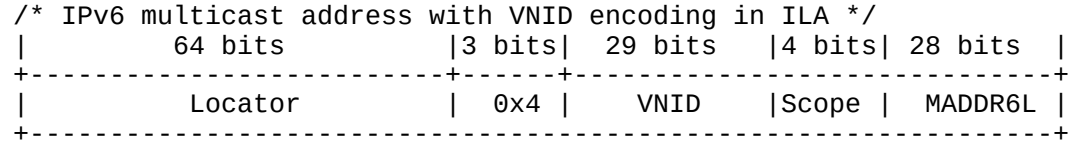


This encoding is reversible, given an ILA address, the virtual address visible to the tenant can be deduced:



3.7.2 Virtual networking identifiers for IPv6 multicast

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



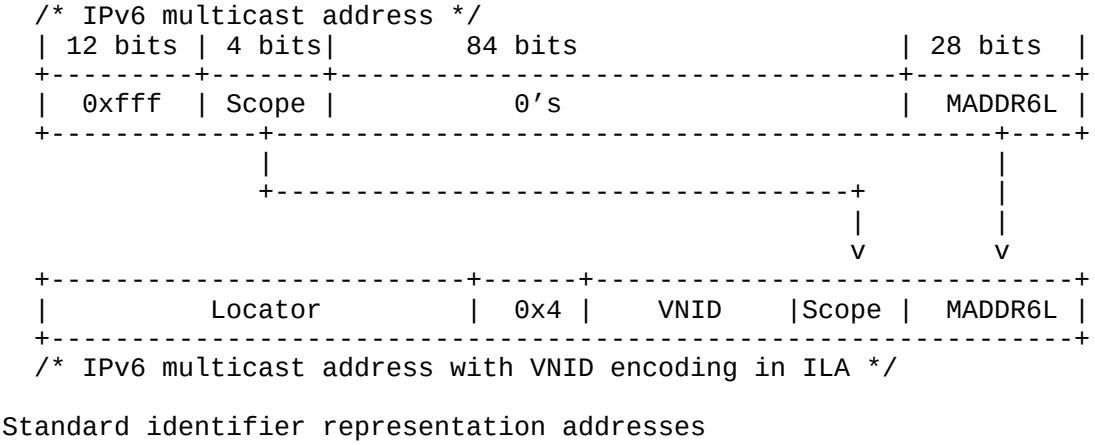
This format encodes an IPv6 multicast address in an identifier. The scope indicates multicast address scope as defined in [RFC7346]. MADDR6L is the low order 28 bits of the multicast address. The full multicast address is thus:

ff0<Scope>::0<MADDRL6 high 12 bits>:<MADDRL6 low 16 bits>

And so can encode multicast addresses of the form:

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

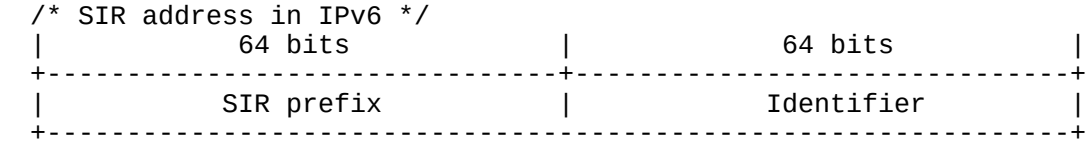
The figure below illustrates encoding a tenant IPv6 virtual multicast address into an ILA address.



3.8 Standard identifier representation addresses

An identifier serves as the external representation of a network node. 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 nominally refer to 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.



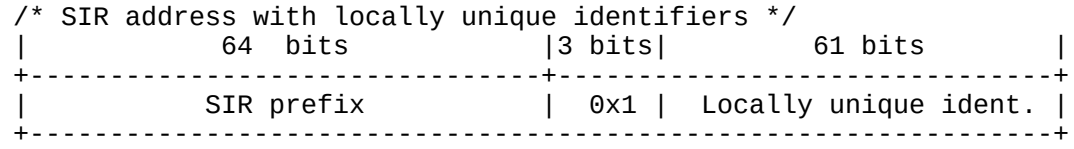
A SIR prefix may be site-local, or globally routable. A globally routable SIR prefix facilitates connectivity between hosts on the Internet and ILA endpoints. A gateway between a site's network and the Internet can translate between SIR prefix and locator for an

identifier. A network may have multiple SIR prefixes, and may also allow tenant specific SIR prefixes in network virtualization. Note that if there are multiple SIR prefixes they would share the same identifier space.

The 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.

3.8.1 SIR for locally unique identifiers

The SIR address for a locally unique identifier has format:



When using ILA with locally unique identifiers a flow tuple logically has the form:

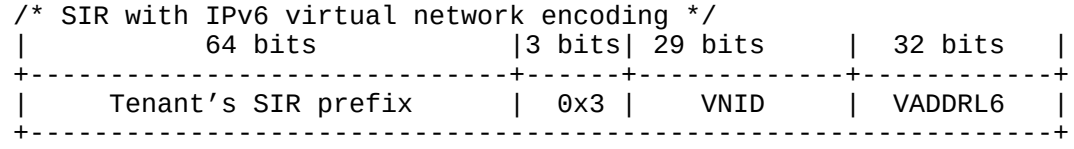
(source identifier, source port,
destination identifier, destination port)

Using standard identifier representation the flow is then represented with IPv6 addresses:

(source SIR address, source port,
destination SIR address, destination port)

3.8.2 SIR for virtual addresses

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



In a tenant system, a flow tuple would have the form:

(local VADDR, local port, remote VADDR, remote port)

After translating packets for the flow into ILA, the flow would be identified on-the-wire as:

((local VNID, local VADDR), local port,
(remote VNID, remote VADDR), remote port

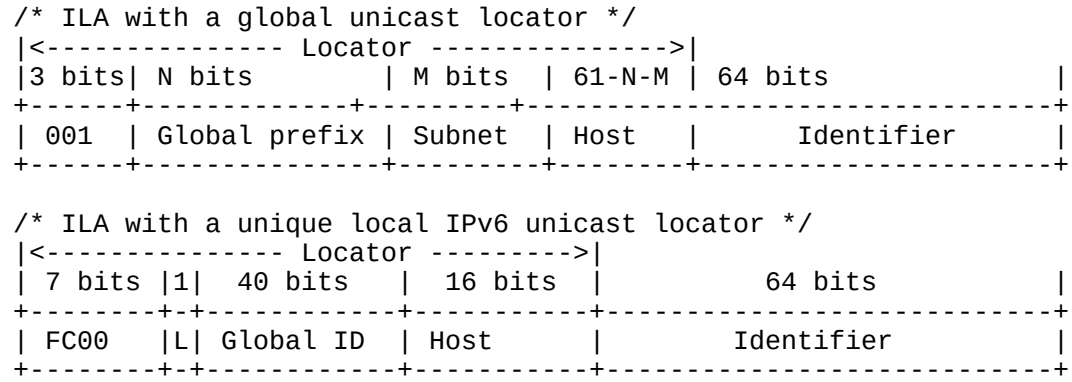
A tenant may communicate with a peer in the network which is not in its virtual network, for instance to reach a network service (see below). In this case the flow tuple at the peer may be:

(local SIR address, local port,
remote SIR address, remote port)

In this example, the remote SIR address is a SIR address for a virtual networking identifier, however from peer's connectivity perspective this is not distinguishable from a SIR address with a locally unique identifier or even a non-ILA address.

3.9 Locators

Locators are routable network address prefixes that address 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].



4 Operation

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

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 identifier is mapped to a locator. The mappings are not expected to change frequently, so it is likely that locator mappings can be cached in the flow contexts.

Identifier to locator mapping is nearly identical to the mechanism needed in virtual networking to map a virtual network and virtual host address to a physical host. These mechanisms should leverage a common solution.

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 done on a destination address as a form of source routing.

4.2.1 SIR to ILA address translation

When transmitting a packet, the locator for the destination ILA address might need to be set before the packet is sent on the wire. In the case that packet was created using a standard identifier representation, the SIR prefix is overridden with a locator. Since this operation is potentially done for every packet the process should be very efficient. Presumably, a host will maintain a cache of identifier locator mappings with a fast lookup function. If there is a connection state associated with the communication, the locator information may be cached with the connection state to obviate the need to perform a lookup per packet.

The typical steps to transmit a packet using ILA are:

1) Stack creates a packet with source address set to a SIR address for the local identity, and the destination address is set to the SIR address for the peer. The peer SIR address may have been discovered through DNS or other means.

2) Stack overwrites the SIR prefix in the destination address with a locator for the peer. This locator is discovered by a lookup in the locator to identifier mappings.

3) If a transport checksum includes a pseudo header that covered the original addresses, the checksum needs to be updated accordingly.

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

4.2.2 ILA to SIR address translation

Upon reception, an ILA address must be translated back to a SIR address before upper layer processing.

Receive processing may be:

1) Packet is received, the destination locator matches an interface address prefix on the host.

2) A lookup is performed on the destination identifier to find if it addresses a local identifier. If match is found, a SIR address can be created for the destination (overwrite locator with a SIR prefix).

3) Perform any checks as necessary. Validate locators, identifiers, and check that packet is not illegitimately crossing virtual networks (see below).

4) Forward packet to application processing. If necessary, the addresses in the packet can be converted to SIR addresses in place. Changing the addresses may also entail updating the checksum to reflect that (similar to a NAT translation).

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. 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).

4.3.2 IPv4/IPv6 protocol translation

An IPv4 tenant may send a packet that is converted to an IPv6 packet with ILA addresses having IPv4 virtual networking identifiers. 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].

4.4 Checksum handling

TCP and UDP checksums include a pseudo header checksum that covers the IP addresses in a packet. In the case of identifier-locator addressing the checksum must include the actual addresses set in the packet on the wire. So when creating a checksum for transmit, or verifying a checksum on receive, identifier-locator addressing must be taken into account.

4.4.1 Transmit checksum

If the source and destination locators are available when the transport checksum is being set, these can be used to calculate the pseudo checksum for the packet. This might be applicable in cases where locator information is cached within the context for a transport connection.

If the locators are set after the transport layer processing, the checksum can be updated following NAT procedures for address translation.

4.4.2 Receive checksum

Similar to the transmit case, if address translation occurs before transport layer processing the checksum must be adjusted per NAT. An implementation may verify a transport checksum before converting addresses to standard identifier representation to potentially obviate modifying the transport checksum to account for translation.

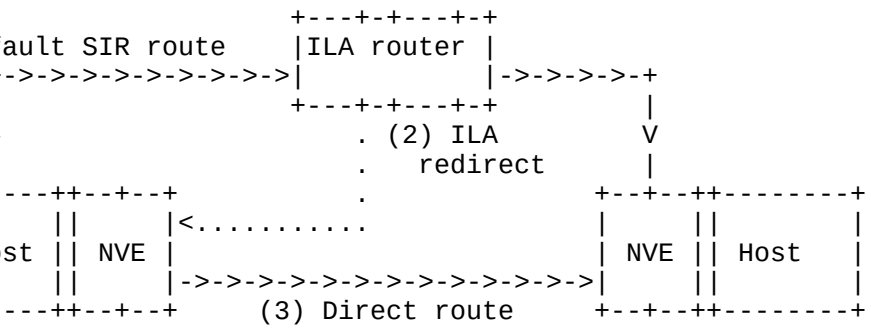
4.5 Address selection

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 selection of an identifier occurs at flow creation and must be invariant for the duration of the flow. Locator selection should be done once per flow, however may change (in the case of a migrating connection it will change). ILA address selection should follow guidelines in Default Address Selection for Internet Protocol Version 6 (IPv6) ([RFC6724]).

4.6 SIR address routing

ILA is intended to be sufficiently lightweight so that all the hosts in a data center 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 topology is illustrated below.



An ILA router is a node that implements both NVE and NVA (Network Virtulization Authority). When an ILA router receives a SIR addressed packet it will perform the ILA translation and send the ILA addressed packet to the destination NVE.

Host NVEs might not be initialized with ILA identifier to locator mappings. When a host sends a SIR addressed packet, the packet is routed to an ILA router based on the SIR prefix. The ILA router provides ILA translation for the SIR prefix (this is shown in (1) above). In addition to forwarding the ILA packet, the ILA router may send an "ILA redirect" back to the source (at (2) above). The redirect indicates the locator to use for the associated identifier. Subsequently, the NVE at the source host can perform ILA translation to send directly to the destination NVE thus eliminating triangular routing (as shown in (3)). The specification of the ILA redirect message is outside the scope of this document.

4.7 Duplicate identifier detection

As part of implementing the locator to identifier mapping, duplicate identifier detection may 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.

5. Communication scenarios

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

5.1 Terminology

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 transport endpoint IPv6 address with a locally unique identifier with 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 transport visible addresses to ILA addresses for transmission on the network and back to transport endpoint addresses at the receiver has notation:

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

5.2 Identifier objects

Identifier-locator addressing is broad enough in scope to address many different types of networking objects within a data center. For descriptive purposes we classify these objects as tasks or tenant systems.

A task is a unit of execution that runs in the data center networks. These do not run in a virtual machine, but typically run in the native host context perhaps within containers. Tasks are the execution mechanism for native jobs in the data center.

A network service is a task that provides some network wide service such as DNS, remote storage, remote logging, etc. A network service may be accessed by tenant systems as well as other tasks.

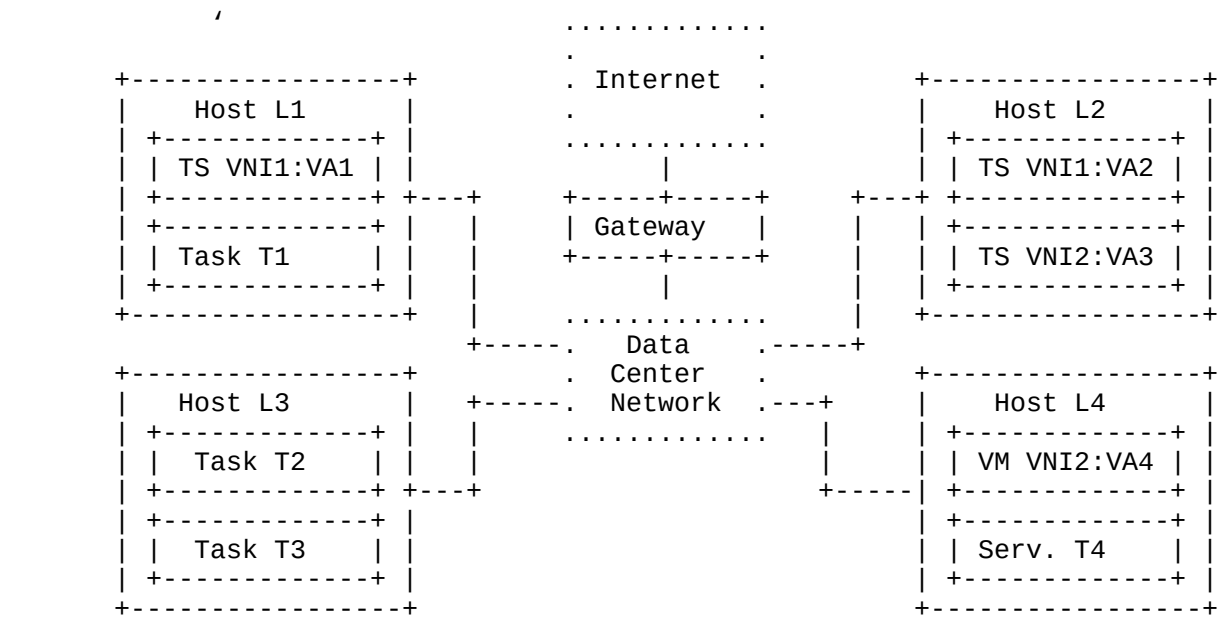
A tenant system, or TS, is a unit of execution which runs on behalf of a tenant in network virtualization. A TS may be implemented as a virtual machine or possibly using containers mechanisms. In either case, a virtual overlay network is implemented on behalf of a tenant, and isolation between tenants' virtual networks is paramount.

5.3 Reference network for scenarios

Several communication scenarios can be considered:

1) Task to task (service)
2) Task to Internet
3) Internet to task
4) TS to service
5) Task to TS
6) TS to Internet
7) Internet to TS
8) IPv4 TS to service
9) TS to TS in same virtual network using IPv6
10) TS to TS in same virtual network using IPv4
11) TS to TS in different virtual network using IPv6
12) TS to TS in different virtual network using IPv4
13) IPv4 TS to IPv6 TS in different virtual networks

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 tasks with identifiers T1, T2, and T3, as well as a networking service task with identifier T4. The identifiers for these tasks may be locally unique identifiers. 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.



5.4 Scenario 1: Task to task

The transport endpoints for task to task communication are the SIR addresses for the tasks. 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 task T1 is communicating with task T2, the ILA translation sequence would be:

```
SIR:T1,SIR:T2 ->                    // Transport endpoints on T1
SIR:T1,L3:T2 ->                     // ILA used on the wire
SIR:T1,SIR:T2                       // Received at T2
```

5.5 Scenario 2: Task to Internet

Communication from a task 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 task T1 is sending to an address Iaddr on the Internet, the packet addresses would be:

```
SIR:T1,Iaddr
```

5.6 Scenario 3: Internet to task

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

If a host on the Internet with address Iaddr sends a packet to task T3, the ILA translation sequence would be:

```
Iaddr,SIR:T3 ->                    // Transport endpoint at Iaddr
Iaddr,L1:T3 ->                     // On the wire in data center
Iaddr,SIR:T3                       // Received at T3
```

5.7 Scenario 4: TS to service task

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

If TS VA1 is communicating with service task T4, the ILA translation sequence would be:

```
VNET:VA1,SIR:T4->                  // Transport endpoints in TS
VNET:VA1,L3:T4->                   // On the wire
VNET:VA1,SIR:T4                    // Received at T4
```

Where VNET is the address prefix for the tenant.

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.

5.8 Scenario 5: Task to TS

A task can communicate with a TS through it’s externally visible address.

If task T2 is communicating with TS VA4, the ILA translation sequence would be:

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

5.9 Scenario 6: TS 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
```

5.10 Scenario 7: Internet to TS

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 data center. 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 data center
Iaddr,VNET:VA4                     // Received at TS
```

5.11 Scenario 8: IPv4 TS to service

A TS that is IPv4-only may communicate with a data center network service using protocol translation. The network service 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 service task T4, the ILA translation sequence would be:

```
VA2,ADDR4 ->                       // IPv4 endpoints at TS
SIR:(VNI1:VA2),L4:T4 ->            // On the wire in data center
SIR:(VNI1:VA2),SIR:T4              // 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 service task T4 to communicate with TS VA2, the ILA translation sequence would be:

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

5.12 TS to TS 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.

5.12.1 Scenario 9: TS to TS in 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
```

5.12.2 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
```

5.13 TS to TS in a 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.

5.13.1 Scenario 11: TS to TS in a 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.

5.13.2 Scenario 12: TS to TS in a 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 the 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.

5.13.3 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 VM VA4 using IPv4 to communicate with VM 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),SIR:(VNI1,VAX1)         // Received at VA1
```

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

6 Security Considerations

Security must be considered when using identifier-locator addressing. In particular, the risk of address spoofing or address corruption must be addressed. To classify this risk the set possible destinations for a packet are classified as trusted or untrusted. The set of possible destinations includes those that a packet may inadvertently be sent due to address or header corruption.

If the set of possible destinations are trusted then packet misdelivery is considered relatively innocuous. This might be the case in a data center if all nodes were tightly controlled under single management. Identifier-locator addressing can be used in this case without further additional security.

If the set of possible destinations contains untrusted hosts, then packet misdelivery could be a risk. This may be the case that virtual machines with untrusted third party applications or OSes are running in the network. A malicious user may be snooping for misdelivered packets, or may attempt to spoof addresses. Identifier-locator

addressing should be used with stronger security and isolation mechanisms such as IPsec or GUESEC.

7 IANA Considerations

There are no IANA considerations in this specification.

8 References

8.1 Normative References

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

[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.

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

8.2 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

9 Acknowledgments

The author would like to thank Mark Smith, Lucy Yong, Erik Kline, Saleem Bhatti, 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: Task identifier generation

Potentially every task in a data center could be migratable as long as each task is assigned a unique identifier. Since an ILA identifier is sixty-one bits it is conceivable that identifiers could be allocated using a shared counter or based on a timestamp.

A.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 61-bit length.

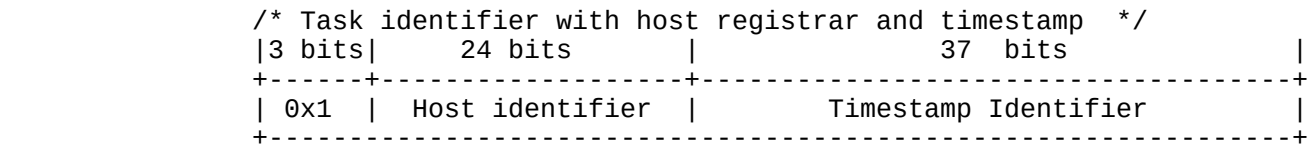
Identifiers	Probability of Collision
1000	2.1684*10^-13
10000	2.1684*10^-11
100000	2.1684*10^-09
1000000	2.1684*10^-07

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.

A.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 task identifiers in sixty-one 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 a task identifier could be created on the host that initially runs a task. The identifier might be composed of a twenty-four bit host identifier followed by a thirty-seven bit timestamp. Assuming that a host can start up to 100 tasks per second, this allows 43.5 years before wrap around.



Appendix B: Task migration considerations

B.1 Address migration

ILA facilitates address (specifically identifier) 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 an ILA redirect 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 redirect mechanism.

B.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
Facebook
1 Hacker Way
Menlo Park, CA
EMail: tom@herbertland.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 20, 2016

N. Matsuhira
Fujitsu Limited

October 18, 2015

Multiple Ethernet - IPv6 address mapping encapsulation - fixed prefix draft-matsuhira-me6e-fp-00

Abstract

This document specifies Multiple Ethernet - IPv6 address mapping encapsulation - fixed prefix (ME6E-FP) base specification. ME6E-FP makes expantion ethernet network over IPv6 backbone network with encapsuation technoogy. And also, E6ME-FP can stack multiple Ethernet networks. ME6E-FP work on own routing domain.

Requirements Language

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

Status of this Memo

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

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

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

This Internet-Draft will expire on April 20, 2016.

Copyright Notice

Table of Contents

1. Introduction

This document provides Multiple Ethernet - IPv6 address mapping encapsulation - fixed prefix (ME6E-FP) base specification.

ME6E-FP make many virtual ethernet network over IPv6 network with unicast base technology.

ME6E-FP can use on own routing domain, i.e. can advertise routes to the network.

2. Basic Network Configuration

Figure 1 shows network configuration with ME6E-FP. The network consists of three parts. IPv6 network, Nodes (Host or Router) , and ME6E-FP.

Backbone network is operated with Dual Stack or IPv6 only. Node may physical node or virtual node, and have Ethernet Interface.

ME6E-FP connects IPv6 network and nodes. ME6E-FP connect to node with Ethernet (Layer2), and ME6E-FP connect to IPv6 network with IPv6 (Layer3).

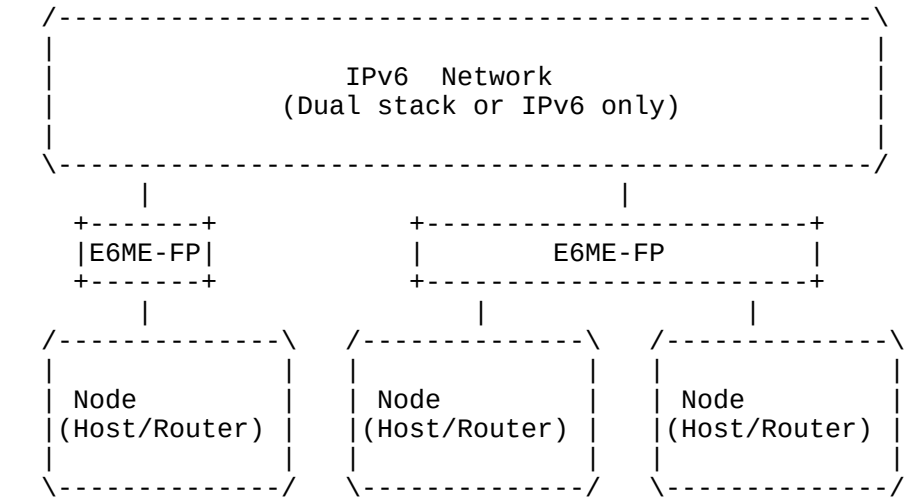


Figure 1

3. Basic Function of ME6E-FP

ME6E-FP has mainly two function. One is encapsulate from Ethernet frame to IPv6 packet, and decapsulate from IPv6 packet to Ethernet frame. Another is advertise route corresponding to Ethernet MAC address.

3.1. Ethernet over IPv6 Encapsulation

ME6E-FP encapsulates ethernet frame to IPv6 packet from node to IPv6 network, and decapsulates IPv6 packet to ethernet frame from IPv6 network to node. Figure 2 shows encapsulation and decapsulation of Ethernet frame and IPv6 packet

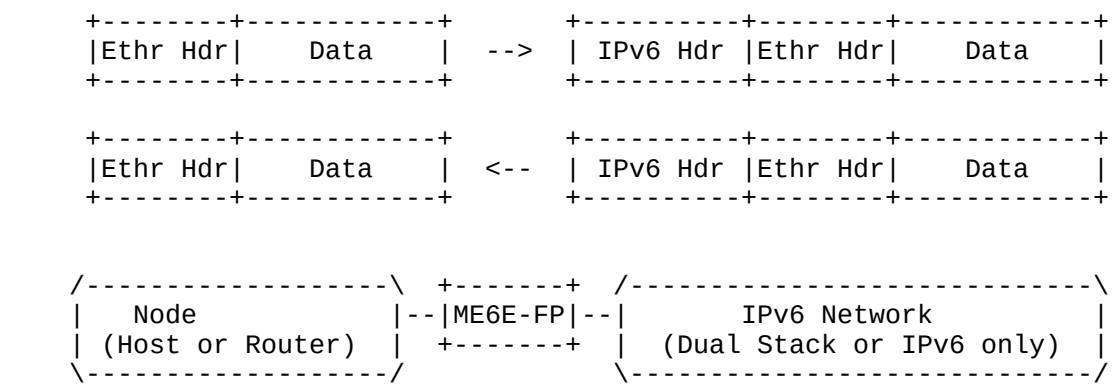


Figure 2

The value of next header field of IPv6 header is TBD. The value of EtherIP [RFC3378] may used, however new value for this protocol may assigned.

When encapsulated IPv6 Packet size exceed path MTU , ME6E-FP fragment Ethernet frame, and then send them.

3.2. Multiple Ethernet - IPv6 mapped (ME6) address architecture

ME6 address is a IPv6 address used in outer IPv6 header which encapsulate ethernet frame by ME6E-FP.

Figure 3 shows ME6 address architecture

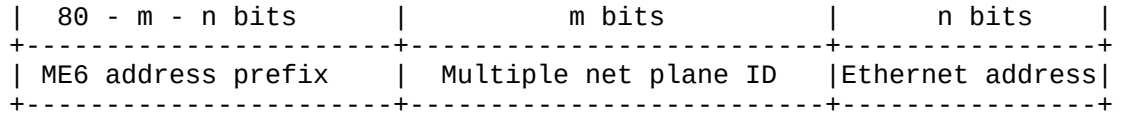


Figure 3

ME6 address consists of three parts as follows.

ME6 address prefix

ME6 address prefix . This value is preconfigured to all ME6E-FP in the IPv6 networks.

Multiple network plane ID

Multiple network plane ID is an identifier of Ethernet network over IPv6 backbone network. This value is preconfigured depend on the ME6E-FP belong which ethernet network plane. This value is just like VLAN-ID of IEEE802.1Q, tag VLAN.

Ethernet address

Ethernet MAC address in inner Ethernet frame. EUI-48 address or EUI-64 address.

ME6 address is resolved by copying ethernet MAC address in inner ethernet frame, and preconfigured values, ME6 prefix and multiple network plane ID.

3.3. Route Advertisement

ME6E-FP advertises ME6 address host route to the IPv6 network. The number of the route of ME6 addresses is the same as the number of MAC address table.

In the IPv6 network, usual dynamic routing protocol for IPv6 can be used such as RIPng [RFC2080], OSPFv3 [RFC2740] and IS-IS [RFC5308] .

4. ME6E-FP address format

ME6E-FP can be used closely in the IPv6 network, so ME6 address does not be advertised outside of the IPv6 network, and IPv6 packet which contains ME6 address does not be forwarded outside of the backbone network.

So, ME6 address format and ME6 address prefix can be decided each

IPv6 network. Some example are shown as follows. These address is based on EUI-48 MAC address. EUI-64 address is the future study.

4.1. IPv6 Global Unicast Address

This example is based on IPv6 Global Unicast Address Format [RFC3587].

Figure 4 shows IPv6 Global Unicast Address Format.

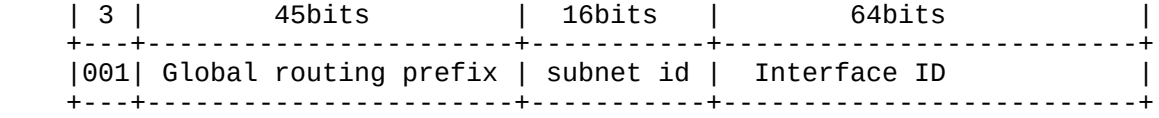


Figure 4

4.2. 16bits plane ID ME6 address

Figure 5 shows ME6 address format with 16bits multiple network plane ID using part of IPv6 Global Unicast Address.

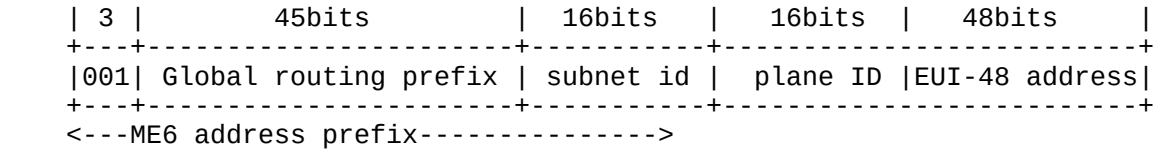


Figure 5

Where:

Global routing prefix

global routing prefix

subnet id

indication for ME6 prefix.

multiple network plane id

ethernet network plane ID.

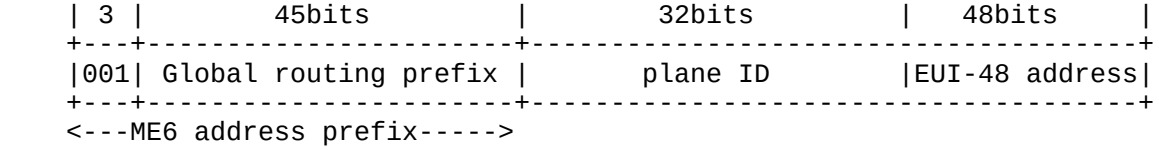
EUI-48 address

EUI-48 MAC address of inner Iethernet frame.

16bits plane ID can represent 65535 ethernet network plane.

4.3. 32bits plane ID ME6 address

Figure 6shows ME6 address format with 32bits plane ID using part of IPv6 Global Unicast Address.



Where:

Global routing prefix

global routing prefix

multiple network plane id

ethernet network plane ID.

EUI-48 address

EUI-48 MAC address of inner Iethernet frame

32bits plane ID can represent about 4.3 billion ethernet network plane.

4.4. mixture and renumbering of ME6 address

If ME6 address prefix does not overrap, ME6 address can co-existing. And also, ME6 address prefix may renumber, that mean, small start with 16bits plane ID ME6 address, then renumber to 32bits plane ID ME6 address.

ME6E-FP provide flexible operation for scalability of multiple network plane id.

5. Configuration of ME6E-FP

Configuration of ME6E-FP require just three information, ME6 address prefix, multiple Network plane ID, and prefix length of ME6E-FP route. These information could explain just only one line, "<ME6E-FP address prefix><multiple network plane ID>/ prefix length of ME6E-FP route".

6. Characteristic

ME6E-FP has following useful characteristics.

- o can operate unicast routing domain
- o TBD

7. IANA Considerations

This document makes no request of IANA if using EtherIP Header.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

ME6E-FP use automatic tunneling technologies. Security consideration related tunneling technologies are discussed in RFC2893 [RFC2893], RFC2267 [RFC2267], etc.

9. References

9.1. Normative References

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

[RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, August 2003.

9.2. References

[RFC2080] Malkin, G. and R. Minnear, "RIPng for IPv6", RFC 2080, January 1997.

[RFC2267] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", RFC 2267, January 1998.

[RFC2740] Coltun, R., Ferguson, D., and J. Moy, "OSPF for IPv6", RFC 2740, December 1999.

[RFC2893] Gilligan, R. and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers", RFC 2893, August 2000.

[RFC3378] Housley, R. and S. Hollenbeck, "EtherIP: Tunneling Ethernet Frames in IP Datagrams", RFC 3378, September 2002.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, October 2008.

Author’s Address

Naoki Matsuhira
Fujitsu Limited
17-25, Shinkamata 1-chome, Ota-ku
Tokyo, 144-8588
Japan

Phone: +81-3-3735-1111
Fax:
Email: matsuhira@jp.fujitsu.com

NVO3
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

Junying Pang
Jie Cao
Dapeng Liu
Dacheng Zhang
Alibaba
Yizhou Li
Hao Chen
Huawei Technologies
David Zhou
BoJian Wang
Deepak Kumar
Cisco Systems
Ruichang Gao
Yan Qiao
H3C

October 19 2015

Path Detection in VXLAN Overlay Network
draft-pang-nvo3-vxlan-path-detection-01

Abstract

In VXLAN overlay networks, Operation and Management(OAM)functions are important for fault management and performance monitoring. Path Detection(PD) is one critical OAM function which is applied to monitor and/or diagnose the potential paths between two VTEPs or between two Tenant System. In addition, it can assist to identify the locations of failures on data transmission paths.

This document specifies a method of PD method for VXLAN Overlay Networks by using a centralized controller. However,the method can be easily extended to support the overlay networks without a centrilized controller. It can also be generalized to other overlay technique such as NVGRE.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2016.

Table of Contents

1. Introduction

In VXLAN overlay networks, OAM functions such as fault management should be implemented to prevent the path failure problem [NVO3-OAM-REQ]. Path Detection is one of OAM function which can be used to

detect all available paths between two Tenant Systems or two VTEPs, and so it is widely used to assist the identify of the failure locations along a transmission path.

In this memo, a PD mechanism is specified for VXLAN overlay networks. A centralized Controller is provided as a centralized unit to: 1) construct Path Detection(PD) packets, 2) inject them into the network devices to record information such as device's Ingress/Egress interface number, and 3) collect the PD packets from network devices for further analysis. Therefore, Path Detection such as monitoring and diagnose can be realized more efficient.

Figure 1 shows the architecture of this mechanism:

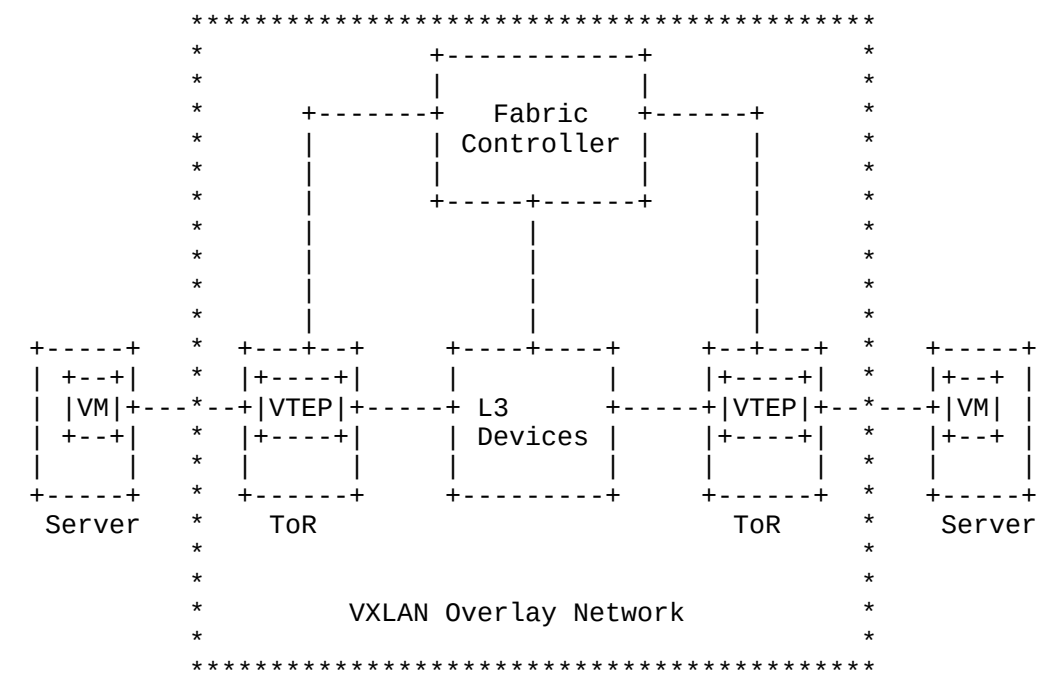


Figure 1: VXLAN Overlay Network with Fabric Controller

This method can be extended to support the overlay network without fabric controller. In this case, it could be regarded as a traditional OAM fault management solution described in draft-tissa-

nvo3-oam-fm-02 [I-D.tissa-nvo3-oam-fm].It can also be generalized to support other overlay technique such as NVGRE [RFC7637].

The following of this document is organized as follows: Section 3 describes the format of PD packets. Section 4 introduces the procedure of Path Detection between VTEPs. Section 5 describes the procedure of Path Detection between Tenant Systems.

1.1. Acronyms and Terminology

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

Because this document reuses most of the terms specified in RFC 7348 [RFC7348] RFC 7364 [RFC7364] and RFC 7365 [RFC7365], this section only defines the key terms used by this document.

NVGRE: Network Virtualization using Generic Routing Encapsulation

OAM: Operations, Administration, and Management

Controller: an entity that generates PD packets and injects them into the overlay network through VTEPs, also collects PD packets from network devices in overlay network.

2. Conventions Used in This Document

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].

3. Path Detection Under the Assistance of Fabric Controller

This section describes the format of PD packet.

To provide accurate monitoring and/or diagnostic services, a PD packet and the corresponding user packets should be transported over the same data path. In addition, PD packets SHOULD NOT be transferred to the outside of the overlay network.

3.1. General Format of PD packet

Figure 2 shows the format of a PD packet:

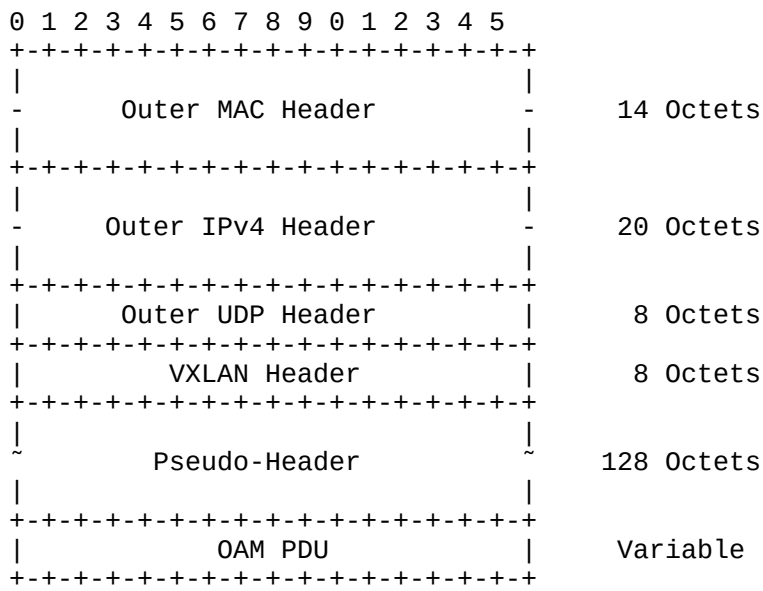


Figure 2: Format of the PD packet

VXLAN Header (8 Octets): A fixed size field, used to carry NVO3 specific information. This work complies with the VXLAN Header specified in Section 5 of [RFC 7348] but uses a reserve field as the flag to distinguish the packets for PD from the normal user packets.

Pseudo-Header (128 Octets): A fixed size field, consists of the information of Ethernet MAC header, IPv4 header, and TCP/UDP header, which is used to identify the packets within the same flow.

OAM PDU (Variable): A variable size field,used to carry the path detection information. An OAM PUB consists of OAM flag, OAM type and Extendable TLV as shown in Section 3.4. For a OAM PDU, 4 Octets alignment MUST be guaranteed.

3.2. Format of VXLAN Header

In this work, the "PD" flag(as with the illustration in Figure 3) MUST be set for all the PD packets.

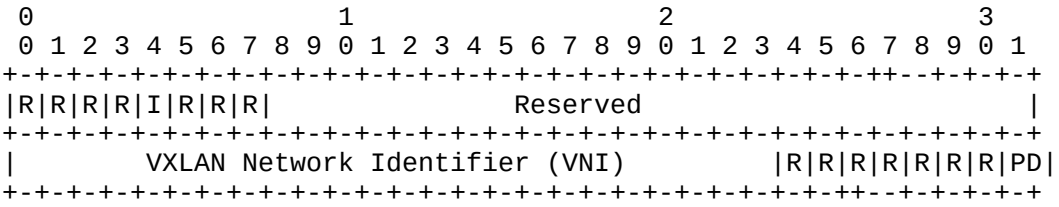


Figure 3: VXLAN header with the PD Flag

PD (1 bit) - Indicates it is a PD packet and needs to be handled as specified in this document.

All other fields comply with what are specified in Section 5 of RFC 7348 [RFC7348].

3.3. Pseudo-Header

The Pseudo-Header is used to ensure that the PD packets are transported along the paths that the service flows actually transported. In order to achieve this, the five-tuples identifying the service flow should be copied directly into associating fields in the Pseudo-header.

3.4. Format of OAM PDU

OAM PDU consists of an OAM flag field, an OAM type field and an Extendable TLV field. This structure is used to identify the type of Path Detection, and records the OAM information along the traverse path at each hop. The information will be report to fabric controller at each hop, in order to depict the complete path information. Following is the format of OAM PDU.

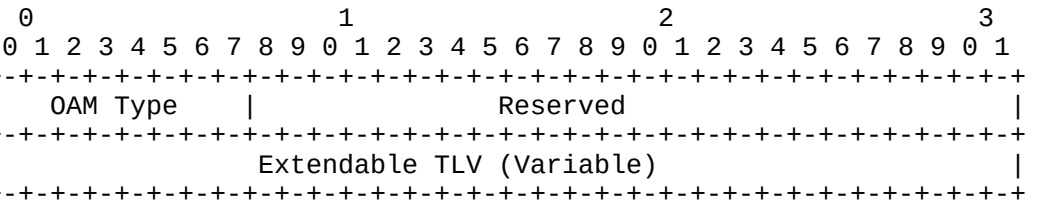


Figure 4: Format of OAM PDU

OAM Type (1 Octet): used to identify the function of PD packets. Currently two functions are specified: path traversal and path tracking.

OAM type	Function
0x01	Path Traversal
0x02	Path Tracking
Other	Reserved

Reserved (3 Octets): padding bits, used to keep the 4 Octets alignment.

Extendable TLV (Variable): used to carry path detection information such as the Ingress/Egress Interface Identifiers of network devices along the path in VXLAN overlay network.

3.5. Format of Extendable OAM TLV

The following figure depicts the general format of an Extendable OAM TLV:

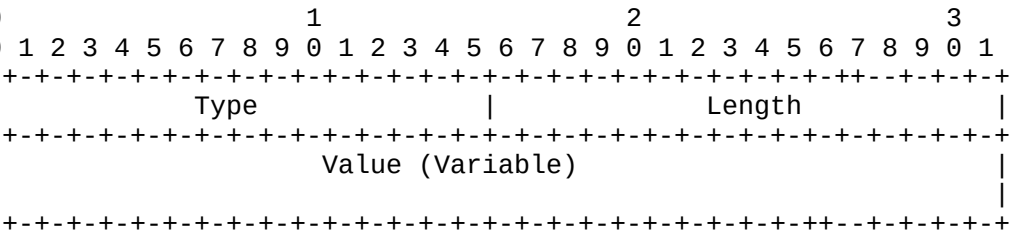


Figure 5: Extendable TLV of OAM PDU

Type (2 Octets): Specifies the Type of the TLV.(see Section 3.5.1 for TLV types)

Length (2 Octets): Specifies the length of the 'Value' field in octets. Length of the 'field' can be either zero or more octets.

Value (Variable): The length and the content of this field depend on the type of the TLV. (see Section 3.5.2 for content of TLV)

3.5.1. TLV Type

This document specifies two type of Extendable OAM TLV: Ingress Interface Identifier (IIID) TLV and Egress Interface Identifier (EIID) TLV. The Type field of each TLV is specified as follows:

Type	TLV Name
0x0001	Ingress Interface Identifier
0x0002	Egress Interface Identifier
0x0003	Transaction Identifier
0x0004	Ingress Interface Name Identifier
0x0005	Egress Interface Name Identifier
0x0006	Authentication

3.5.2. Content of Extendable OAM TLV

For an IIID TLV, the type field is set as 0x0001, the length field is set as 4. The value field is 4 Octets long which contains the device's Ingress Interface Identifier.

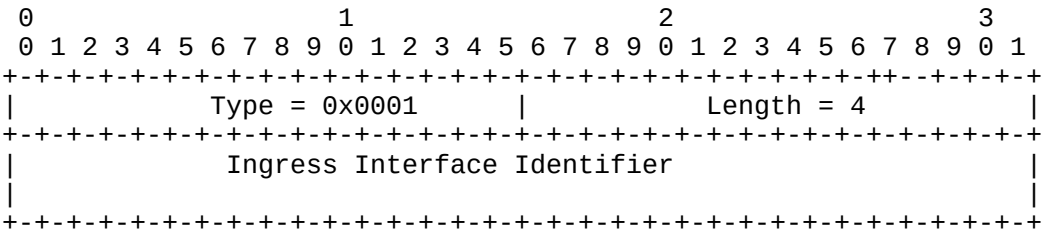


Figure 6: IIID TLV

For an EIID TLV, the type field is set as 0x0002, the length field is set as 4. The value field is 4 Octets long which contains the device's Egress Interface Identifier.

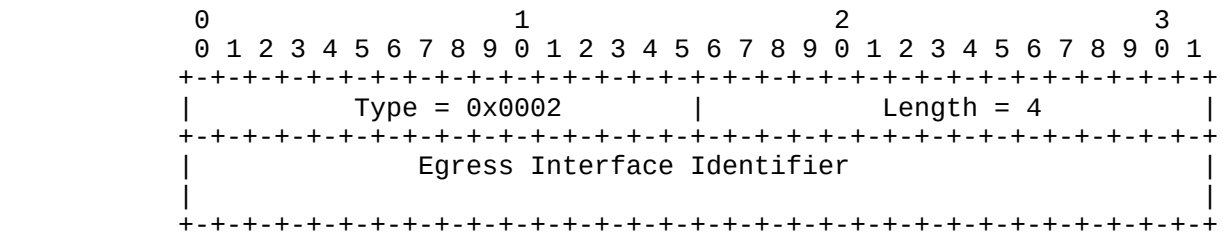


Figure 7: EIID TLV

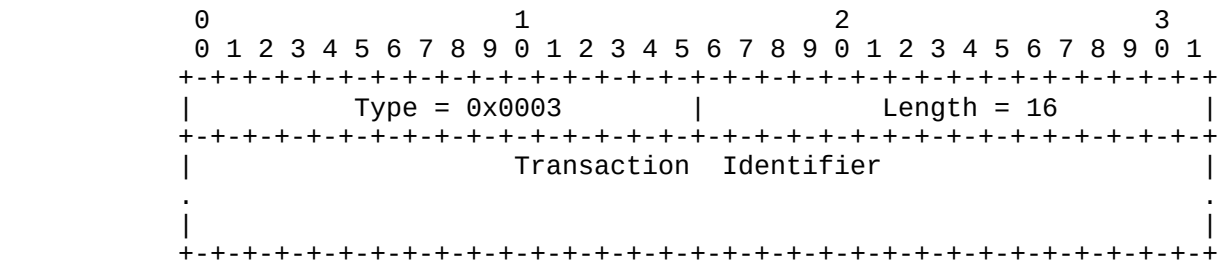


Figure 8: Transaction Identifier TLV

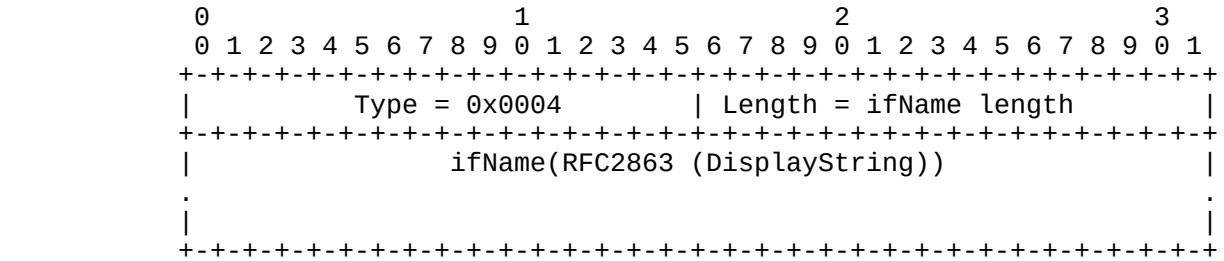


Figure 9: Ingress Interface Name Identifier TLV

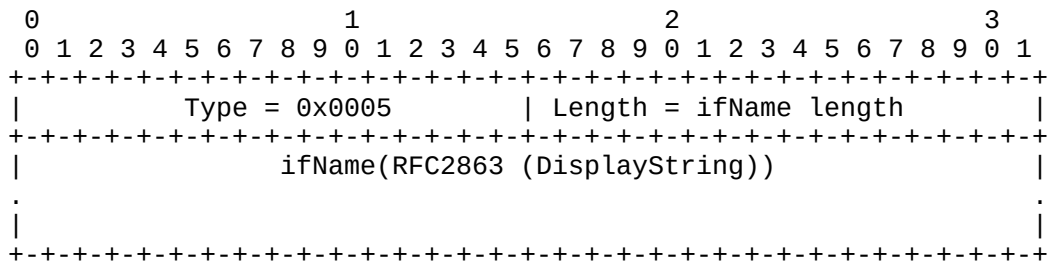
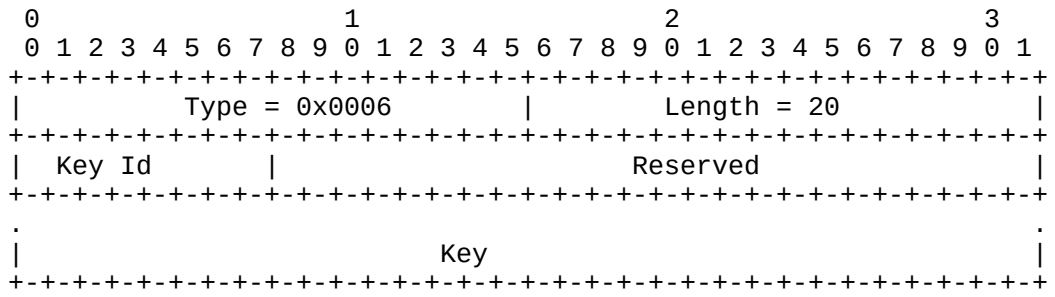


Figure 10: Egress Interface Name Identifier TLV



Key ID: 8 bits. This allows multiple keys to be active
 simultaneously.

Auth Key: 16 octets. This field carries the MD5 [RFC1321] checksum
 for the entire IP packet. When the Auth Key
 is calculated, the shared MD5 key is stored in this field,
 and the checksum fields in the IP header, UDP header are set to zero
.
 The result of the algorithm is placed in the Key field.

Figure 11: Authentication TLV

4. Path Detection between VTEPs

In VXLAN overlay networks, Equal Cost Multi-path(ECMP) may exist between two VTEPs, which may be leveraged to achieve load balance. Link failure and other reasons may lead to the broken of equal cost paths. In order to avoid delivering packets to the broken paths, it’s necessary to detect all the potential paths between the VTEPs. The basic idea is to traversal these paths using the PD packets.

The process of path detection between VTEPs is:

1. The Fabric Controller generates a series of Path Traversal packets targeting to the same Egress VTEP. The outer Source UDP port numbers of the Path Traversal packets keep increased by 1. For example, assume the outer Source UDP port number of a Path Traversal packet is set to 4000. Then the outer Source UDP port numbers of subsequent Path Traversal packets are set as 4001, 4002, 4003, etc. The 'PD bit' in VXLAN header is set to 1 . The content of Pseudo header is left to empty (default value is full-zero), and the OAM type field is set to 0x01 to indicate that it is a Path Traversal packet.

2. After the Ingress VTEP receives the Path Traversal packets from Controller, it then computes the corresponding egress port based on the outer header information and then delivers the packet to that port. By continuous increasing the Source UDP port number, these packets can be distributed to different equal cost paths. Therefore, these Path Traversal packets could go through all the equal cost paths between the two VTEPs.

3. The Extendable TLV field contains multipls TLVs. Transaction Identifier TLV is set by controller and carried in packet without modification in scenario where multiple transactions are initiated by controller between two endpoints. Network device can add Ingress Interface Identifier or Ingress Interface Name Identifier, and Egress Interface Identifier or Egress Interface Name Identifier starting at the end of Transaction Identifier TLV. Both of these TLVs are set by the network devices along the transport path. The TLVs are used to record the identifier of device's Ingress/Egress interface the PD packet goes through. Each network device receives the Path Traversal packet from its upstream device, makes a copy of it and passes the copy to its CPU. After filling the extendable TLVs in this copy, the network device will deliver this copy to the Fabric Controller for further handling.

4. As new TLVs are added by network device in payload section of UDP/Ipv4 packet, it's good practise to update the IP length, UDP length and IP CRC.

5. By gathering all the Path traversal packets from the network devices along the paths, the Controller is able to compute the number of available paths, which could be presented by graphical chart.

5. Path Detection between Tenant Systems

In VXLAN overlay network, link failures are common and it may affect normal operations of up-layer applications. For example, it may lead to service flow interruptions which are unacceptable for most applications.

Path Detection between two End Systems is essential for accurate monitoring and/or diagnostics. The basic idea is to transport the Path Tracking packets right along the path, that the service flow are transport through.

The process of Path Detection between Tenant Systems is:

1. The Fabric Controller generates one Path Tracking packet to Ingress VTEP. The 'PD bit' in the VXLAN header of the packet is set to 1. The content of Pseudo-header is set as the tuple information which are transported over the path being detected. The OAM type field is set to 0x02 to indicate it is a Path Tracking packet.

2. After the Ingress VTEP receives the Path Tracking packets from Fabric Controller, it will firstly compute the outer source UDP port number based on the information form in Pseudo-header. Then it deliveries these packets to the corresponding egress port based on the outer headers information.

3. Each network device receives the Path Tracking packet from its upstream device, makes a copy of it and passes the copy to its CPU. After filling the extendable TLVs in this copy, the network device will deliver this copy to the Controller for further handling. By doing this, each network device along the path will deliver a copy of Path Tracking packets back to Fabric Controller in a hop-by-hop manner.

4. By gathering all the Path traversal packets from network devices along the paths, Fabric Controller is able to accurately monitor the status of each link on the data flow path and locate the point of failure. Fabric Controller may also present the path status using graphical chart.

6. Security Considerations

VXLAN security consideration is discussed in Section 7 of RFC 7348. This document specifies a path failure detection mechanism by extending the VXLAN header. Thus it has the similar vulnerability as VXLAN. For example, attackers can inject spoofed path failure detection packets to the VXLAN overlay network. Administrative

measures, ACL(Access Control List), authentication and encryption etc could be used to mitigate the attack.

In addition, because the controller needs to collect and process the PD packets sent from the network devices. An attacker may perform DDoS attacks to the controller by generating a large amount of PD packets and sent them to a VXLAN overlay network. This issue will be well analyzed in our future work.

As communication between controller and network switch is over internet and it's IP traffic, IPSEC Encryption [RFC 6071] may be used to encrypt the communication.

7. IANA Considerations

TBD.

8. Acknowledgements

The authors would like to specially thank Daolong zhou for his generous help in improving the readability of this document.

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>.

[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>.

9.2. Informative References

[I-D.ashwood-nvo3-oam-requirements]
Chen, H., Ashwood-Smith, P., Xia, L., Iyengar, R., Tsou, T., Sajassi, A., Boucadair, M., Jacquenet, C., Daikoku, M., Ghanwani, A., and R. Krishnan, "NVO3 Operations, Administration, and Maintenance Requirements", draft-ashwood-nvo3-oam-requirements-03 (work in progress), July 2015.

[I-D.tissa-nvo3-oam-fm]
Senevirathne, T., Salam, S., Kumar, D., Finn, N., Eastlake, D., and S. Aldrin, "NVO3 Fault Management", draft-tissa-nvo3-oam-fm-02 (work in progress), June 2015.

[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

Junying Pang
Alibaba

Email: kittypang@alibaba-inc.com

Jie Cao
Alibaba

Email: jie.caojie@alibaba-inc.com

Dapeng Liu
Alibaba

Email: max.ldp@alibaba-inc.com

Dacheng Zhang
Alibaba

Email: dacheng.zdc@alibaba-inc.com

Yizhou Li
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China

Phone: +86-25-56624440
Email: liyizhou@huawei.com

Hao Chen
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China

Phone: +86-25-56624440
Email: philips.chenhao@huawei.com

David Zhou
Cisco Systems
China

BoJian Wang
Cisco Systems
China

Deepak Kumar
Cisco Systems
USA

Email: dekumar@cisco.com

Ruichang Gao
H3C
China

Email: gaoruichang@h3c.com

Yan Qiao
H3C
China

Email: qiaoyan@h3c.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

S. Pallagatti, Ed.
B. Saji
S. Paragiri
Juniper Networks
V. Govindan
M. Mudigonda
Cisco
G. Mirsky
Ericsson

July 6, 2015

BFD for VXLAN
draft-spallagatti-bfd-vxlan-01

Abstract

This document describes use of Bidirectional Forwarding Detection (BFD) protocol for VXLAN . Comments on this draft should be directed to nvo3@ietf.org, rtg-bfd@ietf.org.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 7, 2016.

Table of Contents

1. Introduction

"Virtual eXtensible Local Area Network (VXLAN)" has been defined in [RFC7348] that provides an encapsulation scheme which allows VM's to communicate in data centre network.

VXLAN is typically deployed in data centres on virtualized hosts, which may be spread across multiple racks. The individual racks may be parts of a different Layer 3 network or they could be in a single Layer 2 network. The VXLAN segments/overlay networks are overlaid on top of these Layer 2 or Layer 3 networks.

A VM can communicate with a VM in other host only if they are on same VXLAN. VM's are unaware of VXLAN tunnels as VXLAN tunnel terminates on VTEP (hypervisor/TOR). VTEP (hypervisor/TOR) are responsible for encapsulating and decapsulating frames sent from VM's.

Since underlay is a L3 network, connectivity check for these tunnels becomes important. BFD as defined in [RFC5880] can be used to monitor the VXLAN tunnels.

This draft addresses requirements outlined in [I-D.ashwood-nvo3-operational-requirement]. Specifically with reference to the OAM model to Figure 3 of [I-D.ashwood-nvo3-operational-requirement], this draft outlines proposal to implement the OAM mechanism between the NV Edges using BFD.

2. Use cases

Main use case of BFD for VXLAN is for tunnel connectivity check. There are other use cases such as

Layer 2 VM's:

Most deployments will have VM's with only L2 capabilities and may not understand L3. BFD being a L3 protocol can be used for tunnel connectivity check, where BFD will start and terminate at the NV Edge (VTEPs).

It is possible to aggregate the connectivity checks for multiple tenants by running a BFD session between the VTEPs over VxLAN tunnel. In rest of this document terms NV Edge and VTEP are used interchangeably.

Fault localization:

It is also possible that VM's are L3 aware and can possibly host a BFD session. In these cases BFD sessions can be established between VM's for connectivity check. In addition a BFD session can be established between VTEPs for tunnel connectivity check. Having a hierarchical OAM model helps localize faults.

Service node reachability:

Service node is responsible for sending BUM traffic. In case of service node tunnel terminates at VTEP and it might not even host VM's. If TOR's/Hypervisor wants to check service node reachability then it would like run BFD session over VXLAN tunnel to service node.

3. Deployment

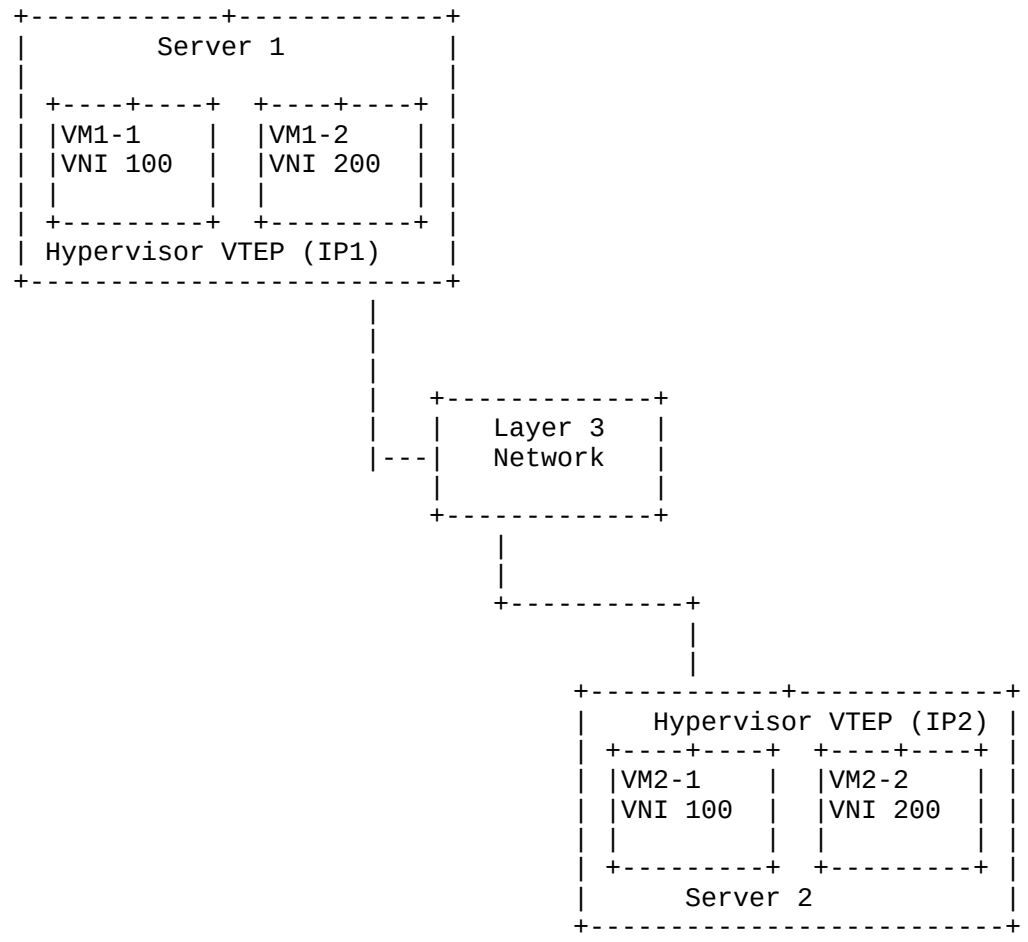


Figure 1

Figure 1 illustrates a scenario where we have two servers, each of them hosting two VMs. These VTEPs terminate two VXLAN tunnels with VNI number 100 and 200 between them. Separate BFD sessions can be established between the VTEPs (IP1 and IP2) for monitoring each of the VXLAN tunnels (VNI 100 and 200). No BFD packet intended to Hypervisor VTEP should be forwarded to VM's as VM's may drop this leading to false negative. This method is also applicable VTEP which are either software or physical device.

4. BFD Packet Encapsulation

VxLAN packet format has been defined in Section 5 of [RFC7348]. The Outer IP/UDP and VXLAN headers MUST be encoded by the sender as per [RFC7348].

If VTEP is equipped with GPE header capitalises and decides to use GPE instead of VXLAN then GPE header MUST be encoded as per Section 3.3 of [I-D.quinn-vxlan-gpe]. Next Protocol Field in GPE header MUST be set to IPv4 or IPv6.

Details of how VTEP decides to use VXLAN or GPE header is outside the scope of this document.

The BFD packet MUST be carried inside the inner MAC frame of the VxLAN packet. The inner MAC frame carrying the BFD payload has the following format:

Ethernet Header:

Destination MAC: This MUST be a well-known MAC [TBD] OR the MAC address of the destination VTEP. The details of how the destination MAC address is obtained is outside the scope of this document.

Source MAC: MAC address of the originating VTEP

IP header:

Source IP: IP address of the originating VTEP.

Destination IP: IP address of the terminating VTEP.

TTL: This MUST be set to 1. This is to ensure that the BFD packet is not routed within the L3 underlay network.

Note: Inner source and destination IP needs more discussion in WG.

The fields of the UDP header and the BFD control packet are encoded as specified in RFC 5881.

5. Reception of BFD packet

Once a packet is received, VTEP MUST validate the packet as described in Section 4.1 of [RFC7348]. If the Destination MAC of the inner MAC frame matches the well-known MAC or the MAC address of the VTEP the packet MUST be processed further.

The UDP destination port and the TTL of the inner MAC frame MUST be validated to determine if the received packet can be processed by BFD. BFD packet with inner MAC set to VTEP or well-known MAC address MUST not be forwarded to VM's.

5.1. Demux of the BFD packet

Demux of IP BFD packet has been defined in Section 3 of [RFC5881]. Since multiple BFD sessions may be running between two VTEPs, there needs to be a mechanism for demultiplexing received BFD packets to the proper session. The procedure for demultiplexing packets with Your Discriminator = 0 is different from [RFC5880]. For such packets, the BFD session is identified using the VNID, the source IP and the destination IP of the packet. If BFD packet is received with non-zero your discriminator then BFD session should be demultiplexed only with your discriminator as the key.

6. Echo BFD

Support for echo BFD is outside the scope of this document.

7. IANA Considerations

The well-known MAC to be used for the Destination MAC address of the inner MAC frame needs to be defined

8. Security Considerations

Document recommends setting of inner IP TTL to 1 which could lead to DDoS attack, implementation MUST have throttling in place. Throttling MAY be relaxed for BFD packeted based on port number.

Other than inner IP TTL set to 1 this specification does not raise any additional security issues beyond those of the specifications referred to in the list of normative references.

9. Contributors

Reshad Rahman
rrahman@cisco.com
Cisco

10. Acknowledgements

Authors would like to thank Jeff Hass of Juniper Networks for his reviews and feedback on this material.

11. Normative References

[I-D.ashwood-nvo3-operational-requirement]
Ashwood-Smith, P., Iyengar, R., Tsou, T., Sajassi, A., Boucadair, M., Jacquenet, C., and M. Daikoku, "NVO3 Operational Requirements", draft-ashwood-nvo3-operational-requirement-03 (work in progress), July 2013.

[I-D.ietf-bfd-seamless-base]
Akiya, N., Pignataro, C., Ward, D., Bhatia, M., and J. Networks, "Seamless Bidirectional Forwarding Detection (S-BFD)", draft-ietf-bfd-seamless-base-05 (work in progress), June 2015.

[I-D.quinn-vxlan-gpe]
Quinn, P., Manur, R., Kreeger, L., Lewis, D., Maino, F., Smith, M., Agarwal, P., Yong, L., Xu, X., Elzur, U., Garg, P., and D. Melman, "Generic Protocol Extension for VXLAN", draft-quinn-vxlan-gpe-04 (work in progress), February 2015.

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

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, June 2010.

[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.

Authors' Addresses

Santosh Pallagatti (editor)
Juniper Networks
Embassy Business Park
Bangalore, KA 560093
India

Email: santoshpk@juniper.net

Basil Saji
Juniper Networks
Embassy Business Park
Bangalore, KA 560093
India

Email: sbasil@juniper.net

Sudarsan Paragiri
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, California 94089-1206
USA

Email: sparagiri@juniper.net

Vengada Prasad Govindan
Cisco

Email: venggovi@cisco.com

Mallik Mudigonda
Cisco

Email: mmudigon@cisco.com

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

NVO3 Working Group
INTERNET-DRAFT
Intended Status: Informational
Expires: January 7, 2016

Y. Li
L. Yong
Huawei Technologies

July 6, 2015

VLAN Configuration Considerations in Split-NVE
draft-yizhou-nvo3-vlan-config-in-split-nve-00

Abstract

In a Split-NVE structure, a control plane protocol between a hypervisor and its associated external NVE(s) to distribute the virtual machine networking state and the relevant attributes. One of the key attributes to be negotiated is VLAN ID which is the most common locally-significant tag for carrying traffic associated with a specific virtual network. This document provides the informational guides on how to configure the VLAN IDs to local networks in Split-NVE structure.

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

The 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 Split-NVE scenario. The protocol requirement [I-D.ietf-nvo3-hpvr2nve-cp-req] presents one of the key requirements which allows the negotiation on a locally-significant tag for carrying traffic associated with a specific virtual network. The tag is commonly a VLAN ID [IEEE 802.1Q]. This document uses the term "VLAN ID" or VID to cover the locally-significant tag. Traffic isolation in overlay network is based on virtual network ID. Before the traffic entering the ingress point of the overlay network, isolation is based on VLAN ID.

A bridged network may connect end Devices to external NVE. We refer it as indirect connection. Another case is direction connect which means end device directly connects to the external NVE without going through any intermediate device. Figure 1 shows the two connection types in local network.

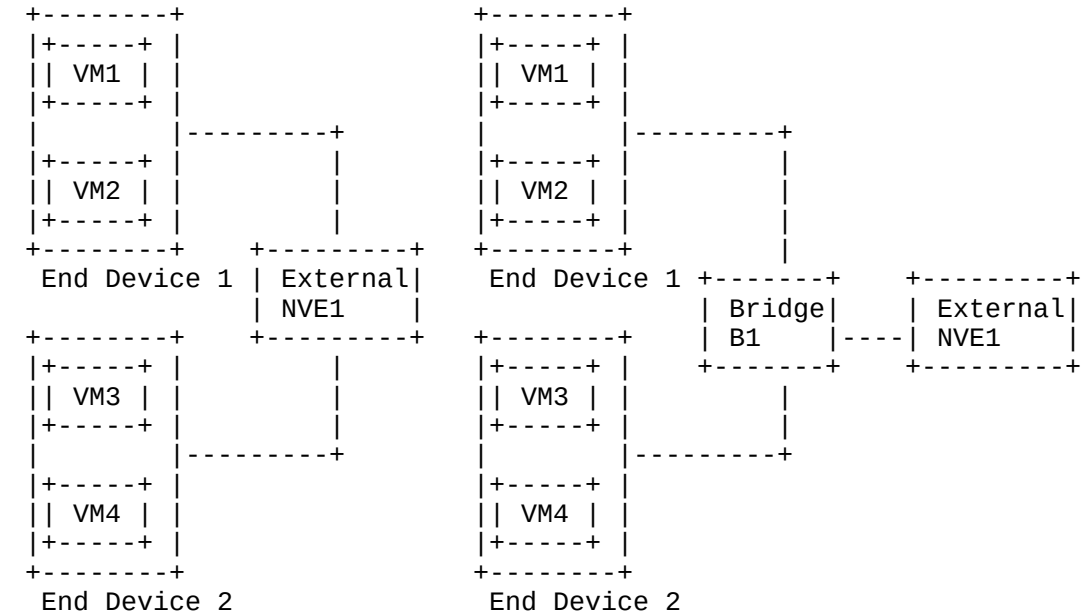


Figure 1 Direct Connection (left) and Indirect Connection (right)

Some scenarios require the switching among virtual machines to be always performed on the external NVE rather than on end device or an intermediate bridge (if any). It helps to ease the policy enforcement. Such forwarding mode is called reflective relay (RR) or hairpin forwarding. A received frame on a port that supports reflective relay mode can be forwarded on the same port on which it was received. Figure 2 and 3 show the expected traffic flow when RR mode is used in direct and indirect connection respectively. The numbers in brackets indicate the expected sequence and the number with a prime indicates simultaneous sequence when the multicast traffic is considered. To achieve the expected local traffic isolation could be tricky especially for that shown in figure 3 if we consider the intermediate bridge is a traditional switch that is only able to identify VLAN tags.

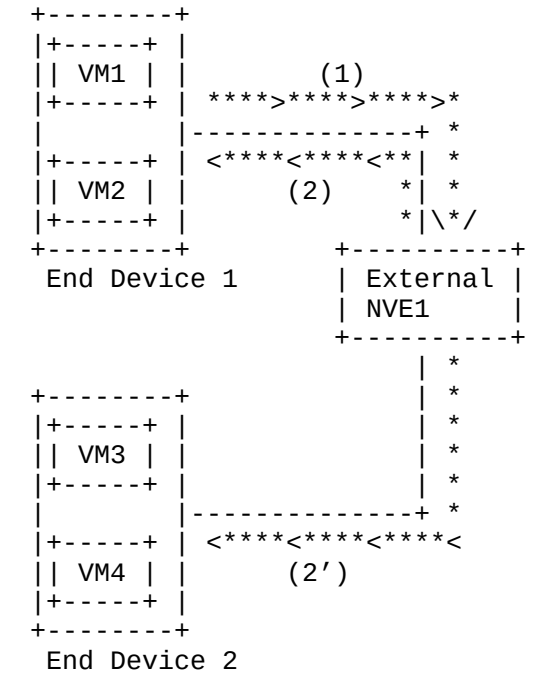


Figure 2 Reflective Relay Mode in Direct Connection

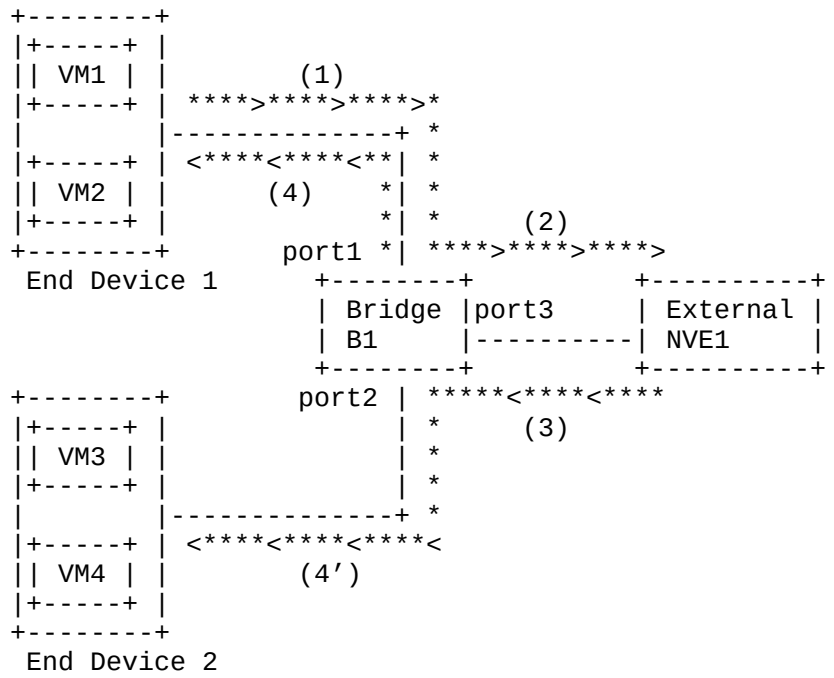


Figure 3 Reflective Relay Mode in Indirect Connection

This document provides the information on how to correctly configure the VLAN IDs to achieve the traffic isolation in local network for either direct or indirect connection and for either RR forwarding or normal forwarding mode.

1.1 Terminology

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

This document uses the same terminology as found in [RFC7365] and [I-D.ietf-nvo3-hpvr2nve-cp-req].

RR - Reflective Relay. A received frame on a port that supports reflective relay mode can be forwarded on the same port on which it was received.

2. VLAN ID Configurations

The most common approach is to configure VLAN on per VN base in the

local network. It works well for most scenarios. If we examine the scenarios from two dimensions, direct or indirect connection, and RR mode or traditional forwarding mode, such VLAN configuration is not applicable to indirect connection and RR mode case for both unicast and multicast. Take figure 3 as example, we assume VM1, VM2 and VM3 are all belonging to the same VN, say VN100. When local VLAN ID is configured based on per VN, the packet from VM1 to VM3 will be forwarded by intermediate bridge B1 directly without NVE1 involved. It violates the expected behaviors in RR mode. If VM1 sends a multicast packet in VN100, intermediate Bridge B1 will forward to port 2 and port 3, NVE1 receives it and hairpins it back to B1. B1 will replicate it to port 1 and port 2. Then VM3 will receive duplicate copies which is not a correct behavior expected.

There are two potential ways to configure VLAN IDs in indirect connection and RR forwarding mode case to fulfil the local traffic isolation requirement.

2.1 VLAN ID per VM

When configuring different VLAN IDs for each VM and let NVE associate these VLAN ID to the same VN, it naturally ensures that the frame from one VM to another is not locally switched at the intermediate bridges. It requires a lot of work at the external NVE. NVE needs to remember the VN to VLAN ID mappings and performs the VLAN ID translations for unicast packet. For multicast traffic, the external NVE needs to replicate the packet to each of the VLANs belonging to the same VN. One way to save such effort for multicast packets is to use per-VN based VLAN ID for downstream multicast traffic. Downstream traffic here refers the multicast packets forwarded by external NVE to potential recipient VMs. Per-VN based VLAN IDs should not overlap with per-VM based VLAN IDs with this approach. Number of VLANs are consumed very quickly in this case.

2.2 Private VLAN configuration per VN

The intermediate bridge can be configured as private VLAN [RFC5517] deployment. Each VN consumes two VLAN IDs in this case. Primary VLAN ID needs to be configured on the uplink port of the intermediate bridge and the port type is set to be Promiscuous Port. Secondary VLAN ID needs to be configured on the down link ports of the intermediate bridge and the port type is set to be Isolated Ports to prohibit the direct communicating between any ports of them. Such setting should be per VN base. The shared VLAN learning (SVL) [IEEE 802.1Q] needs to be enabled for primary and secondary VLAN per VN.

To support RR mode on NVE, the intermediate bridge MUST disable MAC learning on the uplink port. As a result, the frame from a down link

port of the intermediate bridge will be sent to the uplink port as an unknown unicast frame to the external NVE. Such configuration will prevent the MAC learning hopping between the uplink and downlink ports in shared VLAN learning case.

3. Summary

In indirect connection scenarios, the intermediate bridge has to be carefully configured with VLAN IDs especially when RR forwarding is enabled on the external NVE and end device. The protocol running between the hypervisor of the end device and the external NVE does not have the capability to configure the intermediate bridge. Therefore the network management system is required to configure the intermediate bridge when indirect connection has to be used. The MVRP [IEEE802.1ak] may facilitate the auto VLAN ID configuration at the intermediate bridge in some cases.

4. Security Considerations

TBD

5. IANA Considerations

No IANA action is required. RFC Editor: please delete this section before publication.

6. References

6.1 Normative References

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

[RFC5517] HomChaudhuri, S. and M. Foschiano, "Cisco Systems' Private VLANs: Scalable Security in a Multi-Client Environment", RFC 5517, February 2010.

6.2 Informative References

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

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

[I-D.ietf-nvo3-nve-nva-cp-req] Kreeger, L., Dutt, D., Narten, T., and D. Black, "Network Virtualization NVE to NVA Control Protocol Requirements", draft-ietf-nvo3-nve-nva-cp-req-01 (work in progress), October 2014.

[I-D.ietf-nvo3-arch] Black, D., Narten, T., et al, "An Architecture for Overlay Networks (NVO3)", draft-narten-nvo3-arch, work in progress.

[I-D.ietf-nvo3-hpvr2nve-cp-req] Yizhou, L., Yong, L., Kreeger, L., Narten, T., and D. Black, "Hypervisor to NVE Control Plane Requirements", draft-ietf-nvo3-hpvr2nve-cp-req-02 (work in progress), February 2015.

[IEEE 802.1Qbg] IEEE, "Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks - Amendment 21: Edge Virtual Bridging", IEEE Std 802.1Qbg, 2012.

[8021Q] IEEE, "Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks", IEEE Std 802.1Q-2011, August, 2011.

[802.1ak] IEEE, "IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks - Amendment 07: Multiple Registration Protocol", IEEE Std 802.1ak-2007, 2007.

Authors' Addresses

Yizhou Li
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China

Phone: +86-25-56624629
EMail: liyizhou@huawei.com

Lucy Yong
Huawei Technologies, USA

Email: lucy.yong@huawei.com