Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 5, 2019

A. Lindem
Cisco Systems
Y. Qu
Huawei

March 4, 2019

RIB YANG Data Model
draft-acee-rtgwg-yang-rib-extend-10.txt

Abstract

The Routing Information Base (RIB) is a list of routes and their corresponding administrative data and operational state.

RFC 8349 defines the basic building blocks for RIB, and this model augments it to support multiple next-hops (aka, paths) for each route as well as additional attributes.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 5, 2019.

Table of Contents

1. Introduction

This document defines a YANG, [RFC6020][RFC7950], data model which extends the generic data model for RIB by augmenting the ietf-routing model as defined in [RFC8349].

RIB is a collection of best routes from all routing protocols. Within a protocol routes are selected based on the metrics in use by that protocol, and the protocol install its best routes to RIB. RIB selects the best route by comparing the route preference (aka, administrative distance) of the associated protocol.

The augmentations described herein extend the RIB to support multiple paths per route, route metrics, and administrative tags.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Terminology and Notation

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

The following terms are defined in [RFC8342]:

- client
- server
- configuration
- system state
- operational state
- intended configuration

The following terms are defined in [RFC7950]:

- action
- augment
- container
- container with presence
- data model
- data node
- feature
- leaf
- list
- mandatory node
- module
- schema tree

- RPC (Remote Procedure Call) operation

2.1. Glossary of New Terms

Routing Information Base (RIB): An object containing a list of routes, together with other information. See [RFC8349] Section 5.2 for details.

2.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

2.3. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
if	ietf-interfaces	[RFC8343]
rt	ietf-routing	[RFC8349]
v4ur	ietf-ipv4-unicast-routing	[RFC8349]
v6ur	ietf-ipv6-unicast-routing	[RFC8349]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and Corresponding YANG Modules

3. Design of the Model

The YANG definitions in this document augment the ietf-routing model defined in [RFC8349], which provides a basis for routing system data model development. Together with modules defined in [RFC8349], a generic RIB Yang model is defined to implement and monitor RIB.

The models in [RFC8349] also define the basic configuration and operational state for both IPv4 and IPv6 static routes and this document also provides augmentations for static routes to support multiple next-hop and more next-hop attributes.

3.1. RIB Tags and Preference

Individual routes tags will be supported at both the route and next-hop level. A preference per next-hop is also supported for selection of the most preferred reachable static route.

3.2. Multiple next-hops

Both Ipv4 and IPv6 static route configuration defined in [RFC8349] have been augmented with a multi-next-hop option.

A static route/prefix can be configured to have multiple next-hops, each with their own tag and route preference.

In RIB, a route may have multiple next-hops. They can be either equal cost multiple paths (ECMP), or they may have different metrics.

3.3. Repair path

The loop-free alternate (LFA) Fast Reroute (FRR) pre-computes repair paths by routing protocols, and RIB stores the best repair path.

A repair path is augmented in RIB operation state for each path.

4. RIB Model Tree

The tree associated with the "ietf-rib-extension" module follows. The meaning of the symbols can be found in [RFC8340]. The ietf-routing.yang tree with the augmentations herein is included in Appendix A.

```
augment /rt:routing/rt:control-plane-protocols
         /rt:control-plane-protocol/rt:static-routes/v4ur:ipv4
         /v4ur:route/v4ur:next-hop/v4ur:next-hop-options
         /v4ur:simple-next-hop:
 +--rw preference?        uint32
 +--rw tag?               uint32
 +--rw application-tag?   uint32
augment /rt:routing/rt:control-plane-protocols
         /rt:control-plane-protocol/rt:static-routes/v4ur:ipv4
         /v4ur:route/v4ur:next-hop/v4ur:next-hop-options
         /v4ur:next-hop-list/v4ur:next-hop-list/v4ur:next-hop:
 +--rw preference?        uint32
 +--rw tag?               uint32
 +--rw application-tag?   uint32
augment /rt:routing/rt:control-plane-protocols
         /rt:control-plane-protocol/rt:static-routes/v6ur:ipv6
         /v6ur:route/v6ur:next-hop/v6ur:next-hop-options
```

```
          /v6ur:simple-next-hop:
    +--rw preference?        uint32
    +--rw tag?               uint32
    +--rw application-tag?   uint32
   augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol/rt:static-routes/v6ur:ipv6
            /v6ur:route/v6ur:next-hop/v6ur:next-hop-options
            /v6ur:next-hop-list/v6ur:next-hop-list/v6ur:next-hop:
    +--rw preference?        uint32
    +--rw tag?               uint32
    +--rw application-tag?   uint32
   augment /rt:routing/rt:ribs/rt:rib:
    +--ro rib-summary-statistics
       +--ro total-routes?              uint32
       +--ro total-active-routes?       uint32
       +--ro total-route-memory?        uint64
       +--ro protocol-rib-statistics* []
          +--ro rib-protocol?             identityref
          +--ro protocol-total-routes?    uint32
          +--ro protocol-active-routes?   uint32
          +--ro protocol-route-memory?    uint64
   augment /rt:routing/rt:ribs/rt:rib/rt:routes/rt:route:
    +--ro metric?            uint32
    +--ro tag?               uint32
    +--ro application-tag?   uint32
   augment /rt:routing/rt:ribs/rt:rib/rt:routes:
    +--ro repair-route* [id]
       +--ro id          string
       +--ro next-hop
       |  +--ro outgoing-interface?   if:interface-state-ref
       |  +--ro next-hop-address?     inet:ip-address
       +--ro metric?     uint32
   augment /rt:routing/rt:ribs/rt:rib/rt:routes/rt:route
            /rt:next-hop/rt:next-hop-options/rt:simple-next-hop:
    +--ro repair-path?
            -> /rt:routing/ribs/rib/routes/repair-route/id
   augment /rt:routing/rt:ribs/rt:rib/rt:routes/rt:route
            /rt:next-hop/rt:next-hop-options/rt:special-next-hop:
    +--ro repair-path?
            -> /rt:routing/ribs/rib/routes/repair-route/id
   augment /rt:routing/rt:ribs/rt:rib/rt:routes/rt:route
            /rt:next-hop/rt:next-hop-options/rt:next-hop-list
            /rt:next-hop-list/rt:next-hop:
    +--ro repair-path?
            -> /rt:routing/ribs/rib/routes/repair-route/id
```

5. RIB YANG Model

```
<CODE BEGINS> file "ietf-rib-extension@2019-03-01.yang"
module ietf-rib-extension {
  yang-version "1.1";
  namespace "urn:ietf:params:xml:ns:yang:ietf-rib-extension";

  prefix rib-ext;

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

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-routing {
    prefix "rt";
  }

  import ietf-ipv4-unicast-routing {
    prefix "v4ur";
  }

  import ietf-ipv6-unicast-routing {
    prefix "v6ur";
  }

  organization
      "IETF RTGWG - Routing Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/group/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>

     Author:   Acee Lindem
               <mailto:acee@cisco.com>
     Author:   Yingzhen Qu
               <mailto:yingzhen.qu@huawei.com>";

  description
    "This YANG module extends the generic data model for
     RIB by augmenting the ietf-netmod-routing-cfg
     model. It is intended that the module will be extended
     by vendors to define vendor-specific RIB parameters.

     This YANG model conforms to the Network Management
```

```
        Datastore Architecture (NDMA) as described in RFC 8342.

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

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX;
        see the RFC itself for full legal notices.";

    revision 2019-03-01 {
      description
        "Initial RFC Version";
      reference
        "RFC XXXX: A YANG Data Model for RIB Extensions.";
    }

    /* Groupings */
    grouping rib-statistics {
      description "Statistics grouping used for RIB augmentation";
      container rib-summary-statistics {
        config false;
        description "Container for RIB statistics";
        leaf total-routes {
          type uint32;
          description
            "Total routes in the RIB from all protocols";
        }
        leaf total-active-routes {
          type uint32;
          description
            "Total active routes in the RIB";
        }
        leaf total-route-memory {
          type uint64;
          description
            "Total memory for all routes in the RIB from all
             protocol clients";
        }
        list protocol-rib-statistics {
          description "List protocol statistics";
          leaf rib-protocol {
            type identityref {
```

```
            base rt:routing-protocol;
          }
          description "Routing protocol for statistics";
        }
        leaf protocol-total-routes {
          type uint32;
          description
            "Total number routes for protocol in the RIB";
        }
        leaf protocol-active-routes {
          type uint32;
          description
            "Number active routes for protocol in the RIB";
        }
        leaf protocol-route-memory {
          type uint64;
          description
            "Total memory for all routes in the RIB for protocol";
        }
      }
    }
  }

  grouping next-hop {
    description
      "Next-hop grouping";
    leaf interface {
      type if:interface-ref;
      description
        "Outgoing interface";
    }
    leaf address {
      type inet:ip-address;
      description
        "IPv4 or IPv6 Address of the next-hop";
    }
  }

  grouping attributes {
    description
      "Common attributes applicable to all paths";
    leaf metric {
      type uint32;
      description "Route metric";
    }
    leaf tag {
      type uint32;
      description "Route tag";
```

```
      }
      leaf application-tag {
        type uint32;
        description "Additional Application-Specific Route tag";
      }
    }
    grouping path-attribute {
      description
        "Path attribute grouping";
      leaf repair-path {
        type leafref {
          path "/rt:routing/rt:ribs/rt:rib/"
               + "rt:routes/repair-route/id";
        }
        description
          "IP Fast ReRoute (IPFRR) repair path, use a path
           from repair-route list";
      }
    }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/rt:static-routes/v4ur:ipv4/"
          + "v4ur:route/v4ur:next-hop/v4ur:next-hop-options/"
          + "v4ur:simple-next-hop"
    {
      description
        "Augment 'simple-next-hop' case in IPv4 unicast route.";
      leaf preference {
        type uint32;
        default "1";
        description "Route preference - Used to select among multiple
                     static routes with a lower preference next-hop
                     preferred and equal preference paths yielding
                     Equal Cost Multi-Path (ECMP).";
      }
      leaf tag {
        type uint32;
        default "0";
        description "Route tag";
      }
      leaf application-tag {
        type uint32;
        description "Additional Application-Specific Route tag";
      }
    }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/rt:static-routes/v4ur:ipv4/"
```

```
          + "v4ur:route/v4ur:next-hop/v4ur:next-hop-options/"
          + "v4ur:next-hop-list/v4ur:next-hop-list/v4ur:next-hop"
    {
      description
        "Augment static route configuration 'next-hop-list'.";

      leaf preference {
        type uint32;
        default "1";
        description "Route preference - Used to select among multiple
                     static routes with a lower preference next-hop
                     preferred and equal preference paths yielding
                     Equal Cost Multi-Path (ECMP).";
      }
      leaf tag {
        type uint32;
        default "0";
        description "Route tag";
      }
      leaf application-tag {
        type uint32;
        description "Additional Application-Specific Route tag";
      }
  }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/rt:static-routes/v6ur:ipv6/"
          + "v6ur:route/v6ur:next-hop/v6ur:next-hop-options/"
          + "v6ur:simple-next-hop"
    {
      description
        "Augment 'simple-next-hop' case in IPv6 unicast route.";
      leaf preference {
        type uint32;
        default "1";
        description "Route preference - Used to select among multiple
                     static routes with a lower preference next-hop
                     preferred and equal preference paths yielding
                     Equal Cost Multi-Path (ECMP).";
      }
      leaf tag {
        type uint32;
        default "0";
        description "Route tag";
      }
      leaf application-tag {
        type uint32;
        description "Additional Application-Specific Route tag";
```

```
      }
    }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/rt:static-routes/v6ur:ipv6/"
          + "v6ur:route/v6ur:next-hop/v6ur:next-hop-options/"
          + "v6ur:next-hop-list/v6ur:next-hop-list/v6ur:next-hop"
    {
      description
        "Augment static route configuration 'next-hop-list'.";

      leaf preference {
        type uint32;
        default "1";
        description "Route preference - Used to select among multiple
                     static routes with a lower preference next-hop
                     preferred and equal preference paths yielding
                     Equal Cost Multi-Path (ECMP).";
      }
      leaf tag {
        type uint32;
        default "0";
        description "Route tag";
      }
      leaf application-tag {
        type uint32;
        description "Additional Application-Specific Route tag";
      }
    }

    augment "/rt:routing/rt:ribs/rt:rib"
    {
      description "Augment a RIB with statistics";
      uses rib-statistics;
    }

    augment "/rt:routing/rt:ribs/rt:rib/"
          + "rt:routes/rt:route"
    {
      description
        "Augment a route in RIB with tag.";
      uses attributes;
    }

    augment "/rt:routing/rt:ribs/rt:rib/"
          + "rt:routes"
    {
      description
```

```
        "Augment a route with a list of repair-paths.";
      list repair-route {
        key "id";
        description
          "A repair-path entry, which can be referenced
           by a repair-path.";
        leaf id {
          type string;
          description
            "A unique identifier.";
        }

        container next-hop {
                description
                  "Route's next-hop attribute.";
          leaf outgoing-interface {
            type if:interface-state-ref;
            description
              "Name of the outgoing interface.";
          }
          leaf next-hop-address {
            type inet:ip-address;
            description
              "IP address of the next hop.";
          }
        }
        leaf metric {
          type uint32;
          description "Route metric";
        }
      }
    }

    augment "/rt:routing/rt:ribs/rt:rib/"
          + "rt:routes/rt:route/rt:next-hop/rt:next-hop-options/"
          + "rt:simple-next-hop"
    {
       description
         "Add more parameters to a path.";
       uses path-attribute;
    }

    augment "/rt:routing/rt:ribs/rt:rib/"
          + "rt:routes/rt:route/rt:next-hop/rt:next-hop-options/"
          + "rt:special-next-hop"
    {
      description
        "Add more parameters to a path.";
```

```
      uses path-attribute;
    }

    augment "/rt:routing/rt:ribs/rt:rib/"
          + "rt:routes/rt:route/rt:next-hop/rt:next-hop-options/"
          + "rt:next-hop-list/rt:next-hop-list/rt:next-hop"
    {
      description
        "This case augments the 'next-hop-options' in the routing
         model.";
      uses path-attribute;
    }
  }
  <CODE ENDS>
```

6. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in ietf-rib-extensions.yang module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. For these augmentations to ietf-routing.yang, the abiliy to delete, add, and modify IPv4 and IPv6 static routes would allow traffic to be misrouted.

Some of the readable data nodes in the ietf-rib-extensions.yang module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. The exposure of the Routing Information Base (RIB) will expose the routing topology of the network. This may be undesirable since both due to the fact that exposure may facilitate other attacks. Additionally, network operators may consider their topologies to be sensitive confidential data.

All the security considerations for [RFC8349] writable and readable data nodes apply to the augmentations described herein.

7. IANA Considerations

This document registers a URI in the IETF XML registry [XML-REGISTRY]. Following the format in [RFC3688], the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-rib-extension

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-acl namespace: urn:ietf:params:xml:ns:yang:ietf-rib-extension prefix: ietf-rib-ext reference: RFC XXXX

8. References

8.1. Normative References

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://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, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

8.2. Informative References

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[XML-REGISTRY] Mealling, M., "The IETF XML Registry", BCP 81, January 2004.

Appendix A. Combined Tree Diagram

This appendix includes the combined ietf-routing.yang and ietf-rib-extensions.yang tree diagram.

```
module: ietf-routing
+--rw routing
|  +--rw router-id?                 yang:dotted-quad
|  +--ro interfaces
|  |  +--ro interface*   if:interface-ref
|  +--rw control-plane-protocols
|  |  +--rw control-plane-protocol* [type name]
|  |     +--rw type             identityref
|  |     +--rw name             string
|  |     +--rw description?     string
|  |     +--rw static-routes
|  +--rw ribs
|     +--rw rib* [name]
|        +--rw name               string
|        +--rw address-family    identityref
|        +--ro default-rib?      boolean {multiple-ribs}?
|        +--ro routes
|        |  +--ro route* []
|        |     +--ro route-preference?   route-preference
|        |     +--ro next-hop
|        |     |  +--ro (next-hop-options)
|        |     |     +--:(simple-next-hop)
|        |     |     |  +--ro outgoing-interface?   if:interface-ref
|        |     |     +--:(special-next-hop)
|        |     |     |  +--ro special-next-hop?     enumeration
|        |     |     +--:(next-hop-list)
|        |     |        +--ro next-hop-list
|        |     |           +--ro next-hop* []
|        |     |              +--ro outgoing-interface?
|        |     |                      if:interface-ref
|        |     +--ro source-protocol     identityref
|        |     +--ro active?             empty
|        |     +--ro last-updated?       yang:date-and-time
|        +---x active-route
|        |  +--ro output
|        |     +--ro route
|        |        +--ro next-hop
|        |        |  +--ro (next-hop-options)
|        |        |     +--:(simple-next-hop)
|        |        |     |  +--ro outgoing-interface?
|        |        |     |          if:interface-ref
|        |        |     +--:(special-next-hop)
|        |        |     |  +--ro special-next-hop?     enumeration
```

```
|         |         |     +--:(next-hop-list)
|         |         |        +--ro next-hop-list
|         |         |           +--ro next-hop* []
|         |         |              +--ro outgoing-interface?
|         |         |                          if:interface-ref
|         |         +--ro source-protocol    identityref
|         |         +--ro active?            empty
|         |         +--ro last-updated?      yang:date-and-time
|         +--rw description?      string
o--ro routing-state
   +--ro router-id?                  yang:dotted-quad
   o--ro interfaces
   |  o--ro interface*   if:interface-state-ref
   o--ro control-plane-protocols
   |  o--ro control-plane-protocol* [type name]
   |     o--ro type    identityref
   |     o--ro name    string
   o--ro ribs
      o--ro rib* [name]
         o--ro name               string
         +--ro address-family     identityref
         o--ro default-rib?       boolean {multiple-ribs}?
         o--ro routes
         |  o--ro route* []
         |     o--ro route-preference?   route-preference
         |     o--ro next-hop
         |     |  +--ro (next-hop-options)
         |     |     +--:(simple-next-hop)
         |     |     |  +--ro outgoing-interface?   if:interface-ref
         |     |     +--:(special-next-hop)
         |     |     |  +--ro special-next-hop?     enumeration
         |     |     +--:(next-hop-list)
         |     |        +--ro next-hop-list
         |     |           +--ro next-hop* []
         |     |              +--ro outgoing-interface?
         |     |                          if:interface-ref
         |     +--ro source-protocol     identityref
         |     +--ro active?             empty
         |     +--ro last-updated?       yang:date-and-time
         o---x active-route
            +--ro output
               o--ro route
                  o--ro next-hop
                  |  +--ro (next-hop-options)
                  |     +--:(simple-next-hop)
                  |     |  +--ro outgoing-interface?
                  |     |            if:interface-ref
                  |     +--:(special-next-hop)
```

```
                     |      |  +--ro special-next-hop?     enumeration
                     |      +--:(next-hop-list)
                     |         +--ro next-hop-list
                     |            +--ro next-hop* []
                     |               +--ro outgoing-interface?
                     |                       if:interface-ref
                     +--ro source-protocol    identityref
                     +--ro active?            empty
                     +--ro last-updated?      yang:date-and-time

   module: ietf-rib-extension
   augment /rt:routing/rt:control-plane-protocols
             /rt:control-plane-protocol/rt:static-routes/v4ur:ipv4
             /v4ur:route/v4ur:next-hop/v4ur:next-hop-options
             /v4ur:simple-next-hop:
     +--rw preference?        uint32
     +--rw tag?               uint32
     +--rw application-tag?   uint32
   augment /rt:routing/rt:control-plane-protocols
             /rt:control-plane-protocol/rt:static-routes/v4ur:ipv4
             /v4ur:route/v4ur:next-hop/v4ur:next-hop-options
             /v4ur:next-hop-list/v4ur:next-hop-list/v4ur:next-hop:
     +--rw preference?        uint32
     +--rw tag?               uint32
     +--rw application-tag?   uint32
   augment /rt:routing/rt:control-plane-protocols
             /rt:control-plane-protocol/rt:static-routes/v6ur:ipv6
             /v6ur:route/v6ur:next-hop/v6ur:next-hop-options
             /v6ur:simple-next-hop:
     +--rw preference?        uint32
     +--rw tag?               uint32
     +--rw application-tag?   uint32
   augment /rt:routing/rt:control-plane-protocols
             /rt:control-plane-protocol/rt:static-routes/v6ur:ipv6
             /v6ur:route/v6ur:next-hop/v6ur:next-hop-options
             /v6ur:next-hop-list/v6ur:next-hop-list/v6ur:next-hop:
     +--rw preference?        uint32
     +--rw tag?               uint32
     +--rw application-tag?   uint32
   augment /rt:routing/rt:ribs/rt:rib:
     +--ro rib-summary-statistics
        +--ro total-routes?              uint32
        +--ro total-active-routes?       uint32
        +--ro total-route-memory?        uint64
        +--ro protocol-rib-statistics* []
           +--ro rib-protocol?             identityref
           +--ro protocol-total-routes?    uint32
           +--ro protocol-active-routes?   uint32
```

```
         +--ro protocol-route-memory?    uint64
   augment /rt:routing/rt:ribs/rt:rib/rt:routes/rt:route:
     +--ro metric?            uint32
     +--ro tag?               uint32
     +--ro application-tag?   uint32
   augment /rt:routing/rt:ribs/rt:rib/rt:routes:
     +--ro repair-route* [id]
        +--ro id          string
        +--ro next-hop
        |  +--ro outgoing-interface?   if:interface-state-ref
        |  +--ro next-hop-address?     inet:ip-address
        +--ro metric?     uint32
   augment /rt:routing/rt:ribs/rt:rib/rt:routes/rt:route/rt:next-hop
             /rt:next-hop-options/rt:simple-next-hop:
     +--ro repair-path?   -> /rt:routing/ribs/rib/routes/repair-route/id
   augment /rt:routing/rt:ribs/rt:rib/rt:routes/rt:route/rt:next-hop
             /rt:next-hop-options/rt:special-next-hop:
     +--ro repair-path?   -> /rt:routing/ribs/rib/routes/repair-route/id
   augment /rt:routing/rt:ribs/rt:rib/rt:routes/rt:route/rt:next-hop
             /rt:next-hop-options/rt:next-hop-list/rt:next-hop-list
             /rt:next-hop:
     +--ro repair-path?   -> /rt:routing/ribs/rib/routes/repair-route/id
```

Appendix B. ietf-rib-extension.yang examples

Examples will be added in a future version of this document.

Appendix C. Acknowledgments

The RFC text was produced using Marshall Rose's xml2rfc tool.

The authors wish to thank Les Ginsberg, Krishna Deevi, and Suyoung Yoon for their helpful comments and suggestions.

The authors wish to thank Tom Petch and Rob Wilton for review and comments.

Authors' Addresses

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

EMail: acee@cisco.com

Yingzhen Qu
Huawei
2330 Central Expressway
Santa Clara, CA 95050
USA

EMail: yingzhen.qu@huawei.com

Routing Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 2, 2017

Anil Kumar S N
Gaurav Agrawal
Vinod Kumar S
Huawei Technologies
C. Bowers
Juniper Networks

August 29, 2016

Maximally Redundant Trees in Segment Routing
draft-agv-rtgwg-spring-segment-routing-mrt-03

Abstract

[RFC7812] defines an architecture for IP/LDP Fast Reroute using Maximally Redundant Trees (MRT-FRR). This document extends the use of MRT to provide link and node protection for networks that use segment routing. This document makes use of the inherent support in segment routing for associating segments with different algorithms for computing next hops to reach prefixes. It assigns new Segment Routing Algorithms values corresponding to the MRT-Red and MRT-Blue next-hop computations defined in [RFC7811].

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 March 2, 2017.

Table of Contents

1. Introduction

MRT is a fast re-route technology that provides 100% coverage for link and nodes failures. This document describes how to use MRT as a FRR technology in a segment routing network.

2. Requirements Language

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

3. Terminology

Redundant Trees (RT): A pair of trees where the path from any node X to the root R along the first tree is node-disjoint with the path from the same node X to the root along the second tree. These can be computed in 2-connected graphs.

Maximally Redundant Trees (MRT): A pair of trees where the path from any node X to the root R along the first tree and the path from the same node X to the root along the second tree share the minimum number of nodes and the minimum number of links. Each such shared node is a cut-vertex. Any shared links are cut-links. Any RT is an MRT but many MRTs are not RTs. The two MRTs are referred to as MRT-Blue and MRT-Red.

MRT-Red: MRT-Red is used to describe one of the two MRTs; it is used to described the associated forwarding topology and MT-ID. Specifically, MRT-Red is the decreasing MRT where links in the GADAG are taken in the direction from a higher topologically ordered node to a lower one.

MRT-Blue: MRT-Blue is used to describe one of the two MRTs; it is used to described the associated forwarding topology and MT-ID. Specifically, MRT-Blue is the increasing MRT where links in the GADAG are taken in the direction from a lower topologically ordered node to a higher one.

MRT Island: From the computing router, the set of routers that support a particular MRT profile and are connected via MRT-eligible links.

Island Border Router (IBR): A router in the MRT Island that is connected to a router not in the MRT Island and both routers are in a common area or level.

Island Neighbor (IN): A router that is not in the MRT Island but is adjacent to an IBR and in the same area/level as the IBR.

4. MRT for Segment Routing Network

4.1. Overview

Segment Routing (SR) allows a flexible definition of end-to-end paths within IGP topologies by encoding paths as sequences of topological sub-paths, called "segments". These segments are advertised by the link-state routing protocols (IS-IS and OSPF). Prefix segments represent an ECMP-aware shortest-path to a prefix (or a node), as per the state of the IGP topology. Adjacency segments represent a hop over a specific adjacency between two nodes in the IGP.

MRT Fast Reroute requires that packets to be forwarded not only on the shortest-path tree, but also on two Maximally Redundant Trees (MRTs), referred to as the MRT-Blue and the MRT-Red. A router that experiences a local failure must also have predetermined which alternate to use. The MRT algorithm is defined in [RFC7811]. The

Default MRT Profile path calculation uses Lowpoint algorithm to calculate Maximally Redundant Trees.

To use MRT in Segment Routing network below mentioned capabilities needs to be incorporated in SR node.

1. SR nodes MUST support IGP extensions for MRT.

2. SR nodes MUST support MRT-RED and MRT-BLUE Algorithms.

3. SR nodes MUST advertise it’s MRT capability.

4. SR nodes SHOULD send and receive SID/Label for MRT topologies (MRT-RED and MRT-BLUE) for SR segment(s).

5. SR nodes MUST support computation of MRTs.

4.2. IGP extensions for MRT

These extensions are to support the distributed computation of Maximally Redundant Trees (MRT). These extensions indicate the MRT profile(s) each router supports. Different MRT profiles can be defined to support different uses and to allow transition of capabilities.

To support MRT for SR, a new SR MRT Profile is defined (as defined in section 5 of this document). IGP extensions for MRT[I-D.ietf-isis-mrt] / [I-D.ietf-ospf-mrt]MUST carry SR MRT profile.

4.3. SR Algorithm value for MRT-Blue and MRT-Red

Segment routing supports the use of different algorithms for computing paths to reach nodes and prefixes. To accomplish this, every Prefix-SID has a mandatory algorithm field. This Prefix-SID identifies an instruction to forward a packet along the path computed using the algorithm identified in the algorithm field.

[I-D.ietf-spring-segment-routing] defines two algorithms, Shortest Path and Strict Shortest Path.
[I-D.ietf-isis-segment-routing-extensions] and
[I-D.ietf-ospf-segment-routing-extensions] each assign the algorithm values of 0 and 1 to identify these two algorithms.

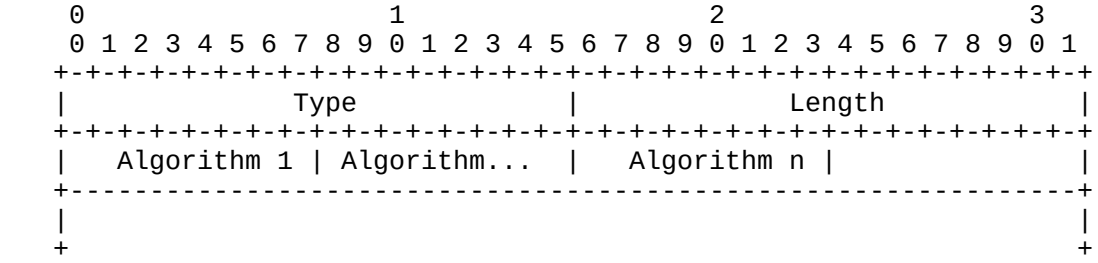
This document defines two additional algorithm values to support MRT-FRR using Segment Routing. The two algorithms correspond to the MRT-Red and MRT-Blue for the Default MRT Profile.

4.4. MRT capability advertisement for SR

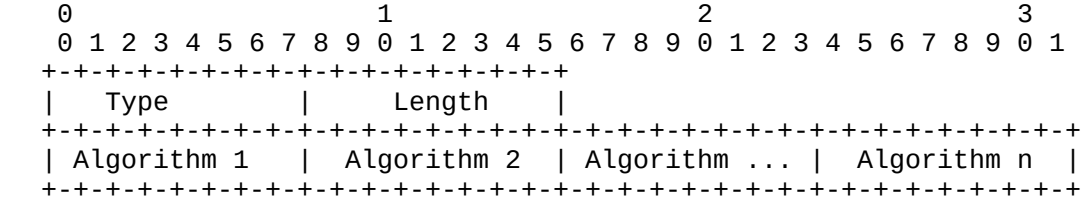
As packets routed on a hop-by-hop basis require that each router compute a shortest-path tree that is consistent, it is necessary for each router to compute the MRT-Blue next hops and MRT-Red next hops in a consistent fashion. For this each SR node needs to know which of the SR nodes in the SR domain supports MRT. This MUST be communicated using SR MRT Capability Advertisement.

Both OSPF [I-D.ietf-ospf-segment-routing-extensions] and ISIS [I-D.ietf-isis-segment-routing-extensions] supports segment routing capabilities advertisement. MRT algorithm capabilities need to be advertised in SR-Algorithm TLV for OSPF extension for segment routing and SR-Algorithm Sub-TLV for ISIS extension for segment routing.

SR-Algorithm TLV [I-D.ietf-ospf-segment-routing-extensions]



SR-Algorithm Sub-TLV[I-D.ietf-isis-segment-routing-extensions]



SR node is considered as MRT capable node if it advertises MRT capability in IGP extension of MRT as well as IGP extension of SR.

4.5. SR MRT SID/Label advertisement

In a link-state IGP domain, an SR-capable IGP node advertises segments for its attached prefixes and adjacencies. These segments are called IGP segments or IGP SIDs. They play a key role in Segment Routing as they enable the expression of any topological path throughout the IGP domain. Such a topological path is either expressed as a single IGP segment or a list of multiple IGP segments.

SR nodes supporting MRT MUST advertise two additional labels corresponding to MRT-BLUE and MRT-RED for each prefix segment.

These labels are carried as a part of prefix SID sub-tlv (OSPF Section 5 of [I-D.ietf-ospf-segment-routing-extensions], ISIS Section 2.1 of [I-D.ietf-isis-segment-routing-extensions]) with algorithm field set to algorithm value corresponding to the MRT forwarding topology as described in section Section 4.3.

4.6. MRT computation for SR

An SR node that advertise support for the Segment Routing MRT Profile MUST support MRT-RED and MRT-BLUE forwarding topology creation and support the computation of FRR paths as per the MRT algorithm described in [RFC7811].

5. MRT-FRR for destination-based and traffic-engineered SR

In addition to the Prefix-SIDs for Shortest Path algorithm, the IGP distributes Prefix-SIDs for MRT-Red and MRT-Blue. This allows each node to install transit forwarding entries for the MRT-Red and MRT-Blue paths for those prefixes. In normal operation, the traffic for a given destination will be forwarded based on the Shortest Path algorithm Prefix-SID for that destination. Upon detecting a link failure, a node can act as a point-of-local repair (PLR) by forwarding the traffic using either the MRT-Red or MRT-Blue Prefix-SID for the same destination. Following the appropriate MRT path, the traffic will the destination without crossing the failed link or the remote node attached to the failed link, if such a path exists.

The description above is independent of the segment routing forwarding plane. With the MRT-Red and MRT-Blue Segment Routing Algorithm values defined in this document, MRT-FRR can provide protection for traffic using either the MPLS or the IPv6 forwarding plane for segment routing. For clarity, we also describe the MRT-FRR mechanism when realized using the MPLS forwarding plane for segment routing.

In the MPLS-specific description, each node uses the index information contained in Prefix-SIDs and the SRGBs advertised by its neighbors to install transit forwarding entries for the Shortest Path, MRT-Red path, and MRT-Blue path for each destination prefix. As an example, the transit forwarding entry for a destination prefix on the MRT-Red path is a label swap operation where the both the incoming and outgoing labels correspond to the MRT-Red labels on the MRT-Red path.

In the absence of failures, traffic flows on transit forwarding entries corresponding to the Shortest Path algorithm where an incoming Shortest Path label is swapped to an outgoing Shortest Path label for a given destination. Upon the failure of a link, the PLR may change some forwarding entries to swap an incoming Shortest Path label to an outgoing MRT-Red or MRT-Blue label.

MRT Prefix Segments are applicable to both destination-based SR as well as traffic-engineered SR. In the case of destination-based SR, the Segment List consists of a single Prefix Segment with an MRT-Red or MRT-Blue algorithm value. In this case Prefix Segment identifies the complete MRT-Red or MRT-Blue path to the destination node or prefix. In the case of traffic-engineered SR, a Prefix Segment with an MRT algorithm value may form part of a larger Segment List. In this case, the MRT Prefix Segment identifies a portion of the entire end-to-end path in the SR domain. That portion of the path corresponds to the MRT-Red or MRT-Blue path for that prefix.

6. SR MRT Profile

The following set of options defines the SR MRT Profile. The SR MRT Profile is indicated by the MRT Profile ID.

MRT Algorithm: MRT Lowpoint algorithm defined in [RFC7811].

MRT-Red SR Algorithm ID: The MRT-Red SR Algorithm ID is indicated by the MRT-Red SR Algorithm ID.

MRT-Blue SR Algorithm ID: The MRT-Blue SR Algorithm ID is indicated by the MRT-BLue SR Algorithm ID.

GADAG Root Selection Policy: Among the routers in the MRT Island with the lowest numerical value advertised for GADAG Root Selection Priority, an implementation MUST pick the router with the highest Router ID to be the GADAG root. Note that a lower numerical value for GADAG Root Selection Priority indicates a higher preference for selection.

Forwarding Mechanisms: MRT SR Label Option 1A

Recalculation: Recalculation of MRTs SHOULD occur as described in Section 12.2 of [RFC7812] with SR label.

Area/Level Border Behavior: As described in Section 10 of [RFC7812], ABRs/LBRs SHOULD ensure that traffic leaving the area also exits the MRT-Red or MRT-Blue forwarding topology.

7. IANA Considerations

IANA is requested to allocate a value from the MRT Profile Identifier Registry for the Segment Routing MRT Profile with a value of TDB-1.

Value	Description	Reference
TBD-1	Segment Routing MRT Profile	This document

Currently, there is no IANA registry defined for SR Algorithm values carried in the Prefix-SID sub-TLV and the SR Algorithm sub-TLV for IS-IS or the Prefix-SID sub-TLV and SR Algorithm TLV for OSPF [I-D.ietf-isis-segment-routing-extensions] and [I-D.ietf-ospf-segment-routing-extensions] define the Segment Routing algorithms for values 0 and 1.

This document requests IANA to create a registry entitled "Segment Routing Algorithm Values". This should appear in the IANA registry under a new top-level entry entitled "IGP-Independent Segment Routing Parameters". The initial registry is shown below.

Value	SR Algorithm	References
0	Shortest Path	I-D.ietf-isis-segment-routing-extensions I-D.ietf-ospf-segment-routing-extensions
1	Strict Shortest Path	I-D.ietf-isis-segment-routing-extensions I-D.ietf-ospf-segment-routing-extensions
TBD-2	MRT-Red	This document
TBD-3	MRT-Blue	This document

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

8. Security Considerations

Security consideration of MRT and SR are applicable here. None of the additional security consideration are identified.

9. Acknowledgements

None

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>.

[RFC7811] Enyedi, G., Csaszar, A., Atlas, A., Bowers, C., and A. Gopalan, "An Algorithm for Computing IP/LDP Fast Reroute Using Maximally Redundant Trees (MRT-FRR)", RFC 7811, DOI 10.17487/RFC7811, June 2016, <http://www.rfc-editor.org/info/rfc7811>.

[RFC7812] Atlas, A., Bowers, C., and G. Enyedi, "An Architecture for IP/LDP Fast Reroute Using Maximally Redundant Trees (MRT-FRR)", RFC 7812, DOI 10.17487/RFC7812, June 2016, <http://www.rfc-editor.org/info/rfc7812>.

10.2. Informative References

[I-D.ietf-isis-mrt]
Li, Z., Wu, N., <>, Q., Atlas, A., Bowers, C., and J. Tantsura, "Intermediate System to Intermediate System (IS-IS) Extensions for Maximally Redundant Trees (MRT)", draft-ietf-isis-mrt-02 (work in progress), May 2016.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-07 (work in progress), June 2016.

[I-D.ietf-ospf-mrt]
Atlas, A., Hegde, S., Bowers, C., Tantsura, J., and Z. Li, "OSPF Extensions to Support Maximally Redundant Trees", draft-ietf-ospf-mrt-02 (work in progress), May 2016.

[I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions-09 (work in progress), July 2016.

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

Authors' Addresses

Anil Kumar S N
Huawei Technologies
Near EPIP Industrial Area, Kundalahalli Village
Whitefield, Bangalore, Karnataka 560066
India

Email: anil.ietf@gmail.com

Gaurav Agrawal
Huawei Technologies
Near EPIP Industrial Area, Kundalahalli Village
Whitefield, Bangalore, Karnataka 560066
India

Email: gaurav.agrawal@huawei.com

Vinod Kumar S
Huawei Technologies
Near EPIP Industrial Area, Kundalahalli Village
Whitefield, Bangalore, Karnataka 560066
India

Email: vinods.kumar@huawei.com

Chris Bowers
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
USA

Email: cbowers@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 23, 2020

A. Choudhary
M. Jethanandani
Cisco Systems
N. Strahle
E. Aries
Juniper Networks
I. Chen
Jabil

Sep 20, 2019

YANG Model for QoS
draft-asechoud-rtgwg-qos-model-11

Abstract

This document describes a YANG model for Quality of Service (QoS) configuration and operational parameters.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 23, 2020.

Copyright Notice

Table of Contents

1. Introduction

This document defines a base YANG [RFC6020] [RFC7950] data module for Quality of Service (QoS) configuration parameters. Differentiated Services (DiffServ) module is an augmentation of the base QoS model. Remote Procedure Calls (RPC) or notification definition is not part of this document. QoS base modules define a basic building blocks to define a classifier, policy, action and target. The base modules have been augmented to include packet match fields and action parameters to define the DiffServ module. Queues and schedulers are stitched as part of diffserv policy itself or separate modules are defined for creating Queue policy and Scheduling policy. The DiffServ model is based on DiffServ architecture, and various references have been made to available standard architecture documents.

DiffServ is a preferred approach for network service providers to offer services to different customers based on their network Quality-of-Service (QoS) objectives. The traffic streams are differentiated based on DiffServ Code Points (DSCP) carried in the IP header of each packet. The DSCP markings are applied by upstream node or by the edge router on entry to the DiffServ network.

Editorial Note: (To be removed by RFC Editor)

This draft contains several placeholder values that need to be replaced with finalized values at the time of publication. Please apply the following replacements: o "XXXX" --> the assigned RFC value for this draft both in this draft and in the YANG models under the revision statement. o The "revision" date in model, in the format XXXX-XX-XX, needs to be updated with the date the draft gets approved.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [RFC8342 [RFC8342]].

1.1. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340 [RFC8340]]

2. Terminology

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

3. QoS Model Design

A classifier consists of packets which may be grouped when a logical set of rules are applied on different packet header fields. The grouping may be based on different values or range of values of same packet header field, presence or absence of some values or range of values of a packet field or a combination thereof. The QoS classifier is defined in the ietf-qos-classifier module.

A classifier entry contains one or more packet conditioning functions. A packet conditioning function is typically based on direction of traffic and may drop, mark or delay network packets. A set of classifier entries with corresponding conditioning functions when arranged in order of priority represents a QoS policy. A QoS

policy may contain one or more classifier entries. These are defined in ietf-qos-policy module.

Actions are configured in line with respect to the policy module. These include marking, dropping or shaping. Actions are defined in the ietf-qos-action module.

A meter qualifies if the traffic arrival rate is based on agreed upon rate and variability. A meter is modeled based on commonly used alogrithms in industry, Single Rate Tri Color Marking (srTCM) [RFC2697] meter, Two Rate Tri Color Marking (trTCM) [RFC2698] meter, and Single Rate Two Color Marking meter. Different vendors can extend it with other types of meters as well.

4. DiffServ Model Design

DiffServ architecture [RFC3289] and [RFC2475] describe the architecture as a simple model where traffic entering a network is classified and possibly conditioned at the boundary of the network and assigned a different Behavior Aggregate (BA). Each BA is identified by a specific value of DSCP, and is used to select a Per Hop Behavior (PHB).

The packet classification policy identifies the subset of traffic which may receive a DiffServ by being conditioned or mapped. Packet classifiers select packets within a stream based on the content of some portion of the packet header. There are two types of classifiers, the BA classifier, and the Multi-Field (MF) classifier which selects packets based on a value which is combination of one or more header fields. In the ietf-diffserv module, this is realized by augmenting the QoS classification module.

Traffic conditioning includes metering, shaping and/or marking. A meter is used to measure the traffic against a given traffic profile. The traffic profile specifies the temporal property of the traffic. A packet that arrives is first determined to be in or out of the profile, which will result in the action of marked, dropped or shaped. This is realized in vendor specific modules based on the parameters defined in action module. The metering parameters are augmented to the QoS policy module when metering is defined inline, and to the metering template when metering profile is referred in policy module.

5. Modules Tree Structure

This document defines seven YANG modules - four QoS base modules, a scheduler policy module, a queuing policy module and one DiffServ module.

ietf-qos-classifier consists of classifier entries identified by a classifier entry name. Each entry MAY contain a list of filter entries. When no filter entry is present in a classifier entry, it matches all traffic.

```
module: ietf-qos-classifier
   +--rw classifiers
      +--rw classifier-entry* [classifier-entry-name]
         +--rw classifier-entry-name                string
         +--rw classifier-entry-descr?              string
         +--rw classifier-entry-filter-operation?   identityref
         +--rw filter-entry* [filter-type filter-logical-not]
            +--rw filter-type           identityref
            +--rw filter-logical-not    boolean
```

An ietf-qos-policy module contains list of policy objects identified by a policy name and policy type which MUST be provided. With different values of policy types, each vendor MAY define their own construct of policy for different QoS functionalities. Each vendor MAY augment classifier entry in a policy definition with a set of actions.

```
module: ietf-qos-policy
   +--rw policies
      +--rw policy-entry* [policy-name policy-type]
         +--rw policy-name         string
         +--rw policy-type         identityref
         +--rw policy-descr?       string
         +--rw classifier-entry* [classifier-entry-name]
            +--rw classifier-entry-name           string
            +--rw classifier-entry-inline?        boolean
            +--rw classifier-entry-filter-oper?   identityref
            +--rw filter-entry* [filter-type filter-logical-not]
                                {policy-inline-classifier-config}?
            |  +--rw filter-type           identityref
            |  +--rw filter-logical-not    boolean
            +--rw classifier-action-entry-cfg* [action-type]
               +--rw action-type    identityref
               +--rw (action-cfg-params)?
```

ietf-qos-action module contains grouping of set of QoS actions. These include metering, marking, dropping and shaping. Marking sets DiffServ codepoint value in the classified packet. Color-aware and Color-blind meters are augmented by vendor specific modules based on the parameters defined in action module.

```
   module: ietf-qos-action
       +--rw meter-template
          +--rw meter-entry* [meter-name] {meter-template-support}?
             +--rw meter-name                  string
             +--rw (meter-type)?
                +--:(one-rate-two-color-meter-type)
                |  +--rw one-rate-two-color-meter
                |     +--rw committed-rate-value?    uint64
                |     +--rw committed-rate-unit?     identityref
                |     +--rw committed-burst-value?   uint64
                |     +--rw committed-burst-unit?    identityref
                |     +--rw conform-action
                |     |  +--rw conform-2color-meter-action-params*
                |     |           [conform-2color-meter-action-type]
                |     |     +--rw conform-2color-meter-action-type
                |     |           identityref
                |     |     +--rw (conform-2color-meter-action-val)?
                |     +--rw exceed-action
                |        +--rw exceed-2color-meter-action-params*
                |                 [exceed-2color-meter-action-type]
                |           +--rw exceed-2color-meter-action-type
                |                 identityref
                |           +--rw (exceed-2color-meter-action-val)?
                +--:(one-rate-tri-color-meter-type)
                |  +--rw one-rate-tri-color-meter
                |     +--rw committed-rate-value?    uint64
                |     +--rw committed-rate-unit?     identityref
                |     +--rw committed-burst-value?   uint64
                |     +--rw committed-burst-unit?    identityref
                |     +--rw excess-burst-value?      uint64
                |     +--rw excess-burst-unit?       identityref
                |     +--rw conform-action
                |     |  +--rw conform-3color-meter-action-params*
                |     |           [conform-3color-meter-action-type]
                |     |     +--rw conform-3color-meter-action-type
                |     |           identityref
                |     |     +--rw (conform-3color-meter-action-val)?
                |     +--rw exceed-action
                |     |  +--rw exceed-3color-meter-action-params*
                |     |           [exceed-3color-meter-action-type]
                |     |     +--rw exceed-3color-meter-action-type
                |     |           identityref
                |     |     +--rw (exceed-3color-meter-action-val)?
                |     +--rw violate-action
                |        +--rw violate-3color-meter-action-params*
                |                 [violate-3color-meter-action-type]
                |           +--rw violate-3color-meter-action-type
                |                 identityref
```

```
                |           +--rw (violate-3color-meter-action-val)?
                +--:(two-rate-tri-color-meter-type)
                   +--rw two-rate-tri-color-meter
                      +--rw committed-rate-value?    uint64
                      +--rw committed-rate-unit?     identityref
                      +--rw committed-burst-value?   uint64
                      +--rw committed-burst-unit?    identityref
                      +--rw peak-rate-value?         uint64
                      +--rw peak-rate-unit?          identityref
                      +--rw peak-burst-value?        uint64
                      +--rw peak-burst-unit?         identityref
                      +--rw conform-action
                      |  +--rw conform-3color-meter-action-params*
                      |           [conform-3color-meter-action-type]
                      |     +--rw conform-3color-meter-action-type
                      |           identityref
                      |     +--rw (conform-3color-meter-action-val)?
                      +--rw exceed-action
                      |  +--rw exceed-3color-meter-action-params*
                      |           [exceed-3color-meter-action-type]
                      |     +--rw exceed-3color-meter-action-type
                      |           identityref
                      |     +--rw (exceed-3color-meter-action-val)?
                      +--rw violate-action
                         +--rw violate-3color-meter-action-params*
                                  [violate-3color-meter-action-type]
                            +--rw violate-3color-meter-action-type
                                  identityref
                            +--rw (violate-3color-meter-action-val)?
```

ietf-qos-target module contains reference of qos-policy and augments ietf-interfaces [RFC8343] module. A single policy of a particular policy-type can be applied on an interface in each direction of traffic. Policy-type is of type identity and is populated in a vendor specific manner. This way it provides greater flexibility for each vendor to define different policy types each with its own capabilities and restrictions.

Classifier, metering and queuing counters are associated with a target.

```
module: ietf-qos-target
augment /if:interfaces/if:interface:
   +--rw qos-target-entry* [direction policy-type]
      +--rw direction      identityref
      +--rw policy-type    identityref
      +--rw policy-name    string
```

Diffserv module augments QoS classifier module. Many of the YANG types defined in [RFC6991] are represented as leafs in the classifier module.

Metering and marking actions are realized by augmenting the QoS policy-module. Any queuing, AQM and scheduling actions are part of vendor specific augmentation. Statistics are realized by augmenting the QoS target module.

```
module: ietf-diffserv
  augment /classifier:classifiers/classifier:classifier-entry +
          /classifier:filter-entry:
    +--rw (filter-param)?
       +--:(dscp)
       |  +--rw dscp-cfg* [dscp-min dscp-max]
       |     +--rw dscp-min    inet:dscp
       |     +--rw dscp-max    inet:dscp
       +--:(source-ipv4-address)
       |  +--rw source-ipv4-address-cfg* [source-ipv4-addr]
       |     +--rw source-ipv4-addr    inet:ipv4-prefix
       +--:(destination-ipv4-address)
       |  +--rw destination-ipv4-address-cfg* [destination-ipv4-addr]
       |     +--rw destination-ipv4-addr    inet:ipv4-prefix
       +--:(source-ipv6-address)
       |  +--rw source-ipv6-address-cfg* [source-ipv6-addr]
       |     +--rw source-ipv6-addr    inet:ipv6-prefix
       +--:(destination-ipv6-address)
       |  +--rw destination-ipv6-address-cfg* [destination-ipv6-addr]
       |     +--rw destination-ipv6-addr    inet:ipv6-prefix
       +--:(source-port)
       |  +--rw source-port-cfg* [source-port-min source-port-max]
       |     +--rw source-port-min    inet:port-number
       |     +--rw source-port-max    inet:port-number
       +--:(destination-port)
       |  +--rw destination-port-cfg*
       |           [destination-port-min destination-port-max]
       |     +--rw destination-port-min    inet:port-number
       |     +--rw destination-port-max    inet:port-number
       +--:(protocol)
       |     +--rw protocol-cfg* [protocol-min protocol-max]
       |     +--rw protocol-min    uint8
       |     +--rw protocol-max    uint8
       +--:(traffic-group)
          +--rw traffic-group-cfg
             +--rw traffic-group-name?   string
  augment /policy:policies/policy:policy-entry +
          /policy:classifier-entry/policy:filter-entry:
    +--rw (filter-params)?
```

```
          +--:(dscp)
          |  +--rw dscp-cfg* [dscp-min dscp-max]
          |     +--rw dscp-min    inet:dscp
          |     +--rw dscp-max    inet:dscp
          +--:(source-ipv4-address)
          |  +--rw source-ipv4-address-cfg* [source-ipv4-addr]
          |     +--rw source-ipv4-addr    inet:ipv4-prefix
          +--:(destination-ipv4-address)
          |  +--rw destination-ipv4-address-cfg* [destination-ipv4-addr]
          |     +--rw destination-ipv4-addr    inet:ipv4-prefix
          +--:(source-ipv6-address)
          |  +--rw source-ipv6-address-cfg* [source-ipv6-addr]
          |     +--rw source-ipv6-addr    inet:ipv6-prefix
          +--:(destination-ipv6-address)
          |  +--rw destination-ipv6-address-cfg* [destination-ipv6-addr]
          |     +--rw destination-ipv6-addr    inet:ipv6-prefix
          +--:(source-port)
          |  +--rw source-port-cfg* [source-port-min source-port-max]
          |     +--rw source-port-min    inet:port-number
          |     +--rw source-port-max    inet:port-number
          +--:(destination-port)
          |  +--rw destination-port-cfg*
          |           [destination-port-min destination-port-max]
          |     +--rw destination-port-min    inet:port-number
          |     +--rw destination-port-max    inet:port-number
          +--:(protocol)
          |  +--rw protocol-cfg* [protocol-min protocol-max]
          |     +--rw protocol-min    uint8
          |     +--rw protocol-max    uint8
          +--:(traffic-group)
             +--rw traffic-group-cfg
                +--rw traffic-group-name?   string
     augment /policy:policies/policy:policy-entry +
             /policy:classifier-entry +
             /policy:classifier-action-entry-cfg +
             /policy:action-cfg-params:
       +--:(dscp-marking)
       |  +--rw dscp-cfg
       |     +--rw dscp?   inet:dscp
       +--:(meter-inline) {action:meter-inline-feature}?
       |  +--rw (meter-type)?
       |     +--:(one-rate-two-color-meter-type)
       |     |  +--rw one-rate-two-color-meter
       |     |     +--rw committed-rate-value?    uint64
       |     |     +--rw committed-rate-unit?     identityref
       |     |     +--rw committed-burst-value?   uint64
       |     |     +--rw committed-burst-unit?    identityref
       |     |     +--rw conform-action
```

```
|     |     |  +--rw conform-2color-meter-action-params*
                  [conform-2color-meter-action-type]
|     |     |     +--rw conform-2color-meter-action-type
                  identityref
|     |     |     +--rw (conform-2color-meter-action-val)?
|     |     +--rw exceed-action
|     |        +--rw exceed-2color-meter-action-params*
                        [exceed-2color-meter-action-type]
|     |           +--rw exceed-2color-meter-action-type
                        identityref
|     |           +--rw (exceed-2color-meter-action-val)?
|     +--:(one-rate-tri-color-meter-type)
|     |  +--rw one-rate-tri-color-meter
|     |     +--rw committed-rate-value?    uint64
|     |     +--rw committed-rate-unit?     identityref
|     |     +--rw committed-burst-value?   uint64
|     |     +--rw committed-burst-unit?    identityref
|     |     +--rw excess-burst-value?      uint64
|     |     +--rw excess-burst-unit?       identityref
|     |     +--rw conform-action
|     |     |  +--rw conform-3color-meter-action-params*
                        [conform-3color-meter-action-type]
|     |     |     +--rw conform-3color-meter-action-type
                        identityref
|     |     |     +--rw (conform-3color-meter-action-val)?
|     |     +--rw exceed-action
|     |     |  +--rw exceed-3color-meter-action-params*
                        [exceed-3color-meter-action-type]
|     |     |     +--rw exceed-3color-meter-action-type
                        identityref
|     |     |     +--rw (exceed-3color-meter-action-val)?
|     |     +--rw violate-action
|     |        +--rw violate-3color-meter-action-params*
                        [violate-3color-meter-action-type]
|     |           +--rw violate-3color-meter-action-type
                        identityref
|     |           +--rw (violate-3color-meter-action-val)?
|     +--:(two-rate-tri-color-meter-type)
|        +--rw two-rate-tri-color-meter
|           +--rw committed-rate-value?    uint64
|           +--rw committed-rate-unit?     identityref
|           +--rw committed-burst-value?   uint64
|           +--rw committed-burst-unit?    identityref
|           +--rw peak-rate-value?         uint64
|           +--rw peak-rate-unit?          identityref
|           +--rw peak-burst-value?        uint64
|           +--rw peak-burst-unit?         identityref
|           +--rw conform-action
```

```
|           |  +--rw conform-3color-meter-action-params*
                        [conform-3color-meter-action-type]
|           |     +--rw conform-3color-meter-action-type
                        identityref
|           |     +--rw (conform-3color-meter-action-val)?
|           +--rw exceed-action
|           |  +--rw exceed-3color-meter-action-params*
                        [exceed-3color-meter-action-type]
|           |     +--rw exceed-3color-meter-action-type
                        identityref
|           |     +--rw (exceed-3color-meter-action-val)?
|           +--rw violate-action
|              +--rw violate-3color-meter-action-params*
                        [violate-3color-meter-action-type]
|                 +--rw violate-3color-meter-action-type
                        identityref
|                 +--rw (violate-3color-meter-action-val)?
+--:(meter-reference) {action:meter-reference-feature}?
|  +--rw meter-reference-cfg
|     +--rw meter-reference-name    string
|     +--rw meter-type              identityref
+--:(traffic-group-marking) {action:traffic-group-feature}?
|  +--rw traffic-group-cfg
|     +--rw traffic-group?   string
+--:(child-policy) {action:child-policy-feature}?
|  +--rw child-policy-cfg {child-policy-feature}?
|     +--rw policy-name?   string
+--:(count) {action:count-feature}?
|  +--rw count-cfg {count-feature}?
|     +--rw count-action?   empty
+--:(named-count) {action:named-counter-feature}?
|  +--rw named-counter-cfg {named-counter-feature}?
|     +--rw count-name-action?   string
+--:(queue-inline) {diffserv-queue-inline-support}?
|  +--rw queue-cfg
|     +--rw priority-cfg
|     |  +--rw priority-level?   uint8
|     +--rw min-rate-cfg
|     |  +--rw rate-value?   uint64
|     |  +--rw rate-unit?    identityref
|     +--rw max-rate-cfg
|     |  +--rw rate-value?    uint64
|     |  +--rw rate-unit?     identityref
|     |  +--rw burst-value?   uint64
|     |  +--rw burst-unit?    identityref
|     +--rw algorithmic-drop-cfg
|        +--rw (drop-algorithm)?
|           +--:(tail-drop)
```

```
      |              +--rw tail-drop-cfg
      |                 +--rw tail-drop-alg?   empty
      +--:(scheduler-inline) {diffserv-scheduler-inline-support}?
         +--rw scheduler-cfg
            +--rw min-rate-cfg
            |  +--rw rate-value?   uint64
            |  +--rw rate-unit?    identityref
            +--rw max-rate-cfg
               +--rw rate-value?    uint64
               +--rw rate-unit?     identityref
               +--rw burst-value?   uint64
               +--rw burst-unit?    identityref

  module: ietf-queue-policy
      +--rw queue-template {queue-policy-support}?
         +--rw name?        string
         +--rw queue-cfg
            +--rw priority-cfg
            |  +--rw priority-level?   uint8
            +--rw min-rate-cfg
            |  +--rw rate-value?   uint64
            |  +--rw rate-unit?    identityref
            +--rw max-rate-cfg
            |  +--rw rate-value?    uint64
            |  +--rw rate-unit?     identityref
            |  +--rw burst-value?   uint64
            |  +--rw burst-unit?    identityref
            +--rw algorithmic-drop-cfg
               +--rw (drop-algorithm)?
                  +--:(tail-drop)
                     +--rw tail-drop-cfg
                        +--rw tail-drop-alg?   empty
    augment /policy:policies/policy:policy-entry +
            /policy:classifier-entry/policy:filter-entry:
      +--rw (filter-params)? {queue-policy-support}?
         +--:(traffic-group-name)
            +--rw traffic-group-reference-cfg
               +--rw traffic-group-name    string
    augment /policy:policies/policy:policy-entry +
            /policy:classifier-entry +
            /policy:classifier-action-entry-cfg +
            /policy:action-cfg-params:
      +--:(queue-template-name)
               {queue-template-support,queue-policy-support}?
      |  +--rw queue-template-reference-cfg
      |     +--rw queue-template-name    string
      +--:(queue-inline)
               {queue-inline-support,queue-policy-support}?
```

```
         +--rw queue-cfg
            +--rw priority-cfg
            |  +--rw priority-level?   uint8
            +--rw min-rate-cfg
            |  +--rw rate-value?   uint64
            |  +--rw rate-unit?    identityref
            +--rw max-rate-cfg
            |  +--rw rate-value?    uint64
            |  +--rw rate-unit?     identityref
            |  +--rw burst-value?   uint64
            |  +--rw burst-unit?    identityref
            +--rw algorithmic-drop-cfg
               +--rw (drop-algorithm)?
                  +--:(tail-drop)
                     +--rw tail-drop-cfg
                        +--rw tail-drop-alg?   empty

   module: ietf-scheduler-policy
     augment /policy:policies/policy:policy-entry +
             /policy:classifier-entry/policy:filter-entry:
       +--rw (filter-params)?
          +--:(filter-match-all)
             +--rw match-all-cfg
                +--rw match-all-action?   empty
     augment /policy:policies/policy:policy-entry +
             /policy:classifier-entry +
             /policy:classifier-action-entry-cfg +
             /policy:action-cfg-params:
       +--:(scheduler)
       |  +--rw scheduler-cfg
       |     +--rw min-rate-cfg
       |     |  +--rw rate-value?   uint64
       |     |  +--rw rate-unit?    identityref
       |     +--rw max-rate-cfg
       |        +--rw rate-value?    uint64
       |        +--rw rate-unit?     identityref
       |        +--rw burst-value?   uint64
       |        +--rw burst-unit?    identityref
       +--:(queue-policy-name)
          +--rw queue-policy-name
             +--rw queue-policy    string
```

6. Modules

6.1. IETF-QOS-CLASSIFIER

```
<CODE BEGINS>file "ietf-qos-classifier@2019-03-13.yang"
module ietf-qos-classifier {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-qos-classifier";
  prefix classifier;

  organization
    "IETF RTG (Routing Area) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>
     WG Chair: Chris Bowers
               <mailto:cbowers@juniper.net>
     WG Chair: Jeff Tantsura
               <mailto:jefftant.ietf@gmail.com>
     Editor:   Aseem Choudhary
               <mailto:asechoud@cisco.com>
     Editor:   Mahesh Jethanandani
               <mailto:mjethanandani@gmail.com>
     Editor:   Norm Strahle
               <mailto:nstrahle@juniper.net>";
  description
    "This module contains a collection of YANG definitions for
     configuring qos specification implementations.
     Copyright (c) 2019 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.
     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).
     This version of this YANG module is part of RFC XXXX; see
     the RFC itself for full legal notices.";

  revision 2019-03-13 {
    description
      "Latest revision of qos base classifier module";
    reference "RFC XXXX: YANG Model for QoS";
  }

  feature policy-inline-classifier-config {
    description
      " This feature allows classifier configuration
        directly under policy.";
  }
```

```
feature classifier-template-feature {
  description
    " This feature allows classifier as template configuration
      in a policy.";
}

feature match-any-filter-type-support {
  description
    " This feature allows classifier configuration
      directly under policy.";
}

identity filter-type {
  description
    "This is identity of base filter-type";
}

identity classifier-entry-filter-operation-type {
  description
    "Classifier entry filter logical operation";
}

identity match-all-filter {
  base classifier-entry-filter-operation-type;
  description
    "Classifier entry filter logical AND operation";
}

identity match-any-filter {
  base classifier-entry-filter-operation-type;
  if-feature "match-any-filter-type-support";
  description
    "Classifier entry filter logical OR operation";
}

grouping filters {
  description
    "Filters types in a Classifier entry";
  leaf filter-type {
    type identityref {
      base filter-type;
    }
    description
      "This leaf defines type of the filter";
  }
  leaf filter-logical-not {
    type boolean;
    description
```

```
            "
             This is logical-not operator for a filter. When true, it
             indicates filter looks for absence of a pattern defined
             by the filter
            ";
        }
      }

      grouping classifier-entry-generic-attr {
        description
          "
           Classifier generic attributes like name,
           description, operation type
          ";
        leaf classifier-entry-name {
          type string;
          description
            "classifier entry name";
        }
        leaf classifier-entry-descr {
          type string;
          description
            "classifier entry description statement";
        }
        leaf classifier-entry-filter-operation {
          type identityref {
            base classifier-entry-filter-operation-type;
          }
          default "match-all-filter";
          description
            "Filters are applicable as match-any or match-all filters";
        }
      }

      grouping classifier-entry-inline-attr {
        description
          "attributes of inline classifier in a policy";
        leaf classifier-entry-inline {
          type boolean;
          default "false";
          description
            "Indication of inline classifier entry";
        }
        leaf classifier-entry-filter-oper {
          type identityref {
            base classifier-entry-filter-operation-type;
          }
          default "match-all-filter";
```

```
        description
          "Filters are applicable as match-any or match-all filters";
      }
      list filter-entry {
        if-feature "policy-inline-classifier-config";
        must " ../classifier-entry-inline = 'true' " {
          description
            "For inline filter configuration, inline attributemust
             be true";
        }
        key "filter-type filter-logical-not";
        uses filters;
        description
          "Filters configured inline in a policy";
      }
    }

    container classifiers {
      if-feature "classifier-template-feature";
      description
        "list of classifier entry";
      list classifier-entry {
        key "classifier-entry-name";
        description
          "each classifier entry contains a list of filters";
        uses classifier-entry-generic-attr;
        list filter-entry {
          key "filter-type filter-logical-not";
          uses filters;
          description
            "Filter entry configuration";
        }
      }
    }
  }
  <CODE ENDS>
```

6.2. IETF-QOS-POLICY

```
  <CODE BEGINS>file "ietf-qos-policy@2019-03-13.yang"
  module ietf-qos-policy {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-qos-policy";
    prefix policy;
    import ietf-qos-classifier {
      prefix classifier;
      reference "RFC XXXX: YANG Model for QoS";
    }
```

```
     organization "IETF RTG (Routing Area) Working Group";
     contact
       "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
        WG List:  <mailto:rtgwg@ietf.org>
        WG Chair: Chris Bowers
                  <mailto:cbowers@juniper.net>
        WG Chair: Jeff Tantsura
                  <mailto:jefftant.ietf@gmail.com>
        Editor:   Aseem Choudhary
                  <mailto:asechoud@cisco.com>
        Editor:   Mahesh Jethanandani
                  <mailto:mjethanandani@gmail.com>
        Editor:   Norm Strahle
                  <mailto:nstrahle@juniper.net>";
     description
       "This module contains a collection of YANG definitions for
        configuring qos specification implementations.
        Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).
        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";
     revision 2019-03-13 {
       description
         "Latest revision of qos policy";
       reference "RFC XXXX: YANG Model for QoS";
     }
     identity policy-type {
       description
         "This base identity type defines policy-types";
     }
     grouping policy-generic-attr {
       description
         "Policy Attributes";
       leaf policy-name {
         type string;
         description
           "policy name";
       }
       leaf policy-type {
         type identityref {
           base policy-type;
         }
```

```
        description
          "policy type";
      }
      leaf policy-descr {
        type string;
        description
          "policy description";
      }
    }
    identity action-type {
      description
        "This base identity type defines action-types";
    }
    grouping classifier-action-entry-cfg {
      description
        "List of Configuration of classifier & associated actions";
      list classifier-action-entry-cfg {
        key "action-type";
        ordered-by user;
        description
          "Configuration of classifier & associated actions";
        leaf action-type {
          type identityref {
            base action-type;
          }
          description
            "This defines action type ";
        }
        choice action-cfg-params {
          description
            "Choice of action types";
        }
      }
    }
    container policies {
      description
        "list of policy templates";
      list policy-entry {
        key "policy-name policy-type";
        description
          "policy template";
        uses policy-generic-attr;
        list classifier-entry {
          key "classifier-entry-name";
          ordered-by user;
          description
            "Classifier entry configuration in a policy";
          leaf classifier-entry-name {
```

```
            type string;
            description
              "classifier entry name";
          }
          uses classifier:classifier-entry-inline-attr;
          uses classifier-action-entry-cfg;
        }
      }
    }
  }
  <CODE ENDS>
```

6.3. IETF-QOS-ACTION

```
  <CODE BEGINS>file "ietf-qos-action@2019-03-13.yang"
  module ietf-qos-action {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-qos-action";
    prefix action;
    import ietf-inet-types {
      prefix inet;
      reference "RFC 6991: Common YANG Data Types";
    }
    import ietf-qos-policy {
      prefix policy;
      reference "RFC XXXX: YANG Model for QoS";
    }
    organization "IETF RTG (Routing Area) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
       WG List:  <mailto:rtgwg@ietf.org>
       WG Chair: Chris Bowers
                 <mailto:cbowers@juniper.net>
       WG Chair: Jeff Tantsura
                 <mailto:jefftant.ietf@gmail.com>
       Editor:   Aseem Choudhary
                 <mailto:asechoud@cisco.com>
       Editor:   Mahesh Jethanandani
                 <mailto:mjethanandani@gmail.com>
       Editor:   Norm Strahle
                 <mailto:nstrahle@juniper.net>";
    description
      "This module contains a collection of YANG definitions for
       configuring qos specification implementations.
       Copyright (c) 2019 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.
       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
```

```
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).
        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";
      revision 2019-03-13 {
        description
          "Latest revision for qos actions";
        reference "RFC XXXX: YANG Model for QoS";
      }
      feature meter-template-support {
        description
          " This feature allows support of meter-template.";
      }
      feature meter-inline-feature {
        description
          "This feature allows support of meter-inline configuration.";
      }
      feature meter-reference-feature {
        description
          "This feature allows support of meter by reference
           configuration.";
      }
      feature queue-action-support {
        description
          " This feature allows support of queue action configuration
            in policy.";
      }
      feature scheduler-action-support {
        description
          " This feature allows support of scheduler configuration
            in policy.";
      }
      feature child-policy-feature {
        description
          " This feature allows configuration of hierarchical policy.";
      }
      feature count-feature {
        description
          "This feature allows action configuration to enable
           counter in a classifier";
      }
      feature named-counter-feature {
        description
          "This feature allows action configuration to enable
           named counter in a classifier";
      }
```

```
feature traffic-group-feature {
  description
    "traffic-group action support";
}
feature burst-time-unit-support {
  description
    "This feature allows burst unit to be configured as
     time duration.";
}

identity rate-unit-type {
  description
    "base rate-unit type";
}
identity bits-per-second {
  base rate-unit-type;
  description
    "bits per second identity";
}
identity kilo-bits-per-second {
  base rate-unit-type;
  description
    "kilo bits per second identity";
}
identity mega-bits-per-second {
  base rate-unit-type;
  description
    "mega bits per second identity";
}
identity giga-bits-per-second {
  base rate-unit-type;
  description
    "mega bits per second identity";
}
identity percent {
  base rate-unit-type;
  description
    "percentage";
}
identity burst-unit-type {
  description
    "base burst-unit type";
}
identity bytes {
  base burst-unit-type;
  description
    "bytes";
}
```

```
identity kilo-bytes {
  base burst-unit-type;
  description
    "kilo bytes";
}
identity mega-bytes {
  base burst-unit-type;
  description
    "mega bytes";
}
identity millisecond {
  base burst-unit-type;
  if-feature burst-time-unit-support;
  description
    "milli seconds";
}
identity microsecond {
  base burst-unit-type;
  if-feature burst-time-unit-support;
  description
    "micro seconds";
}
identity dscp-marking {
  base policy:action-type;
  description
    "dscp marking action type";
}
identity meter-inline {
  base policy:action-type;
  if-feature meter-inline-feature;
  description
    "meter-inline action type";
}
identity meter-reference {
  base policy:action-type;
  if-feature meter-reference-feature;
  description
    "meter reference action type";
}
identity queue {
  base policy:action-type;
  if-feature queue-action-support;
  description
    "queue action type";
}
identity scheduler {
  base policy:action-type;
  if-feature scheduler-action-support;
```

```
      description
        "scheduler action type";
    }
    identity discard {
      base policy:action-type;
      description
        "discard action type";
    }
    identity child-policy {
      base policy:action-type;
      if-feature child-policy-feature;
      description
        "child-policy action type";
    }
    identity count {
      base policy:action-type;
      if-feature count-feature;
      description
        "count action type";
    }
    identity named-counter {
      base policy:action-type;
      if-feature named-counter-feature;
      description
        "name counter action type";
    }

    identity meter-type {
      description
        "This base identity type defines meter types";
    }
    identity one-rate-two-color-meter-type {
      base meter-type;
      description
        "one rate two color meter type";
    }
    identity one-rate-tri-color-meter-type {
      base meter-type;
      description
        "one rate three color meter type";
      reference
          "RFC2697: A Single Rate Three Color Marker";
    }
    identity two-rate-tri-color-meter-type {
      base meter-type;
      description
        "two rate three color meter action type";
      reference
```

```
        "RFC2698: A Two Rate Three Color Marker";
    }

    identity drop-type {
      description
        "drop algorithm";
    }
    identity tail-drop {
      base drop-type;
      description
        "tail drop algorithm";
    }

    identity conform-2color-meter-action-type {
      description
        "action type in a meter";
    }
    identity exceed-2color-meter-action-type {
      description
        "action type in a meter";
    }
    identity conform-3color-meter-action-type {
      description
        "action type in a meter";
    }
    identity exceed-3color-meter-action-type {
      description
        "action type in a meter";
    }
    identity violate-3color-meter-action-type {
      description
        "action type in a meter";
    }

    grouping rate-value-unit {
      leaf rate-value {
        type uint64;
        description
          "rate value";
      }
      leaf rate-unit {
        type identityref {
          base rate-unit-type;
        }
        description
          "rate unit";
      }
      description
```

```
        "rate value and unit grouping";
    }
    grouping burst {
      description
      "burst value and unit configuration";
      leaf burst-value {
        type uint64;
        description
          "burst value";
      }
      leaf burst-unit {
        type identityref {
          base burst-unit-type;
        }
        description
          "burst unit";
      }
    }

    grouping threshold {
      description
        "Threshold Parameters";
      container threshold {
        description
          "threshold";
        choice threshold-type {
          case size {
            leaf threshold-size {
              type uint64;
              units "bytes";
              description
                "Threshold size";
            }
          }
          case interval {
            leaf threshold-interval {
              type uint64;
              units "microsecond";
              description
                "Threshold interval";
            }
          }
          description
            "Choice of threshold type";
        }
      }
    }
```

```
     grouping drop {
       container drop-cfg {
         leaf drop-action {
           type empty;
           description
             "always drop algorithm";
         }
         description
           "the drop action";
       }
       description
         "always drop grouping";
     }

     grouping queuelimit {
       container qlimit-thresh {
         uses threshold;
         description
           "the queue limit";
       }
       description
         "the queue limit beyond which queue will not hold any packet";
     }

     grouping conform-2color-meter-action-params {
       description
         "meter action parameters";
       list conform-2color-meter-action-params {
         key "conform-2color-meter-action-type";
         ordered-by user;
         description
           "Configuration of basic-meter & associated actions";
         leaf conform-2color-meter-action-type {
           type identityref {
             base conform-2color-meter-action-type;
           }
           description
             "meter action type";
         }
         choice conform-2color-meter-action-val {
           description
             " meter action based on choice of meter action type";
         }
       }
     }

     grouping exceed-2color-meter-action-params {
       description
```

```
        "meter action parameters";
      list exceed-2color-meter-action-params {
        key "exceed-2color-meter-action-type";
        ordered-by user;
        description
          "Configuration of basic-meter & associated actions";
        leaf exceed-2color-meter-action-type {
          type identityref {
            base exceed-2color-meter-action-type;
          }
          description
            "meter action type";
        }
        choice exceed-2color-meter-action-val {
          description
            " meter action based on choice of meter action type";
        }
      }
    }


    grouping conform-3color-meter-action-params {
      description
        "meter action parameters";
      list conform-3color-meter-action-params {
        key "conform-3color-meter-action-type";
        ordered-by user;
        description
          "Configuration of basic-meter & associated actions";
        leaf conform-3color-meter-action-type {
          type identityref {
            base conform-3color-meter-action-type;
          }
          description
            "meter action type";
        }
        choice conform-3color-meter-action-val {
          description
            " meter action based on choice of meter action type";
        }
      }
    }

    grouping exceed-3color-meter-action-params {
      description
        "meter action parameters";
      list exceed-3color-meter-action-params {
        key "exceed-3color-meter-action-type";
```

```
        ordered-by user;
        description
          "Configuration of basic-meter & associated actions";
        leaf exceed-3color-meter-action-type {
          type identityref {
            base exceed-3color-meter-action-type;
          }
          description
            "meter action type";
        }
        choice exceed-3color-meter-action-val {
          description
            " meter action based on choice of meter action type";
        }
      }
    }

    grouping violate-3color-meter-action-params {
      description
        "meter action parameters";
      list violate-3color-meter-action-params {
        key "violate-3color-meter-action-type";
        ordered-by user;
        description
          "Configuration of basic-meter & associated actions";
        leaf violate-3color-meter-action-type {
          type identityref {
            base violate-3color-meter-action-type;
          }
          description
            "meter action type";
        }
        choice violate-3color-meter-action-val {
          description
            " meter action based on choice of meter action type";
        }
      }
    }

    grouping one-rate-two-color-meter {
      container one-rate-two-color-meter {
        description
          "single rate two color marker meter";
        leaf committed-rate-value {
          type uint64;
          description
            "committed rate value";
        }
```

```
        leaf committed-rate-unit {
          type identityref {
            base rate-unit-type;
          }
          description
            "committed rate unit";
        }
        leaf committed-burst-value {
          type uint64;
          description
            "burst value";
        }
        leaf committed-burst-unit {
          type identityref {
            base burst-unit-type;
          }
          description
            "committed burst unit";
        }
        container conform-action {
          uses conform-2color-meter-action-params;
          description
            "conform action";
        }
        container exceed-action {
          uses exceed-2color-meter-action-params;
          description
            "exceed action";
        }
      }
      description
        "single rate two color marker meter attributes";
    }

    grouping one-rate-tri-color-meter {
      container one-rate-tri-color-meter {
        description
          "single rate three color meter";
        reference
          "RFC2697: A Single Rate Three Color Marker";
      leaf committed-rate-value {
        type uint64;
        description
          "meter rate";
      }
      leaf committed-rate-unit {
        type identityref {
          base rate-unit-type;
```

```
        }
        description
          "committed rate unit";
      }
      leaf committed-burst-value {
        type uint64;
        description
          "committed burst size";
      }
      leaf committed-burst-unit {
        type identityref {
          base burst-unit-type;
        }
        description
          "committed burst unit";
      }
      leaf excess-burst-value {
        type uint64;
        description
          "excess burst size";
      }
      leaf excess-burst-unit {
        type identityref {
          base burst-unit-type;
        }
        description
          "excess burst unit";
      }
      container conform-action {
        uses conform-3color-meter-action-params;
        description
          "conform, or green action";
      }
      container exceed-action {
        uses exceed-3color-meter-action-params;
        description
          "exceed, or yellow action";
      }
      container violate-action {
        uses violate-3color-meter-action-params;
        description
          "violate, or red action";
      }
    }
      description
        "one-rate-tri-color-meter attributes";
    }
```

```
     grouping two-rate-tri-color-meter {
       container two-rate-tri-color-meter {
         description
           "two rate three color meter";
         reference
           "RFC2698: A Two Rate Three Color Marker";
       leaf committed-rate-value {
         type uint64;
         units "bits-per-second";
         description
           "committed rate";
       }
       leaf committed-rate-unit {
         type identityref {
           base rate-unit-type;
         }
         description
           "committed rate unit";
       }
       leaf committed-burst-value {
         type uint64;
         description
           "commited burst size";
       }
       leaf committed-burst-unit {
         type identityref {
           base burst-unit-type;
         }
         description
           "committed burst unit";
       }
       leaf peak-rate-value {
         type uint64;
         description
           "peak rate";
       }
       leaf peak-rate-unit {
         type identityref {
           base rate-unit-type;
         }
         description
           "committed rate unit";
       }
       leaf peak-burst-value {
         type uint64;
         description
           "commited burst size";
       }
```

```
      leaf peak-burst-unit {
        type identityref {
          base burst-unit-type;
        }
        description
          "peak burst unit";
      }
      container conform-action {
        uses conform-3color-meter-action-params;
        description
          "conform, or green action";
      }
      container exceed-action {
        uses exceed-3color-meter-action-params;
        description
          "exceed, or yellow action";
      }
      container violate-action {
        uses violate-3color-meter-action-params;
        description
          "exceed, or red action";
      }
    }
      description
        "two-rate-tri-color-meter attributes";
    }

    grouping meter {
      choice meter-type {
        case one-rate-two-color-meter-type {
          uses one-rate-two-color-meter;
          description
            "basic meter";
        }
        case one-rate-tri-color-meter-type {
          uses one-rate-tri-color-meter;
          description
            "one rate tri-color meter";
        }
        case two-rate-tri-color-meter-type {
          uses two-rate-tri-color-meter;
          description
            "two rate tri-color meter";
        }
        description
          " meter action based on choice of meter action type";
      }
      description
```

```
        "meter attributes";
    }

    container meter-template {
      description
        "list of meter templates";
      list meter-entry {
        if-feature meter-template-support;
        key "meter-name";
        description
          "meter entry template";
        leaf meter-name {
          type string;
          description
            "meter identifier";
        }
        uses meter;
      }
    }

    grouping meter-reference {
      container meter-reference-cfg {
        leaf meter-reference-name {
          type string ;
          mandatory true;
          description
            "This leaf defines name of the meter referenced";
        }
        leaf meter-type {
          type identityref {
            base meter-type;
          }
          mandatory true;
          description
            "This leaf defines type of the meter";
        }
        description
          "meter reference name";
      }
      description
        "meter reference";
    }

    grouping count {
      container count-cfg {
      if-feature count-feature;
        leaf count-action {
          type empty;
```

```
          description
            "count action";
        }
        description
          "the count action";
      }
      description
        "the count action grouping";
    }

    grouping named-counter {
      container named-counter-cfg {
      if-feature named-counter-feature;
        leaf count-name-action {
          type string;
          description
            "count action";
        }
        description
          "the count action";
      }
      description
        "the count action grouping";
    }

    grouping discard {
      container discard-cfg {
        leaf discard {
          type empty;
          description
            "discard action";
        }
        description
          "discard action";
      }
      description
        "discard grouping";
    }

    grouping priority {
      container priority-cfg {
        leaf priority-level {
          type uint8;
          description
            "priority level";
        }
        description
          "priority attributes";
```

```
      }
      description
        "priority attributes grouping";
    }
    grouping min-rate {
      container min-rate-cfg {
        uses rate-value-unit;
        description
          "min guaranteed bandwidth";
        reference
          "RFC3289, section 3.5.3";
      }
      description
        "minimum rate grouping";
    }
    grouping dscp-marking {
      container dscp-cfg {
        leaf dscp {
          type inet:dscp;
          description
            "dscp marking";
        }
        description
          "dscp marking container";
      }
      description
        "dscp marking grouping";
    }
    grouping traffic-group-marking {
      container traffic-group-cfg {
        leaf traffic-group {
          type string;
          description
            "traffic group marking";
        }
        description
          "traffic group marking container";
      }
      description
        "traffic group marking grouping";
    }
    grouping child-policy {
      container child-policy-cfg {
        if-feature child-policy-feature;
        leaf policy-name {
          type string;
          description
            "Hierarchical Policy";
```

```
        }
        description
          "Hierarchical Policy configuration container";
      }
      description
        "Grouping of Hierarchical Policy configuration";
    }
    grouping max-rate {
      container max-rate-cfg {
        uses rate-value-unit;
        uses burst;
        description
          "maximum rate attributes container";
        reference
          "RFC3289, section 3.5.4";
      }
      description
        "maximum rate attributes";
    }
    grouping queue {
      container queue-cfg {
        uses priority;
        uses min-rate;
        uses max-rate;
        container algorithmic-drop-cfg {
          choice drop-algorithm {
            case tail-drop {
              container tail-drop-cfg {
                leaf tail-drop-alg {
                  type empty;
                  description
                   "tail drop algorithm";
                }
                description
                  "Tail Drop configuration container";
              }
              description
                "Tail Drop choice";
            }
            description
              "Choice of Drop Algorithm";
          }
          description
            "Algorithmic Drop configuration container";
        }
        description
          "Queue configuration container";
      }
```

```
      description
        "Queue grouping";
    }
    grouping scheduler {
      container scheduler-cfg {
        uses min-rate;
        uses max-rate;
        description
          "Schedular configuration container";
      }
      description
        "Schedular configuration grouping";
    }
  }
  <CODE ENDS>
```

6.4. IETF-QOS-TARGET

```
  <CODE BEGINS>file "ietf-qos-target@2019-03-13.yang"
  module ietf-qos-target {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-qos-target";
    prefix target;

    import ietf-interfaces {
      prefix if;
      reference "RFC8343: A YANG Data Model for Interface Management";
    }
    import ietf-qos-policy {
      prefix policy;
      reference "RFC XXXX: YANG Model for QoS";
    }

    organization "IETF RTG (Routing Area) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
       WG List:  <mailto:rtgwg@ietf.org>
       WG Chair: Chris Bowers
                 <mailto:cbowers@juniper.net>
       WG Chair: Jeff Tantsura
                 <mailto:jefftant.ietf@gmail.com>
       Editor:   Aseem Choudhary
                 <mailto:asechoud@cisco.com>
       Editor:   Mahesh Jethanandani
                 <mailto:mjethanandani@gmail.com>
       Editor:   Norm Strahle
                 <mailto:nstrahle@juniper.net>";
    description
```

```
       "This module contains a collection of YANG definitions for
        configuring qos specification implementations.
        Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).
        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     revision 2019-03-13 {
       description
         "Latest revision qos based policy applied to a target";
       reference "RFC XXXX: YANG Model for QoS";
     }

     identity direction {
       description
         "This is identity of traffic direction";
     }

     identity inbound {
       base direction;
       description
         "Direction of traffic coming into the network entry";
     }

     identity outbound {
       base direction;
       description
         "Direction of traffic going out of the network entry";
     }

     augment "/if:interfaces/if:interface" {
       description
         "Augments Diffserv Target Entry to Interface module";
       list qos-target-entry {
         key "direction policy-type";
         description
           "policy target for inbound or outbound direction";
         leaf direction {
           type identityref {
             base direction;
           }
           description
```

```
            "Direction fo the traffic flow either inbound or outbound";
        }
        leaf policy-type {
          type identityref {
            base policy:policy-type;
          }
          description
            "Policy entry type";
        }
        leaf policy-name {
          type string;
          mandatory true;
          description
            "Policy entry name";
        }
      }
    }
  }
  <CODE ENDS>
```

6.5. IETF-DIFFSERV

```
  <CODE BEGINS>file "ietf-diffserv@2019-03-13.yang"
  module ietf-diffserv {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-diffserv";
    prefix diffserv;

    import ietf-qos-classifier {
      prefix classifier;
      reference "RFC XXXX: YANG Model for QoS";
    }
    import ietf-qos-policy {
      prefix policy;
      reference "RFC XXXX: YANG Model for QoS";
    }
    import ietf-qos-action {
      prefix action;
      reference "RFC XXXX: YANG Model for QoS";
    }
    import ietf-inet-types {
      prefix inet;
      reference "RFC 6991: Common YANG Data Types";
    }

    organization "IETF RTG (Routing Area) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
```

```
        WG List:  <mailto:rtgwg@ietf.org>
        WG Chair: Chris Bowers
                  <mailto:cbowers@juniper.net>
        WG Chair: Jeff Tantsura
                  <mailto:jefftant.ietf@gmail.com>
        Editor:   Aseem Choudhary
                  <mailto:asechoud@cisco.com>
        Editor:   Mahesh Jethanandani
                  <mailto:mjethanandani@gmail.com>
        Editor:   Norm Strahle
                  <mailto:nstrahle@juniper.net>";
    description
      "This module contains a collection of YANG definitions for
       configuring diffserv specification implementations.
       Copyright (c) 2019 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.
       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).
       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    revision 2019-03-13 {
      description
        "Latest revision of diffserv based classifier";
      reference "RFC XXXX: YANG Model for QoS";
    }

    feature diffserv-queue-inline-support {
      description
        "Queue inline support in diffserv policy";
    }
    feature diffserv-scheduler-inline-support {
      description
        "scheduler inline support in diffserv policy";
    }
    identity diffserv-policy-type {
      base policy:policy-type;
      description
        "This defines ip policy-type";
    }
    identity ipv4-diffserv-policy-type {
      base policy:policy-type;
      description
        "This defines ipv4 policy-type";
```

```
    }
    identity ipv6-diffserv-policy-type {
      base policy:policy-type;
      description
        "This defines ipv6 policy-type";
    }

    identity dscp {
      base classifier:filter-type;
      description
        "Differentiated services code point filter-type";
    }
    identity source-ipv4-address {
      base classifier:filter-type;
      description
        "source ipv4 address filter-type";
    }
    identity destination-ipv4-address {
      base classifier:filter-type;
      description
        "destination ipv4 address filter-type";
    }
    identity source-ipv6-address {
      base classifier:filter-type;
      description
        "source ipv6 address filter-type";
    }
    identity destination-ipv6-address {
      base classifier:filter-type;
      description
        "destination ipv6 address filter-type";
    }
    identity source-port {
      base classifier:filter-type;
      description
        "source port filter-type";
    }
    identity destination-port {
      base classifier:filter-type;
      description
        "destination port filter-type";
    }
    identity protocol {
      base classifier:filter-type;
      description
        "protocol type filter-type";
    }
    identity traffic-group-name {
```

```
      base classifier:filter-type;
      description
        "traffic-group filter type";
    }

    identity meter-type {
      description
        "This base identity type defines meter types";
    }
    identity one-rate-two-color-meter-type {
      base meter-type;
      description
        "one rate two color meter type";
    }
    identity one-rate-tri-color-meter-type {
      base meter-type;
      description
        "one rate three color meter type";
    }
    identity two-rate-tri-color-meter-type {
      base meter-type;
      description
        "two rate three color meter action type";
    }
    grouping dscp-cfg {
      list dscp-cfg {
        key "dscp-min dscp-max";
        description
          "list of dscp ranges";
        leaf dscp-min {
          type inet:dscp;
          description
            "Minimum value of dscp min-max range";
        }
        leaf dscp-max {
          type inet:dscp;
          description
            "maximum value of dscp min-max range";
        }
      }
      description
        "Filter grouping containing list of dscp ranges";
    }
    grouping source-ipv4-address-cfg {
      list source-ipv4-address-cfg {
        key "source-ipv4-addr";
        description
          "list of source ipv4 address";
```

```
        leaf source-ipv4-addr {
          type inet:ipv4-prefix;
          description
            "source ipv4 prefix";
        }
      }
      description
        "Filter grouping containing list of source ipv4 addresses";
    }
    grouping destination-ipv4-address-cfg {
      list destination-ipv4-address-cfg {
        key "destination-ipv4-addr";
        description
          "list of destination ipv4 address";
        leaf destination-ipv4-addr {
          type inet:ipv4-prefix;
          description
            "destination ipv4 prefix";
        }
      }
      description
        "Filter grouping containing list of destination ipv4 address";
    }
    grouping source-ipv6-address-cfg {
      list source-ipv6-address-cfg {
        key "source-ipv6-addr";
        description
          "list of source ipv6 address";
        leaf source-ipv6-addr {
          type inet:ipv6-prefix;
          description
            "source ipv6 prefix";
        }
      }
      description
        "Filter grouping containing list of source ipv6 addresses";
    }
    grouping destination-ipv6-address-cfg {
      list destination-ipv6-address-cfg {
        key "destination-ipv6-addr";
        description
          "list of destination ipv4 or ipv6 address";
        leaf destination-ipv6-addr {
          type inet:ipv6-prefix;
          description
            "destination ipv6 prefix";
        }
      }
```

```
      description
        "Filter grouping containing list of destination ipv6 address";
    }
    grouping source-port-cfg {
      list source-port-cfg {
        key "source-port-min source-port-max";
        description
          "list of ranges of source port";
        leaf source-port-min {
          type inet:port-number;
          description
            "minimum value of source port range";
        }
        leaf source-port-max {
          type inet:port-number;
          description
            "maximum value of source port range";
        }
      }
      description
        "Filter grouping containing list of source port ranges";
    }
    grouping destination-port-cfg {
      list destination-port-cfg {
        key "destination-port-min destination-port-max";
        description
          "list of ranges of destination port";
        leaf destination-port-min {
          type inet:port-number;
          description
            "minimum value of destination port range";
        }
        leaf destination-port-max {
          type inet:port-number;
          description
            "maximum value of destination port range";
        }
      }
      description
        "Filter grouping containing list of destination port ranges";
    }
    grouping protocol-cfg {
      list protocol-cfg {
        key "protocol-min protocol-max";
        description
          "list of ranges of protocol values";
        leaf protocol-min {
          type uint8 {
```

```
            range "0..255";
          }
          description
            "minimum value of protocol range";
        }
        leaf protocol-max {
          type uint8 {
            range "0..255";
          }
          description
            "maximum value of protocol range";
        }
      }
      description
        "Filter grouping containing list of Protocol ranges";
    }
    grouping traffic-group-cfg {
      container traffic-group-cfg {
        leaf traffic-group-name {
          type string ;
          description
            "This leaf defines name of the traffic group referenced";
        }
      description
        "traffic group container";
      }
      description
        "traffic group grouping";
    }

    augment "/classifier:classifiers/classifier:classifier-entry" +
             "/classifier:filter-entry" {
      choice filter-param {
        description
          "Choice of filter types";
        case dscp {
          uses dscp-cfg;
          description
            "Filter containing list of dscp ranges";
        }
        case source-ipv4-address {
          uses source-ipv4-address-cfg;
          description
            "Filter containing list of source ipv4 addresses";
        }
        case destination-ipv4-address {
          uses destination-ipv4-address-cfg;
          description
```

```
            "Filter containing list of destination ipv4 address";
        }
        case source-ipv6-address {
          uses source-ipv6-address-cfg;
          description
            "Filter containing list of source ipv6 addresses";
        }
        case destination-ipv6-address {
          uses destination-ipv6-address-cfg;
          description
            "Filter containing list of destination ipv6 address";
        }
        case source-port {
          uses source-port-cfg;
          description
            "Filter containing list of source-port ranges";
        }
        case destination-port {
          uses destination-port-cfg;
          description
            "Filter containing list of destination-port ranges";
        }
        case protocol {
          uses protocol-cfg;
          description
            "Filter Type Protocol";
        }
        case traffic-group {
          uses traffic-group-cfg;
          description
            "Filter Type traffic-group";
        }
      }
      description
        "augments diffserv filters to qos classifier";
    }
    augment "/policy:policies/policy:policy-entry" +
            "/policy:classifier-entry/policy:filter-entry" {
      when "../../policy:policy-type =
                    'diffserv:ipv4-diffserv-policy-type' or
            ../../policy:policy-type =
                    'diffserv:ipv6-diffserv-policy-type' or
            ../../policy:policy-type =
                    'diffserv:diffserv-policy-type'" {
        description
          "Filters can be augmented if policy type is
          ipv4, ipv6 or default diffserv policy types ";
      }
```

```
      description
        "Augments Diffserv Classifier with common filter types";
      choice filter-params {
        description
          "Choice of action types";
        case dscp {
          uses dscp-cfg;
          description
            "Filter containing list of dscp ranges";
        }
        case source-ipv4-address {
          when "../../policy:policy-type !=
                              'diffserv:ipv6-diffserv-policy-type'" {
              description
                 "If policy type is v6, this filter cannot be used.";
          }
          uses source-ipv4-address-cfg;
          description
            "Filter containing list of source ipv4 addresses";
        }
        case destination-ipv4-address {
          when "../../policy:policy-type !=
                              'diffserv:ipv6-diffserv-policy-type'" {
              description
                 "If policy type is v6, this filter cannot be used.";
          }
          uses destination-ipv4-address-cfg;
          description
            "Filter containing list of destination ipv4 address";
        }
        case source-ipv6-address {
          when "../../policy:policy-type !=
                              'diffserv:ipv4-diffserv-policy-type'" {
              description
                 "If policy type is v4, this filter cannot be used.";
          }
          uses source-ipv6-address-cfg;
          description
            "Filter containing list of source ipv6 addresses";
        }
        case destination-ipv6-address {
          when "../../policy:policy-type !=
                              'diffserv:ipv4-diffserv-policy-type'" {
              description
                 "If policy type is v4, this filter cannot be used.";
          }
          uses destination-ipv6-address-cfg;
          description
```

```
            "Filter containing list of destination ipv6 address";
        }
        case source-port {
          uses source-port-cfg;
          description
            "Filter containing list of source-port ranges";
        }
        case destination-port {
          uses destination-port-cfg;
          description
            "Filter containing list of destination-port ranges";
        }
        case protocol {
          uses protocol-cfg;
          description
            "Filter Type Protocol";
        }
        case traffic-group {
          uses traffic-group-cfg;
          description
            "Filter Type traffic-group";
        }
      }
    }
    augment "/policy:policies/policy:policy-entry" +
            "/policy:classifier-entry" +
            "/policy:classifier-action-entry-cfg" +
            "/policy:action-cfg-params" {
      when "../../policy:policy-type =
                      'diffserv:ipv4-diffserv-policy-type' or
            ../../policy:policy-type =
                      'diffserv:ipv6-diffserv-policy-type' or
            ../../policy:policy-type =
                      'diffserv:diffserv-policy-type' " {
        description
          "Actions can be augmented if policy type is ipv4,
           ipv6 or default diffserv policy types ";
      }
      description
        "Augments Diffserv Policy with action configuration";
      case dscp-marking {
        uses action:dscp-marking;
      }
      case meter-inline {
        if-feature action:meter-inline-feature;
        uses action:meter;
      }
      case meter-reference {
```

```
        if-feature action:meter-reference-feature;
        uses action:meter-reference;
      }
      case child-policy {
        if-feature action:child-policy-feature;
        uses action:child-policy;
      }
      case count {
        if-feature action:count-feature;
        uses action:count;
      }
      case named-count {
        if-feature action:named-counter-feature;
        uses action:named-counter;
      }
      case queue-inline {
        if-feature diffserv-queue-inline-support;
        uses action:queue;
      }
      case scheduler-inline {
        if-feature diffserv-scheduler-inline-support;
        uses action:scheduler;
      }
    }
  }
  <CODE ENDS>
```

6.6. IETF-QUEUE-POLICY

```
  <CODE BEGINS>file "ietf-queue-policy@2019-03-13.yang"
  module ietf-queue-policy {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-queue-policy";
    prefix queue-policy;

    import ietf-qos-policy {
      prefix policy;
      reference "RFC XXXX: YANG Model for QoS";
    }
    import ietf-qos-action {
      prefix action;
      reference "RFC XXXX: YANG Model for QoS";
    }
    import ietf-diffserv {
      prefix diffserv;
      reference "RFC XXXX: YANG Model for QoS";
    }
```

```
     organization "IETF RTG (Routing Area) Working Group";
     contact
       "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
        WG List:  <mailto:rtgwg@ietf.org>
        WG Chair: Chris Bowers
                  <mailto:cbowers@juniper.net>
        WG Chair: Jeff Tantsura
                  <mailto:jefftant.ietf@gmail.com>
        Editor:   Aseem Choudhary
                  <mailto:asechoud@cisco.com>
        Editor:   Mahesh Jethanandani
                  <mailto:mjethanandani@gmail.com>
        Editor:   Norm Strahle
                  <mailto:nstrahle@juniper.net>";
     description
       "This module contains a collection of YANG definitions for
        configuring diffserv specification implementations.
        Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).
        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     revision 2019-03-13 {
       description
         "Latest revision of queuing policy module";
       reference "RFC XXXX: YANG Model for QoS";
     }

     feature queue-policy-support {
       description
         " This feature allows queue policy configuration
           as a separate policy type support.";
     }

     feature queue-inline-support {
       description
         "Queue inline support in Queue policy";
     }

     feature queue-template-support {
       description
         "Queue template support in Queue policy";
```

```
    }

    identity queue-policy-type {
      base policy:policy-type;
      description
        "This defines queue policy-type";
    }

    augment "/policy:policies/policy:policy-entry" +
            "/policy:classifier-entry/policy:filter-entry" {
      when "../../policy:policy-type =
                          'queue-policy:queue-policy-type'" {
        description
          "If policy type is v6, this filter cannot be used.";
      }
      if-feature queue-policy-support;
      choice filter-params {
        description
          "Choice of action types";
        case traffic-group-name {
          uses diffserv:traffic-group-cfg;
          description
            "traffic group name";
        }
      }
      description
        "Augments Queue policy Classifier with common filter types";
    }

    identity queue-template-name {
      base policy:action-type;
      description
        "queue template name";
    }

    grouping queue-template-reference {
      container queue-template-reference-cfg {
        leaf queue-template-name {
          type string ;
          mandatory true;
          description
            "This leaf defines name of the queue template referenced";
        }
      description
        "queue template reference";
      }
      description
        "queue template reference grouping";
```

```
    }

    container queue-template {
      if-feature queue-policy-support;
      description
        "Queue template";
      leaf name {
        type string;
        description
          "A unique name identifying this queue template";
      }
      uses action:queue;
    }


    augment "/policy:policies/policy:policy-entry" +
            "/policy:classifier-entry" +
            "/policy:classifier-action-entry-cfg" +
            "/policy:action-cfg-params" {
    when "../../policy:policy-type =
                         'queue-policy:queue-policy-type'" {
        description
          "queue policy actions.";
      }
      if-feature queue-policy-support;
        case queue-template-name {
          if-feature queue-template-support;
          uses queue-template-reference;
        }
        case queue-inline {
          if-feature queue-inline-support;
          uses action:queue;
        }
      description
        "augments queue template reference to queue policy";
    }
  }
  <CODE ENDS>
```

6.7. IETF-SCHEDULER-POLICY

```
  <CODE BEGINS>file "ietf-scheduler-policy@2019-03-13.yang"
  module ietf-scheduler-policy {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-scheduler-policy";
    prefix scheduler-policy;

    import ietf-qos-classifier {
```

```
      prefix classifier;
      reference "RFC XXXX: YANG Model for QoS";
    }
    import ietf-qos-policy {
      prefix policy;
      reference "RFC XXXX: YANG Model for QoS";
    }
    import ietf-qos-action {
      prefix action;
      reference "RFC XXXX: YANG Model for QoS";
    }

    organization "IETF RTG (Routing Area) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
       WG List:  <mailto:rtgwg@ietf.org>
       WG Chair: Chris Bowers
                 <mailto:cbowers@juniper.net>
       WG Chair: Jeff Tantsura
                 <mailto:jefftant.ietf@gmail.com>
       Editor:   Norm Strahle
                 <mailto:nstrahle@juniper.net>
       Editor:   Aseem Choudhary
                 <mailto:asechoud@cisco.com>";
    description
      "This module contains a collection of YANG definitions for
       configuring diffserv specification implementations.
       Copyright (c) 2019 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.
       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).
       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    revision 2019-03-13 {
      description
        "Latest revision of scheduler policy module";
      reference "RFC XXXX: YANG Model for QoS";
    }
    feature scheduler-policy-support {
      description
        " This feature allows sheduler policy configuration
          as a separate policy type support.";
    }
```

```
identity scheduler-policy-type {
  base policy:policy-type;
  description
    "This defines scheduler policy-type";
}

identity filter-match-all {
  base classifier:filter-type;
  description
    "Traffic-group filter type";
}

grouping filter-match-all-cfg {
  container match-all-cfg {
    leaf match-all-action {
      type empty;
      description
        "match all packets";
    }
    description
      "the match-all action";
  }
  description
    "the match-all filter grouping";
}

augment "/policy:policies/policy:policy-entry" +
        "/policy:classifier-entry/policy:filter-entry" {
      when "../../policy:policy-type =
            'scheduler-policy:scheduler-policy-type'" {
          description
             "Only when policy type is scheduler-policy";
      }
  choice filter-params {
    description
      "Choice of action types";
    case filter-match-all {
      uses filter-match-all-cfg;
      description
        "filter match-all";
    }
  }
  description
    "Augments Queue policy Classifier with common filter types";
}

identity queue-policy-name {
  base policy:action-type;
```

```
      description
        "queue policy name";
    }

    grouping queue-policy-name-cfg {
      container queue-policy-name {
        leaf queue-policy {
          type string ;
          mandatory true;
          description
            "This leaf defines name of the queue-policy";
        }
      description
        "container for queue-policy name";
      }
      description
        "queue-policy name grouping";
    }


    augment "/policy:policies/policy:policy-entry" +
            "/policy:classifier-entry" +
            "/policy:classifier-action-entry-cfg" +
            "/policy:action-cfg-params" {
          when "../../policy:policy-type =
                  'scheduler-policy:scheduler-policy-type'" {
              description
                 "Only when policy type is scheduler-policy";
          }
        case scheduler {
          uses action:scheduler;
        }
        case queue-policy-name {
          uses queue-policy-name-cfg;
        }
      description
        "augments scheduler template reference to scheduler policy";
    }
  }
  <CODE ENDS>
```

7. IANA Considerations

TBD

8. Security Considerations

9. Acknowledgement

The authors wish to thank Ruediger Geib, Fred Baker, Greg Misky, Tom Petch, many others for their helpful comments.

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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, DOI 10.17487/RFC2697, September 1999, <https://www.rfc-editor.org/info/rfc2697>.

[RFC2698] Heinanen, J. and R. Guerin, "A Two Rate Three Color Marker", RFC 2698, DOI 10.17487/RFC2698, September 1999, <https://www.rfc-editor.org/info/rfc2698>.

[RFC3289] Baker, F., Chan, K., and A. Smith, "Management Information Base for the Differentiated Services Architecture", RFC 3289, DOI 10.17487/RFC3289, May 2002, <https://www.rfc-editor.org/info/rfc3289>.

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

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

10.2. Informative References

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <https://www.rfc-editor.org/info/rfc2475>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Appendix A. Company A, Company B and Company C examples

Company A, Company B and Company C Diffserv modules augments all the filter types of the QoS classifier module as well as the QoS policy module that allow it to define marking, metering, min-rate, max-rate actions. Queuing and metering counters are realized by augmenting of the QoS target module.

A.1. Example of Company A Diffserv Model

The following Company A vendor example augments the qos and diffserv model, demonstrating some of the following functionality:

- use of template based classifier definitions

- use of single policy type modelling queue, scheduler policy, and a filter policy. All of these policies either augment the qos policy or the diffserv modules

- use of inline actions in a policy

- flexibility in marking dscp or metadata at ingress and/or egress.

```
module example-compa-diffserv {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:example-compa-diffserv";
  prefix example;
```

```
        import ietf-qos-classifier {
          prefix classifier;
          reference "RFC XXXX: YANG Model for QoS";
        }
        import ietf-qos-policy {
          prefix policy;
          reference "RFC XXXX: YANG Model for QoS";
        }
        import ietf-qos-action {
          prefix action;
          reference "RFC XXXX: YANG Model for QoS";
        }
        import ietf-diffserv {
          prefix diffserv;
          reference "RFC XXXX: YANG Model for QoS";
        }

        organization "Company A";
        contact
          "Editor:   XYZ
                    <mailto:xyz@compa.com>";
        description
          "This module contains a collection of YANG definitions of
           companyA diffserv specification extension.";
           Copyright (c) 2019 IETF Trust and the persons identified as
           authors of the code.  All rights reserved.
           Redistribution and use in source and binary forms, with or
           without modification, is permitted pursuant to, and subject
           to the license terms contained in, the Simplified BSD License
           set forth in Section 4.c of the IETF Trust's Legal Provisions
           Relating to IETF Documents
           (http://trustee.ietf.org/license-info).

           This version of this YANG module is part of RFC XXXX; see
           the RFC itself for full legal notices.";

        revision 2019-03-13 {
          description
            "Initial revision for diffserv actions on network packets";
          reference
            "RFC 6020: YANG - A Data Modeling Language for the
                       Network Configuration Protocol (NETCONF)";
        }

        identity default-policy-type {
          base policy:policy-type;
          description
            "This defines default policy-type";
```

```
        }

        identity qos-group {
          base classifier:filter-type;
          description
            "qos-group filter-type";
        }

        grouping qos-group-cfg {
          list qos-group-cfg {
            key "qos-group-min qos-group-max";
            description
              "list of dscp ranges";
            leaf qos-group-min {
              type uint8;
              description
                "Minimum value of qos-group range";
            }
            leaf qos-group-max {
              type uint8;
              description
                "maximum value of qos-group range";
            }
          }
          description
            "Filter containing list of qos-group ranges";
        }

        grouping wred-threshold {
          container wred-min-thresh {
            uses action:threshold;
            description
              "Minimum threshold";
          }
          container wred-max-thresh {
            uses action:threshold;
            description
              "Maximum threshold";
          }
          leaf mark-probability {
            type uint32 {
              range "1..1000";
            }
            description
              "Mark probability";
          }
          description
            "WRED threshold attributes";
```

```
        }

        grouping randomdetect {
          leaf exp-weighting-const {
            type uint32;
            description
              "Exponential weighting constant factor for wred profile";
          }
          uses wred-threshold;
          description
            "Random detect attributes";
        }

        augment "/classifier:classifiers/" +
                "classifier:classifier-entry/" +
                "classifier:filter-entry/diffserv:filter-param" {
          case qos-group {
            uses qos-group-cfg;
            description
              "Filter containing list of qos-group ranges.
               Qos-group represent packet metadata information
               in a device. ";
          }
          description
            "augmentation of classifier filters";
        }
        augment "/policy:policies/policy:policy-entry/" +
                "policy:classifier-entry/" +
                "policy:classifier-action-entry-cfg/" +
                "policy:action-cfg-params" {
          case random-detect {
            uses randomdetect;
          }
          description
            "Augment the actions to policy entry";
        }

        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:one-rate-two-color-meter-type" +
                "/diffserv:one-rate-two-color-meter" +
                "/diffserv:conform-action" +
                "/diffserv:conform-2color-meter-action-params" +
```

```
              "/diffserv:conform-2color-meter-action-val" {

          description
                "augment the one-rate-two-color meter conform
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:one-rate-two-color-meter-type" +
                "/diffserv:one-rate-two-color-meter" +
                "/diffserv:exceed-action" +
                "/diffserv:exceed-2color-meter-action-params" +
                "/diffserv:exceed-2color-meter-action-val" {

          description
                "augment the one-rate-two-color meter exceed
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
```

```
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:one-rate-tri-color-meter-type" +
                "/diffserv:one-rate-tri-color-meter" +
                "/diffserv:conform-action" +
                "/diffserv:conform-3color-meter-action-params" +
                "/diffserv:conform-3color-meter-action-val" {

          description
                "augment the one-rate-tri-color meter conform
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:one-rate-tri-color-meter-type" +
                "/diffserv:one-rate-tri-color-meter" +
                "/diffserv:exceed-action" +
                "/diffserv:exceed-3color-meter-action-params" +
                "/diffserv:exceed-3color-meter-action-val" {

          description
                "augment the one-rate-tri-color meter exceed
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
```

```
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:one-rate-tri-color-meter-type" +
                "/diffserv:one-rate-tri-color-meter" +
                "/diffserv:violate-action" +
                "/diffserv:violate-3color-meter-action-params" +
                "/diffserv:violate-3color-meter-action-val" {
          description
                "augment the one-rate-tri-color meter conform
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }

        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:two-rate-tri-color-meter-type" +
                "/diffserv:two-rate-tri-color-meter" +
                "/diffserv:conform-action" +
                "/diffserv:conform-3color-meter-action-params" +
                "/diffserv:conform-3color-meter-action-val" {

          description
                "augment the one-rate-tri-color meter conform
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
```

```
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:two-rate-tri-color-meter-type" +
                "/diffserv:two-rate-tri-color-meter" +
                "/diffserv:exceed-action" +
                "/diffserv:exceed-3color-meter-action-params" +
                "/diffserv:exceed-3color-meter-action-val" {

          description
                "augment the two-rate-tri-color meter exceed
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:two-rate-tri-color-meter-type" +
                "/diffserv:two-rate-tri-color-meter" +
                "/diffserv:violate-action" +
                "/diffserv:violate-3color-meter-action-params" +
                "/diffserv:violate-3color-meter-action-val" {
          description
                "augment the two-rate-tri-color meter violate
```

```
                with actions";
          case meter-action-drop {
            description
                "meter drop";
                uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:one-rate-two-color-meter-type" +
                "/diffserv:one-rate-two-color-meter" {
          description
                "augment the one-rate-two-color meter with" +
                "color classifiers";
            container conform-color {
              uses classifier:classifier-entry-generic-attr;
              description
                "conform color classifier container";
            }
            container exceed-color {
              uses classifier:classifier-entry-generic-attr;
              description
                "exceed color classifier container";
            }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:one-rate-tri-color-meter-type" +
                "/diffserv:one-rate-tri-color-meter" {
          description
                "augment the one-rate-tri-color meter with" +
                "color classifiers";
            container conform-color {
```

```
            uses classifier:classifier-entry-generic-attr;
            description
              "conform color classifier container";
          }
          container exceed-color {
            uses classifier:classifier-entry-generic-attr;
            description
              "exceed color classifier container";
          }
          container violate-color {
            uses classifier:classifier-entry-generic-attr;
            description
              "violate color classifier container";
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:two-rate-tri-color-meter-type" +
                "/diffserv:two-rate-tri-color-meter" {
          description
                "augment the two-rate-tri-color meter with" +
                "color classifiers";
          container conform-color {
            uses classifier:classifier-entry-generic-attr;
            description
              "conform color classifier container";
          }
          container exceed-color {
            uses classifier:classifier-entry-generic-attr;
            description
              "exceed color classifier container";
          }
          container violate-color {
            uses classifier:classifier-entry-generic-attr;
            description
              "violate color classifier container";
          }
        }
      }
```

A.2. Example of Company B Diffserv Model

The following vendor example augments the qos and diffserv model, demonstrating some of the following functionality:

- use of inline classifier definitions (defined inline in the policy vs referencing an externally defined classifier)

- use of mulitple policy types, e.g. a queue policy, a scheduler policy, and a filter policy. All of these policies either augment the qos policy or the diffserv modules

- use of a queue module, which uses and extends the queue grouping from the ietf-qos-action module

- use of meter templates (v.s. meter inline)

- use of internal meta data for classification and marking

```
module example-compb-diffserv-filter-policy {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:" +
               "example-compb-diffserv-filter-policy";
     prefix compb-filter-policy;

  import ietf-qos-classifier {
    prefix classifier;
    reference "RFC XXXX: YANG Model for QoS";
  }
  import ietf-qos-policy {
    prefix policy;
    reference "RFC XXXX: YANG Model for QoS";
  }
  import ietf-qos-action {
    prefix action;
    reference "RFC XXXX: YANG Model for QoS";
  }
  import ietf-diffserv {
    prefix diffserv;
    reference "RFC XXXX: YANG Model for QoS";
  }

  organization "Company B";
  contact
    "Editor:   XYZ
              <mailto:xyz@compb.com>";

  description
```

```
       "This module contains a collection of YANG definitions for
        configuring diffserv specification implementations.
        Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     revision 2019-03-13 {
       description
         "Latest revision of diffserv policy";
       reference "RFC XXXX";
     }


     /***********************************************
      * Classification types
      ***********************************************/

     identity forwarding-class {
         base classifier:filter-type;
         description
           "Forwarding class filter type";
     }

     identity internal-loss-priority  {
        base classifier:filter-type;
        description
           "Internal loss priority filter type";
     }

     grouping forwarding-class-cfg {
       list forwarding-class-cfg {
         key "forwarding-class";
         description
           "list of forwarding-classes";
         leaf forwarding-class {
           type string;
           description
             "Forwarding class name";
         }
       }
```

```
      description
        "Filter containing list of forwarding classes";
    }

    grouping loss-priority-cfg {
      list loss-priority-cfg {
        key "loss-priority";
        description
          "list of loss-priorities";
        leaf loss-priority {
          type enumeration {
            enum high {
             description "High Loss Priority";
            }
            enum medium-high {
             description "Medium-high Loss Priority";
            }
            enum medium-low {
             description "Medium-low Loss Priority";
            }
            enum low {
             description "Low Loss Priority";
            }
          }
          description
            "Loss-priority";
        }
      }
      description
        "Filter containing list of loss priorities";
    }

    augment "/policy:policies" +
            "/policy:policy-entry" +
            "/policy:classifier-entry" +
            "/policy:filter-entry" +
            "/diffserv:filter-params" {
      case forwarding-class {
          uses forwarding-class-cfg;
        description
          "Filter Type Internal-loss-priority";
      }
      case internal-loss-priority {
          uses loss-priority-cfg;
        description
          "Filter Type Internal-loss-priority";
      }
      description
```

```
        "Augments Diffserv Classifier with vendor" +
        " specific types";
    }

    /***********************************************
     * Actions
     ***********************************************/

    identity mark-fwd-class {
      base policy:action-type;
      description
        "mark forwarding class action type";
    }

    identity mark-loss-priority {
      base policy:action-type;
      description
        "mark loss-priority action type";
    }

    grouping mark-fwd-class {
      container mark-fwd-class-cfg {
        leaf forwarding-class {
          type string;
          description
            "Forwarding class name";
        }
        description
          "mark-fwd-class container";
      }
      description
        "mark-fwd-class grouping";
    }

    grouping mark-loss-priority {
      container mark-loss-priority-cfg {
        leaf loss-priority {
          type enumeration {
            enum high {
             description "High Loss Priority";
            }
            enum medium-high {
             description "Medium-high Loss Priority";
            }
            enum medium-low {
             description "Medium-low Loss Priority";
            }
            enum low {
```

```
            description "Low Loss Priority";
          }
        }
        description
          "Loss-priority";
      }
      description
        "mark-loss-priority container";
    }
    description
      "mark-loss-priority grouping";
  }

  identity exceed-2color-meter-action-drop {
    base action:exceed-2color-meter-action-type;
    description
      "drop action type in a meter";
  }

  identity meter-action-mark-fwd-class {
    base action:exceed-2color-meter-action-type;
    description
      "mark forwarding class action type";
  }

  identity meter-action-mark-loss-priority {
    base action:exceed-2color-meter-action-type;
    description
      "mark loss-priority action type";
  }

  identity violate-3color-meter-action-drop {
    base action:violate-3color-meter-action-type;
    description
      "drop action type in a meter";
  }

  augment "/policy:policies/policy:policy-entry/" +
            "policy:classifier-entry/" +
            "policy:classifier-action-entry-cfg/" +
            "policy:action-cfg-params" {
    case mark-fwd-class {
      uses mark-fwd-class;
      description
        "Mark forwarding class in the packet";
    }
    case mark-loss-priority {
        uses mark-loss-priority;
```

```
          description
            "Mark loss priority in the packet";
        }
        case discard {
            uses action:discard;
          description
            "Discard action";
        }
        description
          "Augments common diffserv policy actions";
        }


      augment "/action:meter-template" +
              "/action:meter-entry" +
              "/action:meter-type" +
              "/action:one-rate-tri-color-meter-type" +
              "/action:one-rate-tri-color-meter" {
         leaf one-rate-color-aware {
          type boolean;
          description
            "This defines if the meter is color-aware";
        }
      }
      augment "/action:meter-template" +
              "/action:meter-entry" +
              "/action:meter-type" +
              "/action:two-rate-tri-color-meter-type" +
              "/action:two-rate-tri-color-meter" {
         leaf two-rate-color-aware {
          type boolean;
          description
            "This defines if the meter is color-aware";
        }
      }

      /* example of augmenting a meter template with a
      /* vendor specific action */
      augment "/action:meter-template" +
              "/action:meter-entry" +
              "/action:meter-type" +
              "/action:one-rate-two-color-meter-type" +
              "/action:one-rate-two-color-meter" +
              "/action:exceed-action" +
              "/action:exceed-2color-meter-action-params" +
              "/action:exceed-2color-meter-action-val" {

        case exceed-2color-meter-action-drop {
```

```
        description
            "meter drop";
             uses action:drop;
      }
      case meter-action-mark-fwd-class {
        uses mark-fwd-class;
        description
          "Mark forwarding class in the packet";
      }
      case meter-action-mark-loss-priority {
          uses mark-loss-priority;
        description
          "Mark loss priority in the packet";
      }
    }

    augment "/action:meter-template" +
            "/action:meter-entry" +
            "/action:meter-type" +
            "/action:two-rate-tri-color-meter-type" +
            "/action:two-rate-tri-color-meter" +
            "/action:violate-action" +
            "/action:violate-3color-meter-action-params" +
            "/action:violate-3color-meter-action-val" {
      case exceed-3color-meter-action-drop {
        description
            "meter drop";
        uses action:drop;
      }

      description
        "Augment the actions to the two-color meter";
    }

    augment "/action:meter-template" +
            "/action:meter-entry" +
            "/action:meter-type" +
            "/action:one-rate-tri-color-meter-type" +
            "/action:one-rate-tri-color-meter" +
            "/action:violate-action" +
            "/action:violate-3color-meter-action-params" +
            "/action:violate-3color-meter-action-val" {
      case exceed-3color-meter-action-drop {
        description
            "meter drop";
        uses action:drop;
      }
```

```
      description
        "Augment the actions to basic meter";
    }

  }
  module example-compb-queue-policy {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:example-compb-queue-policy";
    prefix queue-plcy;

    import ietf-qos-classifier {
      prefix classifier;
      reference "RFC XXXX: YANG Model for QoS";
    }
    import ietf-qos-policy {
      prefix policy;
      reference "RFC XXXX: YANG Model for QoS";
    }

    organization "Company B";
    contact
      "Editor:   XYZ
                <mailto:xyz@compb.com>";

    description
      "This module defines a queue policy. The classification
       is based on aforwarding class, and the actions are queues.
       Copyright (c) 2019 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.
       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).
       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    revision 2019-03-13 {
      description
        "Latest revision of diffserv policy";
      reference "RFC XXXX";
    }

    identity forwarding-class {
      base classifier:filter-type;
      description
         "Forwarding class filter type";
```

```
    }

    grouping forwarding-class-cfg {
      leaf forwarding-class-cfg {
        type string;
        description
          "forwarding-class name";
      }
      description
        "Forwarding class filter";
    }

    augment "/policy:policies" +
            "/policy:policy-entry" +
            "/policy:classifier-entry" +
            "/policy:filter-entry" {
      /* Does NOT support "logical-not" of forwarding class.
         Use "must"? */
      choice filter-params {
        description
          "Choice of filters";
        case forwarding-class-cfg {
          uses forwarding-class-cfg;
          description
            "Filter Type Internal-loss-priority";
        }
      }
      description
        "Augments Diffserv Classifier with fwd class filter";
    }

    identity compb-queue {
      base policy:action-type;
      description
        "compb-queue action type";
    }

    grouping compb-queue-name {
      container queue-name {
        leaf name {
          type string;
          description
            "Queue class name";
        }
        description
          "compb queue container";
      }
      description
```

```
        "compb-queue grouping";
    }

    augment "/policy:policies" +
            "/policy:policy-entry" +
            "/policy:classifier-entry" +
            "/policy:classifier-action-entry-cfg" {
      choice action-cfg-params {
        description
          "Choice of action types";
        case compb-queue {
          uses compb-queue-name;
        }
      }
      description
        "Augment the queue actions to queue policy entry";
    }
  }

  module example-compb-queue {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-compb-queue";
    prefix compb-queue;

    import ietf-qos-action {
      prefix action;
      reference "RFC XXXX: YANG Model for QoS";
    }

    organization "Company B";
    contact
      "Editor:   XYZ
                <mailto:xyz@compb.com>";

    description
      "This module describes a compb queue module. This is a
       template for a queue within a queue policy, referenced
       by name.

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    revision 2019-03-13 {
      description
        "Latest revision of diffserv based classifier";
      reference "RFC XXXX";
    }
```

```
     container compb-queue {
       description
         "Queue used in compb architecture";
         leaf name {
             type string;
             description
               "A unique name identifying this queue";
         }
         uses action:queue;
         container excess-rate {
           choice excess-rate-type {
              case percent {
                leaf excess-rate-percent {
                  type uint32 {
                    range "1..100";
                  }
                  description
                    "excess-rate-percent";
                }
              }
              case proportion {
                leaf excess-rate-poroportion {
                  type uint32 {
                    range "1..1000";
                  }
                  description
                    "excess-rate-poroportion";
                }
              }
              description
                "Choice of excess-rate type";
           }
           description
               "Excess rate value";
         }
         leaf excess-priority {
           type enumeration {
             enum high {
              description "High Loss Priority";
             }
             enum medium-high {
              description "Medium-high Loss Priority";
             }
             enum medium-low {
              description "Medium-low Loss Priority";
             }
             enum low {
              description "Low Loss Priority";
```

```
            }
            enum none {
             description "No excess priority";
            }
          }
          description
            "Priority of excess (above guaranted rate) traffic";
        }
        container buffer-size {
          choice buffer-size-type {
             case percent {
               leaf buffer-size-percent {
                 type uint32 {
                   range "1..100";
                 }
                 description
                   "buffer-size-percent";
               }
             }
             case temporal {
               leaf buffer-size-temporal {
                 type uint64;
                 units "microsecond";
                 description
                   "buffer-size-temporal";
               }
             }
             case remainder {
               leaf buffer-size-remainder {
                 type empty;
                 description
                   "use remaining of buffer";
               }
             }
             description
               "Choice of buffer size type";
          }
          description
              "Buffer size value";
        }
      }

      augment
        "/compb-queue" +
        "/queue-cfg" +
        "/algorithmic-drop-cfg" +
        "/drop-algorithm" {
        case random-detect {
```

```
        list drop-profile-list {
          key "priority";
          description
            "map of priorities to drop-algorithms";
          leaf priority {
            type enumeration {
              enum any {
                description "Any priority mapped here";
              }
              enum high {
                description "High Priority Packet";
              }
              enum medium-high {
                description "Medium-high Priority Packet";
              }
              enum medium-low {
                description "Medium-low Priority Packet";
              }
              enum low {
                description "Low Priority Packet";
              }
            }
            description
              "Priority of guaranteed traffic";
          }
          leaf drop-profile {
            type string;
            description
              "drop profile to use for this priority";
          }
        }
      }
      description
        "compb random detect drop algorithm config";
    }
  }

  module example-compb-scheduler-policy {
       yang-version 1.1;
       namespace "urn:ietf:params:xml:ns:yang:" +
                 "example-compb-scheduler-policy";
       prefix scheduler-plcy;

    import ietf-qos-action {
      prefix action;
      reference "RFC XXXX: YANG Model for QoS";
    }
```

```
     import ietf-qos-policy {
       prefix policy;
       reference "RFC XXXX: YANG Model for QoS";
     }

     organization "Company B";
     contact
       "Editor:   XYZ
                 <mailto:xyz@compb.com>";

      description
        "This module defines a scheduler policy. The classification
         is based on classifier-any, and the action is a scheduler.";

      revision 2019-03-13 {
        description
          "Latest revision of diffserv policy";
        reference "RFC XXXX";
      }

      identity queue-policy {
        base policy:action-type;
        description
          "forwarding-class-queue action type";
      }

     grouping queue-policy-name {
       container compb-queue-policy-name {
         leaf name {
           type string;
           description
             "Queue policy name";
         }
         description
           "compb-queue-policy container";
       }
       description
         "compb-queue policy grouping";
     }

     augment "/policy:policies" +
             "/policy:policy-entry" +
             "/policy:classifier-entry" +
             "/policy:classifier-action-entry-cfg" {
       choice action-cfg-params {
         case schedular {
           uses action:schedular;
         }
```

```
        case queue-policy {
          uses queue-policy-name;
        }
        description
          "Augment the scheduler policy with a queue policy";
      }
    }
  }
```

A.3. Example of Company C Diffserv Model

Company C vendor augmentation is based on Ericsson's implementation differentiated QoS. This implementation first sorts traffic based on a classifier, which can sort traffic into one or more traffic forwarding classes. Then, a policer or meter policy references the classifier and its traffic forwarding classes to specify different service levels for each traffic forwarding class.

Because each classifier sorts traffic into one or more traffic forwarding classes, this type of classifier does not align with ietf-qos-classifier.yang, which defines one traffic forwarding class per classifier. Additionally, Company C's policing and metering policies relies on the classifier's pre-defined traffic forwarding classes to provide differentiated services, rather than redefining the patterns within a policing or metering policy, as is defined in ietf-diffserv.yang.

Due to these differences, even though Company C uses all the building blocks of classifier and policy, Company C's augmentation does not use ietf-diffserv.yang to provide differentiated service levels. Instead, Company C's augmentation uses the basic building blocks, ietf-qos-policy.yang to provide differentiated services.

```
module example-compc-qos-policy {
  yang-version 1.1;
  namespace "urn:example-compc-qos-policy";
  prefix "compcqos";

  import ietf-qos-policy {
    prefix "pol";
    reference "RFC XXXX: YANG Model for QoS";
  }

  import ietf-qos-action {
    prefix "action";
    reference "RFC XXXX: YANG Model for QoS";
  }
```

```
    organization "";
    contact "";
    description "";

    revision 2019-03-13 {
      description "";
      reference "";
    }

    /* identities */

    identity compc-qos-policy {
      base pol:policy-type;
    }

    identity mdrr-queuing-policy {
      base compc-qos-policy;
    }

    identity pwfq-queuing-policy {
      base compc-qos-policy;
    }

    identity policing-policy {
      base compc-qos-policy;
    }

    identity metering-policy {
      base compc-qos-policy;
    }

    identity forwarding-policy {
      base compc-qos-policy;
    }

    identity overhead-profile-policy {
      base compc-qos-policy;
    }

    identity resource-profile-policy {
      base compc-qos-policy;
    }

    identity protocol-rate-limit-policy {
      base compc-qos-policy;
    }

    identity compc-qos-action {
```

```
      base pol:action-type;
    }

    /* groupings */

    grouping redirect-action-grp {
      container redirect {
        /* Redirect options */
      }
    }

    /* deviations */

    deviation "/pol:policies/pol:policy-entry" {
      deviate add {
        must "pol:type = compc-qos-policy" {
          description
            "Only policy types drived from compc-qos-policy " +
            "are supported";
        }
      }
    }

    deviation "/pol:policies/pol:policy-entry/pol:classifier-entry" {
      deviate add {
        must "../per-class-action = 'true'" {
          description
            "Only policies with per-class actions have classifiers";
        }
        must "((../sub-type != 'mdrr-queuing-policy') and " +
             " (../sub-type != 'pwfq-queuing-policy')) or " +
             "(((../sub-type = 'mdrr-queuing-policy') or " +
             "  (../sub-type = 'pwfq-queueing-policy')) and " +
             " ((classifier-entry-name = '0') or " +
             "  (classifier-entry-name = '1') or " +
             "  (classifier-entry-name = '2') or " +
             "  (classifier-entry-name = '3') or " +
             "  (classifier-entry-name = '4') or " +
             "  (classifier-entry-name = '5') or " +
             "  (classifier-entry-name = '6') or " +
             "  (classifier-entry-name = '7') or " +
             "  (classifier-entry-name = '8')))" {
          description
            "MDRR queuing policy's or PWFQ queuing policy's " +
            "classifier-entry-name is limited to the listed values";
        }
      }
    }
```

```
    deviation "/pol:policies/pol:policy-entry/pol:classifier-entry" +
              "/pol:classifier-action-entry-cfg" {
      deviate add {
        max-elements 1;
        must "action-type = 'compc-qos-action'" {
          description
            "Only compc-qos-action is allowed";
        }
      }
    }

    /* augments */

    augment "/pol:policies/pol:policy-entry" {
      when "pol:type = 'compc-qos-policy')" {
        description
          "Additional nodes only for diffserv-policy";
      }
      leaf sub-type {
        type identityref {
          base compc-qos-policy;
        }
        mandatory true;
        /* The value of this leaf must not change once configured */
      }
      leaf per-class-action {
        mandatory true;
        type boolean;
        must "(((. = 'true') and " +
             "  ((../sub-type = 'policing-policy') or " +
             "   (../sub-type = 'metering-policy') or " +
             "   (../sub-type = 'mdrr-queuing-policy') or " +
             "   (../sub-type = 'pwfq-queuing-policy') or " +
             "   (../sub-type = 'forwarding-policy'))) or " +
             " ((. = 'false') and " +
             "  ((../sub-type = 'overhead-profile-policy') or " +
             "   (../sub-type = 'resource-profile-policy') or " +
             "   (../sub-type = 'protocol-rate-limit-policy')))" {
          description
            "Only certain policies have per-class action";
        }
      }
      container traffic-classifier {
        presence true;
        when "../sub-type = 'policing-policy' or " +
             "../sub-type = 'metering-policy' or " +
             "../sub-type = 'forwarding-policy'" {
          description
```

```
            "A classifier for policing-policy or metering-policy";
          }
          leaf name {
            type string;
            mandatory true;
            description
              "Traffic classifier name";
          }
          leaf type {
            type enumeration {
              enum 'internal-dscp-only-classifier' {
                value 0;
                description
                  "Classify traffic based on (internal) dscp only";
              }
              enum 'ipv4-header-based-classifier' {
                value 1;
                description
                  "Classify traffic based on IPv4 packet header fields";
              }
              enum 'ipv6-header-based-classifier' {
                value 2;
                description
                  "Classify traffic based on IPv6 packet header fields";
              }
            }
            mandatory true;
            description
              "Traffic classifier type";
          }
        }
        container traffic-queue {
          when "(../sub-type = 'mdrr-queuing-policy') or " +
               "(../sub-type = 'pwfq-queuing-policy')" {
            description
              "Queuing policy properties";
          }
          leaf queue-map {
            type string;
            description
              "Traffic queue map for queuing policy";
          }
        }
        container overhead-profile {
          when "../sub-type = 'overhead-profile-policy'" {
            description
              "Overhead profile policy properties";
          }
```

```
      }
      container resource-profile {
        when "../sub-type = 'resource-profile-policy'" {
          description
            "Resource profile policy properties";
        }
      }
      container protocol-rate-limit {
        when "../sub-type = 'protocol-rate-limit-policy'" {
          description
            "Protocol rate limit policy properties";
        }
      }
    }

    augment "/pol:policies/pol:policy-entry/pol:classifier-entry" +
            "/pol:classifier-action-entry-cfg/pol:action-cfg-params" {
      when "../../../pol:type = 'compc-qos-policy')" {
        description
          "Configurations for a classifier-policy-type policy";
      }
      case metering-or-policing-policy {
        when "../../../sub-type = 'policing-policy' or "
           + "../../../sub-type = 'metering-policy'" {
        }
        container dscp-marking {
          uses action:dscp-marking;
        }
        container precedence-marking {
          uses action:dscp-marking;
        }
        container priority-marking {
          uses action:priority;
        }
        container rate-limiting {
          uses action:one-rate-two-color-meter;
        }
      }
      case mdrr-queuing-policy {
        when "../../../sub-type = 'mdrr-queuing-policy'" {
          description
            "MDRR queue handling properties for the traffic " +
            "classified into current queue";
        }
        leaf mdrr-queue-weight {
          type uint8 {
            range "20..100";
          }
```

```
          units percentage;
        }
      }
      case pwfq-queuing-policy {
        when "../../../sub-type = 'pwfq-queuing-policy'" {
          description
            "PWFQ queue handling properties for traffic " +
            "classified into current queue";
        }
        leaf pwfq-queue-weight {
          type uint8 {
            range "20..100";
          }
          units percentage;
        }
        leaf pwfq-queue-priority {
          type uint8;
        }
        leaf pwfq-queue-rate {
          type uint8;
        }
      }
      case forwarding-policy {
        when "../../../sub-type = 'forwarding-policy'" {
          description
            "Forward policy handling properties for traffic " +
            "in this classifier";
        }
        uses redirect-action-grp;
      }
      description
        "Add the classify action configuration";
    }

  }
```

Authors' Addresses

Aseem Choudhary
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
US

Email: asechoud@cisco.com

Mahesh Jethanandani
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
US

Email: mjethanandani@gmail.com

Norm Strahle
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089
US

Email: nstrahle@juniper.net

Ebben Aries
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089
US

Email: exa@juniper.net

Ing-Wher Chen
Jabil

Email: ing-wher_chen@jabil.com

Routing Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

F. Baker
Cisco Systems
C. Bowers
Juniper Networks
J. Linkova
Google

October 31, 2016

Enterprise Multihoming using Provider-Assigned Addresses without Network Prefix Translation: Requirements and Solution
draft-bowbakova-rtgwg-enterprise-pa-multihoming-01

Abstract

Connecting an enterprise site to multiple ISPs using provider-assigned addresses is difficult without the use of some form of Network Address Translation (NAT). Much has been written on this topic over the last 10 to 15 years, but it still remains a problem without a clearly defined or widely implemented solution. Any multihoming solution without NAT requires hosts at the site to have addresses from each ISP and to select the egress ISP by selecting a source address for outgoing packets. It also requires routers at the site to take into account those source addresses when forwarding packets out towards the ISPs.

This document attempts to define a complete solution to this problem. It covers the behavior of routers to forward traffic taking into account source address, and it covers the behavior of host to select appropriate source addresses. It also covers any possible role that routers might play in providing information to hosts to help them select appropriate source addresses. In the process of exploring potential solutions, this documents also makes explicit requirements for how the solution would be expected to behave from the perspective of an enterprise site network administrator .

Status of This Memo

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

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

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

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

This Internet-Draft will expire on May 4, 2017.

Table of Contents

1. Introduction

Site multihoming, the connection of a subscriber network to multiple upstream networks using redundant uplinks, is a common enterprise architecture for improving the reliability of its Internet connectivity. If the site uses provider-independent (PI) addresses, all traffic originating from the enterprise can use source addresses from the PI address space. Site multihoming with PI addresses is commonly used with both IPv4 and IPv6, and does not present any new technical challenges.

It may be desirable for an enterprise site to connect to multiple ISPs using provider-assigned (PA) addresses, instead of PI addresses. Multihoming with provider-assigned addresses is typically less expensive for the enterprise relative to using provider-independent

addresses. PA multihoming is also a practice that should be facilitated and encouraged because it does not add to the size of the Internet routing table, whereas PI multihoming does. Note that PA is also used to mean "provider-aggregatable". In this document we assume that provider-assigned addresses are always provider-aggregatable.

With PA multihoming, for each ISP connection, the site is assigned a prefix from within an address block allocated to that ISP by its National or Regional Internet Registry. In the simple case of two ISPs (ISP-A and ISP-B), the site will have two different prefixes assigned to it (prefix-A and prefix-B). This arrangement is problematic. First, packets with the "wrong" source address may be dropped by one of the ISPs. In order to limit denial of service attacks using spoofed source addresses, BCP38 [RFC2827] recommends that ISPs filter traffic from customer sites to only allow traffic with a source address that has been assigned by that ISP. So a packet sent from a multihomed site on the uplink to ISP-B with a source address in prefix-A may be dropped by ISP-B.

However, even if ISP-B does not implement BCP38 or ISP-B adds prefix-A to its list of allowed source addresses on the uplink from the multihomed site, two-way communication may still fail. If the packet with source address in prefix-A was sent to ISP-B because the uplink to ISP-A failed, then if ISP-B does not drop the packet and the packet reaches its destination somewhere on the Internet, the return packet will be sent back with a destination address in prefix-A. The return packet will be routed over the Internet to ISP-A, but it will not be delivered to the multihomed site because its link with ISP-A has failed. Two-way communication would require some arrangement for ISP-B to advertise prefix-A when the uplink to ISP-A fails.

Note that the same may be true with a provider that does not implement BCP 38, if his upstream provider does, or has no corresponding route. The issue is not that the immediate provider implements ingress filtering; it is that someone upstream does, or lacks a route.

With IPv4, this problem is commonly solved by using [RFC1918] private address space within the multi-homed site and Network Address Translation (NAT) or Network Address/Port Translation (NAPT) on the uplinks to the ISPs. However, one of the goals of IPv6 is to eliminate the need for and the use of NAT or NAPT. Therefore, requiring the use of NAT or NAPT for an enterprise site to multihome with provider-assigned addresses is not an attractive solution.

[RFC6296] describes a translation solution specifically tailored to meet the requirements of multi-homing with provider-assigned IPv6 addresses. With the IPv6-to-IPv6 Network Prefix Translation (NPTv6) solution, within the site an enterprise can use Unique Local Addresses [RFC4193] or the prefix assigned by one of the ISPs. As traffic leaves the site on an uplink to an ISP, the source address gets translated to an address within the prefix assigned by the ISP on that uplink in a predictable and reversible manner. [RFC6296] is currently classified as Experimental, and it has been implemented by several vendors. See Section 5.2, for more discussion of NPTv6.

This document defines routing requirements for enterprise multihoming using provider-assigned IPv6 addresses. We have made no attempt to write these requirements in a manner that is agnostic to potential solutions. Instead, this document focuses on the following general class of solutions.

Each host at the enterprise has multiple addresses, at least one from each ISP-assigned prefix. Each host, as discussed in Section 4.1 and [RFC6724], is responsible for choosing the source address applied to each packet it sends. A host SHOULD be able respond dynamically to the failure of an uplink to a given ISP by no longer sending packets with the source address corresponding to that ISP. Potential mechanisms for the communication of changes in the network to the host are Neighbor Discovery Router Advertisements, DHCPv6, and ICMPv6.

The routers in the enterprise network are responsible for ensuring that packets are delivered to the "correct" ISP uplink based on source address. This requires that at least some routers in the site network are able to take into account the source address of a packet when deciding how to route it. That is, some routers must be capable of some form of Source Address Dependent Routing (SADR), if only as described in [RFC3704]. At a minimum, the routers connected to the ISP uplinks (the site exit routers or SERs) must be capable of Source Address Dependent Routing. Expanding the connected domain of routers capable of SADR from the site exit routers deeper into the site network will generally result in more efficient routing of traffic with external destinations.

The document first looks in more detail at the enterprise networking environments in which this solution is expected to operate. It then discusses existing and proposed mechanisms for hosts to select the source address applied to packets. Finally, it looks at the requirements for routing that are needed to support these enterprise network scenarios and the mechanisms by which hosts are expected to select source addresses dynamically based on network state.

2. Enterprise Multihoming Requirements

2.1. Simple ISP Connectivity with Connected SERs

We start by looking at a scenario in which a site has connections to two ISPs, as shown in Figure 1. The site is assigned the prefix 2001:db8:0:a000::/52 by ISP-A and prefix 2001:db8:0:b000::/52 by ISP-B. We consider three hosts in the site. H31 and H32 are on a LAN that has been assigned subnets 2001:db8:0:a010::/64 and 2001:db8:0:b010::/64. H31 has been assigned the addresses 2001:db8:0:a010::31 and 2001:db8:0:b010::31. H32 has been assigned 2001:db8:0:a010::32 and 2001:db8:0:b010::32. H41 is on a different subnet that has been assigned 2001:db8:0:a020::/64 and 2001:db8:0:b020::/64.

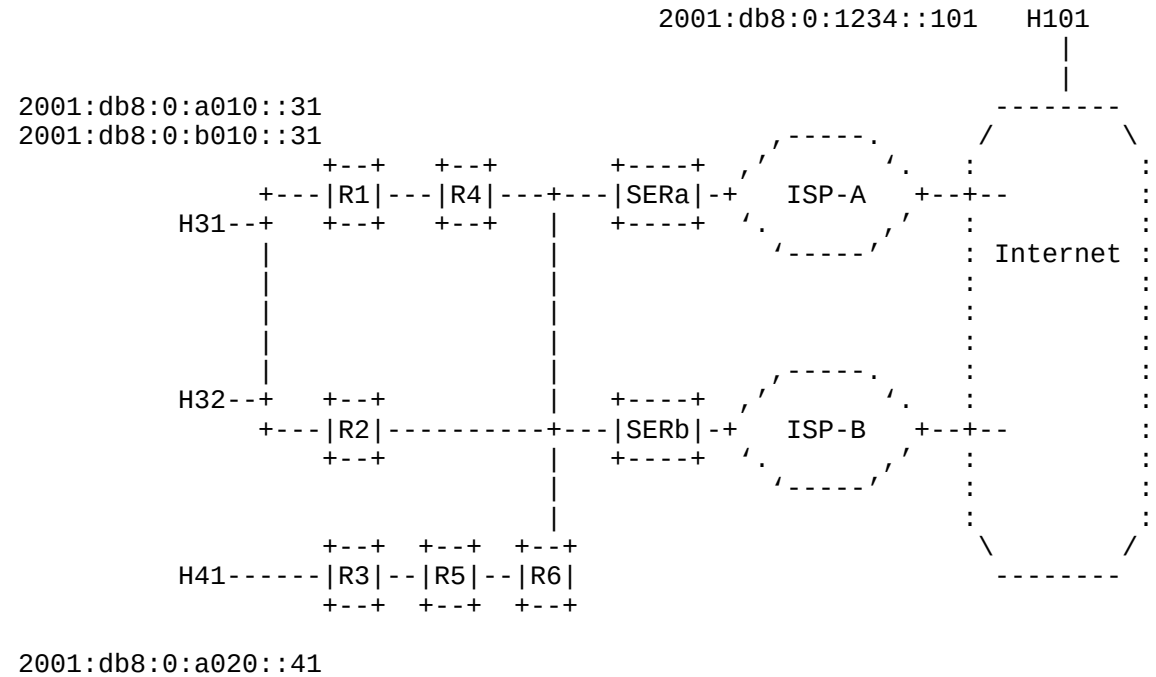


Figure 1: Simple ISP Connectivity With Connected SERs

We refer to a router that connects the site to an ISP as a site edge router(SER). Several other routers provide connectivity among the internal hosts (H31, H32, and H41), as well as connecting the internal hosts to the Internet through SERa and SERb. In this

example SERa and SERb share a direct connection to each other. In Section 2.2, we consider a scenario where this is not the case.

For the moment, we assume that the hosts are able to make good choices about which source addresses through some mechanism that doesn't involve the routers in the site network. Here, we focus on primary task of the routed site network, which is to get packets efficiently to their destinations, while sending a packet to the ISP that assigned the prefix that matches the source address of the packet. In Section 4, we examine what role the routed network may play in helping hosts make good choices about source addresses for packets.

With this solution, routers will need form of Source Address Dependent Routing, which will be new functionality. It would be useful if an enterprise site does not need to upgrade all routers to support the new SADR functionality in order to support PA multi-homing. We consider if this is possible and what are the tradeoffs of not having all routers in the site support SADR functionality.

In the topology in Figure 1, it is possible to support PA multihoming with only SERa and SERb being capable of SADR. The other routers can continue to forward based only on destination address, and exchange routes that only consider destination address. In this scenario, SERa and SERb communicate source-scoped routing information across their shared connection. When SERa receives a packet with a source address matching prefix 2001:db8:0:b000::/52 , it forwards the packet to SERb, which forwards it on the uplink to ISP-B. The analogous behaviour holds for traffic that SERb receives with a source address matching prefix 2001:db8:0:a000::/52.

In Figure 1, when only SERa and SERb are capable of source address dependent routing, PA multi-homing will work. However, the paths over which the packets are sent will generally not be the shortest paths. The forwarding paths will generally be more efficient as more routers are capable of SADR. For example, if R4, R2, and R6 are upgraded to support SADR, then can exchange source-scoped routes with SERa and SERb. They will then know to send traffic with a source address matching prefix 2001:db8:0:b000::/52 directly to SERb, without sending it to SERa first.

2.2. Simple ISP Connectivity Where SERs Are Not Directly Connected

In Figure 2, we modify the topology slightly by inserting R7, so that SERa and SERb are no longer directly connected. With this topology, it is not enough to just enable SADR routing on SERa and SERb to support PA multi-homing. There are two solutions to ways to enable PA multihoming in this topology.

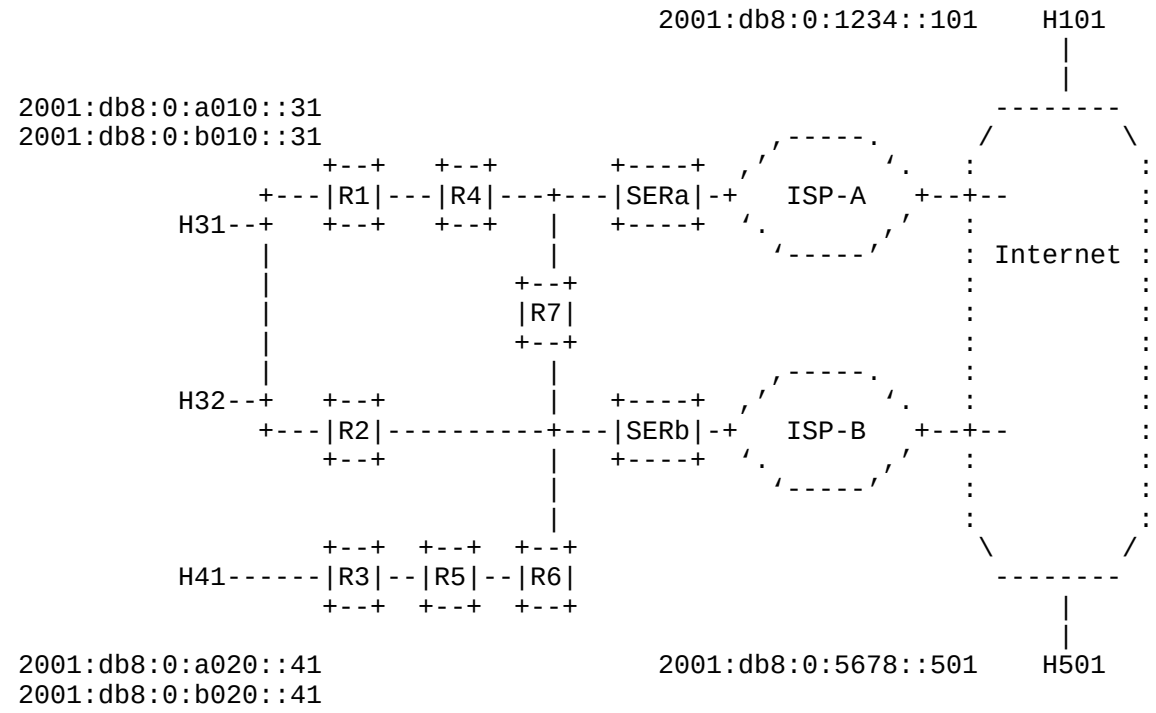


Figure 2: Simple ISP Connectivity Where SERs Are Not Directly Connected

One option is to effectively modify the topology by creating a logical tunnel between SERa and SERb, using GRE for example. Although SERa and SERb are not directly connected physically in this topology, they can be directly connected logically by a tunnel.

The other option is to enable SADR functionality on R7. In this way, R7 will exchange source-scoped routes with SERa and SERb, making the three routers act as a single SADR domain. This illustrates the basic principle that the minimum requirement for the routed site network to support PA multi-homing is having all of the site exit routers be part of a connected SADR domain. Extending the connected SADR domain beyond that point can produce more efficient forwarding paths.

2.3. Enterprise Network Operator Expectations

Before considering a more complex scenario, let’s look in more detail at the reasonably simple multihoming scenario in Figure 2 to understand what can reasonably be expected from this solution. As a

general guiding principle, we assume an enterprise network operator will expect a multihomed network to behave as close as to a single-homed network as possible. So a solution that meets those expectations where possible is a good thing.

For traffic between internal hosts and traffic from outside the site to internal hosts, an enterprise network operator would expect there to be no visible change in the path taken by this traffic, since this traffic does not need to be routed in a way that depends on source address. It is also reasonable to expect that internal hosts should be able to communicate with each other using either of their source addresses without restriction. For example, H31 should be able to communicate with H41 using a packet with S=2001:db8:0:a010::31, D=2001:db8:0:b010::41, regardless of the state of uplink to ISP-B.

These goals can be accomplished by having all of the routers in the network continue to originate normal unscoped destination routes for their connected networks. If we can arrange so that these unscoped destination routes get used for forwarding this traffic, then we will have accomplished the goal of keeping forwarding of traffic destined for internal hosts, unaffected by the multihoming solution.

For traffic destined for external hosts, it is reasonable to expect that traffic with an source address from the prefix assigned by ISP-A to follow the path to that the traffic would follow if there is no connection to ISP-B. This can be accomplished by having SERa originate a source-scoped route of the form (S=2001:db8:0:a000::/52, D=::/0) . If all of the routers in the site support SADR, then the path of traffic exiting via ISP-A can match that expectation. If some routers don’t support SADR, then it is reasonable to expect that the path for traffic exiting via ISP-A may be different within the site. This is a tradeoff that the enterprise network operator may decide to make.

It is important to understand how this multihoming solution behaves when an uplink to one of the ISPs fails. To simplify this discussion, we assume that all routers in the site support SADR. We first start by looking at how the network operates when the uplinks to both ISP-A and ISP-B are functioning properly. SERa originates a source-scoped route of the form (S=2001:db8:0:a000::/52, D=::/0), and SERb is originates a source-scoped route of the form (S=2001:db8:0:b000::/52, D=::/0). These routes are distributed through the routers in the site, and they establish within the routers two set of forwarding paths for traffic leaving the site. One set of forwarding paths is for packets with source address in 2001:db8:0:a000::/52. The other set of forwarding paths is for packets with source address in 2001:db8:0:b000::/52. The normal destination routes which are not scoped to these two source prefixes

play no role in the forwarding. Whether a packet exits the site via SERa or via SERb is completely determined by the source address applied to the packet by the host. So for example, when host H31 sends a packet to host H101 with (S=2001:db8:0:a010::31, D=2001:db8:0:1234::101), the packet will only be sent out the link from SERa to ISP-A.

Now consider what happens when the uplink from SERa to ISP-A fails. The only way for the packets from H31 to reach H101 is for H31 to start using the source address for ISP-B. H31 needs to send the following packet: (S=2001:db8:0:b010::31, D=2001:db8:0:1234::101).

This behavior is very different from the behavior that occurs with site multihoming using PI addresses or with PA addresses using NAT. In these other multi-homing solutions, hosts do not need to react to network failures several hops away in order to regain Internet access. Instead, a host can be largely unaware of the failure of an uplink to an ISP. When multihoming with PA addresses and NAT, existing sessions generally need to be re-established after a failure since the external host will receive packets from the internal host with a new source address. However, new sessions can be established without any action on the part of the hosts.

Another example where the behavior of this multihoming solution differs significantly from that of multihoming with PI address or with PA addresses using NAT is in the ability of the enterprise network operator to route traffic over different ISPs based on destination address. We still consider the fairly simple network of Figure 2 and assume that uplinks to both ISPs are functioning. Assume that the site is multihomed using PA addresses and NAT, and that SERa and SERb each originate a normal destination route for D=::/0, with the route origination dependent on the state of the uplink to the respective ISP.

Now suppose it is observed that an important application running between internal hosts and external host H101 experience much better performance when the traffic passes through ISP-A (perhaps because ISP-A provides lower latency to H101.) When multihoming this site with PI addresses or with PA addresses and NAT, the enterprise network operator can configure SERa to originate into the site network a normal destination route for D=2001:db8:0:1234::/64 (the destination prefix to reach H101) that depends on the state of the uplink to ISP-A. When the link to ISP-A is functioning, the destination route D=2001:db8:0:1234::/64 will be originated by SERa, so traffic from all hosts will use ISP-A to reach H101 based on the longest destination prefix match in the route lookup.

Implementing the same routing policy is more difficult with the PA multihoming solution described in this document since it doesn't use NAT. By design, the only way to control where a packet exits this network is by setting the source address of the packet. Since the network cannot modify the source address without NAT, the host must set it. To implement this routing policy, each host needs to use the source address from the prefix assigned by ISP-A to send traffic destined for H101. Mechanisms have been proposed to allow hosts to choose the source address for packets in a fine grained manner. We will discuss these proposals in Section 4. However, interacting with host operating systems in some manner to ensure a particular source address is chosen for a particular destination prefix is not what an enterprise network administrator would expect to have to do to implement this routing policy.

2.4. More complex ISP connectivity

The previous sections considered two variations of a simple multihoming scenario where the site is connected to two ISPs offering only Internet connectivity. It is likely that many actual enterprise multihoming scenarios will be similar to this simple example. However, there are more complex multihoming scenarios that we would like this solution to address as well.

It is fairly common for an ISP to offer a service in addition to Internet access over the same uplink. Two variation of this are reflected in Figure 3. In addition to Internet access, ISP-A offers a service which requires the site to access host H51 at 2001:db8:0:5555::51. The site has a single physical and logical connection with ISP-A, and ISP-A only allows access to H51 over that connection. So when H32 needs to access the service at H51 it needs to send packets with (S=2001:db8:0:a010::32, D=2001:db8:0:5555::51) and those packets need to be forward out the link from SERa to ISP-A.

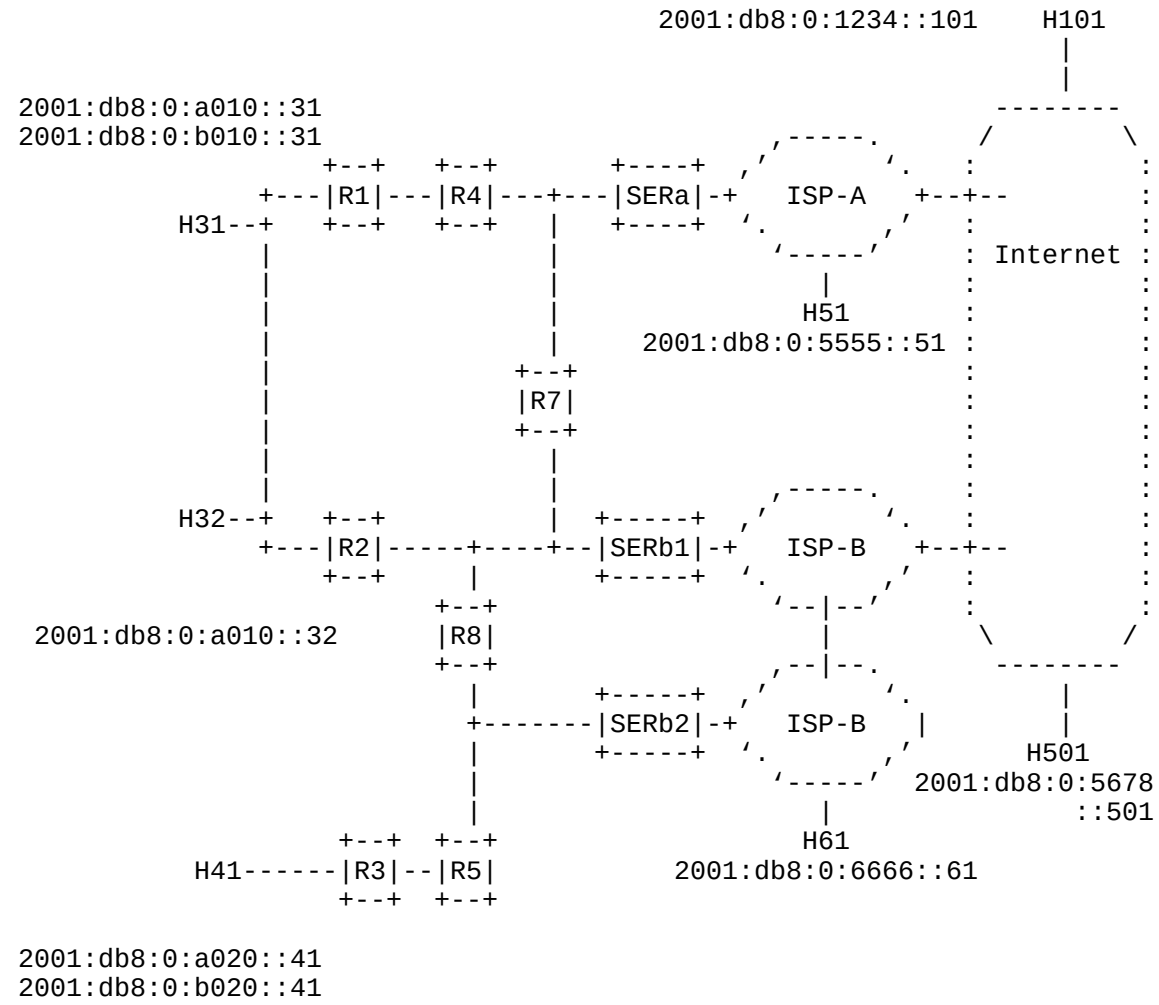


Figure 3: Internet access and services offered by ISP-A and ISP-B

ISP-B illustrates a variation on this scenario. In addition to Internet access, ISP-B also offers a service which requires the site to access host H61. The site has two connections to two different parts of ISP-B (shown as SERb1 and SERb2 in Figure 3). ISP-B expects Internet traffic to use the uplink from SERb1, while it expects it expects traffic destined for the service at H61 to use the uplink from SERb2. For either uplink, ISP-B expects the ingress traffic to have a source address matching the prefix it assigned to the site, 2001:db8:0:b000::/52.

As discussed before, we rely completely on the internal host to set the source address of the packet properly. In the case of a packet sent by H31 to access the service in ISP-B at H61, we expect the packet to have the following addresses: (S=2001:db8:0:b010::31, D=2001:db8:0:6666::61). The routed network has two potential ways of distributing routes so that this packet exits the site on the uplink at SERb2.

We could just rely on normal destination routes, without using source-prefix scoped routes. If we have SERb2 originate a normal unscoped destination route for D=2001:db8:0:6666::/64, the packets from H31 to H61 will exit the site at SERb2 as desired. We should not have to worry about SERa needing to originate the same route, because ISP-B should choose a globally unique prefix for the service at H61.

The alternative is to have SERb2 originate a source-prefix-scoped destination route of the form (S=2001:db8:0:b000::/52, D=2001:db8:0:6666::/64). From a forwarding point of view, the use of the source-prefix-scoped destination route would result in traffic with source addresses corresponding only to ISP-B being sent to SERb2. Instead, the use of the unscoped destination route would result in traffic with source addresses corresponding to ISP-A and ISP-B being sent to SERb2, as long as the destination address matches the destination prefix. It seems like either forwarding behavior would be acceptable.

However, from the point of view of the enterprise network administrator trying to configure, maintain, and trouble-shoot this multihoming solution, it seems much clearer to have SERb2 originate the source-prefix-scoped destination route correspond to the service offered by ISP-B. In this way, all of the traffic leaving the site is determined by the source-prefix-scoped routes, and all of the traffic within the site or arriving from external hosts is determined by the unscoped destination routes. Therefore, for this multihoming solution we choose to originate source-prefix-scoped routes for all traffic leaving the site.

2.5. ISPs and Provider-Assigned Prefixes

While we expect that most site multihoming involves connecting to only two ISPs, this solution allows for connections to an arbitrary number of ISPs to be supported. However, when evaluating scalable implementations of the solution, it would be reasonable to assume that the maximum number of ISPs that a site would connect to is five.

It is also useful to note that the prefixes assigned to the site by different ISPs will not overlap. This must be the case , since the provider-assigned addresses have to be globally unique.

2.6. Simplified Topologies

The topologies of many enterprise sites using this multihoming solution may in practice be simpler than the examples that we have used. The topology in Figure 1 could be further simplified by having all hosts directly connected to the LAN connecting the two site exit routers, SERa and SERb. The topology could also be simplified by having the uplinks to ISP-A and ISP-B both connected to the same site exit router. However, it is the aim of this draft to provide a solution that applies to a broad a range of enterprise site network topologies, so this draft focuses on providing a solution to the more general case. The simplified cases will also be supported by this solution, and there may even be optimizations that can be made for simplified cases. This solution however needs to support more complex topologies.

We are starting with the basic assumption that enterprise site networks can be quite complex from a routing perspective. However, even a complex site network can be multihomed to different ISPs with PA addresses using IPv4 and NAT. It is not reasonable to expect an enterprise network operator to change the routing topology of the site in order to deploy IPv6.

3. Generating Source-Prefix-Scoped Forwarding Tables

So far we have described in general terms how the routers in this solution that are capable of Source Address Dependent Routing will forward traffic using both normal unscoped destination routes and source-prefix-scoped destination routes. Here we give a precise method for generating a source-prefix-scoped forwarding table on a router that supports SADR.

1. Compute the next-hops for the source-prefix-scoped destination prefixes using only routers in the connected SADR domain. These are the initial source-prefix-scoped forwarding table entries.

2. Compute the next-hops for the unscoped destination prefixes using all routers in the IGP. This is the unscoped forwarding table.

3. Augment each source-prefix-scoped forwarding table with unscoped forwarding table entries based on the following rule. If the destination prefix of the unscoped forwarding entry exactly matches the destination prefix of an existing source-prefix-scoped forwarding entry (including destination prefix length),

then do not add the unscoped forwarding entry. If the destination prefix does NOT match an existing entry, then add the entry to the source-prefix-scoped forwarding table.

The forward tables produced by this process are used in the following way to forward packets.

1. If the source address of the packet matches one of the source prefixes, then look up the destination address of the packet in the corresponding source-prefix-scoped forwarding table to determine the next-hop for the packet.

2. If the source address of the packet does NOT match one of the source prefixes, then look up the destination address of the packet in unscoped forwarding table to determine the next-hop for the packet.

The following example illustrates how this process is used to create a forwarding table for each provider-assigned source prefix. We consider the multihomed site network in Figure 3. Initially we assume that all of the routers in the site network support SADR. Figure 4 shows the routes that are originated by the routers in the site network.

```
Routes originated by SERa:
(S=2001:db8:0:a000::/52, D=2001:db8:0:5555/64)
(S=2001:db8:0:a000::/52, D=::/0)
(D=2001:db8:0:5555::/64)
(D=::/0)

Routes originated by SERb1:
(S=2001:db8:0:b000::/52, D=::/0)
(D=::/0)

Routes originated by SERb2:
(S=2001:db8:0:b000::/52, D=2001:db8:0:6666::/64)
(D=2001:db8:0:6666::/64)

Routes originated by R1:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R2:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R3:
(D=2001:db8:0:a020::/64)
(D=2001:db8:0:b020::/64)
```

Figure 4: Routes Originated by Routers in the Site Network

Each SER originates destination routes which are scoped to the source prefix assigned by the ISP that the SER connects to. Note that the SERs also originate the corresponding unscoped destination route. This is not needed when all of the routers in the site support SADR. However, it is required when some routers do not support SADR. This will be discussed in more detail later.

We focus on how R8 constructs its source-prefix-scoped forwarding tables from these route advertisements. R8 computes the next hops for destination routes which are scoped to the source prefix 2001:db8:0:a000::/52. The results are shown in the first table in Figure 5. (In this example, the next hops are computed assuming that all links have the same metric.) Then, R8 computes the next hops for destination routes which are scoped to the source prefix 2001:db8:0:b000::/52. The results are shown in the second table in Figure 5 . Finally, R8 computes the next hops for the unscoped destination prefixes. The results are shown in the third table in Figure 5.

```
forwarding entries scoped to
source prefix = 2001:db8:0:a000::/52
==========================================
D=2001:db8:0:5555/64      NH=R7
D=::/0                    NH=R7

forwarding entries scoped to
source prefix = 2001:db8:0:b000::/52
==========================================
D=2001:db8:0:6666/64      NH=SERb2
D=::/0                    NH=SERb1

unscoped forwarding entries
==========================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=SERb1
```

Figure 5: Forwarding Entries Computed at R8

The final step is for R8 to augment the source-prefix-scoped forwarding entries with unscoped forwarding entries. If an unscoped forwarding entry has the exact same destination prefix as an source-prefix-scoped forwarding entry (including destination prefix length), then the source-prefix-scoped forwarding entry wins.

As as an example of how the source scoped forwarding entries are augmented with unscoped forwarding entries, we consider how the two entries in the first table in Figure 5 (the table for source prefix = 2001:db8:0:a000::/52) are augmented with entries from the third table in Figure 5 (the table of unscoped forwarding entries). The first four unscoped forwarding entries (D=2001:db8:0:a010::/64, D=2001:db8:0:b010::/64, D=2001:db8:0:a020::/64, and D=2001:db8:0:b020::/64) are not an exact match for any of the existing entries in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, these four entries are added to the final forwarding table for source prefix 2001:db8:0:a000::/52. The result of adding these entries is reflected in first four entries the first table in Figure 6.

The next unscoped forwarding table entry is for D=2001:db8:0:5555::/64. This entry is an exact match for the existing entry in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, we do not replace the existing

entry with the entry from the unscoped forwarding table. This is reflected in the fifth entry in the first table in Figure 6. (Note that since both scoped and unscoped entries have R7 as the next hop, the result of applying this rule is not visible.)

The next unscoped forwarding table entry is for D=2001:db8:0:6666::/64. This entry is not an exact match for any existing entries in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, we add this entry. This is reflected in the sixth entry in the first table in Figure 6.

The next unscoped forwarding table entry is for D=::/0. This entry is an exact match for the existing entry in the forwarding table for source prefix 2001:db8:0:a000::/52. Therefore, we do not overwrite the existing source-prefix-scoped entry, as can be seen in the last entry in the first table in Figure 6.

```
if source address matches 2001:db8:0:a000::/52
then use this forwarding table
==========================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=R7

else if source address matches 2001:db8:0:b000::/52
then use this forwarding table
==========================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=SERb1

else use this forwarding table
==========================================
D=2001:db8:0:a010::/64    NH=R2
D=2001:db8:0:b010::/64    NH=R2
D=2001:db8:0:a020::/64    NH=R5
D=2001:db8:0:b020::/64    NH=R5
D=2001:db8:0:5555::/64    NH=R7
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=SERb1
```

Figure 6: Complete Forwarding Tables Computed at R8

The forwarding tables produced by this process at R8 have the desired properties. A packet with a source address in 2001:db8:0:a000::/52 will be forwarded based on the first table in Figure 6. If the packet is destined for the Internet at large or the service at D=2001:db8:0:5555/64, it will be sent to R7 in the direction of SERa. If the packet is destined for an internal host, then the first four entries will send it to R2 or R5 as expected. Note that if this packet has a destination address corresponding to the service offered by ISP-B (D=2001:db8:0:5555::/64), then it will get forwarded to SERb2. It will be dropped by SERb2 or by ISP-B, since it the packet has a source address that was not assigned by ISP-B. However, this is expected behavior. In order to use the service offered by ISP-B, the host needs to originate the packet with a source address assigned by ISP-B.

In this example, a packet with a source address that doesn't match 2001:db8:0:a000::/52 or 2001:db8:0:b000::/52 must have originated from an external host. Such a packet will use the unscoped forwarding table (the last table in Figure 6). These packets will flow exactly as they would in absence of multihoming.

We can also modify this example to illustrate how it supports deployments where not all routers in the site support SADR. Continuing with the topology shown in Figure 3, suppose that R3 and R5 do not support SADR. Instead they are only capable of understanding unscoped route advertisements. The SADR routers in the network will still originate the routes shown in Figure 4. However, R3 and R5 will only understand the unscoped routes as shown in Figure 7.

```
Routes originated by SERa:
(D=2001:db8:0:5555::/64)
(D=::/0)

Routes originated by SERb1:
(D=::/0)

Routes originated by SERb2:
(D=2001:db8:0:6666::/64)

Routes originated by R1:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R2:
(D=2001:db8:0:a010::/64)
(D=2001:db8:0:b010::/64)

Routes originated by R3:
(D=2001:db8:0:a020::/64)
(D=2001:db8:0:b020::/64)
```

Figure 7: Routes Advertisements Understood by Routers that do no Support SADR

With these unscoped route advertisements, R5 will produce the forwarding table shown in Figure 8.

```
forwarding table
=========================================
D=2001:db8:0:a010::/64    NH=R8
D=2001:db8:0:b010::/64    NH=R8
D=2001:db8:0:a020::/64    NH=R3
D=2001:db8:0:b020::/64    NH=R3
D=2001:db8:0:5555::/64    NH=R8
D=2001:db8:0:6666::/64    NH=SERb2
D=::/0                    NH=R8
```

Figure 8: Forwarding Table For R5, Which Doesn't Understand Source-Prefix-Scoped Routes

Any traffic that needs to exit the site will eventually hit a SADR-capable router. Once that traffic enters the SADR-capable domain, then it will not leave that domain until it exits the site. This property is required in order to guarantee that there will not be routing loops involving SADR-capable and non-SADR-capable routers.

Note that the mechanism described here for converting source-prefix-scoped destination prefix routing advertisements into forwarding state is somewhat different from that proposed in [I-D.ietf-rtgwg-dst-src-routing]. The method described in this document is intended to be easy to understand for network enterprise operators while at the same time being functionally correct. Another difference is that the method in this document assumes that source prefix will not overlap. Other differences between the two approaches still need to be understood and reconciled.

An interesting side-effect of deploying SADR is if all routers in a given network support SADR and have a scoped forwarding table, then the unscoped forwarding table can be eliminated which ensures that packets with legitimate source addresses only can leave the network (as there are no scoped forwarding tables for spoofed/bogon source addresses). It would prevent accidental leaks of ULA/reserved/link-local sources to the Internet as well as ensures that no spoofing is possible from the SADR-enabled network.

4. Mechanisms For Hosts To Choose Good Source Addresses In A Multihomed Site

Until this point, we have made the assumption that hosts are able to choose the correct source address using some unspecified mechanism. This has allowed us to just focus on what the routers in a multihomed site network need to do in order to forward packets to the correct ISP based on source address. Now we look at possible mechanisms for hosts to choose the correct source address. We also look at what

role, if any, the routers may play in providing information that helps hosts to choose source addresses.

Any host that needs to be able to send traffic using the uplinks to a given ISP is expected to be configured with an address from the prefix assigned by that ISP. The host will control which ISP is used for its traffic by selecting one of the addresses configured on the host as the source address for outgoing traffic. It is the responsibility of the site network to ensure that a packet with the source address from an ISP is not sent on an uplink to that ISP.

If all of the ISP uplinks are working, the choice of source address by the host may be driven by the desire to load share across ISP uplinks, or it may be driven by the desire to take advantage of certain properties of a particular uplink or ISP. If any of the ISP uplinks is not working, then the choice of source address by the host can determine if packets get dropped.

How a host should make good decisions about source address selection in a multihomed site is not a solved problem. We do not attempt to solve this problem in this document. Instead we discuss the current state of affairs with respect to standardized solutions and implementation of those solutions. We also look at proposed solutions for this problem.

An external host initiating communication with a host internal to a PA multihomed site will need to know multiple addresses for that host in order to communicate with it using different ISPs to the multihomed site. These addresses are typically learned through DNS. (For simplicity, we assume that the external host is single-homed.) The external host chooses the ISP that will be used at the remote multihomed site by setting the destination address on the packets it transmits. For a sessions originated from an external host to an internal host, the choice of source address used by the internal host is simple. The internal host has no choice but to use the destination address in the received packet as the source address of the transmitted packet.

For a session originated by a host internal to the multi-homed site, the decision of what source address to select is more complicated. We consider three main methods for hosts to get information about the network. The two proactive methods are Neighbor Discovery Router Advertisements and DHCPv6. The one reactive method we consider is ICMPv6. Note that we are explicitly excluding the possibility of having hosts participate in or even listen directly to routing protocol advertisements.

First we look at how a host is currently expected to select the source and destination address with which it sends a packet.

4.1. Source Address Selection Algorithm on Hosts

[RFC6724] defines the algorithms that hosts are expected to use to select source and destination addresses for packets. It defines an algorithm for selecting a source address and a separate algorithm for selecting a destination address. Both of these algorithms depend on a policy table. [RFC6724] defines a default policy which produces certain behavior.

The rules in the two algorithms in [RFC6724] depend on many different properties of addresses. While these are needed for understanding how a host should choose addresses in an arbitrary environment, most of the rules are not relevant for understanding how a host should choose among multiple source addresses in mulitihomed envinronment when sending a packet to a remote host. Returning to the example in Figure 3, we look at what the default algorithms in [RFC6724] say about the source address that internal host H31 should use to send traffic to external host H101, somewhere on the Internet. Let’s look at what rules in [RFC6724] are actually used by H31 in this case.

There is no choice to be made with respect to destination address. H31 needs to send a packet with D=2001:db8:0:1234::101 in order to reach H101. So H31 have to choose between using S=2001:db8:0:a010::31 or S=2001:db8:0:b010::31 as the source address for this packet. We go through the rules for source address selection in Section 5 of [RFC6724]. Rule 1 (Prefer same address) is not useful to break the tie between source addresses, because neither the candidate source addresses equals the destination address. Rule 2 (Prefer appropriate scope) is also not used in this scenario, because both source addresses and the destination address have global scope.

Rule 3 (Avoid deprecated addresses) applies to an address that has been autoconfigured by a host using stateless address autoconfiguration as defined in [RFC4862]. An address autoconfigured by a host has a preferred lifetime and a valid lifetime. The address is preferred until the preferred lifetime expires, after which it becomes deprecated. A deprecated address can still be used, but it is better to use a preferred address. When the valid lifetime expires, the address cannot be used at all. The preferred and valid lifetimes for an autoconfigured address are set based on the corresponding lifetimes in the Prefix Information Option in Neighbor Discovery Router Advertisements. So a possible tool to control source address selection in this scenario would be to a host to make an address deprecated by having routers on that link, R1 and R2 in

Figure 3, send Prefix Information Option messages with the preferred lifetime for the source prefix to be discouraged (or prohibited) set to zero. This is a rather blunt tool, because it discourages or prohibits the use of that source prefix for all destinations. However, it may be useful in some scenarios.

Rule 4 (Avoid home addresses) does not apply here because we are not considering Mobile IP.

Rule 5 (Prefer outgoing interface) is not useful in this scenario, because both source addresses are assigned to the same interface.

Rule 5.5 (Prefer addresses in a prefix advertised by the next-hop) is not useful in the scenario when both R1 and R2 will advertise both source prefixes. However potentially this rule may allow a host to select the correct source prefix by selecting a next-hop. The most obvious way would be to make R1 to advertise itself as a default router and send PIO for 2001:db8:0:a010::/64, while R2 is advertising itself as a default router and sending PIO for 2001:db8:0:b010::/64. We'll discuss later how Rule 5.5 can be used to influence a source address selection in single-router topologies (e.g. when H41 is sending traffic using R3 as a default gateway).

Rule 6 (Prefer matching label) refers to the Label value determined for each source and destination prefix as a result of applying the policy table to the prefix. With the default policy table defined in Section 2.1 of [RFC6724], Label(2001:db8:0:a010::31) = 5, Label(2001:db8:0:b010::31) = 5, and Label(2001:db8:0:1234::101) = 5. So with the default policy, Rule 6 does not break the tie. However, the algorithms in [RFC6724] are defined in such as way that non-default address selection policy tables can be used. [RFC7078] defines a way to distribute a non-default address selection policy table to hosts using DHCPv6. So even though the application of rule 6 to this scenario using the default policy table is not useful, rule 6 may still be a useful tool.

Rule 7 (Prefer temporary addresses) has to do with the technique described in [RFC4941] to periodically randomize the interface portion of an IPv6 address that has been generated using stateless address autoconfiguration. In general, if H31 were using this technique, it would use it for both source addresses, for example creating temporary addresses 2001:db8:0:a010:2839:9938:ab58:830f and 2001:db8:0:b010:4838:f483:8384:3208, in addition to 2001:db8:0:a010::31 and 2001:db8:0:b010::31. So this rule would prefer the two temporary addresses, but it would not break the tie between the two source prefixes from ISP-A and ISP-B.

Rule 8 (Use longest matching prefix) dictates that between two candidate source addresses the one which has longest common prefix length with the destination address. For example, if H31 were selecting the source address for sending packets to H101, this rule would not be a tie breaker as for both candidate source addresses 2001:db8:0:a101::31 and 2001:db8:0:b101::31 the common prefix length with the destination is 48. However if H31 were selecting the source address for sending packets H41 address 2001:db8:0:a020::41, then this rule would result in using 2001:db8:0:a101::31 as a source (2001:db8:0:a101::31 and 2001:db8:0:a020::41 share the common prefix 2001:db8:0:a000::/58, while for '2001:db8:0:b101::31 and 2001:db8:0:a020::41 the common prefix is 2001:db8:0:a000::/51). Therefore rule 8 might be useful for selecting the correct source address in some but not all scenarios (for example if ISP-B services belong to 2001:db8:0:b000::/59 then H31 would always use 2001:db8:0:b010::31 to access those destinations).

So we can see that of the 8 source selection address rules from [RFC6724], five actually apply to our basic site multihoming scenario. The rules that are relevant to this scenario are summarized below.

- Rule 3: Avoid deprecated addresses.
- Rule 5.5: Prefer addresses in a prefix advertised by the next-hop.
- Rule 6: Prefer matching label.
- Rule 8: Prefer longest matching prefix.

The two methods that we discuss for controlling the source address selection through the four relevant rules above are SLAAC Router Advertisement messages and DHCPv6.

We also consider a possible role for ICMPv6 for getting traffic-driven feedback from the network. With the source address selection algorithm discussed above, the goal is to choose the correct source address on the first try, before any traffic is sent. However, another strategy is to choose a source address, send the packet, get feedback from the network about whether or not the source address is correct, and try another source address if it is not.

We consider four scenarios where a host needs to select the correct source address. The first is when both uplinks are working. The second is when one uplink has failed. The third one is a situation when one failed uplink has recovered. The last one is failure of both (all) uplinks.

4.2. Selecting Source Address When Both Uplinks Are Working

Again we return to the topology in Figure 3. Suppose that the site administrator wants to implement a policy by which all hosts need to use ISP-A to reach H01 at D=2001:db8:0:1234::101. So for example, H31 needs to select S=2001:db8:0:a010::31.

4.2.1. Distributing Address Selection Policy Table with DHCPv6

This policy can be implemented by using DHCPv6 to distribute an address selection policy table that assigns the same label to destination address that match 2001:db8:0:1234::/64 as it does to source addresses that match 2001:db8:0:a000::/52. The following two entries accomplish this.

Prefix	Precedence	Label
2001:db8:0:1234::/64	50	33
2001:db8:0:a000::/52	50	33

Figure 9: Policy table entries to implement a routing policy

This requires that the hosts implement [RFC6724], the basic source and destination address framework, along with [RFC7078], the DHCPv6 extension for distributing a non-default policy table. Note that it does NOT require that the hosts use DHCPv6 for address assignment. The hosts could still use stateless address autoconfiguration for address configuration, while using DHCPv6 only for policy table distribution (see [RFC3736]). However this method has a number of disadvantages:

- DHCPv6 support is not a mandatory requirement for IPv6 hosts, so this method might not work for all devices.
- Network administrators are required to explicitly configure the desired network access policies on DHCPv6 servers.

4.2.2. Controlling Source Address Selection With Router Advertisements

Neighbor Discovery currently has two mechanisms to communicate prefix information to hosts. The base specification for Neighbor Discovery (see [RFC4861]) defines the Prefix Information Option (PIO) in the Router Advertisement (RA) message. When a host receives a PIO with the A-flag set, it can use the prefix in the PIO as source prefix from which it assigns itself an IP address using stateless address autoconfiguration (SLAAC) procedures described in [RFC4862]. In the example of Figure 3, if the site network is using SLAAC, we would expect both R1 and R2 to send RA messages with PIOs for both source prefixes 2001:db8:0:a010::/64 and 2001:db8:0:b010::/64 with the

A-flag set. H31 would then use the SLAAC procedure to configure itself with the 2001:db8:0:a010::31 and 2001:db8:0:b010::31.

Whereas a host learns about source prefixes from PIO messages, hosts can learn about a destination prefix from a Router Advertisement containing Route Information Option (RIO), as specified in [RFC4191]. The destination prefixes in RIOs are intended to allow a host to choose the router that it uses as its first hop to reach a particular destination prefix.

As currently standardized, neither PIO nor RIO options contained in Neighbor Discovery Router Advertisements can communicate the information needed to implement the desired routing policy. PIO's communicate source prefixes, and RIO communicate destination prefixes. However, there is currently no standardized way to directly associate a particular destination prefix with a particular source prefix.

[I-D.pfister-6man-sadr-ra] proposes a Source Address Dependent Route Information option for Neighbor Discovery Router Advertisements which would associate a source prefix and with a destination prefix. The details of [I-D.pfister-6man-sadr-ra] might need tweaking to address this use case. However, in order to be able to use Neighbor Discovery Router Advertisements to implement this routing policy, an extension that allows a R1 and R2 to explicitly communicate to H31 an association between S=2001:db8:0:a000::/52 D=2001:db8:0:1234::/64 would be needed.

However the Rule 5.5 of the source address selection (discussed above) together with default router preference (specified in [RFC4191]) and RIO can be used to influence a source address selection on a host as described below. Let's look at source address selection on the host H41. It receives RAs from R3 with PIOs for 2001:db8:0:a020::/64 and 2001:db8:0:b020::/64. At that point all traffic would use the same next-hop (R3 link-local address) so Rule 5.5 does not apply. Now let's assume that R3 supports SADR and has two scoped forwarding tables, one scoped to S=2001:db8:0:a000::/52 and another scoped to S=2001:db8:0:b000::/52. If R3 generates two different link-local addresses for its interface facing H41 (one for each scoped forwarding table, LLA_A and LLA_B) and starts sending two different RAs: one is sent from LLA_A and includes PIO for 2001:db8:0:a020::/64, another us sent from LLA_B and includes PIO for 2001:db8:0:b020::/64. Now it is possible to influence H41 source address selection for destinations which follow the default route by setting default router preference in RAs. If it is desired that H41 reaches H101 (or any destinations in the Internet) via ISP-A, then RAs sent from LLA_A should have default router preference set to 01 (high priority), while RAs sent from LLA_B should have preference set

to 11 (low). Then LLA_A would be chosen as a next-hop for H101 and therefore (as per rule 5.5) 2001:db8:0:a020::41 would be selected as the source address. If, at the same time, it is desired that H61 is accessible via ISP-B then R3 should include a RIO for 2001:db8:0:6666::/64 to its RA sent from LLA_B. H41 would chose LLA_B as a next-hop for all traffic to H61 and then as per Rule 5.5, 2001:db8:0:b020::41 would be selected as a source address.

If in the above mentioned scenario it is desirable that all Internet traffic leaves the network via ISP-A and the link to ISP-B is used for accessing ISP-B services only (not as ISP-A link backup), then RAs sent by R3 from LLA_B should have Router Lifetime set to 0 and should include RIOs for ISP-B address space. It would instruct H41 to use LLA_A for all Internet traffic but use LLA_B as a next-hop while sending traffic to ISP-B addresses.

The proposed solution relies on SADR support by first-hop routers as well as SERs.

4.2.3. Controlling Source Address Selection With ICMPv6

We now discuss how one might use ICMPv6 to implement the routing policy to send traffic destined for H101 out the uplink to ISP-A, even when uplinks to both ISPs are working. If H31 started sending traffic to H101 with S=2001:db8:0:b010::31 and D=2001:db8:0:1234::101, it would be routed through SER-b1 and out the uplink to ISP-B. SERb1 could recognize that this is traffic is not following the desired routing policy and react by sending an ICMPv6 message back to H31.

In this example, we could arrange things so that SERb1 drops the packet with S=2001:db8:0:b010::31 and D=2001:db8:0:1234::101, and then sends to H31 an ICMPv6 Destination Unreachable message with Code 5 (Source address failed ingress/egress policy). When H31 receives this packet, it would then be expected to try another source address to reach the destination. In this example, H31 would then send a packet with S=2001:db8:0:a010::31 and D=2001:db8:0:1234::101, which will reach SERa and be forwarded out the uplink to ISP-A.

However, we would also want it to be the case that SERb1 does not enforce this routing policy when the uplink from SERa to ISP-A has failed. This could be accomplished by having SERa originate a source-prefix-scoped route for (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64) and have SERb1 monitor the presence of that route. If that route is not present (because SERa has stopped originating it), then SERb1 will not enforce the routing policy, and it will forward packets with S=2001:db8:0:b010::31 and D=2001:db8:0:1234::101 out its uplink to ISP-B.

We can also use this source-prefix-scoped route originated by SERa to communicate the desired routing policy to SERb1. We can define an EXCLUSIVE flag to be advertised together with the IGP route for (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64). This would allow SERa to communicate to SERb that SERb should reject traffic for D=2001:db8:0:1234::/64 and respond with an ICMPv6 Destination Unreachable Code 5 message, as long as the route for (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64) is present.

Finally, if we are willing to extend ICMPv6 to support this solution, then we could create a mechanism for SERb1 to tell the host what source address it should be using to successfully forward packets that meet the policy. In its current form, when SERb1 sends an ICMPv6 Destination Unreachable Code 5 message, it is basically saying, "This source address is wrong. Try another source address." It would be better is if the ICMPv6 message could say, "This source address is wrong. Instead use a source address in S=2001:db8:0:a000::/52."

However using ICMPv6 for signalling source address information back to hosts introduces new challenges. Most routers currently have software or hardware limits on generating ICMP messages. An site administrator deploying a solution that relies on the SERs generating ICMP messages could try to improve the performance of SERs for generating ICMP messages. However, in a large network, it is still likely that ICMP message generation limits will be reached. As a result hosts would not receive ICMPv6 back which in turns leads to traffic blackholing and poor user experience. To improve the scalability of ICMPv6-based signalling hosts SHOULD cache the preferred source address (or prefix) for the given destination. In addition, the same source prefix SHOULD be used for other destinations in the same /64 as the original destination address. The source prefix SHOULD have a specific lifetime. Expiration of the lifetime SHOULD trigger the source address selection algorithm again.

Using ICMPv6 Code 5 message for influencing source address selection allows an attacker to exhaust the list of candidate source addresses on the host by sending spoofed ICMPv6 Code 5 for all prefixes known on the network (therefore preventing a victim from establishing a communication with the destination host). To protect from such attack hosts SHOULD verify that the original packet header included into ICMPv6 error message was actually sent by the host.

4.2.4. Summary of Methods For Controlling Source Address Selection To Implement Routing Policy

So to summarize this section, we have looked at three methods for implementing a simple routing policy where all traffic for a given destination on the Internet needs to use a particular ISP, even when the uplinks to both ISPs are working.

The default source address selection policy cannot distinguish between the source addresses needed to enforce this policy, so a non-default policy table using associating source and destination prefixes using Label values would need to be installed on each host. A mechanism exists for DHCPv6 to distribute a non-default policy table but such solution would heavily rely on DHCPv6 support by host operating system. Moreover there is no mechanism to translate desired routing/traffic engineering policies into policy tables on DHCPv6 servers. Therefore using DHCPv6 for controlling address selection policy table is not recommended and SHOULD NOT be used.

At the same time Router Advertisements provide a reliable mechanism to influence source address selection process via PIO, RIO and default router preferences. As all those options have been standardized by IETF and are supported by various operating systems, no changes are required on hosts. First-hop routers in the enterprise network need to be able of sending different RAs for different SLAAC prefixes (either based on scoped forwarding tables or based on pre-configured policies).

SERs can enforce the routing policy by sending ICMPv6 Destination Unreachable messages with Code 5 (Source address failed ingress/ egress policy) for traffic that is being sent with the wrong source address. The policy distribution can be automated by defining an EXCLUSIVE flag for the source-prefix-scoped route which can be set on the SER that originates the route. As ICMPv6 message generation can be rate-limited on routers, it SHOULD NOT be used as the only mechanism to influence source address selection on hosts. While hosts SHOULD select the correct source address for a given destination the network SHOULD signal any source address issues back to hosts using ICMPv6 error messages.

4.3. Selecting Source Address When One Uplink Has Failed

Now we discuss if DHCPv6, Neighbor Discovery Router Advertisements, and ICMPv6 can help a host choose the right source address when an uplink to one of the ISPs has failed. Again we look at the scenario in Figure 3. This time we look at traffic from H31 destined for external host H501 at D=2001:db8:0:5678::501. We initially assume

that the uplink from SERa to ISP-A is working and that the uplink from SERb1 to ISP-B is working.

We assume there is no particular routing policy desired, so H31 is free to send packets with S=2001:db8:0:a010::31 or S=2001:db8:0:b010::31 and have them delivered to H501. For this example, we assume that H31 has chosen S=2001:db8:0:b010::31 so that the packets exit via SERb to ISP-B. Now we see what happens when the link from SERb1 to ISP-B fails. How should H31 learn that it needs to start sending the packet to H501 with S=2001:db8:0:a010::31 in order to start using the uplink to ISP-A? We need to do this in a way that doesn't prevent H31 from still sending packets with S=2001:db8:0:b010::31 in order to reach H61 at D=2001:db8:0:6666::61.

4.3.1. Controlling Source Address Selection With DHCPv6

For this example we assume that the site network in Figure 3 has a centralized DHCP server and all routers act as DHCP relay agents. We assume that both of the addresses assigned to H31 were assigned via DHCP.

We could try to have the DHCP server monitor the state of the uplink from SERb1 to ISP-B in some manner and then tell H31 that it can no longer use S=2001:db8:0:b010::31 by settings its valid lifetime to zero. The DHCP server could initiate this process by sending a Reconfigure Message to H31 as described in Section 19 of [RFC3315]. Or the DHCP server can assign addresses with short lifetimes in order to force clients to renew them often.

This approach would prevent H31 from using S=2001:db8:0:b010::31 to reach the a host on the Internet. However, it would also prevent H31 from using S=2001:db8:0:b010::31 to reach H61 at D=2001:db8:0:6666::61, which is not desirable.

Another potential approach is to have the DHCP server monitor the uplink from SERb1 to ISP-B and control the choice of source address on H31 by updating its address selection policy table via the mechanism in [RFC7078]. The DHCP server could initiate this process by sending a Reconfigure Message to H31. Note that [RFC3315] requires that Reconfigure Message use DHCP authentication. DHCP authentication could be avoided by using short address lifetimes to force clients to send Renew messages to the server often. If the host is not obtaining its IP addresses from the DHCP server, then it would need to use the Information Refresh Time option defined in [RFC4242].

If the following policy table can be installed on H31 after the failure of the uplink from SERb1, then the desired routing behavior

should be achieved based on source and destination prefix being matched with label values.

Prefix	Precedence	Label
::/0	50	44
2001:db8:0:a000::/52	50	44
2001:db8:0:6666::/64	50	55
2001:db8:0:b000::/52	50	55

Figure 10: Policy Table Needed On Failure Of Uplink From SERb1

The described solution has a number of significant drawbacks, some of them already discussed in Section 4.2.1.

- o DHCPv6 support is not required for an IPv6 host and there are operating systems which do not support DHCPv6. Besides that, it does not appear that [RFC7078] has been widely implemented on host operating systems.

- o [RFC7078] does not clearly specify this kind of a dynamic use case where address selection policy needs to be updated quickly in response to the failure of a link. In a large network it would present scalability issues as many hosts need to be reconfigured in very short period of time.

- o No mechanism exists for making DHCPv6 servers aware of network topology/routing changes in the network. In general DHCPv6 servers monitoring network-related events sounds like a bad idea as completely new functionality beyond the scope of DHCPv6 role is required.

4.3.2. Controlling Source Address Selection With Router Advertisements

The same mechanism as discussed in Section 4.2.2 can be used to control the source address selection in the case of an uplink failure. If a particular prefix should not be used as a source for any destinations, then the router needs to send RA with Preferred Lifetime field for that prefix set to 0.

Let’s consider a scenario when all uplinks are operational and H41 receives two different RAs from R3: one from LLA_A with PIO for 2001:db8:0:a020::/64, default router preference set to 11 (low) and another one from LLA_B with PIO for 2001:db8:0:a020::/64, default router preference set to 01 (high) and RIO for 2001:db8:0:6666::/64. As a result H41 is using 2001:db8:0:b020::41 as a source address for all Internet traffic and those packets are sent by SERs to ISP-B. If SERb1 uplink to ISP-B failed, the desired behavior is that H41 stops

using 2001:db8:0:b020::41 as a source address for all destinations but H61. To achieve that R3 should react to SERb1 uplink failure (which could be detected as the scoped route (S=2001:db8:0:b000::/52, D=::/0) disappearance) by withdrawing itself as a default router. R3 sends a new RA from LLA_B with Router Lifetime value set to 0 (which means that it should not be used as default router). That RA still contains PIO for 2001:db8:0:b020::/64 (for SLAAC purposes) and RIO for 2001:db8:0:6666::/64 so H41 can reach H61 using LLA_B as a next-hop and 2001:db8:0:b020::41 as a source address. For all traffic following the default route, LLA_A will be used as a next-hop and 2001:db8:0:a020::41 as a source address.

If all uplinks to ISP-B have failed and therefore source addresses from ISP-B address space should not be used at all, the forwarding table scoped S=2001:db8:0:b000::/52 contains no entries. Hosts can be instructed to stop using source addresses from that block by sending RAs containing PIO with Preferred Lifetime set to 0.

4.3.3. Controlling Source Address Selection With ICMPv6

Now we look at how ICMPv6 messages can provide information back to H31. We assume again that at the time of the failure H31 is sending packets to H501 using (S=2001:db8:0:b010::31, D=2001:db8:0:5678::501). When the uplink from SERb1 to ISP-B fails, SERb1 would stop originating its source-prefix-scoped route for the default destination (S=2001:db8:0:b000::/52, D=::/0) as well as its unscoped default destination route. With these routes no longer in the IGP, traffic with (S=2001:db8:0:b010::31, D=2001:db8:0:5678::501) would end up at SERa based on the unscoped default destination route being originated by SERa. Since that traffic has the wrong source address to be forwarded to ISP-A, SERa would drop it and send a Destination Unreachable message with Code 5 (Source address failed ingress/egress policy) back to H31. H31 would then know to use another source address for that destination and would try with (S=2001:db8:0:a010::31, D=2001:db8:0:5678::501). This would be forwarded to SERa based on the source-prefix-scoped default destination route still being originated by SERa, and SERa would forward it to ISP-A. As discussed above, if we are willing to extend ICMPv6, SERa can even tell H31 what source address it should use to reach that destination. The expected host behaviour has been discussed in Section 4.2.3. Potential issue with using ICMPv6 for signalling source address issues back to hosts is that uplink to an ISP-B failure immediately invalidates source addresses from 2001:db8:0:b000::/52 for all hosts which triggers a large number of ICMPv6 being sent back to hosts - the same scalability/rate limiting issues discussed in Section 4.2.3 would apply.

4.3.4. Summary Of Methods For Controlling Source Address Selection On The Failure Of An Uplink

It appears that DHCPv6 is not particularly well suited to quickly changing the source address used by a host in the event of the failure of an uplink, which eliminates DHCPv6 from the list of potential solutions. On the other hand Router Advertisements provides a reliable mechanism to dynamically provide hosts with a list of valid prefixes to use as source addresses as well as prevent particular prefixes to be used. While no additional new features are required to be implemented on hosts, routers need to be able to send RAs based on the state of scoped forwarding tables entries and to react to network topology changes by sending RAs with particular parameters set.

The use of ICMPv6 Destination Unreachable messages generated by the SER (or any SADR-capable) routers seem like they have the potential to provide a support mechanism together with RAs to signal source address selection errors back to hosts, however scalability issues may arise in large networks in case of sudden topology change. Therefore it is highly desirable that hosts are able to select the correct source address in case of uplinks failure with ICMPv6 being an additional mechanism to signal unexpected failures back to hosts.

The current behavior of different host operating system when receiving ICMPv6 Destination Unreachable message with code 5 (Source address failed ingress/egress policy) is not clear to the authors. Information from implementers, users, and testing would be quite helpful in evaluating this approach.

4.4. Selecting Source Address Upon Failed Uplink Recovery

The next logical step is to look at the scenario when a failed uplink on SERb1 to ISP-B is coming back up, so hosts can start using source addresses belonging to 2001:db8:0:b000::/52 again.

4.4.1. Controlling Source Address Selection With DHCPv6

The mechanism to use DHCPv6 to instruct the hosts (H31 in our example) to start using prefixes from ISP-B space (e.g. S=2001:db8:0:b010::31 for H31) to reach hosts on the Internet is quite similar to one discussed in Section 4.3.1 and shares the same drawbacks.

4.4.2. Controlling Source Address Selection With Router Advertisements

Let’s look at the scenario discussed in Section 4.3.2. If the uplink(s) failure caused the complete withdrawal of prefixes from 2001:db8:0:b000::/52 address space by setting Preferred Lifetime value to 0, then the recovery of the link should just trigger new RA being sent with non-zero Preferred Lifetime. In another scenario discussed in Section 4.3.2, the SERb1 uplink to ISP-B failure leads to disappearance of the (S=2001:db8:0:b000::/52, D=::/0) entry from the forwarding table scoped to S=2001:db8:0:b000::/52 and, in turn, caused R3 to send RAs from LLA_B with Router Lifetime set to 0. The recovery of the SERb1 uplink to ISP-B leads to (S=2001:db8:0:b000::/52, D=::/0) scoped forwarding entry re-appearance and instructs R3 that it should advertise itself as a default router for ISP-B address space domain (send RAs from LLA_B with non-zero Router Lifetime).

4.4.3. Controlling Source Address Selection With ICMP

It looks like ICMPv6 provides a rather limited functionality to signal back to hosts that particular source addresses have become valid again. Unless the changes in the uplink state a particular (S,D) pair, hosts can keep using the same source address even after an ISP uplink has come back up. For example, after the uplink from SERb1 to ISP-B had failed, H31 received ICMPv6 Code 5 message (as described in Section 4.3.3) and allegedly started using (S=2001:db8:0:a010::31, D=2001:db8:0:5678::501) to reach H501. Now when the SERb1 uplink comes back up, the packets with that (S,D) pair are still routed to SERa1 and sent to the Internet. Therefore H31 is not informed that it should stop using 2001:db8:0:a010::31 and start using 2001:db8:0:b010::31 again. Unless SERa has a policy configured to drop packets (S=2001:db8:0:a010::31, D=2001:db8:0:5678::501) and send ICMPv6 back if SERb1 uplink to ISP-B is up, H31 will be unaware of the network topology change and keep using S=2001:db8:0:a010::31 for Internet destinations, including H51.

One of the possible option may be using a scoped route with EXCLUSIVE flag as described in Section 4.2.3. SERa1 uplink recovery would cause (S=2001:db8:0:a000::/52, D=2001:db8:0:1234::/64) route to reappear in the routing table. In the absence of that route packets to H101 which were sent to ISP-B (as ISP-A uplink was down) with source addresses from 2001:db8:0:b000::/52. When the route re-appears SERb1 would reject those packets and sends ICMPv6 back as discussed in Section 4.2.3. Practically it might lead to scalability issues which have been already discussed in Section 4.2.3 and Section 4.4.3.

4.4.4. Summary Of Methods For Controlling Source Address Selection Upon Failed Uplink Recovery

Once again DHCPv6 does not look like reasonable choice to manipulate source address selection process on a host in the case of network topology changes. Using Router Advertisement provides the flexible mechanism to dynamically react to network topology changes (if routers are able to use routing changes as a trigger for sending out RAs with specific parameters). ICMPv6 could be considered as a supporting mechanism to signal incorrect source address back to hosts but should not be considered as the only mechanism to control the address selection in multihomed environments.

4.5. Selecting Source Address When All Uplinks Failed

One particular tricky case is a scenario when all uplinks have failed. In that case there is no valid source address to be used for any external destinations while it might be desirable to have intra-site connectivity.

4.5.1. Controlling Source Address Selection With DHCPv6

From DHCPv6 perspective uplinks failure should be treated as two independent failures and processed as described in Section 4.3.1. At this stage it is quite obvious that it would result in quite complicated policy table which needs to be explicitly configured by administrators and therefore seems to be impractical.

4.5.2. Controlling Source Address Selection With Router Advertisements

As discussed in Section 4.3.2 an uplink failure causes the scoped default entry to disappear from the scoped forwarding table and triggers RAs with zero Router Lifetime. Complete disappearance of all scoped entries for a given source prefix would cause the prefix being withdrawn from hosts by setting Preferred Lifetime value to zero in PIO. If all uplinks (SERa, SERb1 and SERb2) failed, hosts either lost their default routers and/or have no global IPv6 addresses to use as a source. (Note that ’uplink failure’ might mean ’IPv6 connectivity failure with IPv4 still being reachable’, in which case hosts might fall back to IPv4 if there is IPv4 connectivity to destinations). As a results intra-site connectivity is broken. One of the possible way to solve it is to use ULAs.

All hosts have ULA addresses assigned in addition to GUAs and used for intra-site communication even if there is no GUA assigned to a host. To avoid accidental leaking of packets with ULA sources SADR-capable routers SHOULD have a scoped forwarding table for ULA source for internal routes but MUST NOT have an entry for D=::/0 in that

table. In the absence of (S=ULA_Prefix; D=::/0) first-hop routers will send dedicated RAs from a unique link-local source LLA_ULA with PIO from ULA address space, RIO for the ULA prefix and Router Lifetime set to zero. The behaviour is consistent with the situation when SERb1 lost the uplink to ISP-B (so there is no Internet connectivity from 2001:db8:0:b000::/52 sources) but those sources can be used to reach some specific destinations. In the case of ULA there is no Internet connectivity from ULA sources but they can be used to reach another ULA destinations. Note that ULA usage could be particularly useful if all ISPs assign prefixes via DHCP-PD. In the absence of ULAs uplinks failure hosts would lost all their GUAs upon prefix lifetime expiration which again makes intra-site communication impossible.

4.5.3. Controlling Source Address Selection With ICMPv6

In case of all uplinks failure all SERs will drop outgoing IPv6 traffic and respond with ICMPv6 error message. In the large network when many hosts are trying to reach Internet destinations it means that SERs need to generate an ICMPv6 error to every packet they receive from hosts which presents the same scalability issues discussed in Section 4.3.3

4.5.4. Summary Of Methods For Controlling Source Address Selection When All Uplinks Failed

Again, combining SADR with Router Advertisements seems to be the most flexible and scalable way to control the source address selection on hosts.

4.6. Summary Of Methods For Controlling Source Address Selection

To summarize the scenarios and options discussed above:

While DHCPv6 allows administrators to manipulate source address selection policy tables, this method has a number of significant disadvantages which eliminates DHCPv6 from a list of potential solutions:

1. It required hosts to support DHCPv6 and its extension (RFC7078);

2. DHCPv6 server need to monitor network state and detect routing changes.

3. Network topology/routing policy changes could trigger simultaneous re-configuration of large number of hosts which present serious scalability issues.

The use of Router Advertisements to influence the source address selection on hosts seem to be the most reliable, flexible and scalable solution. It has the following benefits:

1. no new (non-standard) functionality needs to be implemented on hosts (except for [RFC4191] support);

2. no changes in RA format;

3. Routers can react to routing table changes by sending RAs which would minimize the failover time in the case of network topology changes;

4. information required for source address selection is broadcast to all affected hosts in case of topology change event which improves the scalability of the solution (comparing to DHCPv6 reconfiguration or ICMPv6 error messages).

To fully benefit from the RA-based solution, first-hop routers need to implement SADR and be able to send dedicated RAs per scoped forwarding table as discussed above, reacting to network changes with sending new RAs. It should be noted that the proposed solution would work even if first-hop routers are not SADR-capable but still able to send individual RAs for each ISP prefix and react to topology changes as discussed above

The RA-based solution relies heavily on hosts correctly implementing default address selection algorith as defined in [RFC6724] and in particular, Rule 5.5. There are some evidences that not all host OSes have that rule implemented currenly (it should be noted that [I-D.ietf-6man-multi-homed-host] states that Rule 5.5 SHOULD be implemented.

ICMPv6 Code 5 error message SHOULD be used to complement RA-based solution to signal incorrect source address selection back to hosts, but it SHOULD NOT be considered as the stand-alone solution. To prevent scenarios when hosts in multihomed envinronments incorrectly identify onlink/offlink destimations, hosts should treat ICMPv6 Redirects as discussed in [I-D.ietf-6man-multi-homed-host].

4.7. Other Configuration Parameters

4.7.1. DNS Configuration

In mutihomed envinronment each ISP might provide their own list of DNS servers. E.g. in the topology show on Figure 3, ISP-A might provide recursive DNS server H51 2001:db8:0:5555::51, while ISP-B might provide H61 2001:db8:0:6666::61 as a recursive DNS server. If

the multihomed enterprise network is not running their own recursive resolver then hosts need to be configured with DNS server IPv6 addresses. [RFC6106] defines IPv6 Router Advertisement options to allow IPv6 routers to advertise a list of DNS recursive server addresses and a DNS Search List to IPv6 hosts. Using RDNSS together with 'scoped' RAs as described above would allow a first-hop router (R3 in the Figure 3) to send DNS server addresses and search lists provided by each ISPs.

As discussed in Section 4.5.2, failure of all ISP uplinks would cause depreaction of all addresses assigned to a host from ISPs address space. Most likely intra-site IPv6 connectivity would be still desirable so Section 4.5.2 proposes a usage of ULAs to enable intra-site communication. In such scenario the enterprise network should run its own recursive DNS server(s) and provide its ULA addresses to hosts via RDNSS mechanism in RAs send for ULA-scoped forwarding table as described in Section 4.5.2.

It should be noted that [RFC6106] explicitly prohibits using DNS information if the RA router Lifetime expired: "An RDNSS address or a DNSSL domain name MUST be used only as long as both the RA router Lifetime (advertised by a Router Advertisement message) and the corresponding option Lifetime have not expired.". Therefore hosts might ignore RDNSS information provided in ULA-scoped RAs as those RAs would have router lifetime set to 0. However the updated version of RFC6106 ([I-D.ietf-6man-rdnss-rfc6106bis]) has that requirement removed.

5. Other Solutions

5.1. Shim6

The Shim6 working group specified the Shim6 protocol [RFC5533] which allows a host at a multihomed site to communicate with an external host and exchange information about possible source and destination address pairs that they can use to communicate. It also specified the REAP protocol [RFC5534] to detect failures in the path between working address pairs and find new working address pairs. A fundamental requirement for Shim6 is that both internal and external hosts need to support Shim6. That is, both the host internal to the multihomed site and the host external to the multihomed site need to support Shim6 in order for there to be any benefit for the internal host to run Shim6. The Shim6 protocol specification was published in 2009, but it has not been implemented on widely used operating systems.

We do not consider Shim6 to be a viable solution. It suffers from the fact that it requires widespread deployment of Shim6 on hosts all

over the Internet before the host at a PA multihomed site sees significant benefit. However, there appears to be no motivation for the vast majority of hosts on the Internet (which are not at PA multihomed sites) to deploy Shim6. This may help explain why Shim6 has not been widely implemented.

5.2. IPv6-to-IPv6 Network Prefix Translation

IPv6-to-IPv6 Network Prefix Translation (NPTv6) [RFC6296] is not the focus of this document. This document describes a solution where a host in a multihomed site determines which ISP a packet will be sent to based on the source address it applies to the packet. This solution has many moving parts. It requires some routers in the enterprise site to support some form of Source Address Dependent Routing (SADR). It requires a host to be able to learn when the uplink to an ISP fails so that it can stop using the source address corresponding to that ISP. Ongoing work to create mechanisms to accomplish this are discussed in this document, but they are still a work in progress.

This document attempts to create a PA multihoming solution that is as easy as possible for an enterprise to deploy. However, the success of this solution will depend greatly on whether or not the mechanisms for hosts to select source addresses based on the state of ISP uplinks gets implemented across a wide range of operating systems as the default mode of operation. Until that occurs, NPTv6 should still be considered a viable option to enable PA multihoming for enterprises.

6. IANA Considerations

This memo asks the IANA for no new parameters.

7. Security Considerations

7.1. Privacy Considerations

8. Acknowledgements

The original outline was suggested by Ole Troan.

9. References

9.1. Normative References

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <http://www.rfc-editor.org/info/rfc1123>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

[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>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3582] Abley, J., Black, B., and V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC 3582, DOI 10.17487/RFC3582, August 2003, <http://www.rfc-editor.org/info/rfc3582>.

[RFC4116] Abley, J., Lindqvist, K., Davies, E., Black, B., and V. Gill, "IPv4 Multihoming Practices and Limitations", RFC 4116, DOI 10.17487/RFC4116, July 2005, <http://www.rfc-editor.org/info/rfc4116>.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <http://www.rfc-editor.org/info/rfc4191>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <http://www.rfc-editor.org/info/rfc4193>.

[RFC4218] Nordmark, E. and T. Li, "Threats Relating to IPv6 Multihoming Solutions", RFC 4218, DOI 10.17487/RFC4218, October 2005, <http://www.rfc-editor.org/info/rfc4218>.

[RFC4219] Lear, E., "Things Multihoming in IPv6 (MULTI6) Developers Should Think About", RFC 4219, DOI 10.17487/RFC4219, October 2005, <http://www.rfc-editor.org/info/rfc4219>.

[RFC4242] Venaas, S., Chown, T., and B. Volz, "Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 4242, DOI 10.17487/RFC4242, November 2005, <http://www.rfc-editor.org/info/rfc4242>.

[RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, DOI 10.17487/RFC6106, November 2010, <http://www.rfc-editor.org/info/rfc6106>.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <http://www.rfc-editor.org/info/rfc6296>.

[RFC7157] Troan, O., Ed., Miles, D., Matsushima, S., Okimoto, T., and D. Wing, "IPv6 Multihoming without Network Address Translation", RFC 7157, DOI 10.17487/RFC7157, March 2014, <http://www.rfc-editor.org/info/rfc7157>.

9.2. Informative References

[I-D.baker-ipv6-isis-dst-src-routing]
Baker, F. and D. Lamparter, "IPv6 Source/Destination Routing using IS-IS", draft-baker-ipv6-isis-dst-src-routing-06 (work in progress), October 2016.

[I-D.baker-rtgwg-src-dst-routing-use-cases]
Baker, F., Xu, M., Yang, S., and J. Wu, "Requirements and Use Cases for Source/Destination Routing", draft-baker-rtgwg-src-dst-routing-use-cases-02 (work in progress), April 2016.

[I-D.boutier-babel-source-specific]
Boutier, M. and J. Chroboczek, "Source-Specific Routing in Babel", draft-boutier-babel-source-specific-01 (work in progress), May 2015.

[I-D.huitema-shim6-ingress-filtering]
Huitema, C., "Ingress filtering compatibility for IPv6 multihomed sites", draft-huitema-shim6-ingress-filtering-00 (work in progress), September 2005.

[I-D.ietf-6man-multi-homed-host]
Baker, F. and B. Carpenter, "First-hop router selection by hosts in a multi-prefix network", draft-ietf-6man-multi-homed-host-10 (work in progress), October 2016.

[I-D.ietf-6man-rdnss-rfc6106bis]
Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", draft-ietf-6man-rdnss-rfc6106bis-14 (work in progress), June 2016.

[I-D.ietf-mif-mpvd-arch]
Anipko, D., "Multiple Provisioning Domain Architecture", draft-ietf-mif-mpvd-arch-11 (work in progress), March 2015.

[I-D.ietf-mptcp-experience]
Bonaventure, O., Paasch, C., and G. Detal, "Use Cases and Operational Experience with Multipath TCP", draft-ietf-mptcp-experience-07 (work in progress), October 2016.

[I-D.ietf-rtgwg-dst-src-routing]
Lamparter, D. and A. Smirnov, "Destination/Source Routing", draft-ietf-rtgwg-dst-src-routing-02 (work in progress), May 2016.

[I-D.pfister-6man-sadr-ra]
Pfister, P., "Source Address Dependent Route Information Option for Router Advertisements", draft-pfister-6man-sadr-ra-01 (work in progress), June 2015.

[I-D.xu-src-dst-bgp]
Xu, M., Yang, S., and J. Wu, "Source/Destination Routing Using BGP-4", draft-xu-src-dst-bgp-00 (work in progress), March 2016.

[PATRICIA]
Morrison, D., "Practical Algorithm to Retrieve Information Coded in Alphanumeric", Journal of the ACM 15(4) pp514-534, October 1968.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <http://www.rfc-editor.org/info/rfc3704>.

[RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, DOI 10.17487/RFC3736, April 2004, <http://www.rfc-editor.org/info/rfc3736>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <http://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, DOI 10.17487/RFC5533, June 2009, <http://www.rfc-editor.org/info/rfc5533>.

[RFC5534] Arkko, J. and I. van Beijnum, "Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming", RFC 5534, DOI 10.17487/RFC5534, June 2009, <http://www.rfc-editor.org/info/rfc5534>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

[RFC7078] Matsumoto, A., Fujisaki, T., and T. Chown, "Distributing Address Selection Policy Using DHCPv6", RFC 7078, DOI 10.17487/RFC7078, January 2014, <http://www.rfc-editor.org/info/rfc7078>.

[RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <http://www.rfc-editor.org/info/rfc7788>.

Appendix A. Change Log

Initial Version: July 2016

Authors' Addresses

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

Email: fred@cisco.com

Chris Bowers
Juniper Networks
Sunnyvale, California 94089
USA

Email: cbowers@juniper.net

Jen Linkova
Google
Mountain View, California 94043
USA

Email: furry@google.com

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

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

July 2, 2017

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

Abstract

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

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

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

2. Conventions

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

Abbreviations used in this document:

E2E Edge to Edge

Geneve: Generic Network Virtualization Encapsulation [I-D.ietf-nvo3-geneve]

IOAM: In-situ Operations, Administration, and Maintenance

MTU: Maximum Transmit Unit

NSH: Network Service Header [I-D.ietf-sfc-nsh]

OAM: Operations, Administration, and Maintenance

POT: Proof of Transit

SFC: Service Function Chain

SID: Segment Identifier

SR: Segment Routing

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

3. Scope, Applicability, and Assumptions

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

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

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

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

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

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

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

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

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

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

4. IOAM Data Types and Formats

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

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

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

4.1. IOAM Tracing Options

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

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

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

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

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

Every node data entry is to hold information for a particular IOAM transit node that is traversed by a packet. The in-situ OAM decapsulating node removes the IOAM data and processes and/or exports the metadata. IOAM data uses its own name-space for information such as node identifier or interface identifier. This allows for a domain-specific definition and interpretation. For example: In one case an interface-id could point to a physical interface (e.g., to understand which physical interface of an aggregated link is used when receiving or transmitting a packet) whereas in another case it could refer to a logical interface (e.g., in case of tunnels).

The following IOAM data is defined for IOAM tracing:

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

- Identification of the interface that a packet was received on, i.e. ingress interface.

- Identification of the interface that a packet was sent out on, i.e. egress interface.

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

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

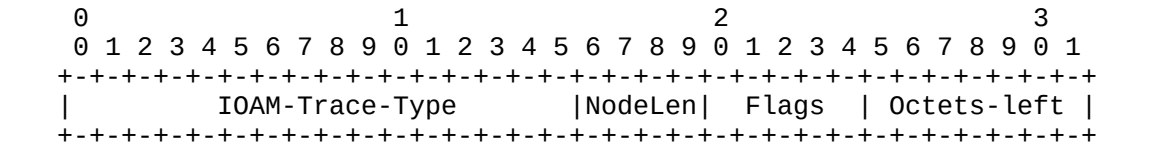
- A mechanism to detect whether IOAM trace data was added at every hop or whether certain hops in the domain weren't in-situ OAM transit nodes.

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

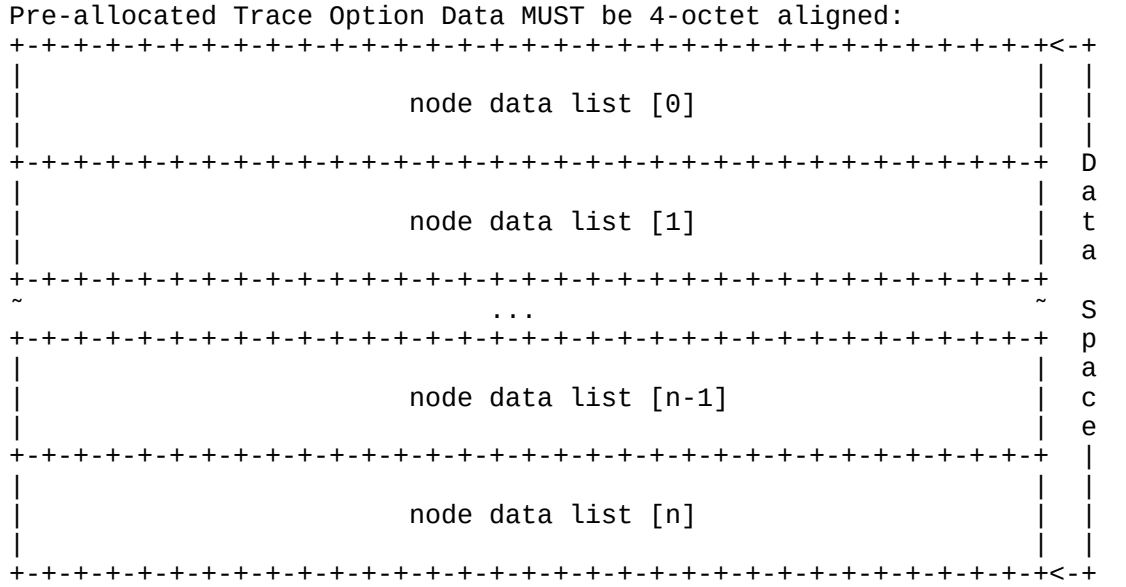
4.1.1. Pre-allocated Trace Option

In-situ OAM pre-allocated trace option:

Pre-allocated trace option header:



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



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

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

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

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

Bit 2 When set indicates presence of timestamp seconds in the node data

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

Bit 4 When set indicates presence of transit delay in the node data.

Bit 5 When set indicates presence of app_data (short format) in the node data.

Bit 6 When set indicates presence of queue depth in the node data.

Bit 7 When set indicates presence of variable length Opaque State Snapshot field.

Bit 8 When set indicates presence of Hop_Lim and node_id in wide format in the node data.

Bit 9 When set indicates presence of ingress_if_id and egress_if_id in wide format in the node data.

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

Bit 11 When set indicates presence of the Checksum Complement node data.

Bit 12-15 Undefined in this draft.

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

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

Flags 5-bit field. Following flags are defined:

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

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

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

Bit 2-4 Reserved: Must be zero.

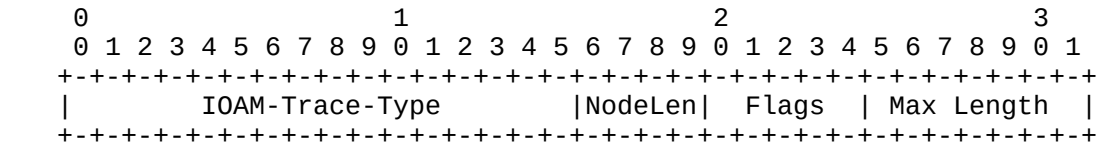
Octets-left: 7-bit unsigned integer. It is the data space in multiples of 4-octets remaining for recording the node data. This is used as an offset in data space to record the node data element.

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

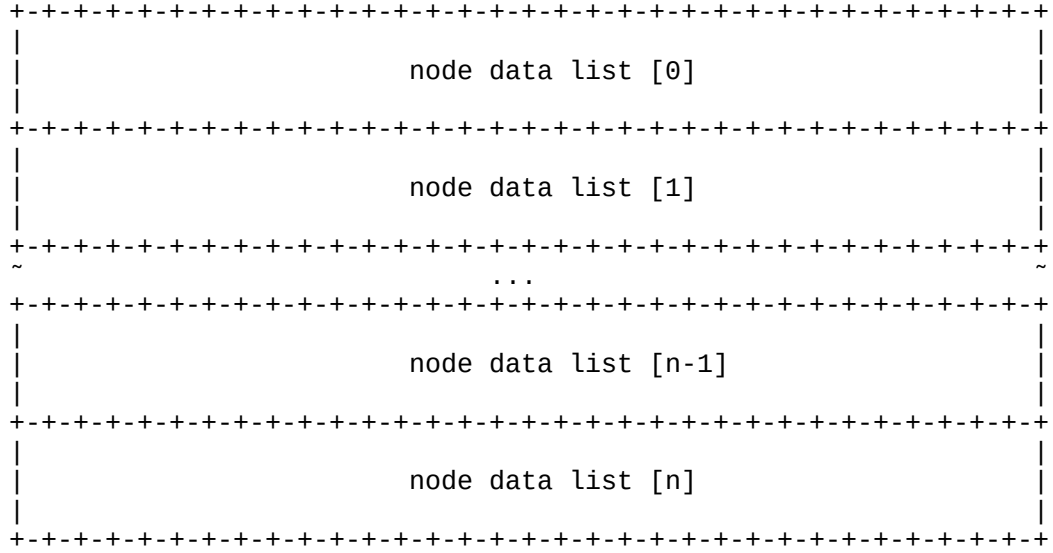
4.1.2. Incremental Trace Option

In-situ OAM incremental trace option:

In-situ OAM incremental trace option Header:



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



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

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

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

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

Bit 2 When set indicates presence of timestamp seconds in the node data.

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

Bit 4 When set indicates presence of transit delay in the node data.

Bit 5 When set indicates presence of app_data in the node data.

Bit 6 When set indicates presence of queue depth in the node data.

Bit 7 When set indicates presence of variable length Opaque State Snapshot field.

Bit 8 When set indicates presence of Hop_Lim and node_id wide in the node data.

Bit 9 When set indicates presence of ingress_if_id and egress_if_id in wide format in the node data.

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

Bit 11 When set indicates presence of the Checksum Complement node data.

Bit 12-15 Undefined in this draft.

Section 4.1.3 describes the IOAM data types and their formats.

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

Flags 5-bit field. Following flags are defined:

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

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

Bit 2-4 Reserved. Must be zero.

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

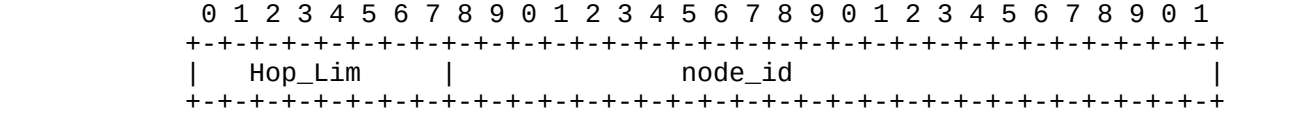
Node data List [n]: Variable-length field. The type of which is determined by the OAM Type representing the n-th node data in the node data list. The node data list is encoded starting from the last node data of the path. The first element of the node data list (node data list [0]) contains the last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

4.1.3. IOAM node data fields and associated formats

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

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

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

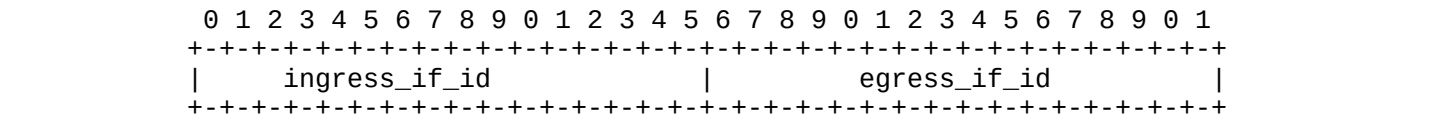


Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop Limit information is used to identify the location of the node

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

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

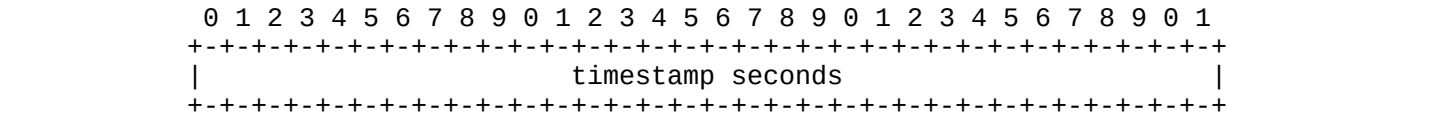
ingress_if_id and egress_if_id: 4-octet field defined as follows: When this field is part of the data field but a node populating the field is not able to fill it, the position in the field must be filled with value 0xFFFFFFFF to mean not populated.



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

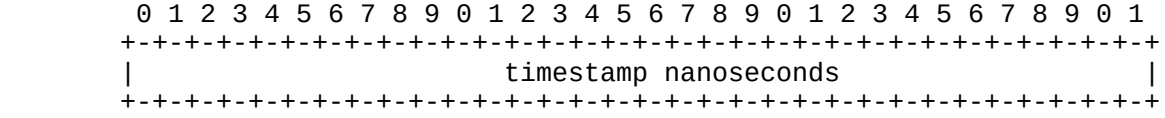
egress_if_id: 2-octet unsigned integer. Interface identifier to record the egress interface the packet is forwarded out of.

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

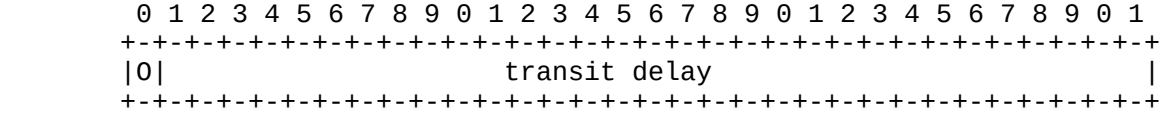


timestamp nanoseconds: 4-octet unsigned integer in the range 0 to 10^9-1. This timestamp specifies the fractional part of the wall clock time at which the packet was received by the node in units of nanoseconds. This field is identical to the 32 least significant bits of the [IEEE1588v2] timestamp. This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

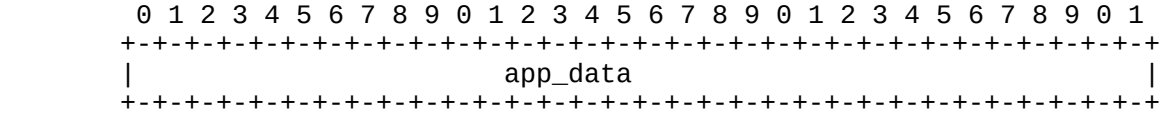
allows for delay computation between any two nodes in the network when the nodes are time synchronized. When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.



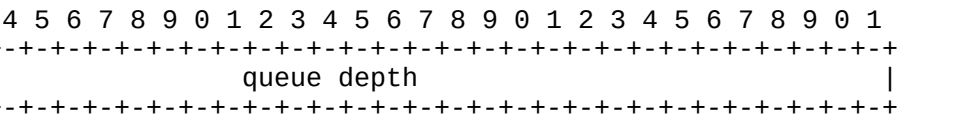
transit delay: 4-octet unsigned integer in the range 0 to 2^30-1. It is the time in nanoseconds the packet spent in the transit node. This can serve as an indication of the queuing delay at the node. If the transit delay exceeds 2^30-1 nanoseconds then the top bit 'O' is set to indicate overflow. When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFF to mean not populated.



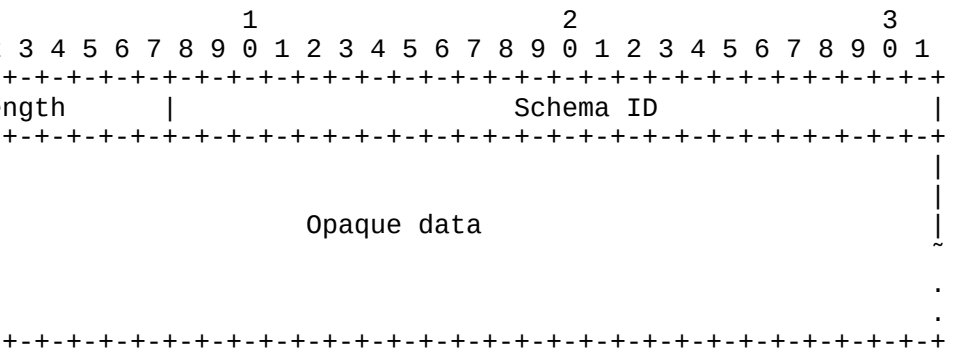
app_data: 4-octet placeholder which can be used by the node to add application specific data. App_data represents a "free-format" 4-octet bit field with its semantics defined by a specific deployment.



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



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

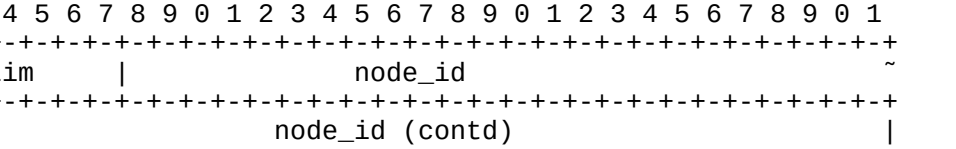


Length: 1-octet unsigned integer. It is the length in octets of the Opaque data field that follows Schema Id. It MUST always be a multiple of 4.

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

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

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

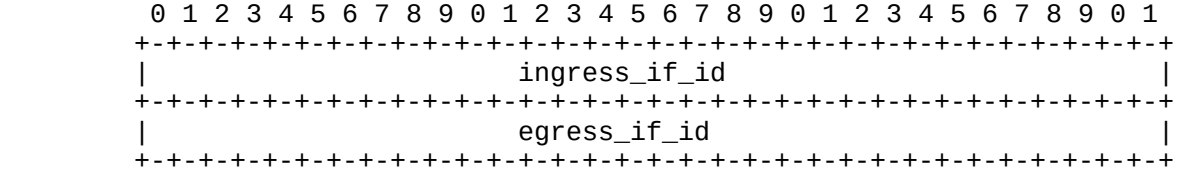


Hop_Lim: 1-octet unsigned integer. It is set to the Hop Limit value in the packet at the node that records this data. Hop

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

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

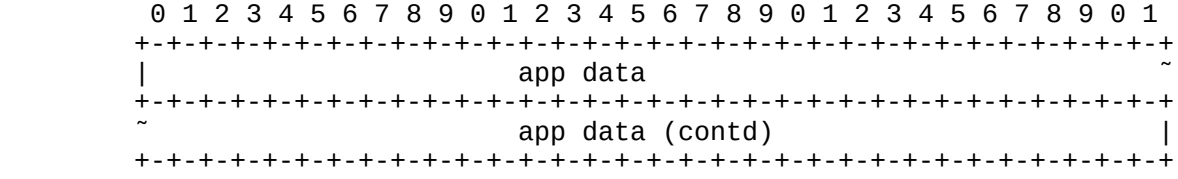
ingress_if_id and egress_if_id wide: 8-octet field defined as follows: When this field is part of the data field but a node populating the field is not able to fill it, the field position in the field must be filled with value 0xFFFFFFFFFFFFFFFF to mean not populated.



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

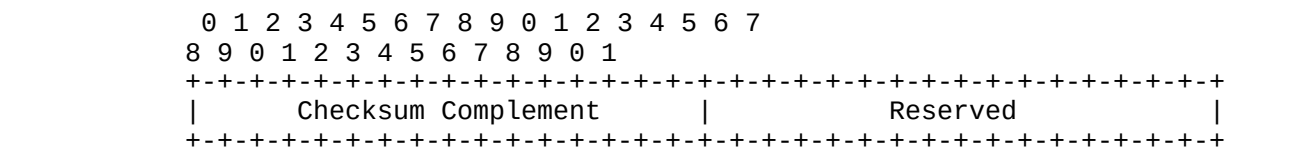
egress_if_id: 4-octet unsigned integer. Interface identifier to record the egress interface the packet is forwarded out of.

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



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

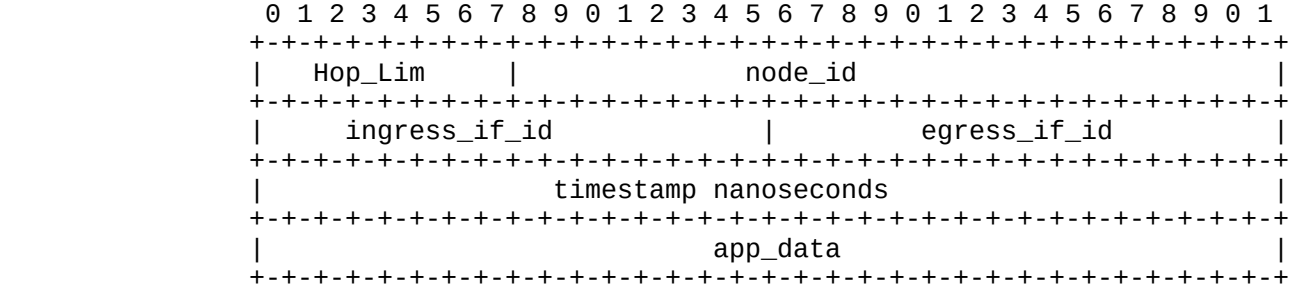
or Geneve. In this case, incorporating the IOAM node data requires the UDP Checksum field to be updated. Rather than to recompute the Chekcsum field, a node can use the Checksum Complement to make a checksum-neutral update in the UDP payload; the Checksum Complement is assigned a value that complements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



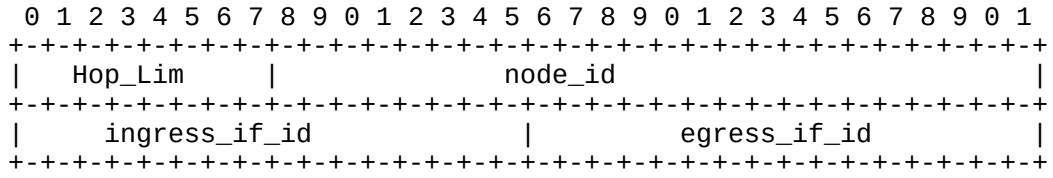
4.1.4. Examples of IOAM node data

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

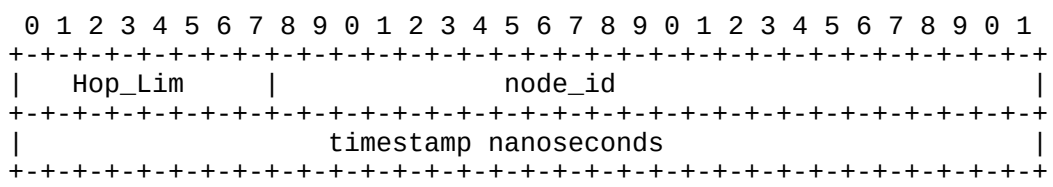
0x002B: IOAM-Trace-Type is 0x2B then the format of node data is:



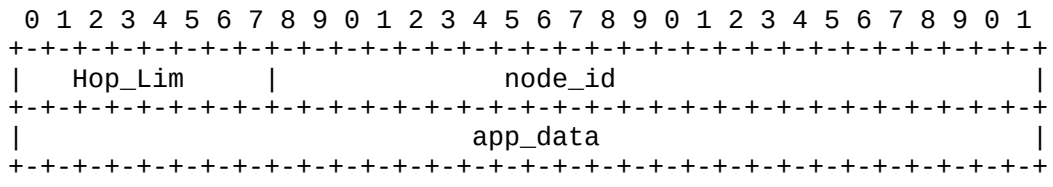
0x0003: IOAM-Trace-Type is 0x0003 then the format is:



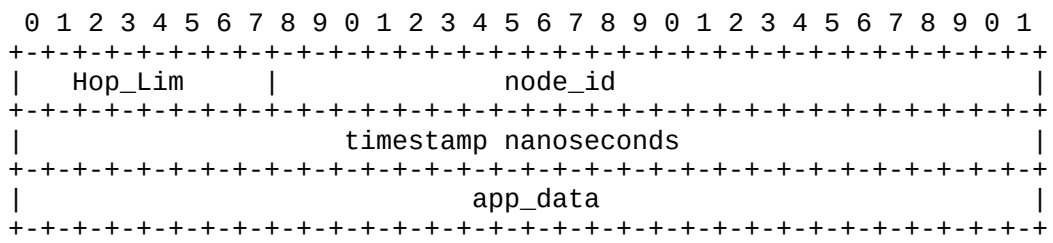
0x0009: IOAM-Trace-Type is 0x0009 then the format is:



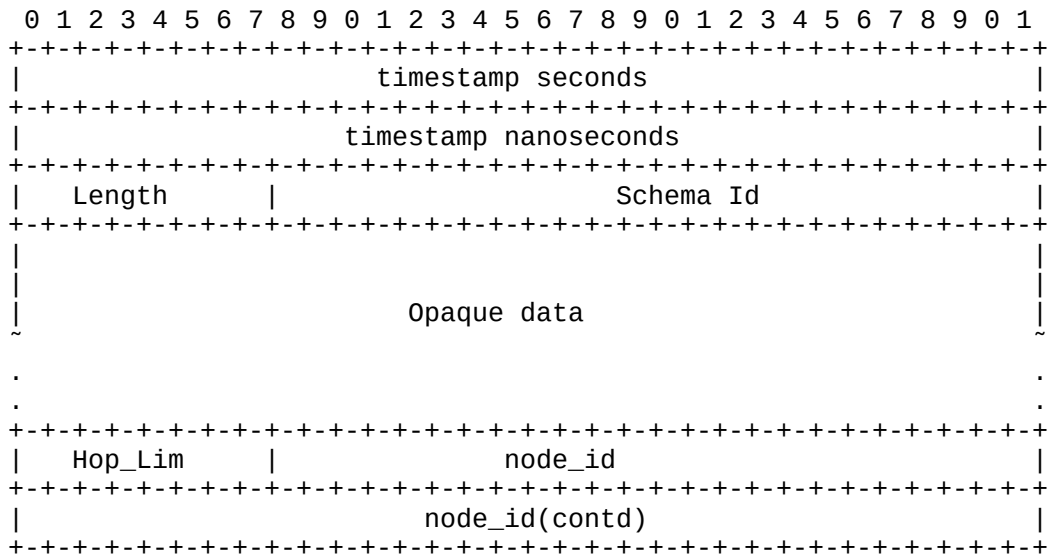
0x0021: IOAM-Trace-Type is 0x0021 then the format is:



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



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



4.2. IOAM Proof of Transit Option

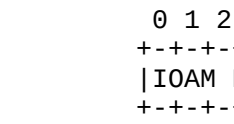
IOAM Proof of Transit data is to support the path or service function chain [RFC7665] verification use cases. Proof-of-transit uses methods like nested hashing or nested encryption of the IOAM data or mechanisms such as Shamir's Secret Sharing Schema (SSSS). While details on how the IOAM data for the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

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

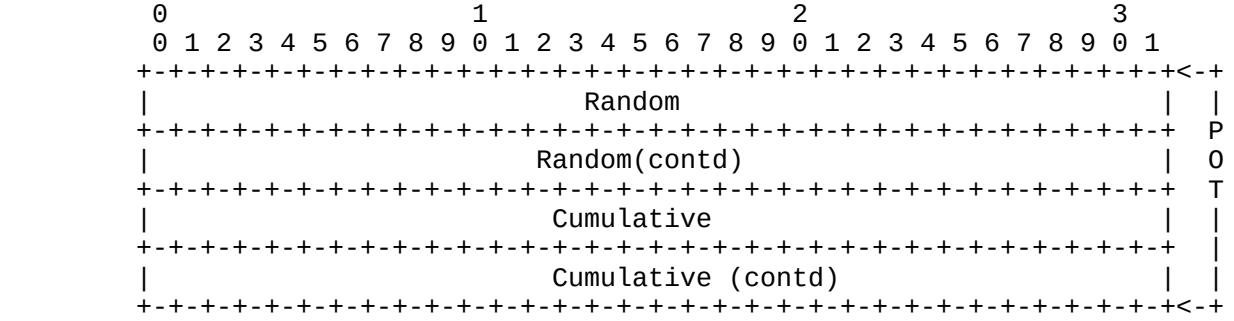
- Cumulative: Information which is handed from node to node and updated by every node according to a verification algorithm.

IOAM proof of transit option:

IOAM proof of transit option header:



IOAM proof of transit option data MUST be 4-octet aligned:



IOAM POT Type: 7-bit identifier of a particular POT variant that dictates the POT data that is included. This document defines POT Type 0:

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

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

Random: 64-bit Per packet Random number.

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

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

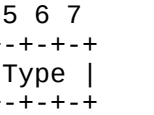
4.3. IOAM Edge-to-Edge Option

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

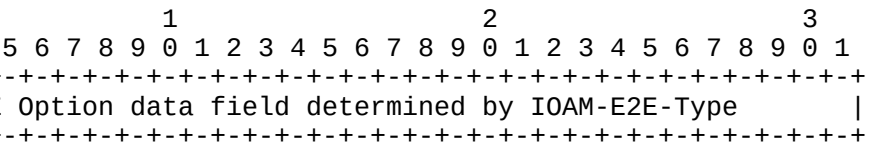
Currently only sequence numbers use the IOAM edge-to-edge option. In order to detect packet loss, packet reordering, or packet duplication in an in-situ OAM-domain, sequence numbers can be added to packets of a particular tube (see [I-D.hildebrand-spud-prototype]). Each tube leverages a dedicated namespace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

IOAM edge-to-edge option:

IOAM edge-to-edge option header:



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



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

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

5. IOAM Data Export

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

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

6. IANA Considerations

This document requests the following IANA Actions.

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

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

IOAM Trace Type

IOAM Trace flags

IOAM POT Type

IOAM E2E Type

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

The subsequent sub-sections detail the registries herein contained.

6.2. IOAM Trace Type Registry

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

6.3. IOAM Trace Flags Registry

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

6.4. IOAM POT Type Registry

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

6.5. IOAM E2E Type Registry

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

7. Manageability Considerations

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

8. Security Considerations

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

9. Acknowledgements

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

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

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

10. References

10.1. Normative References

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

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

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <http://www.rfc-editor.org/info/rfc8126>.

10.2. Informative References

[I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., <>, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.brockners-inband-oam-transport]
Brockners, F., Bhandari, S., Govindan, V., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., and R. <>, "Encapsulations for In-situ OAM Data", draft-brockners-inband-oam-transport-03 (work in progress), March 2017.

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

[I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-04 (work in progress), March 2017.

[I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work in progress), April 2017.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-13 (work in progress), June 2017.

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

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

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

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <http://www.rfc-editor.org/info/rfc7799>.

[RFC7820] Mizrahi, T., "UDP Checksum Complement in the One-Way Active Measurement Protocol (OWAMP) and Two-Way Active Measurement Protocol (TWAMP)", RFC 7820, DOI 10.17487/RFC7820, March 2016, <http://www.rfc-editor.org/info/rfc7820>.

[RFC7821] Mizrahi, T., "UDP Checksum Complement in the Network Time Protocol (NTP)", RFC 7821, DOI 10.17487/RFC7821, March 2016, <http://www.rfc-editor.org/info/rfc7821>.

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 2066721
Israel

Email: talmi@marvell.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

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

Email: petr@fb.com

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

Daniel
Bell Canada

Email: daniel.bernier@bell.ca

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

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

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

Abstract

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

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

This document discusses requirements for "in-situ" Operations, Administration, and Maintenance (OAM) mechanisms. In this context, "in-situ OAM" refers to the concept of directly encoding telemetry information within the data packet as it traverses the network or telemetry domain. Mechanisms which add tracing or other types of telemetry information to the regular data traffic, sometimes also referred to as "in-band" OAM can complement active, probe-based mechanisms such as ping or traceroute, which are sometimes considered as "out-of-band", because the messages are transported independently from regular data traffic. In terms of "active" or "passive" OAM, "in-situ" OAM can be considered a hybrid OAM type. While no extra packets are sent, in-situ OAM adds information to the packets therefore cannot be considered passive. In terms of the classification given in [RFC7799] in-situ OAM could be portrayed as "hybrid OAM, type 1". "In-situ" mechanisms do not require extra packets to be sent and hence don't change the packet traffic mix within the network. Traceroute and ping for example use ICMP messages: New packets are injected to get tracing information. Those add to the number of messages in a network, which already might be highly loaded or suffering performance issues for a particular path or traffic type.

A number of in-situ as well as in-band OAM mechanisms have been discussed, such as the INT spec for the P4 programming language [P4] or the SPUD prototype [I-D.hildebrand-spud-prototype]. The SPUD prototype uses a similar logic that allows network devices on the path between endpoints to participate explicitly in the tube outside the end-to-end context. Even the IPv4 route-record option defined in [RFC0791] can be considered an in-situ OAM mechanism. Per what was already stated, in-situ OAM complements "out-of-band" mechanisms such as ping or traceroute, or more recent active probing mechanisms, as described in [I-D.lapukhov-dataplane-probe]. In-situ OAM mechanisms can be leveraged where current out-of-band mechanisms do not apply or do not offer the desired characteristics or requirements, such as

proving that a certain set of traffic takes a pre-defined path, strict congruency between overlay and underlay transports is in place, checking service level agreements for the live data traffic, detailed statistics or verification of path selections within a domain, or scenarios where probe traffic is potentially handled differently from regular data traffic by the network devices. [RFC7276] presents an overview of OAM tools.

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

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

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

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

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

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

f. The ability to carry in-situ OAM data in various different transport protocols.

2. Conventions

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

Abbreviations used in this document:

ECMP: Equal Cost Multi-Path

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

MTU: Maximum Transmit Unit

NSH: Network Service Header

NFV: Network Function Virtualization

OAM: Operations, Administration, and Maintenance

PMTU: Path MTU

SFC: Service Function Chain

SLA: Service Level Agreement

SR: Segment Routing

SID: Segment Identifier

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

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

3. Motivation for in-situ OAM

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

3.1. Path Congruency Issues with Dedicated OAM Packets

Packet scheduling algorithms, especially for balancing traffic across equal cost paths or links, often leverage information contained within the packet, such as protocol number, IP-address or MAC-address. Probe packets would thus either need to be sent from the exact same endpoints with the exact same parameters, or probe packets would need to be artificially constructed as "fake" packets and

inserted along the path. Both approaches are often not feasible from an operational perspective, be it that access to the end-system is not feasible, or that the diversity of parameters and associated probe packets to be created is simply too large. An in-situ mechanism is an alternative in those cases.

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

3.2. Results Sent to a System Other Than the Sender

Traditional ping and traceroute tools return the OAM results to the sender of the probe. Even when the ICMP messages that are used with these tools are enhanced, and additional telemetry is collected (e.g., ICMP Multi-Part [RFC4884] supporting MPLS information [RFC4950], Interface and Next-Hop Identification [RFC5837], etc.), it would be advantageous to separate the sending of an OAM probe from the receiving of the telemetry data. In this context, it is helpful to eliminate the requirement that there be a working bidirectional path.

3.3. Overlay and Underlay Correlation

Several network deployments leverage tunneling mechanisms to create overlay or service-layer networks. Examples include VXLAN-GPE, GRE, or LISP. One often observed attribute of overlay networks is that they do not offer the user of the overlay any insight into the underlay network. This means that the path that a particular tunneled packet takes, nor other operational details such as the per-hop delay/jitter in the underlay are visible to the user of the overlay network, giving rise to diagnosis and debugging challenges in case of connectivity or performance issues. The scope of OAM tools like ping or traceroute is limited to either the overlay or the underlay which means that the user of the overlay has typically no access to OAM in the underlay, unless specific operational procedures are put in place. With in-situ OAM the operator of the underlay can offer details of the connectivity in the underlay to the user of the overlay. This could include the ability to find out which underlay elements are shared by overlays and ability to know which overlays are mapped to the same underlay elements. Deployment dependent

underlay transit nodes can be configured to update OAM information in the overlay transport encapsulation. The operator of the egress tunnel router could choose to share the recorded information about the path with the user of the overlay.

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

3.4. SLA Verification

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

3.5. Analytics and Diagnostics

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

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

system, without a need to correlate/aggregate statistics from every router in the system. Transit devices are off-loaded from the task of gathering packet statistics.

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

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

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

3.7. Proof of Transit

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

is delivered to the ingress interface of a service appliance and received back from an egress interface of a service appliance. This "wiring" is verified and trusted. The evolution to Network Function Virtualization (NFV) and modern service chaining concepts (using technologies such as Locator/ID Separation Protocol (LISP), Network Service Header (NSH), Segment Routing (SR), etc.) blurs the line between the different trust domains, because the hand-off-points are no longer clearly defined physical interfaces, but are virtual interfaces. Because of that very reason, networks operators require that different trust layers not to be mixed in the same device. For an NFV scenario a different proof is required. Offering a proof that a packet traversed a specific set of service functions would allow network operators to move away from the above described indirect methods of proving that a service chain is in place for a particular application.

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

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

3.8. Use Cases

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

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

- Flow Debugging: Discover which path(s) a particular set of traffic (identified by an n-tuple) takes in the network. Such a procedure

is particularly useful in case traffic is balanced across multiple paths, like with link aggregation (LACP) or equal cost multi-pathing (ECMP).

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

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

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

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

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

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

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

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

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

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

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

4. Considerations for In-situ OAM

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

4.1. Type of Information to be Recorded

The information gathered for in-situ OAM can be categorized into three main categories: Information with a per-hop scope, such as path tracing; information which applies to a specific set of hops, such as path or service chain verification; information which only applies to the edges of a domain, such as sequence numbers. Note that a single network device could comprise several in-situ OAM hops, for example in case one wants to trace the path of a packet through that device.

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

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

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

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

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

4.2. MTU and Packet Size

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

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

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

- Pre-allocated: In this case, the encapsulating node inserts empty data fields into the packet to cover the entire domain. The data fields will be incrementally updated/filled as the packet progresses through the network. With pre-allocation the packet size is only changed at the encapsulating node and is kept constant throughout the domain. The pre-allocated approach is beneficial for software data-plane implementations where allocating the required space only once and index into the array to populate the data during transit avoids copy operations at every hop.

- Incremental: Every node that desires to include in-situ OAM information extends the packet as needed. The incremental approach is beneficial for hardware data-plane implementations as it eliminates the need for the transit nodes to read the full array and lookup the pointer in the option prior to updating the data fields contents.

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

4.3. Administrative Boundaries

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

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

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

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

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

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

4.3.1. Layered In-Situ OAM Domains

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

Consider the following examples:

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

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

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

4.4. Selective Enablement

The ability to selectively enable in-situ OAM is valuable. While it may be desirable to enable data collection on all traffic or devices, this may not always be feasible. In-situ OAM collection may also come with a performance impact to forwarding rates or feature capabilities, which may be acceptable in only some locations. For example, the SPUD prototype uses the notion of "pipes" to describe the portion of the traffic that could be subject to in-path inspection. Mechanisms to decide which traffic would be subject to in-situ OAM are outside the scope of this document.

4.5. Forwarding Behavior

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

4.6. Optimization of Node and Interface Identifiers

Since packets have a finite maximum size, the data recording or carrying capacity of one packet in which the in-situ OAM metadata is present is limited. In-situ OAM should use its own dedicated

namespace (confined to the domain in-situ OAM operates in) to represent node and interface IDs to save space in the header. Generic representations of node and interface identifiers which are globally unique (such as a UUID) would consume significantly more bits of in-situ OAM data.

4.7. Loop Communication Path (IPv6-specifics)

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

One way to carry the in-situ OAM data back to the source is to utilize an ICMP Echo Request/Reply (ping) or ICMPv6 Echo Request/ Reply (ping6) mechanism. In order to run the in-situ OAM mechanism appropriately on the ping/ping6 mechanism, the following two operations should be implemented by the ping/ping6 target node:

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

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

5. Requirements for In-situ OAM Data Types

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

5.1. Generic Requirements

REQ-G1: Classification: It should be possible to enable in-situ OAM on a selected set of traffic (e.g., per interface, based on an access control list specifying a specific set of traffic, etc.) The selected set of traffic can also be all traffic.

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

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

REQ-G4: Layering: It should be possible to have in-situ OAM information for different transport protocol layers be present in several fields within a single packet. This could for example be the case when tunnels are employed and in-situ OAM information is to be gathered for both the underlay as well as the overlay network. Layering support should not be limited to just underlay and overlay, but include more than two layers.

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

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

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

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

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

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

REQ-G8: Combination with active OAM mechanisms: In-situ OAM should be usable for active network probing, like for example a customized version of traceroute. Decapsulating in-situ OAM nodes may have an ability to send the in-situ OAM

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

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

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

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

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

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

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

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

REQ-H5: Time-related requirements

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

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

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

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

REQ-H5.4: Time precision: Time with different precision should be supported. Use-case dependent, the required precision could e.g., be nanoseconds, microseconds, milliseconds, or seconds.

REQ-H6: Generic data fields (like e.g., GPS/Geo-location information): It should be possible to add user-defined OAM data at select hops to the packet. The semantics of the data are defined by the user.

5.3. In-situ OAM with Selected Hop Scope

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

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

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

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

REQ-E1: Sequence numbering:

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

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

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

6. Security Considerations and Requirements

6.1. General considerations

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

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

6.2. Proof of Transit

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

REQ-SEC1: Sound Proof of Transit: A valid and verifiable proof that the packet definitively traversed through all the nodes as expected. Probabilistic methods to achieve this should be avoided, as the same could be exploited by an attacker.

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

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

REQ-SEC4: Pre-play Attacks: A active attacker should not be able to generate or reuse valid POT bits from legitimate packets, in order to prove to the verifier as valid packets. This

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

REQ-SEC5: Recycle Secrets: Any configuration of the secrets (like cryptographic keys, initialization vectors etc.) either in the controller or service functions should be re-configurable. Solution approach should enable controls, API calls etc. needed in order to perform such recycling. It is desirable to provide recommendations on the duration of rotation cycles needed for the secure functioning of the overall system.

REQ-SEC6: Secret storage and distribution: Secrets should be shared with the devices over secure channels. Methods should be put in place so that secrets cannot be retrieved by non-authorized personnel from the devices.

7. IANA Considerations

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

This document has no IANA actions.

8. Acknowledgements

The authors would like to thank Jen Linkova, LJ Wobker, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Ignas Bagdonas, LJ Wobker, Erik Nordmark, Vengada Prasad Govindan, and Andrew Yourtchenko for the comments and advice. This document leverages and builds on top of several concepts described in [I-D.kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

9. References

9.1. Normative References

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

9.2. Informative References

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

[I-D.brockners-proof-of-transit]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Leddy, J., Youell, S., Mozes, D., and T. Mizrahi, "Proof of Transit", draft-brockners-proof-of-transit-02 (work in progress), October 2016.

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

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-10 (work in progress), November 2016.

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

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

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

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

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

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

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

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

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

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

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <http://www.rfc-editor.org/info/rfc7799>.

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: sadara@cisco.com

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

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

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

URI: petr@fb.com

Remy Chang
Barefoot Networks

Email: remy@barefootnetworks.com

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

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

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

Abstract

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

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

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

deployments and packet transport formats. For details, see the open source code in the FD.io [FD.io].

2. Conventions

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

Abbreviations used in this document:

IOAM: In-situ Operations, Administration, and Maintenance

MTU: Maximum Transmit Unit

NSH: Network Service Header

OAM: Operations, Administration, and Maintenance

POT: Proof of Transit

SFC: Service Function Chain

SID: Segment Identifier

SR: Segment Routing

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

3. In-Situ OAM Metadata Transport in IPv6

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

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

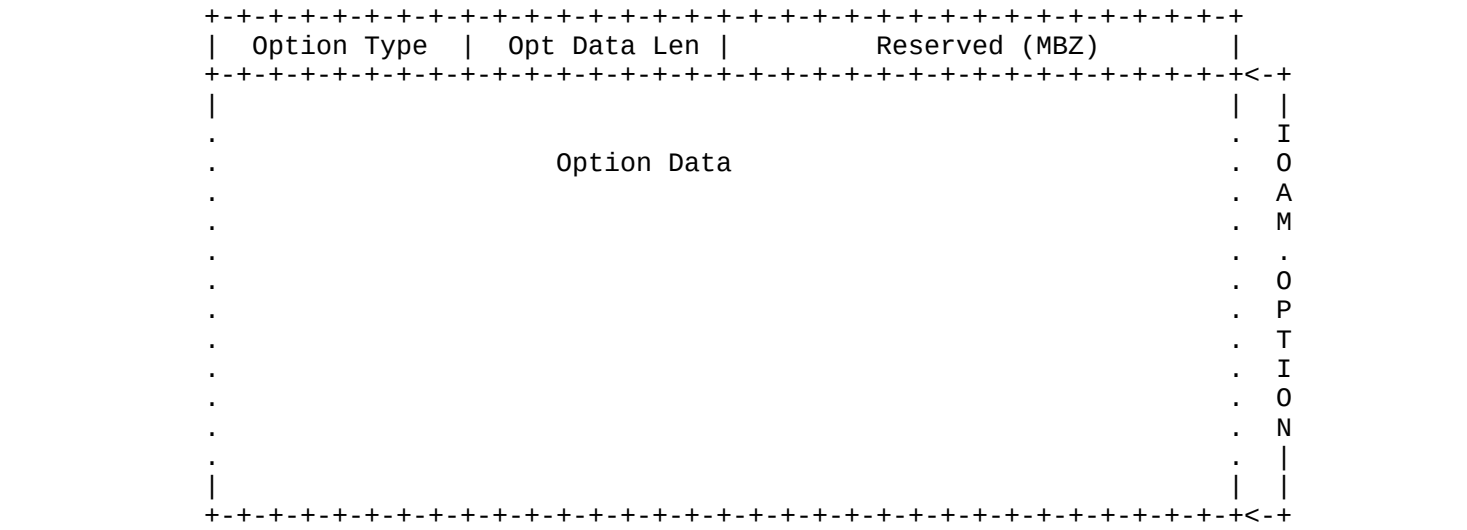
The in-situ OAM options translate into options for an IPv6 hop by hop extension header. The extension header would be inserted by either a host source of the packet, or by a transit/domain-edge node. If the addition of the in-situ OAM Hop-by-Hop Option header would lead to the packet exceeding the MTU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

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

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

3.1.1. In-situ OAM Hop by Hop Options

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



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

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

Reserved (MBZ) 16-bit field MUST be filled with zeroes.

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

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

1. Pre-allocated Tracing Option: The in-situ OAM Preallocated Tracing option defined in [I-D.brockners-inband-oam-data] is represented as a IPv6 option in hop by hop extension header by allocating following type:

   Option Type: 001xxxxxx 8-bit identifier of the type of option.
   xxxxxx=TBD_IANA_PRE_TRACE_OPTION_IPV6.

2. Incremental Tracing Option: The in-situ OAM Incremental Tracing option defined in [I-D.brockners-inband-oam-data] is represented as a IPv6 option in hop by hop extension header by allocating following type:

   Option Type: 001xxxxxx 8-bit identifier of the type of option.
   xxxxxx=TBD_IANA_INCR_TRACE_OPTION_IPV6.

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

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

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

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

4. In-situ OAM Metadata Transport in IPv4

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

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

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

For IOAM the following new GRE protocol types are requested:

1. IOAM_Trace_Preallocated: TBD_IANA_ETHERNET_NUMBER_IOAM_TRACE_PREALLOCATED

2. IOAM_Trace_Incremental: TBD_IANA_ETHERNET_NUMBER_IOAM_TRACE_INCREMENTAL

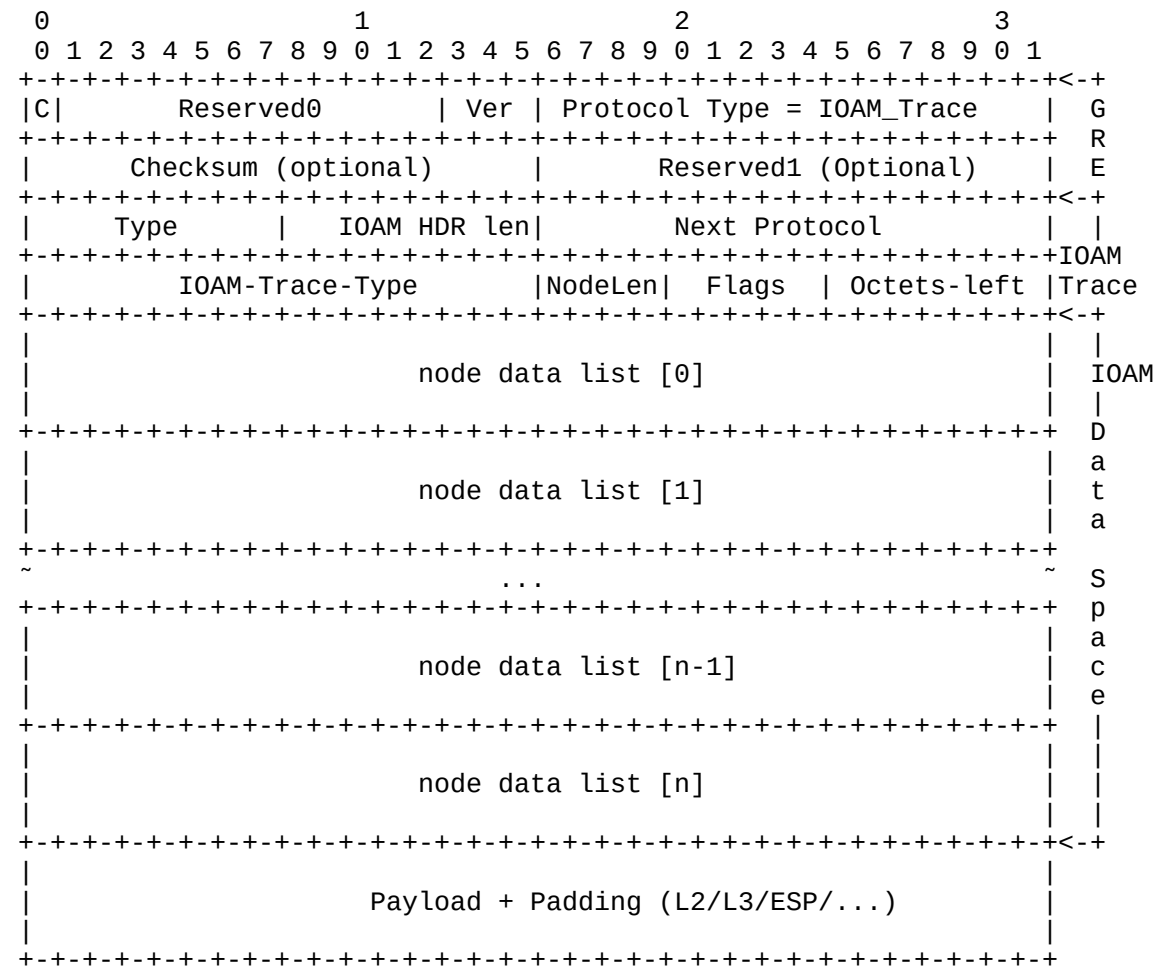
3. IOAM_POT: TBD_IANA_ETHERNET_NUMBER_IOAM_POT

4. IOAM_End-to_End: TBD_IANA_ETHERNET_NUMBER_IOAM_E2E

4.1. In-situ OAM Tracing in GRE

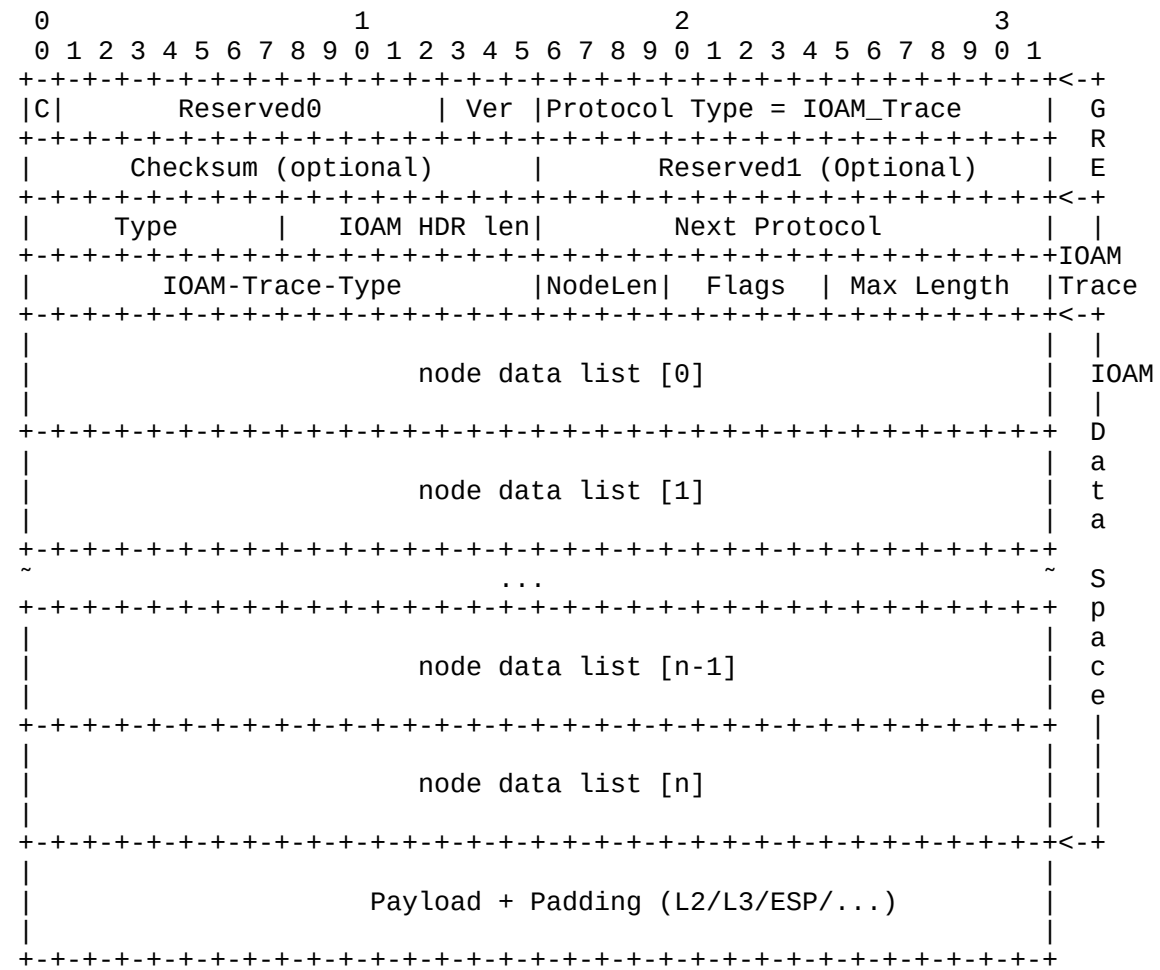
The packet formats of the pre-allocated IOAM trace and incremental IOAM trace when transported using GRE are defined as below. See [I-D.brockners-inband-oam-data] for details about pre-allocated and incremental IOAM trace options.

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



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

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



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

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

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

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

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

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

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

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

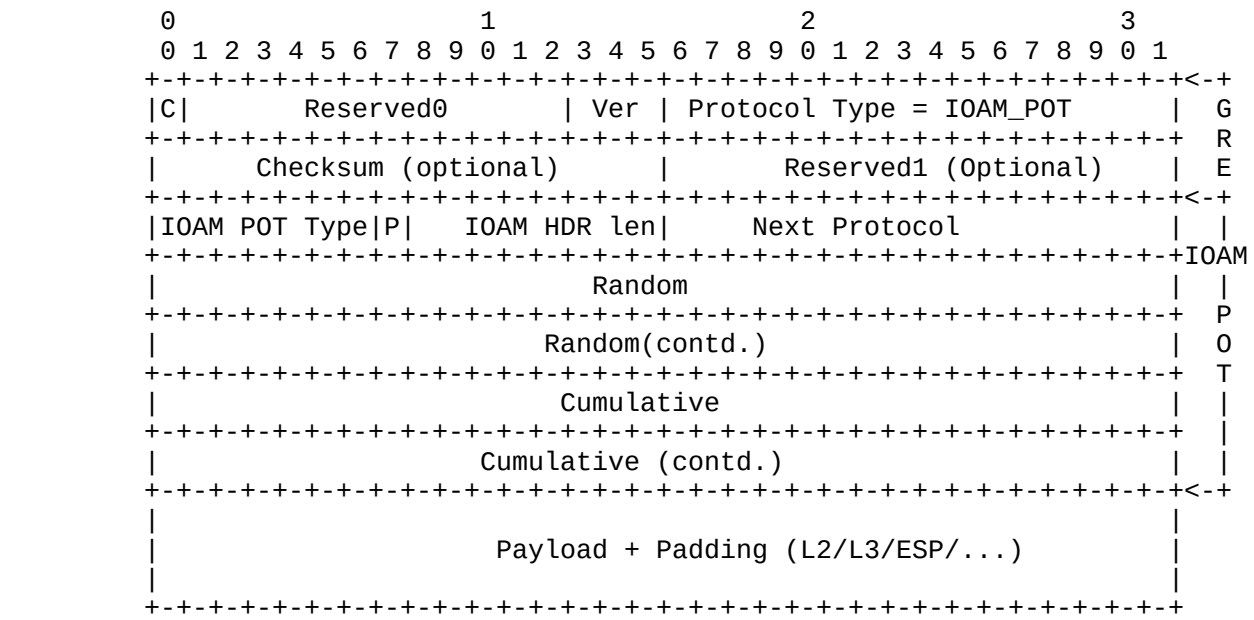
Octets-left: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

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

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

4.2. In-situ OAM POT in GRE

In-situ OAM POT header following GRE header:



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

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

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

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

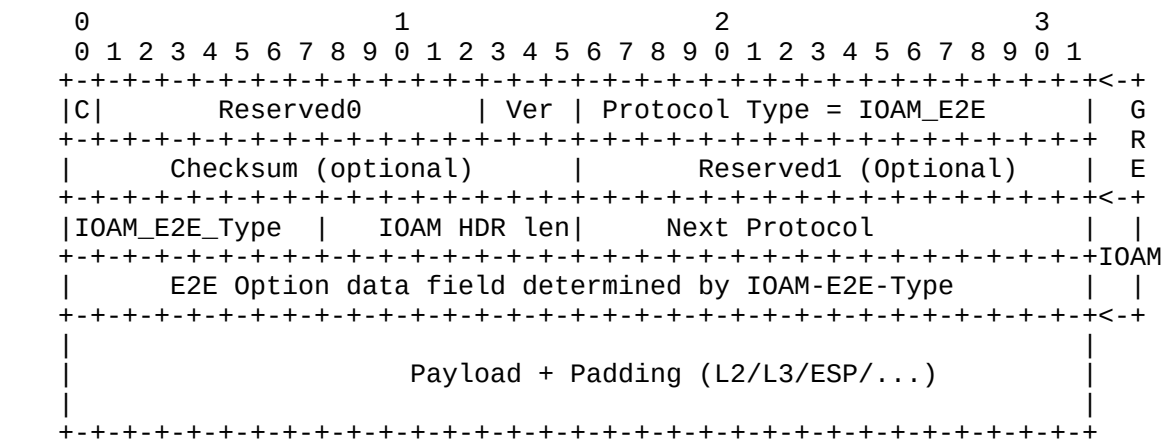
Next Protocol: 16 bits Next Protocol Type field contains the protocol type of the packet following IOAM protocol header. These Protocol Types are defined in [RFC3232] as "ETHER TYPES" and in [ETYPES]. An implementation receiving a packet containing a Protocol Type which is not listed in [RFC3232] or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

Random: 64-bit Per-packet random number.

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

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

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



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

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

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

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

5. In-situ OAM Metadata Transport in VXLAN-GPE

VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe] encapsulation is somewhat similar to IPv6 extension headers in that a series of headers can be contained in the header as a linked list. The different iIOAM types are added as options within a new IOAM protocol header in VXLAN GPE. In an administrative domain where IOAM is used, insertion of the IOAM

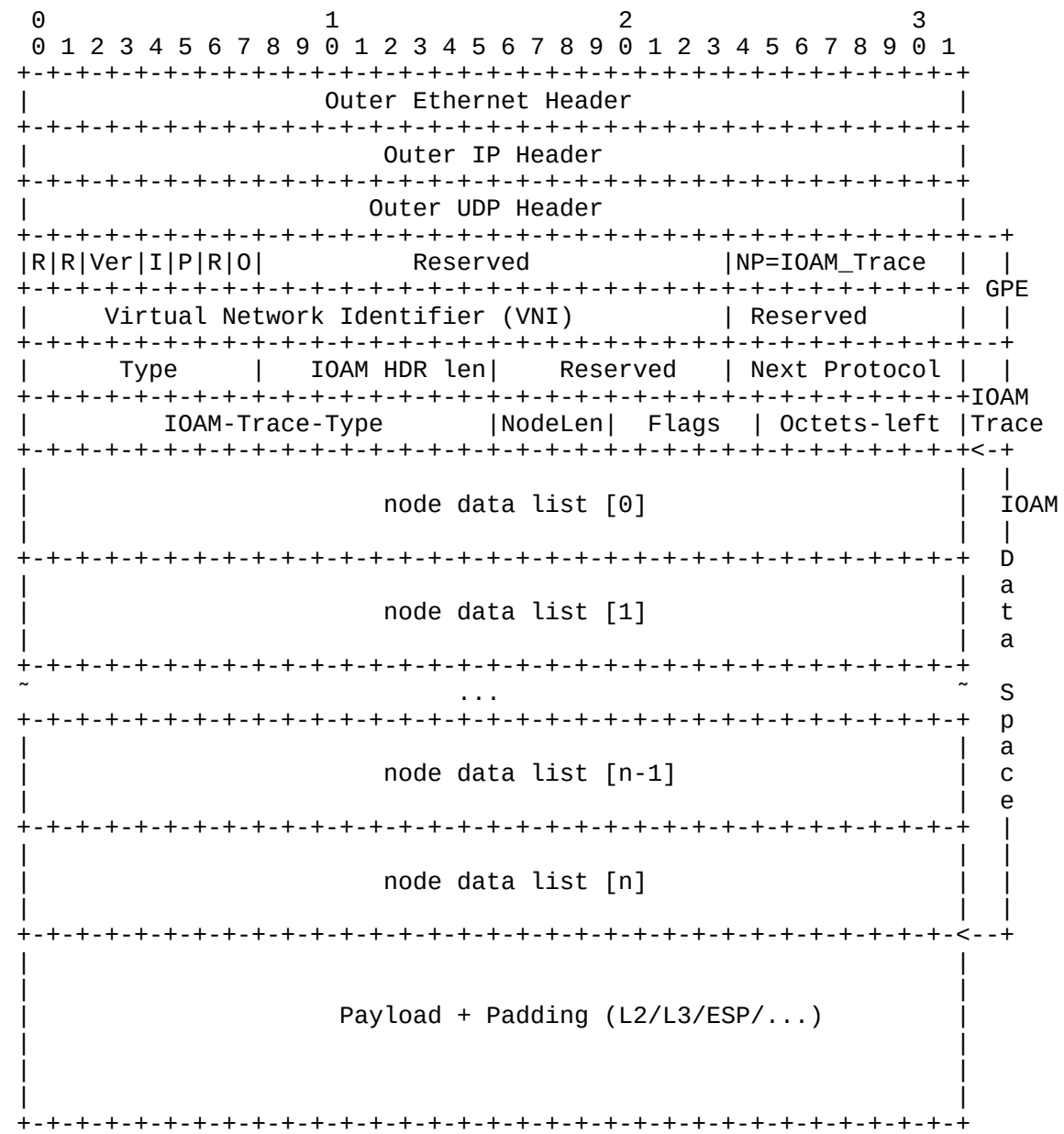
protocol header in VXLAN GPE is enabled at the VXLAN GPE tunnel endpoint which also serve as IOAM encapsulating/decapsulating nodes by means of configuration.

5.1. In-situ OAM Tracing in VXLAN-GPE

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

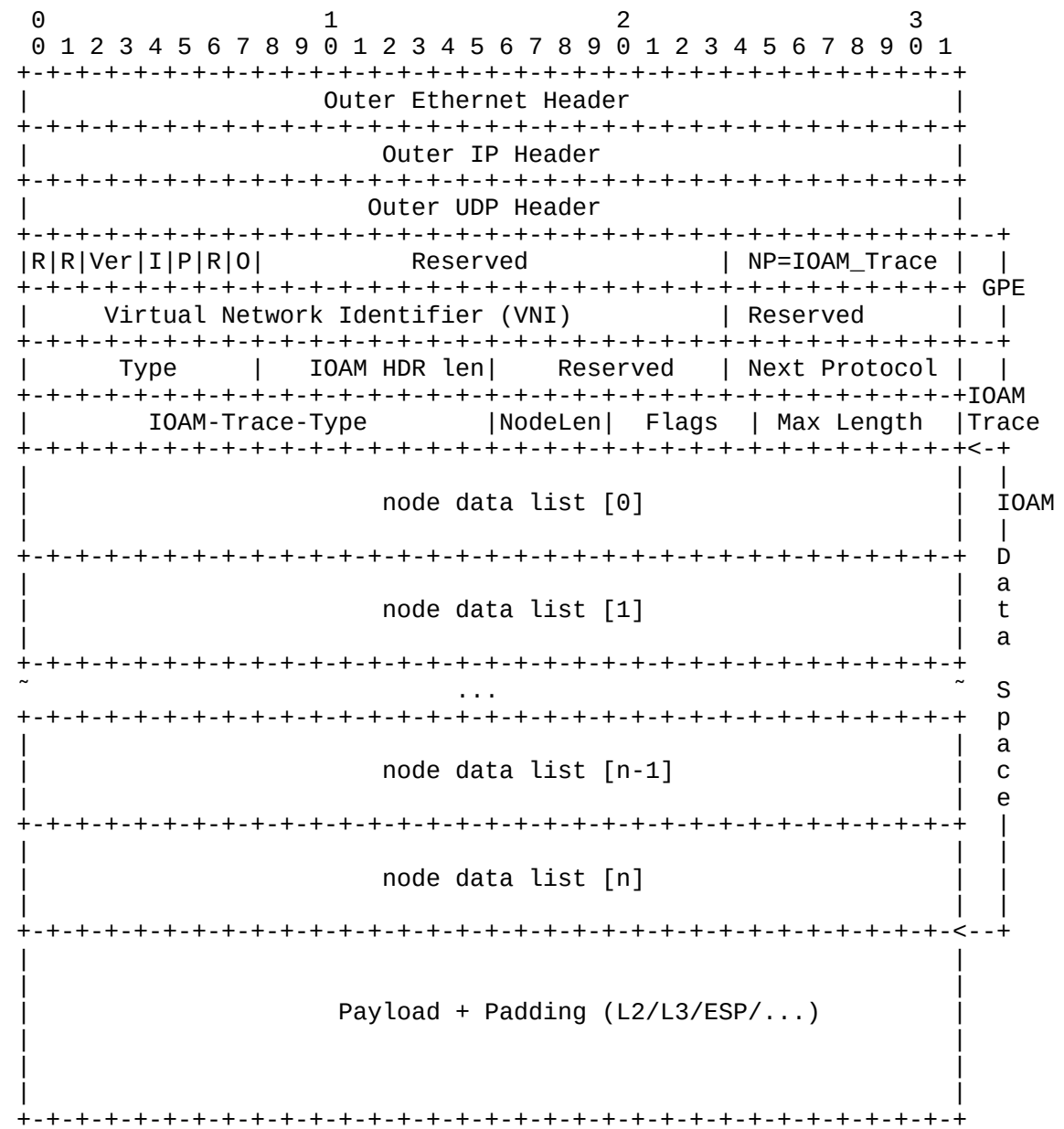
The VXLAN-GPE header and fields are defined in [I-D.ietf-nvo3-vxlan-gpe]. IOAM specific fields and header are defined here:

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



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

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



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

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

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

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

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

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

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

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

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

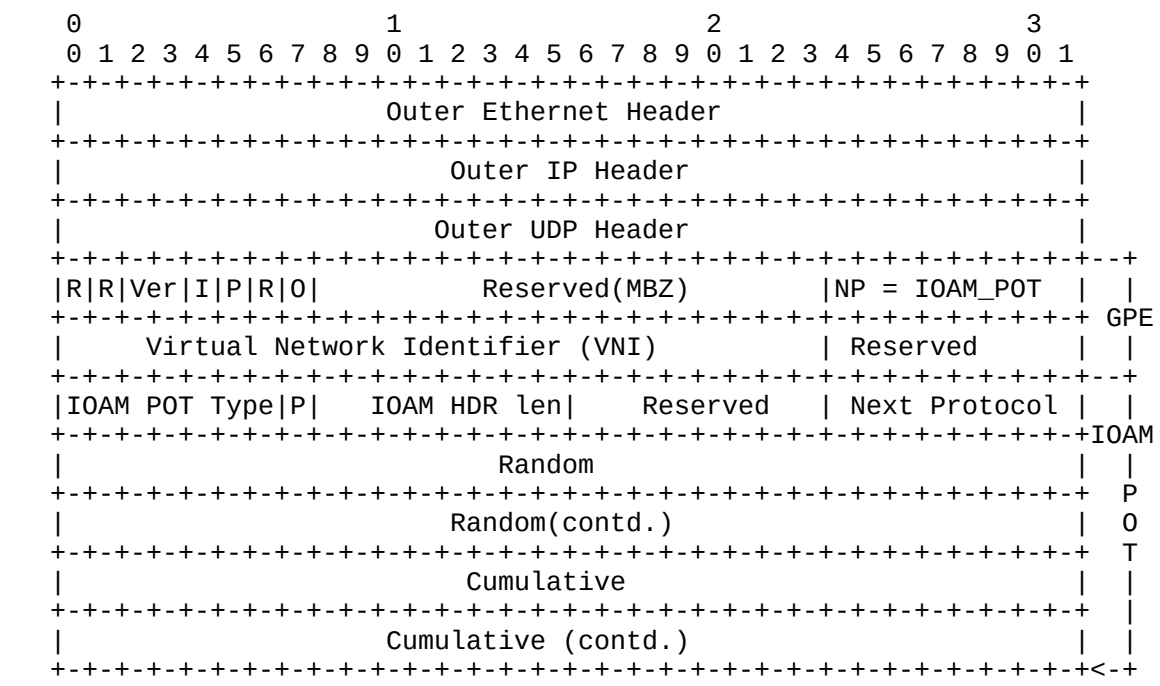
Maximum-length: 7-bit unsigned integer as defined in [I-D.brockners-inband-oam-data].

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

5.2. In-situ OAM POT in VXLAN-GPE

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

In-situ OAM POT header following VXLAN GPE header:



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

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

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

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

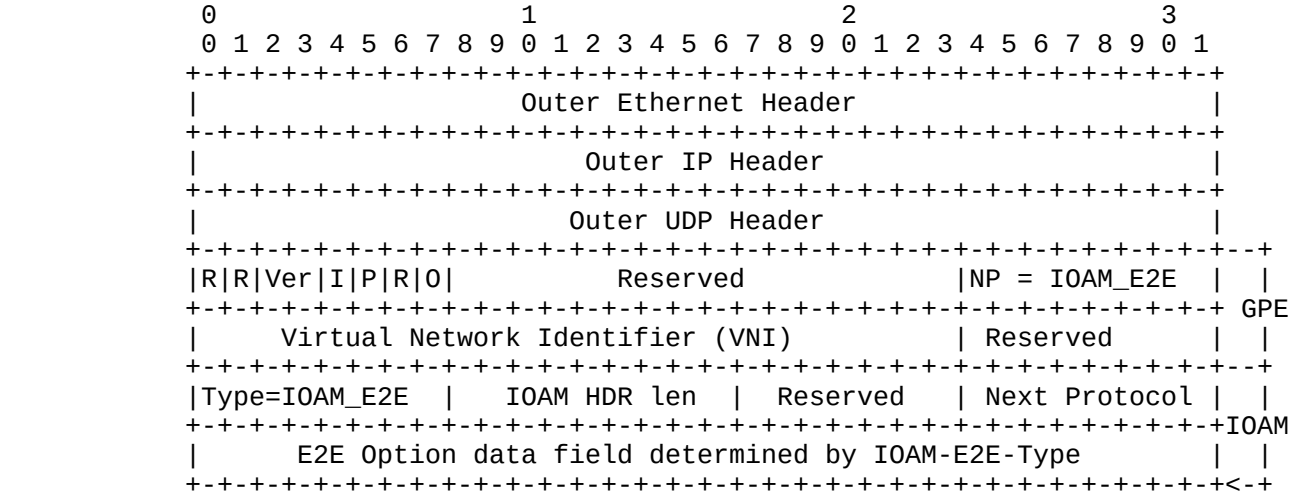
Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The value is from the IANA registry setup for VXLAN GPE Next Protocol defined in [I-D.ietf-nvo3-vxlan-gpe].

Random: 64-bit Per-packet random number.

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

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

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



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

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

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

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

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

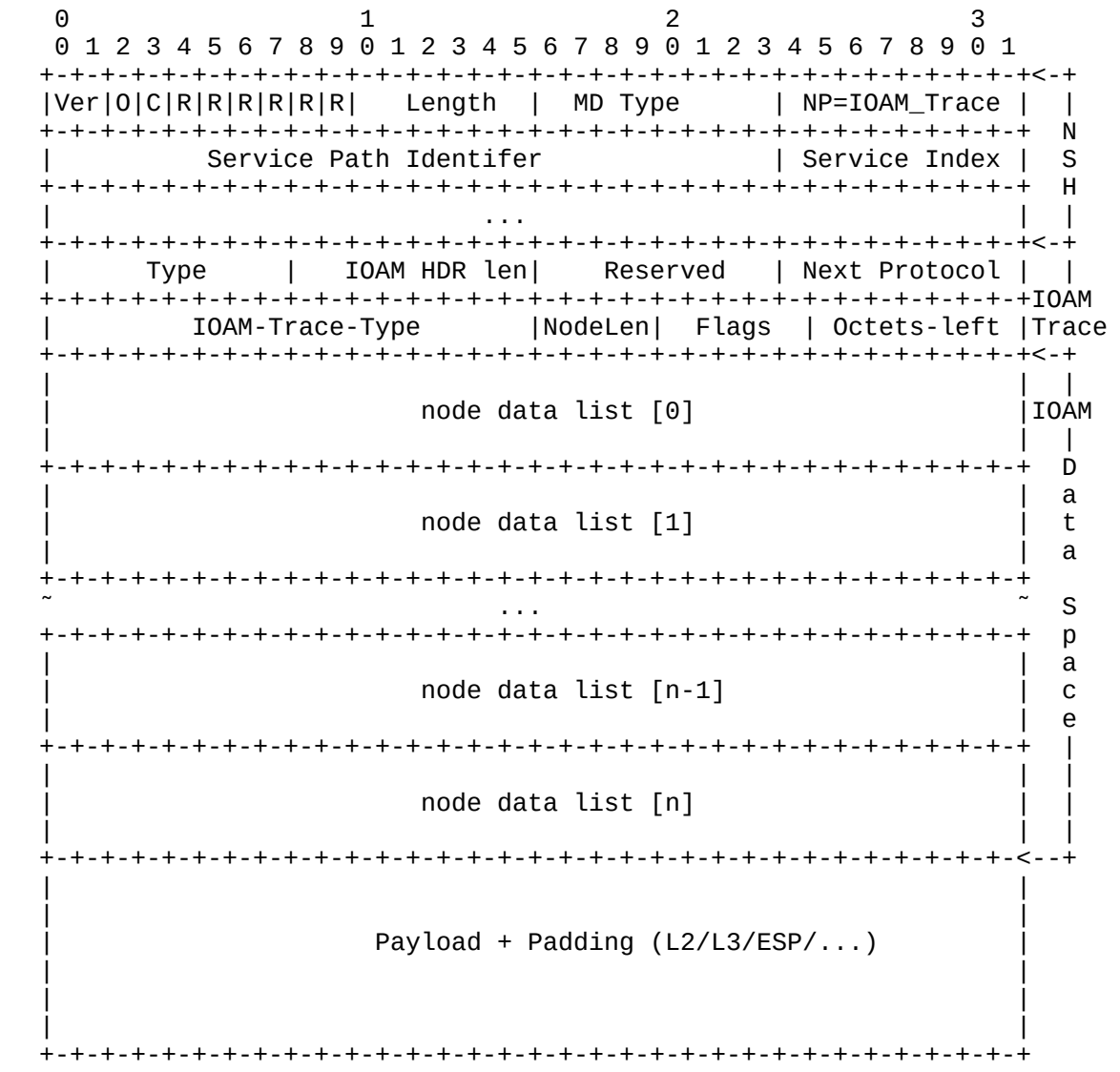
6. In-situ OAM Metadata Transport in NSH

6.1. In-situ OAM Tracing in NSH

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

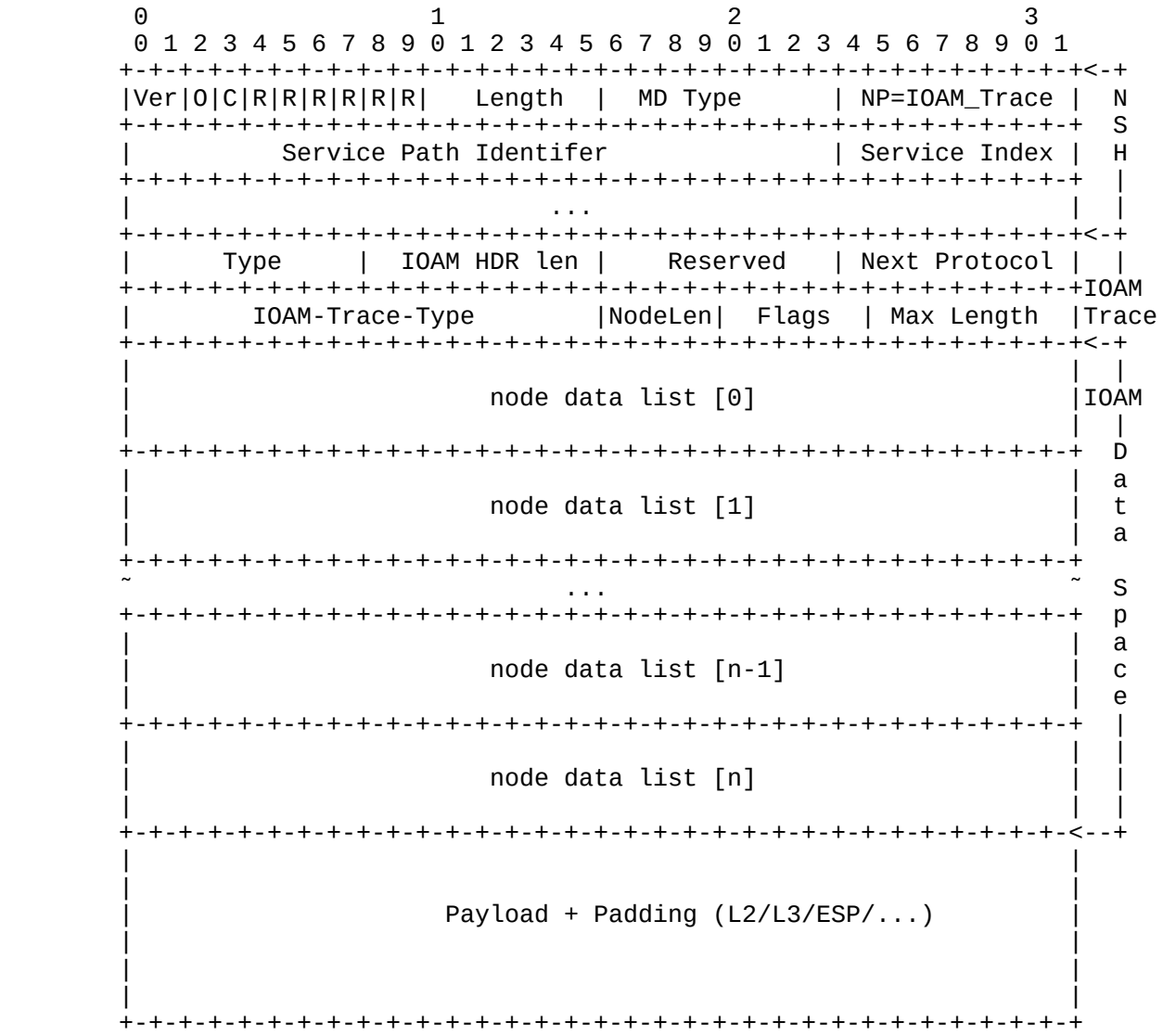
In Service Function Chaining (SFC) [RFC7665], the Network Service Header (NSH) [I-D.ietf-sfc-nsh] already includes path tracing capabilities [I-D.penno-sfc-trace]. Tracing information can be carried in-situ as IOAM data fields following NSH MDx metadata TLVs.

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



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

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



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

Next Protocol of NSH: TBD value for IOAM_Trace.

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

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

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

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

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

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

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

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

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

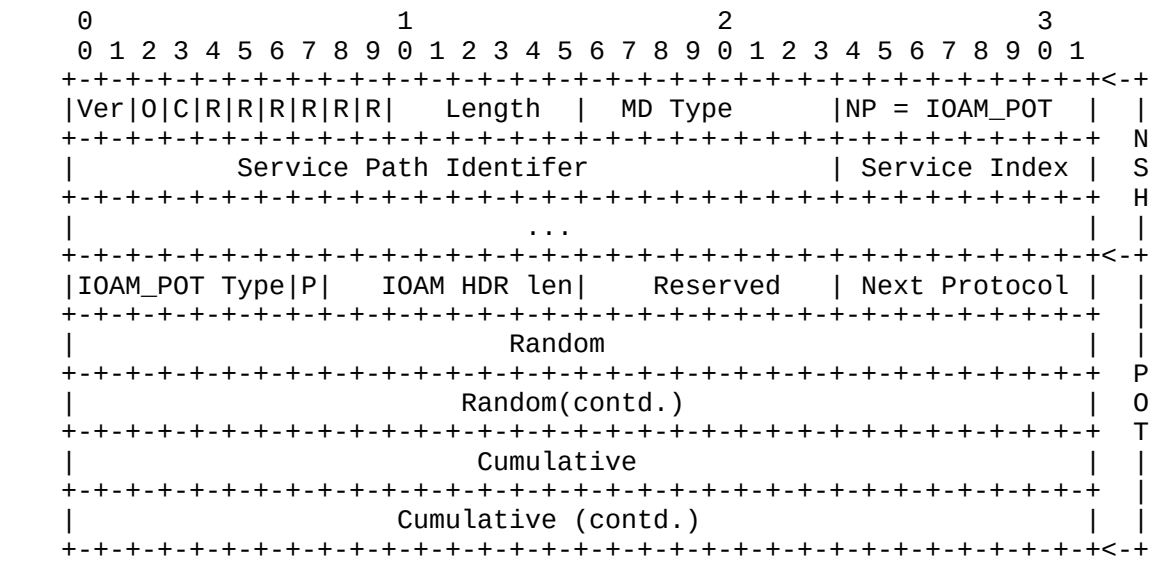
Node data List [n]: Variable-length field as defined in [I-D.brockners-inband-oam-data].

6.2. In-situ OAM POT in NSH

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

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

In-situ OAM POT header following NSH MDx header:



Next Protocol of NSH: TBD value for IOAM_POT.

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

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

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

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

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

Random: 64-bit Per-packet random number.

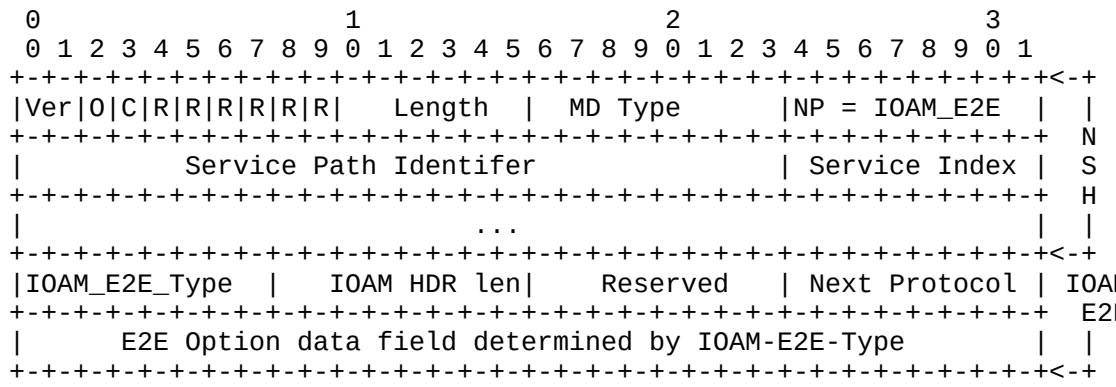
Cumulative: 64-bit Cumulative value that is updated by the Service Functions.

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

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

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

In-situ OAM E2E header following NSH MDx header:



Next Protocol of NSH: TBD value for IOAM_E2E.

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

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

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

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

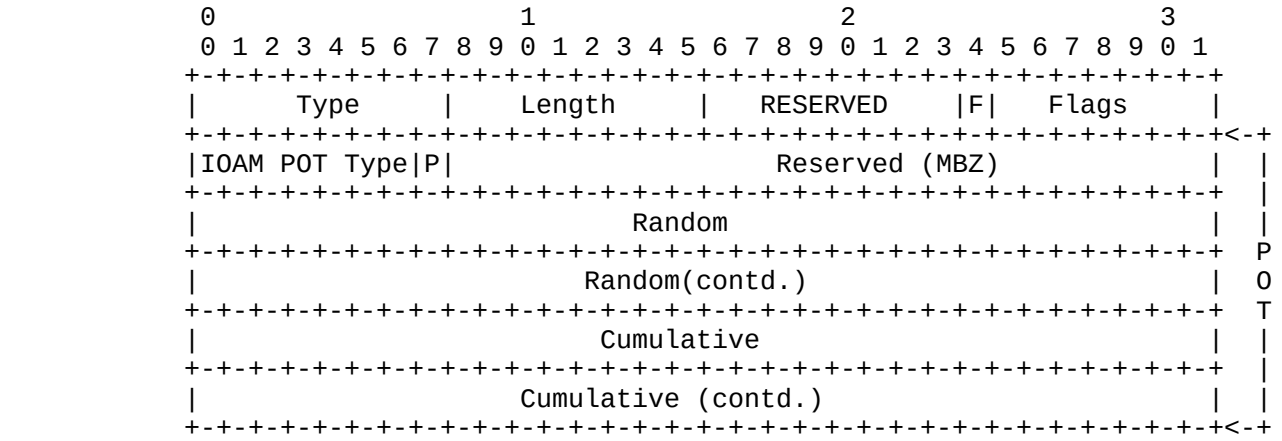
E2E Option data field: Variable length field as defined in [I-D.brockners-inband-oam-data] IOAM E2E Option.

7. In-situ OAM Metadata Transport in Segment Routing

7.1. In-situ OAM in SR with IPv6 Transport

Similar to NSH, a policy defined using Segment Routing for IPv6 can be verified using the in-situ OAM "Proof of Transit" approach. The Segment Routing Header (SRH) for IPv6 offers the ability to transport TLV structured data, similar to what NSH does (see [I-D.ietf-6man-segment-routing-header]). In an domain where in-situ OAM is used, insertion of the in-situ OAM data is enabled at the required edge nodes (i.e. at the in-situ OAM encapsulating/ decapsulating nodes) by means of configuration.

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



Type: To be assigned by IANA.

Length: 20.

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

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

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

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

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

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

Random: 64-bit per-packet random number.

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

7.2. In-situ OAM in SR with MPLS Transport

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

8. IANA Considerations

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

9. Manageability Considerations

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

10. Security Considerations

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

11. Acknowledgements

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

12. References

12.1. Normative References

[ETYPES] "IANA Ethernet Numbers", <https://www.iana.org/assignments/ethernet-numbers/ethernet-numbers.xhtml>.

[I-D.brockners-inband-oam-data] Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., <>, R., and d. daniel.bernier@bell.ca, "Data Fields for In-situ OAM", draft-brockners-inband-oam-data-05 (work in progress), May 2017.

[I-D.brockners-inband-oam-requirements] Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., <>, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.ietf-6man-segment-routing-header] Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-06 (work in progress), March 2017.

[I-D.ietf-nvo3-vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work in progress), April 2017.

[I-D.ietf-sfc-nsh] Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-13 (work in progress), June 2017.

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

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <http://www.rfc-editor.org/info/rfc2784>.

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

12.2. Informative References

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

[I-D.brockners-proof-of-transit] Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Leddy, J., Youell, S., Mozes, D., and T. Mizrahi, "Proof of Transit", draft-brockners-proof-of-transit-03 (work in progress), March 2017.

[I-D.ietf-ippm-6man-pdm-option] Elkins, N., Hamilton, R., and m. mackermann@bcbsm.com, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", draft-ietf-ippm-6man-pdm-option-13 (work in progress), June 2017.

[I-D.ietf-spring-segment-routing] Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-12 (work in progress), June 2017.

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

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

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

Vengada Prasad Govindan
Cisco Systems, Inc.

Email: venggovi@cisco.com

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

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

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

Email: petr@fb.com

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

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

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

T. Mizrahi
Marvell

May 7, 2018

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

Abstract

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

Status of This Memo

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

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

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

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

Table of Contents

1. Introduction

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

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

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

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

2. Conventions

Abbreviations used in this document:

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

IOAM: In-situ Operations, Administration, and Maintenance

LISP: Locator/ID Separation Protocol

LPC: Lagrange Polynomial Constants

MTU: Maximum Transmit Unit

NFV: Network Function Virtualization

NSH: Network Service Header

POT: Proof of Transit

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

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

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

SFC: Service Function Chain

SR: Segment Routing

3. Proof of Transit

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

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

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

3.1. Basic Idea

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

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

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

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

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

3.2. Solution Approach

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

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

3.2.1. Setup

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

3.2.2. In Transit

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

3.2.3. Verification

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

3.3. Illustrative Example

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

3.3.1. Basic Version

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

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

3.3.1.1. Secret Shares

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

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

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

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

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

3.3.1.2. Lagrange Polynomials

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

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

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

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

3.3.1.3. LPC Computation

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

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

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

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

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

3.3.1.4. Reconstruction

Reconstruction of the polynomial is well-defined as

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

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

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

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

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

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

3.3.1.5. Verification

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

3.3.2. Enhanced Version

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

3.3.2.1. Random Polynomial

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

3.3.2.2. Reconstruction

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

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

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

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

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

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

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

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

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

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

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

The verification using CML is discussed in next section.

3.3.2.3. Verification

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

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

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

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

3.3.3. Final Version

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

3.4. Operational Aspects

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

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

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

2. A pre-evaluated portion of the public polynomial for each of the nodes is calculated and added to the POT-profile. Without this all the coefficients of the public polynomial had to be added to the POT profile and each node had to evaluate them. As stated before, the public portion is only the constant coefficient RND value, the pre-evaluated portion for each node should be kept secret as well.

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

3.5. Alternative Approach

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

3.5.1. Basic Idea

1. The controller provisions all the nodes with their respective secret keys.

2. The controller provisions the verifier with all the secret keys of the nodes.

3. For each packet, the ingress node generates a random number RND and encrypts it with its secret key to generate CML value

4. Each subsequent node on the path encrypts CML with their respective secret key and passes it along

5. The verifier is also provisioned with the expected sequence of nodes in order to verify the order

6. The verifier receives the CML, RND values, re-encrypts the RND with keys in the same order as expected sequence to verify.

3.5.2. Pros

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

3.5.3. Cons

1. Standard AES encryption would need 128 bits of RND, CML. This results in a 256 bits of additional overhead is added per packet

2. In hardware platforms that do not support native encryption capabilities like (AES-NI). This approach would have considerable impact on the computational latency

4. Sizing the Data for Proof of Transit

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

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

2. CML: Cumulative

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

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

Table assumes 64 octet packets

Table 1: Proof of transit data sizing

5. Node Configuration

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

5.1. Procedure

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

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

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

5.2. YANG Model

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

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

```
module ietf-pot-profile {

  yang-version 1;

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

  prefix ietf-pot-profile;

  organization "IETF xxx Working Group";
```

```
     contact "";

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

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

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


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

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

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

         leaf secret-share {
```

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

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

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

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

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

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

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

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

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

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

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

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

6. IANA Considerations

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

7. Manageability Considerations

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

8. Security Considerations

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

8.1. Proof of Transit

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

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

- Precisely three values are kept secret by individual nodes. Share of SECRET (i.e. points on POLY-1), Share of POLY-2, LPC, P. Note that only constant coefficient, RND, of POLY-2 is public. x values and non-constant coefficient of POLY-2 are secret

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

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

8.2. Cryptanalysis

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

- Inter-Node: A passive attacker observing CML values across nodes (i.e., as the packets entering and leaving), cannot perform differential analysis to construct the points on POLY-1. This is because at each point there are four unknowns (i.e. Share(POLY-1), Share(Poly-2) LPC and prime number P) and three known values (i.e. RND, CML-before, CML-after).

- Inter-Packets: A passive attacker could observe CML values across packets (i.e., values of PKT-1 and subsequent PKT-2), in order to predict the secrets. Differential analysis across packets could be mitigated using a good PRNG for generating RND. Note that if constant coefficient is a sequence number than CML values become quite predictable and the scheme would be broken.

8.3. Anti-Replay

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

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

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

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

8.4. Anti-Preplay

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

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

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

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

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

8.5. Anti-Tampering

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

8.6. Recycling

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

8.7. Redundant Nodes and Failover

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

8.8. Controller Operation

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

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

Controller would only be used for the initial configuration of the POT-profiles.

8.9. Verification Scope

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

8.9.1. Node Ordering

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

8.9.2. Stealth Nodes

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

9. Acknowledgements

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

10. References

10.1. Normative References

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

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

10.2. Informative References

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

Authors' Addresses

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

Email: fbrockne@cisco.com

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

Email: shwethab@cisco.com

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

Email: sadara@cisco.com

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

Email: cpignata@cisco.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

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

Email: stephen.youell@jpmorgan.com

David Mozes

Email: mosesster@gmail.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 3, 2018

S. Bryant
Huawei Technologies
A. Atlas
C. Bowers
Juniper Networks

August 02, 2017

Routing Timer Parameter Synchronization
draft-bryant-rtgwg-param-sync-03

Abstract

This document describes a mechanism for a link state routing protocol to coordinate the value of a routing timer parameter amongst routers in the flooding domain. The document also defines the solution to one specific case: the agreement of a common convergence timer value for use by routers in network convergence.

Status of This Memo

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

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

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

This Internet-Draft will expire on February 3, 2018.

Copyright Notice

Table of Contents

1. Introduction

There exist use cases where it desirable for a network to use a common value for a routing timer parameter across all nodes. In the past, these types of use case have been addressed by setting the parameter to a constant value in the protocol definition itself, or by requiring that the same value of the parameter be configured at every node.

Setting the routing timer parameter to a constant value in the protocol definition makes it difficult to change the parameter, since a change would require formal modification to the protocol. In practice, such a change is impractical, so the constant value needs to be chosen conservatively. This may impose a fundamental restriction on the eventual use of the protocol.

Manual or "static" configuration of the timer parameter is fraught for two reasons. First, it is can be difficult to ensure that the correct, identical, value is installed in all of the routers. Second, if any change is introduced into the network that results in a need to change the value (for example due to a change in hardware or software version) then all of the routers need to be reconfigured

to use the new timer parameter value. Such consistency may be ensured by deploying automated means such as enforcing the new value by invoking the management interface of all involved routers. For example, a central management entity may be responsible for communicating the new configuration value by means of vendor-specific command line interface (CLI), NETCONF[RFC6241], etc. This approach may be attractive if all involved nodes expose technology-agnostic and vendor-independent interfaces to modify a given network-wide configuration parameter.

This document describes a protocol extension that propagates a routing timer parameter throughout the flooding domain, which can be used as an alternative to the centralized approach described above. The method of choosing between one or more different advertised values, the flooding scope, and the action to be taken when the parameter changes MUST be provided in the definition of the parameter type.

This document also creates one parameter type: Convergence Timer intended for use in IP Fast-reroute applications [RFC5714] [RFC5715].

Note that this protocol is only intended to be used for the propagation of parameters needed to support the operation of the routing system. It MUST NOT be used as a general purpose parameter exchange protocol, and in particular it MUST NOT be used as a parameter negotiation protocol, since such use may degrade the ability of the underlying link-state routing protocol to carry our its essential purpose.

2. Requirements Language

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

3. Overview of Mechanism

Routing Timer parameter values that can be disseminated by means of the attribute defined in this specification MUST be defined as a configurable parameter or a default parameter in the corresponding specification.

A new information element is introduced into the routing protocol that specifies the parameter. Each router taking part in the parameter synchronization is expected to advertise a specific value of the parameter, which that router determines based mainly on considerations local to that router. In general, different routers in the flooding domain may advertise different values of the

parameter. How the values advertised by a router are determined is out of scope of this document.

A router receiving the parameter values advertised by all routers in the flooding domain will use a well-defined method to select the operational value of the parameter that it uses in the running of the protocol. All routers MUST use the same method applied to the same set of advertised parameter values. All routers SHALL therefore choose the same operational value for the parameter.

Note the operational value for the parameter selected SHOULD NOT directly affect the value for the parameter advertised by a router, since this introduces a form of negotiation leading to additional routing protocol traffic and possibly to instability in the routing protocol.

The method of selecting from a range of advertised parameter values MUST be provided in the parameter definition.

The definition of the parameter MUST specify the action to be taken when a new parameter value is advertised that would cause a change in the selected value.

The definition of the parameter MUST specify the action to be taken in the legacy/migration case, where not all routers advertise the parameter.

4. Protocol Details

This section describes the protocol extensions needed to implement this functionality.

4.1. ISIS

A new Routing Timer Parameter Synchronization (RTPS) sub-TLV is introduced into the IS-IS Router CAPABILITY TLV (TLV #242 defined in [RFC7981]). The setting of the S-bit in TLV #242 (indicating whether the parameter should be leaked between levels) MUST be included in the specific routing parameter definition.

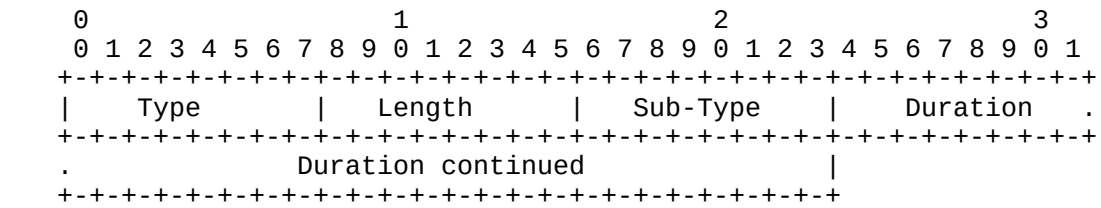


Figure 1: Routing Timer Parameter Synchronization ISIS Sub-TLV

The Type (1 octet) of this sub-TLV to be assigned by IANA.

Length is variable (minimum value 5, multiple of 5 octets) and represents the total Length of the field.

Sub-Type consists of a one octet identifier of the timer type.

Duration is a 32 bit value representing is the length of the timer in milliseconds. This is capable of expressing a time in the range 1ms to just under 50 days.

4.2. OSPF

A new Routing Timer Parameter Synchronization TLV is defined for the OSPF Router Information LSA. This new TLV may be carried in a type 10 or type 11 OSPF Opaque LSA depending on the required flooding scope.
The specification of a the specific routing timer parameter MUST define the appropriate flooding scope(s) for that parameter.

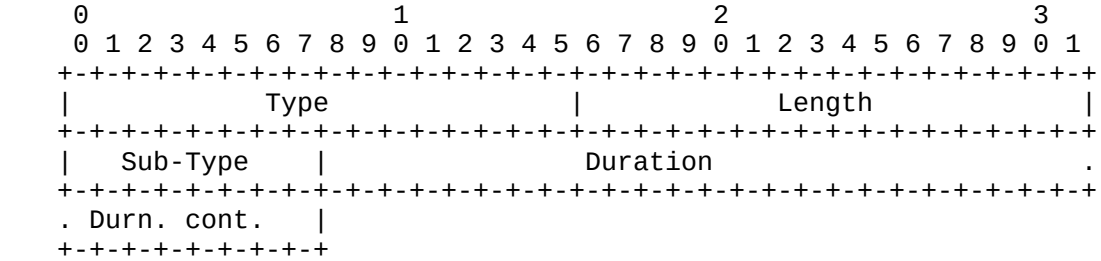


Figure 2: Routing Timer Parameter Synchronization OSPF TLV

The Type (2 octets) of this sub-TLV to be assigned by IANA.

Length is variable (minimum value 5, multiple of 5 octets) and represents the total Length of the field.

Sub-Type consists of a one octet identifier of the timer type.

Duration is a 32 bit value representing is the length of the timer in milliseconds. This is capable of expressing a time in the range 1ms to just under 50 days.

5. Convergence Time

Routers running a fast-reroute mechanism such as Maximally Redundant Tree (MRT) [RFC7812] fast re-route require a network wide convergence time value so that know how long they need continue using the repair path before it is safe to use the base path. This time is set to be the worst case time that any router will take to calculate the new topology, and to make the necessary changes to the FIB.

The time taken by a router to complete each phase of the transition will be dependent on the size of the network and the design and implementation of the router. It can therefore be expected that the optimum delay will need to be tuned from time to time as the network evolves.

5.1. Required Properties

The Convergence Time mechanism MUST have the following properties:

- The operational convergence delay time MUST be consistent among all routers that are converging on the new topology.
- The operational convergence delay time MUST be the highest delay time advertised by any router in the new topology.
- The mechanism MUST increase the delay when a new router in introduced to the network that requires a higher delay than is currently in use.
- When the router that had the longest delay requirements is removed from the topology, the convergence delay timer value MUST, within some reasonable time, be reduced to the longest delay required by the remaining routers.
- It MUST be possible for a router to change the convergence delay timer value that it requires.
- A router which is in multiple routing areas, or is running multiple routing protocols MAY signal a different loop-free convergence delay for each area.

How a router determines the time that it needs to execute each convergence phase is an implementation issue, and outside the scope of this specification. However a router that dynamically determines

its proposed delay value must do so in such a way that it does not cause the synchronized value to continually fluctuate.

5.2. Definition of the Convergence Timer

It is RECOMMENDED that the routing convergence timer be limited to a maximum of 60 seconds.

The routing convergence timer value selected is the largest value advertised.

If a routing protocol message is issued that changes the Convergence Timer value, but does not change the topology, the new timer value MUST be taken into consideration during the next network transition, but MUST NOT instigate a new transition.

If a routing protocol message is issued that changes both the Convergence Timer value and the topology, a transition is instigated and the new timer value MUST be taken into consideration.

The convergence mechanism MUST specify the action to be taken if a timer change (only) message and a topology change message are independently generated during the hold-off time.

A router running ISIS that supports convergence timer synchronization SHOULD advertise the Routing Timer Parameter Synchronization sub-TLV with the value of the Convergence Timer in the Duration field of this sub-TLV. The S-bit is set to zero indicating that the Convergence Timer RPTS sub-TLV MUST NOT be leaked between levels.

A router running OSPF that supports convergence timer synchronization SHOULD advertise the Routing Timer Parameter Synchronization TLV with the value of the Convergence Timer in the Duration field of this TLV. The TLV SHOULD only be carried in the type 10 Opaque LSA which prevents propagation outside the OSPF area.

6. IANA considerations

6.1. ISIS

IANA is requested to allocate a new Sub-TLVs for TLV 242 from the IS-IS TLV Codepoints name space.

Value	Description	Reference
TBD	Routing Timer Parameter Synchronization	This Document

6.2. OSPF

IANA is requested to allocate a new OSPF Router Information (RI) TLV from the Open Shortest Path First (OSPF) Parameters name space

Value	TLV Name	Reference
TBD	Routing Timer Parameter Synchronization	This document

A value in the range 12 to 32767 is requested.

6.3. Routing Timer Parameter Synchronization Registry

IANA is requested to create a new Routing Timer Parameter Synchronization Registry within its own name space, and to allocate one value from it.

Value	Name	Reference
0	Reserved	This document
1	Convergence Timer	This document
2..255	Reserved	This document

Allocations within this registry require IETF Consensus. This link state protocol extension MUST NOT be used for any purpose other than one associated with the routing system timer parameters.

7. Security Considerations

The introduction of this parameter advertising mechanism does not introduce a significant vulnerability into the base routing protocol and is secured in exactly the same way as the other TLVs that are carried.

In specifying a new parameter, consideration must be given to the impact of the additional parameter, and in particular the rate of change of that parameter, on the dynamics of the link-state routing protocol in use. In the specific case of the Convergence Timer, the amount of data being carried and the rate of change of the parameter value will have a negligible impact on the link-state routing protocol in use.

A rouge router deliberately introducing an anomalous parameter value is just as capable of introducing many other anomalies into the routing domain.

As far as possible, care should be taken to validate that the parameter is reasonable.

In the specific case of the Convergence Time RTPS, the following considerations apply.

If an abnormally large timer value is proposed by a router, the there is a danger that the convergence process will take an excessive time. If during that time the routing protocol signals the need for another transition, the transition will be abandoned and the default best case (traditional) convergence mechanism used.

It is RECOMMENDED that implementations prohibit the configuration of a router convergence timer value in excess of 60 seconds.

8. Acknowledgments

The authors thank Les Ginsberg and the other authors of [I-D.ietf-isis-segment-routing-msd] and [I-D.ietf-ospf-segment-routing-msd] , Mohamed Boucadair for their review comments and proposed text, and Tom Petch for his review comments.

9. Contributing Authors

Mike Shand
Independent
mike@mshand.org.uk

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>.

[RFC7981] Ginsberg, L., Previdi, S., and M. Chen, "IS-IS Extensions for Advertising Router Information", RFC 7981, DOI 10.17487/RFC7981, October 2016, <http://www.rfc-editor.org/info/rfc7981>.

10.2. Informative References

[I-D.ietf-isis-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling MSD (Maximum SID Depth) using IS-IS", draft-ietf-isis-segment-routing-msd-04 (work in progress), June 2017.

[I-D.ietf-ospf-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling MSD (Maximum SID Depth) using OSPF", draft-ietf-ospf-segment-routing-msd-05 (work in progress), June 2017.

[RFC5714] Shand, M. and S. Bryant, "IP Fast Reroute Framework", RFC 5714, DOI 10.17487/RFC5714, January 2010, <http://www.rfc-editor.org/info/rfc5714>.

[RFC5715] Shand, M. and S. Bryant, "A Framework for Loop-Free Convergence", RFC 5715, DOI 10.17487/RFC5715, January 2010, <http://www.rfc-editor.org/info/rfc5715>.

[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>.

[RFC7812] Atlas, A., Bowers, C., and G. Enyedi, "An Architecture for IP/LDP Fast Reroute Using Maximally Redundant Trees (MRT-FRR)", RFC 7812, DOI 10.17487/RFC7812, June 2016, <http://www.rfc-editor.org/info/rfc7812>.

Authors' Addresses

Stewart Bryant
Huawei Technologies

Email: stewart.bryant@gmail.com

Alia Atlas
Juniper Networks

Email: akatlas@gmail.com

Chris Bowers
Juniper Networks

Email: cbowers@juniper.net

Network Working Group
Internet-Draft
Intended status: Informational

L. Dunbar
L. Yong
Song Xiao Lin
Huawei

Expires: April 2017

October 31, 2016

Client Defined Private Networks laid over Thin CPEs
draft-dunbar-opsawg-private-networks-over-thin-cpe-01

Abstract

This document specifies a type of private networks that interconnect thin CPEs at multiple client sites by IP tunnels, or more specifically, lay over multiple client sites' Thin CPEs via IP tunnels. Those private overlay networks not only interconnect those sites by secure IP tunnels but can also enforce the client specified policies to govern how applications or hosts within those sites communicate and how to access public internet.

Hosts or applications in those sites can be interconnected by Layer 2 networks or/and by Layer 3 networks. The network that the IP tunnels are traversing can be IPv4 or IPv6 networks. This document describes the special properties of the client defined networks over Thin CPEs.

A separate draft will describes the special features that those IP tunnels need to have in order to interconnect multiple sites as if those sites are directly connected by wires and how communication policies are enforced.

Status of This Document

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

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

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

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

This Internet-Draft will expire on April 31, 2017.

Table of Contents

1. Introduction

This document specifies a type of private networks that interconnect thin CPEs at multiple client sites by IP tunnels, or more specifically, lay over multiple client sites' Thin CPEs via IP tunnels. Those private overlay networks not only interconnect those sites by secure IP tunnels but can also enforce the client specified policies to govern how applications or hosts within those sites communicate and how to access public internet.

Hosts or applications in those sites can be interconnected by Layer 2 networks or/and by Layer 3 networks. The network that the IP tunnels are traversing can be IPv4 or IPv6 networks. This document describes the special properties of the client defined networks over Thin CPEs.

For ease of description, the "Client Defined Private Overlay Network" is also called the client's "Overlay Private Network" or "Overlay Virtual Private Network (Overlay-VPN)" throughout this document.

A separate draft will describes the special features that those IP tunnels need to have in order to interconnect multiple sites as if those sites are directly connected by wires and how communication policies are enforced.

2. Terminology

2.1. Requirements Language

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

2.2. Terms defined in this document

Internet Gateway: a network function, which can be a physical device in the provider site or a virtual function instantiated to connect client site traffic to the public internet, and can enforce client specified policies.

Overlay Private Network: private network over a set of thin CPEs at multiple sites created by clients or users, who don't need to worry

about how thin CPEs are connected nor the protocol setting at network side. The "Overlay Private Network" not only interconnects multiple sites by (secure) IP tunnels but can also enforce the client specified policies to govern how applications or hosts within those sites communicate and how to access public internet.

Overlay-VPN: Overlay Private Network.

Provider site: the location where the provider have access to the devices or equipment.

Site: A place that contains switches, routers, services, appliances and these devices are configured to form L2 domain (s) or L3 domain. For example an Enterprise company data center, a college campus network center. For L3 subnets, either private IPv4 or IPv6 address or public IPv4 or Ipv6 address can be used.

SITE: Site Interconnection Tunnel Encapsulation Protocol

Thin CPE: a simple device at a customer premise that maps the site local traffic to either the IP tunnels connected to the Internet Gateway, or the IP tunnels connected to the VPN Gateway.

Overlay-VPN Gateway: the function (which can be virtual) that establish private (secure) connections to other sites belonging to the same client.

3. Brief Description of the Private networks laid over Thin CPEs

The following figure depicts multiple overlay private networks that interconnect the client's various sites. Note, the Overlay Private Network is marked as "Overlay" in the figure. The client can create multiple overlay private networks and then assign each site to specific overlay private networks. The client also specify the policies on what traffic to/from the clients can be exchanged with external network, which are enforced by the "Internet gateways" created by the provider.

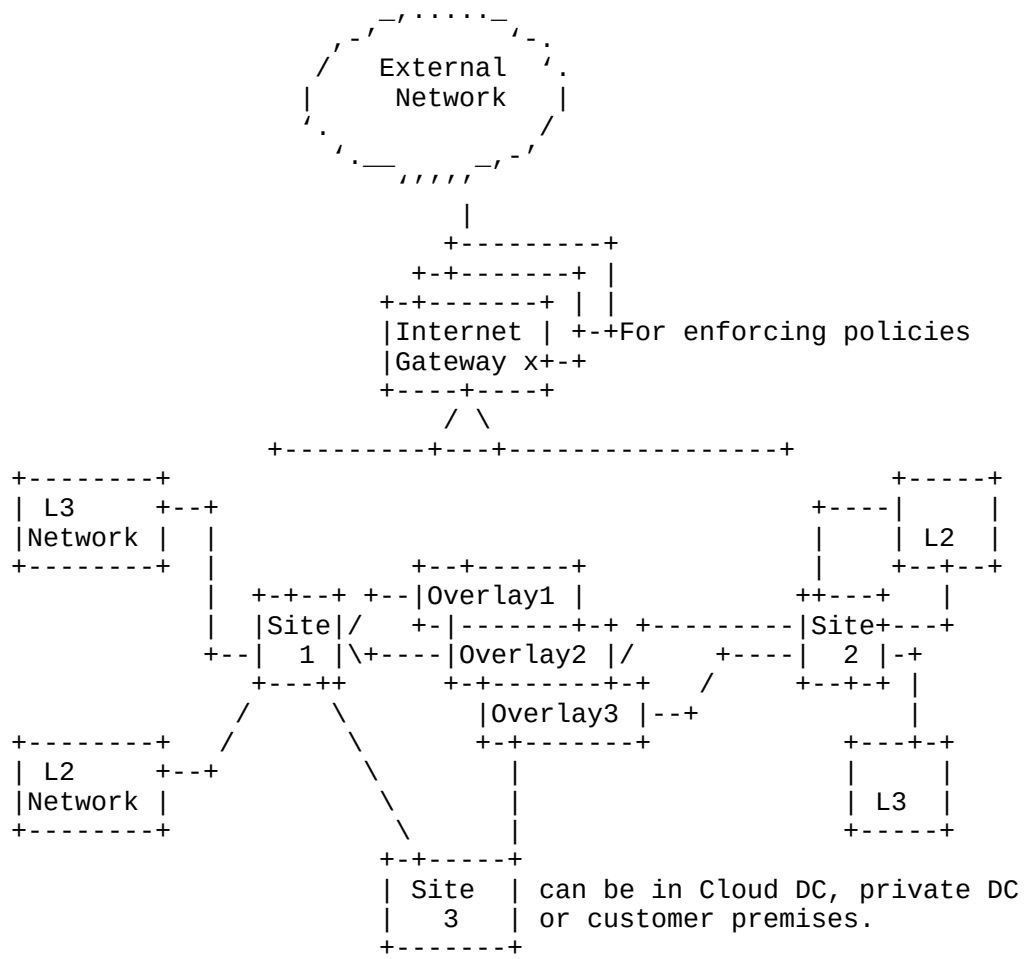


Figure 1 Overlay Private Networks interconnecting sites

Here are some key properties of Client defined Overlay Private Networks:

- Each client "Site" has a Thin CPE that is connected to a VPN gateway which is hosted in the provider site via IP Tunnel (which can be secured per customer request). The Thin CPE can be software image instantiated on virtual machines, physical CPE, or other form factors.

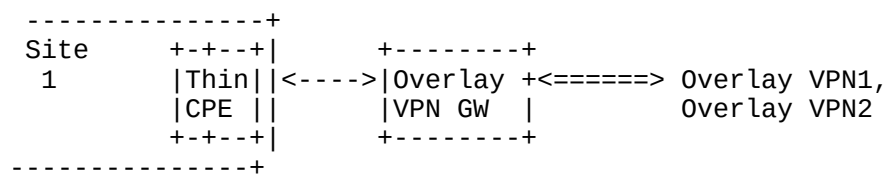


Figure 2 site Thin CPE connect to Overlay GW via IP Tunnel

- Each Thin CPE is connected to an "Internet Gateway" via IP Tunnel (that is automatically created by provider). The "Internet Gateway", virtual or physical, can be located anywhere. An IP Tunnel is created automatically between the Thin CPE and the "Internet Gateway".

- When the provider don’t own the infrastructure to interconnect multiple sites, (secure) IP Tunnels are created among each site’s VPN Gateway, so that each site’s local networks (L2 or L3) attached to the Thin CPEs are interconnected as if those networks are directly connected by physical wire.

- Some traffic between Thin CPE have to go through secure tunnel, e.g. IPSec. Clients can specify what traffic to go through secure tunnels without specifically worrying about how to establish or maintain the secure tunnels. The client traffic can be carried by VxLAN (for interconnecting layer 2 traffic) or GRE (for L3 traffic) over the IPSEc tunnel.

- Client specifies the policies on how/what/when hosts from the interconnected sites can communicate with external peers; E.g. Hosts in one Layer 2 domain from one site may communicate with hosts in different Layer 2 domains in different sites.

The Client Defined Overlay Networks can be viewed by client as their own private networks. For ease of description, the terminology "Overlay Private Network" or "Overlay-VPN" is used throughout this document to refer to this kind of client defined overlay network over Thin CPEs.

"Overlay Private Network" is different from the IETF's L2VPN or L3VPN for the following reasons:

- Overlay-Private-Network is built upon IP network (whereas L2VPN/L3VPN is built upon MPLS network),

- Traffic originated from a client's site (where Thin CPE is instantiated) not only can communicate with hosts in other sites of the client via IP tunnels, but also can communicate with public internet (governed by the policies specified by the client),

- Client's site Thin CPE don't participate in IGP or BGP routing with provider side. Client can specify the prefixes and/or VLANs for each site so that they can be reached by external hosts,

- IP tunnel is automatically created between a Thin CPE and provider site where VPN gateway and internet gateway are instantiated and maintained.

4. Overlay Private Network Configuration from Client Perspective

4.1. Client Defined Overlay Private Networks

The client can specify multiple overlay private networks (a.k.a. Overlay-VPNs). Client can specify which sites connect to which Overlay-VPNs. Each Site can connect to multiple Overlay-VPNs.

As features on Thin CPE are very limited, each Overlay-VPN has its own Overlay VPN gateway in provider site to connect to Thin CPE via IP tunnel, as depicted in Figure 2 above.

4.2. Client's site Configuration

For each site, the client needs to specify:

- Site Identifier (include unique system Identifier, name, etc.)

- VLANs enabled on the site (i.e. the VLANs enabled on the client facing ports of the Thin CPE).

- Subnets from the site (i.e. the subnets enabled on the client facing ports of the Thin CPE)

- IP address for the Overlay-VPN Gateway that connect other sites belonging to the client

- IP address for the Internet Gateway

The configuration on the site is mainly for the Thin CPE instantiated on the site. Therefore, the client also needs to specify which VLANs/subnets are enabled on the ports of the Thin CPE facing the local network on the site.

4.3. Internet Gateway for each Site

Each site is associated with an Internet Gateway, which is automatically created by the provider. The Interconnect gateway can be a physical device on the provider site or a virtual function, to connect client site traffic to the public internet, and can enforce client specified policies.

Considering one client can have multiple sites in different geographic locations, the client can specify different policies for traffic to/from each site.

4.4. Overlay-VPN Gateway

The Overlay-VPN Gateway is on the provider site, connected to Thin CPE via IP tunnel. The purpose of the Overlay-VPN Gateway is to connect a site to its specified Overlay VPNs. Each site can be connected to multiple Overlay VPNs.

For each Overlay-VPN gateway, the client needs to specify:

- Identifier

- Which VPN is the Gateway connected to

- Upstream bandwidth from Thin CPE to the Overlay VPN GW

- Downstream bandwidth from the Overlay VPN GW to the Thin CPE

4.5. Interconnection among Sites

For each Overlay VPN, the Client can choose which sites are connected by specifying the VPN Gateway associated with each site.

5. Protocols needed for the Client Defined Overlay Private Networks

5.1. Thin CPE Auto Instantiation

Thin CPE is a simple device that maps the site local traffic to either the IP tunnels connected to the Internet Gateway, or the IP tunnels connected to the VPN Gateway.

5.2. Network agnostic interworking

IP tunnels are automatically created between Thin CPE and (Internet/VPN) gateways based on the traffic to the access network.

For Layer 2 traffic from the client local site, VxLAN is used to build the IP Tunnels to the site’s Internet gateway or VPN gateway respectively.

For Layer 3 traffic from the client local site, GRE is used to build the IP Tunnels to the site’s Internet gateway or VPN gateway respectively.

If the client specifies secure connection to other sites, IPSec is added to the tunnels between the Thin CPE and the VPN Gateway.

5.3. Gateway Anchor Auto-Selection

For each client site, internet gateway and VPN gateway will be automatically instantiated.

There will be protocol extension needed for the creation/deletion process and how NAT is used for client traffic from each site.

5.4. Middle boxes auto-creation and rules exchanges

To be added

5.5. Thin CPE on Third Party location

Thin CPEs can also be instantiated third party premises, such as cloud data centers. The instantiated Thin CPE can establish IP tunnels with the client's Internet Gateway or VPN Gateway.

5.6. Client Defined Polices for traffic to/from client sites

Depending on the policies specified by the clients, the Thin CPE jointly with the virtual GW will select the appropriate network security functions, i.e. (virtual) FW, IPS, IDS, or others to enforce the policies specified by the clients.

The policies specified by the clients will be more expressed in clients' oriented language, e.g. using client Identifier or virtual addresses (instead of IP addresses of the actual packets traverse the FW). Those policies will be translated to the implementable rules to the chosen network security functions, such as FW.

5.7. QoS policies

To be added

5.8. Explicit Service functions chain specified by clients

Clients can query network service functions available to them and the capabilities of those functions. Then, the client can choose a set of them, either in strict sequence or simply as a set to apply to their traffic.

The policies to service functions can follow the guideline specified by [I2NSF-framework].

5.9. Thin CPE monitoring

5.10. Alarm & Events via Thin CPE

To be added

5.11. Resource management via Thin CPE instantiated in Remote Locations

To be added

5.12. Client traffic flows management, monitoring, and reporting

To be added

6. Networks carried by IP tunnels in conjunction with existing L2VPN/L3VPN

7. IANA Considerations

To be added

8. Security Considerations

To be added.

9. References

9.1. Normative References

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

9.2. Informative Reference

[I2NSF-Framework] Lopez, D, et al, "Framework for Interface to Network security functions", draft-ietf-i2nsf-framework-04, Oct 2016

10. Authors' Addresses

Linda Dunbar
Huawei Technologies
Email: linda.dunbar@huawei.com

Lucy Yong
Huawei Technologies
Email: lucy.yong@huawei.com

Song Xiao Li
Huawei Technologies
Email: sxlin@huawei.com

11. Contributors Addresses

Xuan Ming fu
Huawei Technologies
xuanmingfu@huawei.com

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

A. Lindem, Ed.
Cisco Systems
L. Berger, Ed.
LabN Consulting, L.L.C.
D. Bogdanovic

C. Hopps
Deutsche Telekom

March 13, 2017

Network Device YANG Logical Organization
draft-ietf-rtgwg-device-model-02

Abstract

This document presents an approach for organizing YANG models in a comprehensive logical structure that may be used to configure and operate network devices. The structure is itself represented as an example YANG model, with all of the related component models logically organized in a way that is operationally intuitive, but this model is not expected to be implemented. The identified component modules are expected to be defined and implemented on common network devices.

This document is derived from work submitted to the IETF by members of the informal OpenConfig working group of network operators and is a product of the Routing Area YANG Architecture design team.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 14, 2017.

Table of Contents

1. Introduction

"Operational Structure and Organization of YANG Models" [I-D.openconfig-netmod-model-structure], highlights the value of organizing individual, self-standing YANG [RFC6020] models into a more comprehensive structure. This document builds on that work and presents a derivative logical structure for use in representing the networking infrastructure aspects of physical and virtual devices. [I-D.openconfig-netmod-model-structure] and earlier versions of this document presented a single device-centric model root, this document no longer contains this element. Such an element would have translated to a single device management model that would be the root

of all other models and was judged to be overly restrictive in terms of definition, implementation, and operation.

The document presents a logical network device YANG organizational structure that provides a conceptual framework for the models that may be used to configure and operate network devices. The structure is itself presented as an example YANG module, with all of the related component modules logically organized in a way that is operationally intuitive. This network device model is not expected to be implemented, but rather provide as context for the identified representative component modules with are expected to be defined, and supported on typical network devices.

This document refers to two new modules that are expected to be implemented. These models are defined to support the configuration and operation of network-devices that allow for the partitioning of resources from both, or either, management and networking perspectives. Two forms of resource partitioning are referenced:

The first form provides a logical partitioning of a network device where each partition is separately managed as essentially an independent network element which is 'hosted' by the base network device. These hosted network elements are referred to as logical network elements, or LNEs, and are supported by the logical-network-element module defined in [I-D.ietf-rtgwg-lne-model]. The module is used to identify LNEs and associate resources from the network-device with each LNE. LNEs themselves are represented in YANG as independent network devices; each accessed independently. Optionally, and when supported by the implementation, they may also be accessed from the host system. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric.

The second form provides support what is commonly referred to as Virtual Routing and Forwarding (VRF) instances as well as Virtual Switch Instances (VSI), see [RFC4026]. In this form of resource partitioning multiple control plane and forwarding/bridging instances are provided by and managed via a single (physical or logical) network device. This form of resource partitioning is referred to as Network Instances and are supported by the network-instance module defined in [I-D.ietf-rtgwg-ni-model]. Configuration and operation of each network-instance is always via the network device and the network-instance module.

This document was motivated by, and derived from, [I-D.openconfig-netmod-model-structure]. The requirements from that document have been combined with the requirements from "Consistent Modeling of Operational State Data in YANG",

[I-D.openconfig-netmod-opstate], into "NETMOD Operational State Requirements", [I-D.ietf-netmod-opstate-reqs]. This document is aimed at the requirement related to a common model-structure, currently Requirement 7, and also aims to provide a modeling base for Operational State representation.

The approach taken in this (and the original) document is to organize the models describing various aspects of network infrastructure, focusing on devices, their subsystems, and relevant protocols operating at the link and network layers. The proposal does not consider a common model for higher level network services. We focus on the set of models that are commonly used by network operators, and suggest a corresponding organization.

A significant portion of the text and model contained in this document was taken from the -00 of [I-D.openconfig-netmod-model-structure].

1.1. Status of Work and Open Issues

This version of the document and structure are a product of the Routing Area YANG Architecture design team and is very much a work in progress rather than a final proposal. This version is a major change from the prior version and this change was enabled by the work on the previously mentioned Schema Mount.

Schema Mount enables a dramatic simplification of the presented device model, particularly for "lower-end" devices which are unlikely to support multiple network instances or logical network elements. Should structural-mount/YSDL not be available, the more explicit tree structure presented in earlier versions of this document will need to be utilized.

The top open issues are:

1. This document will need to match the evolution and standardization of [I-D.openconfig-netmod-opstate] or [I-D.ietf-netmod-opstate-reqs] by the Netmod WG.

2. Interpretation of different policy containers requires clarification.

3. It may make sense to use the identityref structuring with hardware and QoS model.

4. Which document(s) define the base System management, network services, and oam protocols modules is TBD. This includes the

possibility of simply using RFC7317 in place of the presented System management module.

5. The model will be updated once the "opstate" requirements are addressed.

2. Module Overview

In this document, we consider network devices that support protocols and functions defined within the IETF Routing Area, e.g, routers, firewalls and hosts. Such devices may be physical or virtual, e.g., a classic router with custom hardware or one residing within a server-based virtual machine implementing a virtual network function (VNF). Each device may sub-divide their resources into logical network elements (LNEs) each of which provides a managed logical device. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric. Each LNE may also support virtual routing and forwarding (VRF) and virtual switching instance (VSI) functions, which are referred to below as a network instances (NIs). This breakdown is represented in Figure 1.

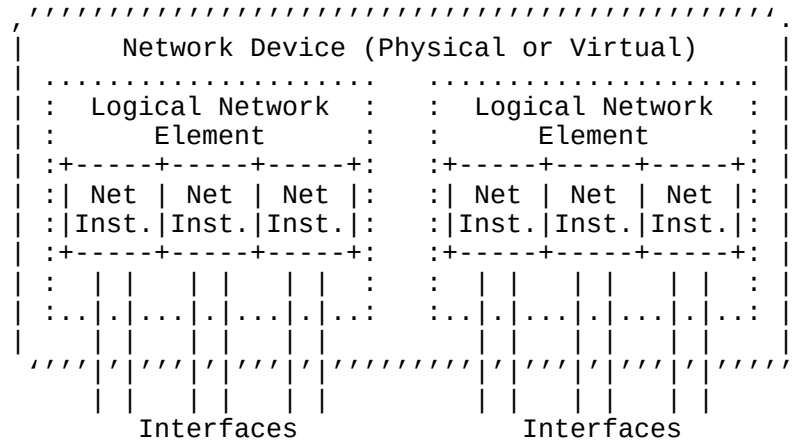


Figure 1: Module Element Relationships

A model for LNEs is described in [I-D.ietf-rtgwg-lne-model] and the model for network instances is covered in [I-D.ietf-rtgwg-ni-model].

The presented notional network device module can itself be thought of as a "meta-model" as it describes the relationships between individual models. We choose to represent it also as a simple YANG module consisting of other models, which are in fact independent top

level individual models. Although it is never expected to be implemented.

The presented modules do not follow the hierarchy of any Particular implementation, and hence is vendor-neutral. Nevertheless, the structure should be familiar to network operators and also readily mapped to vendor implementations.

The overall structure is:

```
    module: example-network-device
       +--rw modules-state             [RFC7895]
       |
       +--rw interfaces                [RFC7223]
       +--rw hardware
       +--rw qos
       |
       +--rw system-management         [RFC7317 or derived]
       +--rw network-services
       +--rw oam-protocols
       |
       +--rw routing                   [I-D.ietf-netmod-routing-cfg]
       +--rw mpls
       +--rw ieee-dot1Q
       |
       +--rw acls                      [I-D.ietf-netmod-acl-model]
       +--rw key-chains                [I-D.ietf-rtgwg-yang-key-chain]
       |
       +--rw logical-network-elements [I-D.ietf-rtgwg-lne-model]
       +--rw network-instances         [I-D.ietf-rtgwg-ni-model]
```

The network device is composed of top level modules that can be used to configure and operate a network device. (This is a significant difference from earlier versions of this document where there was a strict model hierarchy.) Importantly the network device structure is the same for a physical network device or a logical network device, such as those instantiated via the logical-network-element model. Extra spacing is included to denote different types of modules included.

YANG library [RFC7895] is included as it used to identify details of the top level modules supported by the (physical or logical) network device. Th ability to identify supported modules is particularly important for LNEs which may have a set of supported modules which differs from the set supported by the host network device.

The interface management model [RFC7223] is included at the top level. The hardware module is a placeholder for a future device-

specific configuration and operational state data model. For example, a common structure for the hardware model might include chassis, line cards, and ports, but we leave this unspecified. The quality of service (QoS) section is also a placeholder module for device configuration and operational state data which relates to the treatment of traffic across the device. This document references augmentations to the interface module to support LNEs and NIs. Similar elements, although perhaps only for LNEs, may also need to be included as part of the definition of the future hardware and QoS modules.

System management, network services, and oam protocols represent new top level modules that are used to organize data models of similar functions. Additional information on each is provided below.

The routing and MPLS modules provide core support for the configuration and operation of a devices control plane and data plane functions. IEEE dot1Q [IEEE-8021Q] is an example of another module that provides similar functions for VLAN bridging, and other similar modules are also possible. Each of these modules is expected to be LNE and NI unaware, and to be instantiated as needed as part of the LNE and NI configuration and operation supported by the logical-network-element and network-instance modules. (Note that this is a change from [I-D.ietf-netmod-routing-cfg] which is currently defined with VRF/NI semantics.)

The access control list (ACL) and key chain modules are included as examples of other top level modules that may be supported by a network device.

The logical network element and network instance modules enable LNEs and NIs respectively and are defined below.

2.1. Interface Model Components

Interfaces are a crucial part of any network device’s configuration and operational state. They generally include a combination of raw physical interfaces, link-layer interfaces, addressing configuration, and logical interfaces that may not be tied to any physical interface. Several system services, and layer 2 and layer 3 protocols may also associate configuration or operational state data with different types of interfaces (these relationships are not shown for simplicity). The interface management model is defined by [RFC7223].

The logical-network-element and network-instance modules defined in [I-D.ietf-rtgwg-lne-model] and [I-D.ietf-rtgwg-ni-model] augment the existing interface management model in two ways: The first, by the

logical-network-element module, adds an identifier which is used on physical interface types to identify an associated LNE. The second, by the network-instance module, adds a name which is used on interface or sub-interface types to identify an associated network instance. Similarly, this name is also added for IPv4 and IPv6 types, as defined in [RFC7277].

The interface related augmentations are as follows:

```
module: ietf-logical-network-element
augment /if:interfaces/if:interface:
   +--rw bind-lne-name?   string

module: ietf-network-instance
augment /if:interfaces/if:interface:
   +--rw bind-network-instance-name?   string
augment /if:interfaces/if:interface/ip:ipv4:
   +--rw bind-network-instance-name?   string
augment /if:interfaces/if:interface/ip:ipv6:
   +--rw bind-network-instance-name?   string
```

The following is an example of envisioned combined usage. The interfaces container includes a number of commonly used components as examples:

```
            +--rw if:interfaces
            |  +--rw interface* [name]
            |     +--rw name                        string
            |     +--rw lne:bind-lne-name?          string
            |     +--rw ethernet
            |     |  +--rw ni:bind-network-instance-name? string
            |     |  +--rw aggregates
            |     |  +--rw rstp
            |     |  +--rw lldp
            |     |  +--rw ptp
            |     +--rw vlans
            |     +--rw tunnels
            |     +--rw ipv4
            |     |  +--rw ni:bind-network-instance-name? string
            |     |  +--rw arp
            |     |  +--rw icmp
            |     |  +--rw vrrp
            |     |  +--rw dhcp-client
            |     +--rw ipv6
            |        +--rw ni:bind-network-instance-name? string
            |        +--rw vrrp
            |        +--rw icmpv6
            |        +--rw nd
            |        +--rw dhcpv6-client
```

The [RFC7223] defined interface model is structured to include all interfaces in a flat list, without regard to logical or virtual instances (e.g., VRFs) supported on the device. The bind-lne-name and bind-network-instance-name leaves provide the association between an interface and its associated LNE and NI (e.g., VRF or VSI).

2.2. System Management

[Editor’s note: need to discuss and resolve relationship between this structure and RFC7317 and determine if 7317 is close enough to simply use as is.]

System management is expected to reuse definitions contained in [RFC7317]. It is expected to be instantiated per device and LNE. Its structure is shown below:

```
    module: example-network-device
       +--rw system-management
       |  +--rw system-management-global
       |  +--rw system-management-protocol* [type]
       |     +--rw type    identityref
```

System-management-global is used for configuration information and state that is independent of a particular management protocol. System-management-protocol is a list of management protocol specific elements. The type-specific sub-modules are expected to be defined.

The following is an example of envisioned usage:

```
module: example-network-device
   +--rw system-management
      +--rw system-management-global
      |  +--rw statistics-collection
      |  ...
      +--rw system-management-protocol* [type]
      |  +--rw type=syslog
      |  +--rw type=dns
      |  +--rw type=ntp
      |  +--rw type=ssh
      |  +--rw type=tacacs
      |  +--rw type=snmp
      |  +--rw type=netconf
```

2.3. Network Services

A device may provide different network services to other devices, for example a device my act as a DHCP server. The model may be instantiated per device, LNE, and NI. An identityref is used to identify the type of specific service being provided and its associated configuration and state information. The defined structure is as follows:

```
module: example-network-device
   +--rw network-services
   |  +--rw network-service* [type]
   |     +--rw type    identityref
```

The following is an example of envisioned usage: Examples shown below include a device-based Network Time Protocol (NTP) server, a Domain Name System (DNS) server, and a Dynamic Host Configuration Protocol (DHCP) server:

```
module: example-network-device
   +--rw network-services
      +--rw network-service* [type]
         +--rw type=ntp-server
         +--rw type=dns-server
         +--rw type=dhcp-server
```

2.4. OAM Protocols

OAM protocols that may run within the context of a device are grouped within the oam-protocols model. The model may be instantiated per device, LNE, and NI. An identifyref is used to identify the information and state that may relate to a specific OAM protocol. The defined structure is as follows:

```
module: example-network-device
   +--rw oam-protocols
      +--rw oam-protocol* [type]
         +--rw type    identityref
```

The following is an example of envisioned usage. Examples shown below include Bi-directional Forwarding Detection (BFD), Ethernet Connectivity Fault Management (CFM), and Two-Way Active Measurement Protocol (TWAMP):

```
module: example-network-device
   +--rw oam-protocols
      +--rw oam-protocol* [type]
         +--rw type=bfd
         +--rw type=cfm
         +--rw type=twamp
```

2.5. Routing

Routing protocol and IP forwarding configuration and operation information is modeled via a routing model, such as the one defined in [I-D.ietf-netmod-routing-cfg].

The routing module is expected to include all IETF defined control plane protocols, such as BGP, OSPF, LDP and RSVP-TE. It is also expected to support configuration and operation of or more routing information bases (RIB). A RIB is a list of routes complemented with administrative data. Finally, policy is expected to be represented within each control plane protocol and RIB.

The anticipated structure is as follows:

```
module: example-network-device
    +--rw rt:routing                [I-D.ietf-netmod-routing-cfg]
       +--rw control-plane-protocol* [type]
       |  +--rw type      identityref
       |  +--rw policy
       +--rw rib* [name]
          +--rw name           string
          +--rw description?   string
          +--rw policy
```

2.6. MPLS

MPLS data plane related information is grouped together, as with the previously discussed modules, is unaware of VRFs/NIs. The model may be instantiated per device, LNE, and NI. MPLS control plane protocols are expected to be included in Section 2.5. MPLS may reuse and build on [I-D.openconfig-mpls-consolidated-model] or other emerging models and has an anticipated structure as follows:

```
module: example-network-device
     +--rw mpls
        +--rw global
        +--rw lsps* [type]
           +--rw type    identityref
```

Type refers to LSP type, such as static, traffic engineered or routing congruent. The following is an example of such usage:

```
module: example-network-device
     +--rw mpls
        +--rw global
        +--rw lsps* [type]
            +--rw type=static
            +--rw type=constrained-paths
            +--rw type=igp-congruent
```

3. Security Considerations

The network-device model structure described in this document does not define actual configuration and state data, hence it is not directly responsible for security risks.

Each of the component models that provide the corresponding configuration and state data should be considered sensitive from a security standpoint since they generally manipulate aspects of network configurations. Each component model should be carefully evaluated to determine its security risks, along with mitigations to reduce such risks.

LNE portion is TBD

NI portion is TBD

4. IANA Considerations

This YANG model currently uses a temporary ad-hoc namespace. If it is placed or redirected for the standards track, an appropriate namespace URI will be registered in the "IETF XML Registry" [RFC3688]. The YANG structure modules will be registered in the "YANG Module Names" registry [RFC6020].

5. Network Device Model Structure

```
<CODE BEGINS> file "example-network-device@2017-03-13.yang"
module example-network-device {

  yang-version "1";

  // namespace
  namespace "urn:example:network-device";

  prefix "nd";

  // import some basic types

  // meta
  organization "IETF RTG YANG Design Team Collaboration
                with OpenConfig";

  contact
      "Routing Area YANG Architecture Design Team -
       <rtg-dt-yang-arch@ietf.org>";

  description
    "This module describes a model structure for YANG
     configuration and operational state data models. Its intent is
     to describe how individual device protocol and feature models
     fit together and interact.";

  revision "2017-03-13" {
    description
      "IETF Routing YANG Design Team Meta-Model";
    reference "TBD";
  }

  // extension statements
```

```
     // identity statements

     identity oam-protocol-type {
         description
             "Base identity for derivation of OAM protocols";
     }

     identity network-service-type {
         description
             "Base identity for derivation of network services";
     }

      identity system-management-protocol-type {
         description
             "Base identity for derivation of system management
              protocols";
      }

      identity oam-service-type {
         description
             "Base identity for derivation of Operations,
              Administration, and Maintenance (OAM) services.";
      }

      identity control-plane-protocol-type {
         description
             "Base identity for derivation of control-plane protocols";
      }

      identity mpls-lsp-type {
         description
             "Base identity for derivation of MPLS LSP typs";
      }

     // typedef statements

     // grouping statements

     grouping ribs {
       description
         "Routing Information Bases (RIBs) supported by a
          network-instance";
       container ribs {
           description
               "RIBs supported by a network-instance";
           list rib {
               key "name";
               min-elements "1";
```

```
            description
                "Each entry represents a RIB identified by the
               'name' key. All routes in a RIB must belong to the
                same address family.

                For each routing instance, an implementation should
                provide one system-controlled default RIB for each
                supported address family.";
            leaf name {
                type string;
                description
                    "The name of the RIB.";
            }
            reference "draft-ietf-netmod-routing-cfg";
            leaf description {
                type string;
                description
                    "Description of the RIB";
            }
            // Note that there is no list of interfaces within
            container policy {
                description "Policy specific to RIB";
            }
        }
      }
    }

    // top level device definition statements
    container ietf-yang-library {
      description
        "YANG Module Library as defined in
         draft-ietf-netconf-yang-library";
    }

    container interfaces {
      description
       "Interface list as defined by RFC7223/RFC7224";
    }

    container hardware {
      description
        "Hardware / vendor-specific data relevant to the platform.
        This container is an anchor point for platform-specific
        configuration and operational state data.  It may be further
        organized into chassis, line cards, ports, etc.  It is
        expected that vendor or platform-specific augmentations
        would be used to populate this part of the device model";
    }
```

```
    container qos {
      description "QoS features, for example policing, shaping, etc.";
    }

    container system-management {
        description
          "System management for physical or virtual device.";
        container system-management-global {
            description "System management - with reuse of RFC 7317";
        }
        list system-management-protocol {
            key "type";
            leaf type {
                type identityref {
                    base system-management-protocol-type;
                }
                mandatory true;
                description
                    "Syslog, ssh, TACAC+, SNMP, NETCONF, etc.";
            }
            description "List of system management protocol
                         configured for a logical network
                         element.";
        }
    }

    container network-services {
        description
            "Container for list of configured network
             services.";
        list network-service {
            key "type";
            description
                "List of network services configured for a
                 network instance.";
            leaf type {
                type identityref {
                    base network-service-type;
                }
                mandatory true;
                description
                    "The network service type supported within
                     a network instance, e.g., NTP server, DNS
                     server, DHCP server, etc.";
            }
        }
    }
```

```
    container oam-protocols {
        description
            "Container for configured OAM protocols.";
        list oam-protocol {
            key "type";
            leaf type {
                type identityref {
                    base oam-protocol-type;
                }
                mandatory true;
                description
                    "The Operations, Administration, and
                     Maintenance (OAM) protocol type, e.g., BFD,
                     TWAMP, CFM, etc.";
            }
            description
                "List of configured OAM protocols.";
        }
    }

    container routing {
      description
        "The YANG Data Model for Routing Management revised to be
         Network Instance / VRF independent. ";
      // Note that there is no routing or network instance
      list control-plane-protocol {
          key "type";
          description
              "List of control plane protocols configured for
               a network instance.";
          leaf type {
              type identityref {
                  base control-plane-protocol-type;
              }
              mandatory true;
              description
                  "The control plane protocol type, e.g., BGP,
                   OSPF IS-IS, etc";
          }
          container policy {
              description
                  "Protocol specific policy,
                  reusing [RTG-POLICY]";
          }
      }
      list rib {
          key "name";
          description
```

```
                "Each entry represents a RIB identified by the
               'name' key. All routes in a RIB must belong to the
                same address family.

                For each routing instance, an implementation should
                provide one system-controlled default RIB for each
                supported address family.";
            leaf name {
                type string;
                mandatory true;
                description
                    "The name of the RIB.";
            }
            reference "draft-ietf-netmod-routing-cfg";
            leaf description {
                type string;
                description
                    "Description of the RIB";
            }
            // Note that there is no list of interfaces within
            container policy {
                description "Policy specific to RIB";
            }
        }
    }

    container mpls {
        description "MPLS and TE configuration";
        container global {
            description "Global MPLS configuration";
        }
        list lsps {
            key "type";
            description
                "List of LSP types.";
            leaf type {
                type identityref {
                    base mpls-lsp-type;
                }
                mandatory true;
                description
                    "MPLS and Traffic Engineering protocol LSP types,
                     static, LDP/SR (igp-congruent),
                     RSVP TE (constrained-paths) , etc.";
            }
        }
    }
```

```
     container ieee-dot1Q {
       description
         "The YANG Data Model for VLAN bridges as defined by the IEEE";
     }

     container ietf-acl {
       description "Packet Access Control Lists (ACLs) as specified
                      in draft-ietf-netmod-acl-model";
     }

     container ietf-key-chain {
       description "Key chains as specified in
                    draft-ietf-rtgwg-yang-key-chain;";
     }

     container logical-network-element {
       description
         "This module is used to support multiple logical network
          elements on a single physical or virtual system.";
     }

     container network-instance {
       description
         "This module is used to support multiple network instances
          within a single physical or virtual device.  Network
          instances are commonly know as VRFs (virtual routing
          and forwarding) and VSIs (virtual switching instances).";
     }
     // rpc statements

     // notification statements

   }
   <CODE ENDS>
```

6. References

6.1. Normative References

[I-D.ietf-rtgwg-lne-model]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "YANG Logical Network Elements", draft-ietf-rtgwg-lne-model-01 (work in progress), October 2016.

[I-D.ietf-rtgwg-ni-model]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "YANG Network Instances", draft-ietf-rtgwg-ni-model-01 (work in progress), October 2016.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <http://www.rfc-editor.org/info/rfc4026>.

[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>.

[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>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

6.2. Informative References

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-09 (work in progress), October 2016.

[I-D.ietf-netmod-opstate-reqs]
Watsen, K. and T. Nadeau, "Terminology and Requirements for Enhanced Handling of Operational State", draft-ietf-netmod-opstate-reqs-04 (work in progress), January 2016.

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-25 (work in progress), November 2016.

[I-D.ietf-rtgwg-yang-key-chain]
Lindem, A., Qu, Y., Yeung, D., Chen, I., Zhang, Z., and Y. Yang, "Routing Key Chain YANG Data Model", draft-ietf-rtgwg-yang-key-chain-15 (work in progress), February 2017.

[I-D.openconfig-mpls-consolidated-model]
George, J., Fang, L., eric.osborne@level3.com, e., and R. Shakir, "MPLS / TE Model for Service Provider Networks", draft-openconfig-mpls-consolidated-model-02 (work in progress), October 2015.

[I-D.openconfig-netmod-model-structure]
Shaikh, A., Shakir, R., D'Souza, K., and L. Fang, "Operational Structure and Organization of YANG Models", draft-openconfig-netmod-model-structure-00 (work in progress), March 2015.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

[IEEE-8021Q]
Holness, M., "IEEE 802.1Q YANG Module Specifications", IEEE-Draft http://www.ieee802.org/1/files/public/docs2015/new-mholness-yang-8021Q-0515-v04.pdf, May 2015.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <http://www.rfc-editor.org/info/rfc7895>.

Appendix A. Acknowledgments

This document is derived from draft-openconfig-netmod-model-structure-00. We thank the Authors of that document and acknowledge their indirect contribution to this work. The authors include: Anees Shaikh, Rob Shakir, Kevin D'Souza, Luyuan Fang, Qin Wu, Stephane Litkowski and Gang Yan.

This work was discussed in and produced by the Routing Area Yang Architecture design team. Members at the time of writing included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Lou Berger, Qin Wu, Rob Shakir, Stephane Litkowski, and Gang Yan.

The identityref approach was proposed by Mahesh Jethanandani.

The RFC text was produced using Marshall Rose's xml2rfc tool.

Authors' Addresses

Acee Lindem (editor)
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Lou Berger (editor)
LabN Consulting, L.L.C.

Email: lberger@labn.net

Dean Bogdanovic

Email: ivandean@gmail.com

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

RTGWG
Internet-Draft
Intended status: Informational
Expires: May 4, 2017

E. Nordmark (ed)
Arista Networks
A. Tian
Ericsson Inc.
J. Gross
VMware
J. Hudson
Brocade Communications Systems, Inc.
L. Kreeger
Cisco Systems, Inc.
P. Garg
Microsoft
P. Thaler
Broadcom Corporation
T. Herbert
Facebook
October 31, 2016

Encapsulation Considerations
draft-ietf-rtgwg-dt-encap-02

Abstract

The IETF Routing Area director has chartered a design team to look at common issues for the different data plane encapsulations being discussed in the NVO3 and SFC working groups and also in the BIER BoF, and also to look at the relationship between such encapsulations in the case that they might be used at the same time. The purpose of this design team is to discover, discuss and document considerations across the different encapsulations in the different WGs/BoFs so that we can reduce the number of wheels that need to be reinvented in the future.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 4, 2017.

Table of Contents

1. Design Team Charter

There have been multiple efforts over the years that have resulted in new or modified data plane behaviors involving encapsulations. That includes IETF efforts like MPLS, LISP, and TRILL but also industry efforts like VXLAN and NVGRE. These collectively can be seen as a source of insight into the properties that data planes need to meet. The IETF is currently working on potentially new encapsulations in NVO3 and SFC and considering working on BIER. In addition there is work on tunneling in the INT area.

This is a short term design team chartered to collect and construct useful advice to parties working on new or modified data plane behaviors that include additional encapsulations. The goal is for the group to document useful advice gathered from interacting with ongoing efforts. An Internet Draft will be produced for IETF92 to capture that advice, which will be discussed in RTGWG.

Data plane encapsulations face a set of common issues such as:

- How to provide entropy for ECMP
- Issues around packet size and fragmentation/reassembly
- OAM - what support is needed in an encapsulation format?
- Security and privacy.
- QoS
- Congestion Considerations
- IPv6 header protection (zero UDP checksum over IPv6 issue)
- Extensibility - e.g., for evolving OAM, security, and/or congestion control
- Layering of multiple encapsulations e.g., SFC over NVO3 over BIER

The design team will provide advice on those issues. The intention is that even where we have different encapsulations for different purposes carrying different information, each such encapsulation doesn't have to reinvent the wheel for the above common issues.

The design team will look across the routing area in particular at SFC, NVO3 and BIER. It will not be involved in comparing or analyzing any particular encapsulation formats proposed in those WGs and BoFs but instead focus on common advice.

2. Overview

The references provide background information on NVO3, SFC, and BIER. In particular, NVO3 is introduced in [RFC7364], [RFC7365], and [I-D.ietf-nvo3-arch]. SFC is introduced in [I-D.ietf-sfc-architecture] and [I-D.ietf-sfc-problem-statement]. Finally, the information on BIER is in [I-D.shepherd-bier-problem-statement], [I-D.wijnands-bier-architecture], and [I-D.wijnands-mpls-bier-encapsulation]. We assume the reader has some basic familiarity with those proposed encapsulations. The Related Work section points at some prior work that relates to the encapsulation considerations in this document.

Encapsulation protocols typically have some unique information that they need to carry. In some cases that information might be modified along the path and in other cases it is constant. The in-flight modifications has impacts on what it means to provide security for the encapsulation headers.

- NVO3 carries a VNI Identifier edge to edge which is not modified. There has been OAM discussions in the WG and it isn't clear whether some of the OAM information might be modified in flight.
- SFC carries Service Function Path identification and service meta-data. The meta-data might be modified as the packets follow the service path. SFC talks of some loop avoidance mechanism which is likely to result in modifications for for each hop in the service chain even if the meta-data is unmodified.
- BIER carries a bitmap of egress ports to which a packet should be delivered, and as the packet is forwarded down different paths different bits are cleared in that bitmap.

Even if information isn't modified in flight there might be devices that wish to inspect that information. For instance, one can envision future NVO3 security devices which filter based on the virtual network identifier.

The need for extensibility is different across the protocols

- NVO3 might need some extensions for OAM and security.
- SFC consists of Service Function Path identification plus carrying service meta-data along a path, and different services might need different types and amount of meta-data.
- BIER might need variable number of bits in their bitmaps, or other future schemes to scale up to larger network.

The extensibility needs and constraints might be different when considering hardware vs. software implementations of the

encapsulation headers. NIC hardware might have different constraints than switch hardware.

As the IETF designs these encapsulations the different WGs solve the issues for their own encapsulation. But there are likely to be future cases when the different encapsulations are combined in the same header. For instance, NVO3 might be a "transport" used to carry SFC between the different hops in the service chain.

Most of the issues discussed in this document are not new. The IETF and industry as specified and deployed many different encapsulation or tunneling protocols over time, ranging from simple IP-in-IP and GRE encapsulation, IPsec, pseudo-wires, session-based approached like L2TP, and the use of MPLS control and data planes. IEEE 802 has also defined layered encapsulation for Provider Backbone Bridges (PBB) and IEEE 802.1Qbp (ECMP). This document tries to leverage what we collectively have learned from that experience and summarize what would be relevant for new encapsulations like NVO3, SFC, and BIER.

3. Common Issues

[This section is mostly a repeat of the charter but with a few modifications and additions.]

Any new encapsulation protocol would need to address a large set of issues that are not central to the new information that this protocol intends to carry. The common issues explored in this document are:

- How to provide entropy for Equal Cost MultiPath (ECMP) routing
- Issues around packet size and fragmentation/reassembly
- Next header indication - each encapsulation might be able to carry different payloads
- OAM - what support is needed in an encapsulation format?
- Security and privacy
- QoS
- Congestion Considerations
- Header protection
- Extensibility - e.g., for evolving OAM, security, and/or congestion control
- Layering of multiple encapsulations e.g., SFC over NVO3 over BIER
- Importance of being friendly to hardware and software implementations

The degree to which these common issues apply to a particular encapsulation can differ based on the intended purpose of the encapsulation. But it is useful to understand all of them before determining which ones apply.

4. Scope

It is important to keep in mind what we are trying to cover and not cover in this document and effort. This is

- A look across the three new encapsulations, while taking lots of previous work into account
- Focus on the class of encapsulations that would run over IP/UDP. That was done to avoid being distracted by the data-plane and control-plane interaction, which is more significant for protocols that are designed to run over "transports" that maintain session or path state.
- We later expanded the scope somewhat to consider how the encapsulations would play with MPLS "transport", which is important because SFC and BIER seem to target being independent of the underlying "transport"

However, this document and effort is NOT intended to:

- Design some new encapsulation header to rule them all
- Design yet another new NVO3 encapsulation header
- Try to select the best encapsulation header
- Evaluate any existing and proposed encapsulations

While the origin and focus of this document is the routing area and in particular NVO3, SFC, and BIER, the considerations apply to other encapsulations that are being defined in the IETF and elsewhere. There seems to be an increase in the number of encapsulations being defined to run over UDP, where there might already exist an encapsulation over IP or Ethernet. Feedback on how these considerations apply in those contexts is welcome.

5. Assumptions

The design center for the new encapsulations is a well-managed network. That network can be a datacenter network (plus datacenter interconnect) or a service provider network. Based on the existing and proposed encapsulations in those environment it is reasonable to make these assumptions:

- The MTU is carefully managed and configured. Hence an encapsulation protocol can make the packets bigger without resulting in a requirement for fragmentation and reassembly between ingress and egress. (However, it might be useful to detecting MTU misconfigurations.)
- In general an encapsulation needs some approach for congestion management. But the assumptions are different than for arbitrary Internet paths in that the underlay might be well-provisioned and

better policed at the edge, and due to multi-tenancy, the congestion control in the endpoints might be even less trusted than on the Internet at large.

The goal is to implement these encapsulations in hardware and software hence we can't assume that the needs of either implementation approach can trump the needs of the other. In particular, around extensibility the needs and constraints might be quite different.

6. Terminology

The capitalized keyword MUST is used as defined in http://en.wikipedia.org/wiki/Julmust

TBD: Refer to existing documents for at least NVO3 and SFC terminology. We use at least the VNI ID in this document.

7. Entropy

In many cases the encapsulation format needs to enable ECMP in unmodified routers. Those routers might use different fields in TCP/UDP packets to do ECMP without a risk of reordering a flow. Note that the same entropy might also be used at layer 2 e.g. for Link Aggregation (LAG).

The common way to do ECMP-enabled encapsulation over IP today is to add a UDP header and to use UDP with the UDP source port carrying entropy from the inner/original packet headers as in LISP [RFC6830]. The total entropy consists of 14 bits in the UDP source port (using the ephemeral port range) plus the outer IP addresses which seems to be sufficient for entropy; using outer IPv6 headers would give the option for more entropy should it be needed in the future.

In some environments it might be fine to use all 16 bits of the port range. However, middleboxes might make assumptions about the system ports or user ports. But they should not make any assumptions about the ports in the Dynamic and/or Private Port range, which have the two MSBs set to 11b.

The UDP source port might change over the lifetime of an encapsulated flow, for instance for DoS mitigation or re-balancing load across ECMP. Such changes need to consider reordering if there are packets in flight for the flow.

There is some interaction between entropy and OAM and extensibility mechanism. It is desirable to be able to send OAM packets to follow the same path as network packets. Hence OAM packets should use the

same entropy mechanism as data packets. While routers might use information in addition the entropy field and outer IP header, they can not use arbitrary parts of the encapsulation header since that might result in OAM frames taking a different path. Likewise if routers look past the encapsulation header they need to be aware of the extensibility mechanism(s) in the encapsulation format to be able to find the inner headers in the presence of extensions; OAM frames might use some extensions e.g. for timestamps.

Architecturally the entropy and the next header field are really part of enclosing delivery header. UDP with entropy goes hand-in-hand with the outer IP header. Thus the UDP entropy is present for the underlay IP routers the same way that an MPLS entropy label is present for LSRs. The entropy above is all about providing entropy for the outer delivery of the encapsulated packets.

It has been suggested that when IPv6 is used it would not be necessary to add a UDP header for entropy, since the IPv6 flow label can be used for entropy. (This assumes that there is an IP protocol number for the encapsulation in addition to a UDP destination port number since UDP would be used with IPv4 underlay. And any use of UDP checksums would need to be replaced by an encaps-specific checksum or secure hash.) While such an approach would save 8 bytes of headers when the underlay is IPv6, it does assume that the underlay routers use the flow label for ECMP, and it also would make the IPv6 approach different than the IPv4 approach. Currently the leaning is towards recommending using the UDP encapsulation for both IPv4 and IPv6 underlay. The IPv6 flow label can be used for additional entropy if need be. There is more detailed discussion for using the IPv6 flow label for tunnels in [RFC6438].

Note that in the proposed BIER encapsulation [I-D.wijnands-mpls-bier-encapsulation], there is an an 8-bit field which specifies an entropy value that can be used for load balancing purposes. This entropy is for the BIER forwarding decisions, which is independent of any outer delivery ECMP between BIER routers. Thus it is not part of the delivery ECMP discussed in this section.

> [Note: For any given bit in BIER (that identifies an exit from the BIER domain) there might be multiple immediate next hops. The BIER entropy field is used to select that next hop as part of BIER processing. The BIER forwarding process may do equal cost load balancing, but the load balancing procedure MUST choose the same path for any two packets that have the same entropy value.]

In summary:

- o The entropy is associated with the transport, that is an outer IP header or MPLS.
- o In the case of IP transport use 14 or 16 bits of UDP source port, plus outer IPv6 flowid for entropy.

8. Next-protocol indication

Next-protocol indications appear in three different contexts for encapsulations.

Firstly, the transport delivery mechanism for the encapsulations we discuss in this document need some way to indicate which encapsulation header (or other payload) comes next in the packet. Some encapsulations might be identified by a UDP port; others might be identified by an Ethernet type or IP protocol number. Which approach is used is a function of the preceding header the same way as IPv4 is identified by both an Ethernet type and an IP protocol number (for IP-in-IP). In some cases the header type is implicit in some session (L2TP) or path (MPLS) setup. But this is largely beyond the control of the encapsulation protocol. For instance, if there is a requirement to carry the encapsulation after an Ethernet header, then an Ethernet type is needed. If required to be carried after an IP/UDP header, then a UDP port number is needed. For UDP port numbers there are considerations for port number conservation described in [I-D.ietf-tsvwg-port-use].

It is worth mentioning that in the MPLS case of no implicit protocol type many forwarding devices peek at the first nibble of the payload to determine whether to apply IPv4 or IPv6 L3/L4 hashes for load balancing [RFC7325]. That behavior places some constraints on other payloads carried over MPLS and some protocol define an initial control word in the payload with a value of zero in its first nibble [RFC4385] to avoid confusion with IPv4 and IPv6 payload headers.

Secondly, the encapsulation needs to indicate the type of its payload, which is in scope for the design of the encapsulation. We have existing protocols which use Ethernet types (such as GRE). Here each encapsulation header can potentially makes its own choices between:

- o Use the Ethernet type space - makes it easy to carry existing L2 and L3 protocols including IPv4, IPv6, and Ethernet. Disadvantages are that it is a 16 bit number and we probably need far less than 100 values, and the number space is controlled by the IEEE 802 RAC with its own allocation policies.
- o Use the IP protocol number space - makes it easy to carry e.g., ESP in addition to IP and Ethernet but brings in all existing protocol numbers many of which would never be used directly on top

of the encapsulation protocol. IANA managed eight bit values, presumably more difficult to get an assigned number than to get a transport port assignment.

- Define their own next-protocol number space, which can use fewer bits than an Ethernet type and give more flexibility, but at the cost of administering that numbering space (presumably by the IANA).

Thirdly, if the IETF ends up defining multiple encapsulations at about the same time, and there is some chance that multiple such encapsulations can be combined in the same packet, there is a question whether it makes sense to use a common approach and numbering space for the encapsulation across the different protocols. A common approach might not be beneficial as long as there is only one way to indicate e.g., SFC inside NVO3.

Many Internet protocols use fixed values (typically managed by the IANA function) for their next-protocol field. That facilitates interpretation of packets by middleboxes and e.g., for debugging purposes, but might make the protocol evolution inflexible. Our collective experience with MPLS shows an alternative where the label can be viewed as an index to a table containing processing instructions and the table content can be managed in different ways. Encapsulations might want to consider the tradeoffs between such more flexible versus more fixed approaches.

In summary:

- Would it be useful for the IETF come up with a common scheme for encapsulation protocols? If not each encapsulation can define its own scheme.

9. MTU and Fragmentation

A common approach today is to assume that the underlay have sufficient MTU to carry the encapsulated packets without any fragmentation and reassembly at the tunnel endpoints. That is sufficient when the operator of the ingress and egress have full control of the paths between those endpoints. And it makes for simpler (hardware) implementations if fragmentation and reassembly can be avoided.

However, even under that assumption it would be beneficial to be able to detect when there is some misconfiguration causing packets to be dropped due to MTU issues. One way to do this is to have the encapsulator set the don’t-fragment (DF) flag in the outer IPv4 header and receive and log any received ICMP "packet too big" (PTB)

errors. Note that no flag needs to be set in an outer IPv6 header [RFC2460].

Encapsulations could also define an optional tunnel fragmentation and reassembly mechanism which would be useful in the case when the operator doesn't have full control of the path, or when the protocol gets deployed outside of its original intended context. Such a mechanism would be required if the underlay might have a path MTU which makes it impossible to carry at least 1518 bytes (if offering Ethernet service), or at least 1280 (if offering IPv6 service). The use of such a protocol mechanism could be triggered by receiving a PTB. But such a mechanism might not be implemented by all encapsulators and decapsulators. [Aerolink is one example of such a protocol.]

Depending on the payload carried by the encapsulation there are some additional possibilities:

- If payload is IPv4/6 then the underlay path MTU could be used to report end-to-end path MTU.
- If the payload service is Ethernet/L2, then there is no such per destination reporting mechanism. However, there is a LLDP TLV for reporting max frame size; might be useful to report minimum to end stations, but unmodified end stations would do nothing with that TLV since they assume that the MTU is at least 1518.

In summary:

- In some deployments an encapsulation can assume well-managed MTU hence no need for fragmentation and reassembly related to the encapsulation.
- Even so, it makes sense for ingress to track any ICMP packet too big addressed to ingress to be able to log any MTU misconfigurations.
- Should an encapsulation protocol be deployed outside of the original context it might very well need support for fragmentation and reassembly.

10. OAM

The OAM area is seeing active development in the IETF with discussions (at least) in NVO3 and SFC working groups, plus the new LIME WG looking at architecture and YANG models.

The design team has take a narrow view of OAM to explore the potential OAM implications on the encapsulation format.

In terms of what we have heard from the various working groups there seem to be needs to:

- Be able to send out-of-band OAM messages - that potentially should follow the same path through the network as some flow of data packets.
  - Such OAM messages should not accidentally be decapsulated and forwarded to the end stations.
- Be able to add OAM information to data packets that are encapsulated. Discussions have been around:
  - Using a bit in the OAM to synchronize sampling of counters between the encapsulator and decapsulator.
  - Optional timestamps, sequence numbers, etc for more detailed measurements between encapsulator and decapsulator.
- Usable for both proactive monitoring (akin to BFD) and reactive checks (akin to traceroute to pin-point a failure)

To ensure that the OAM messages can follow the same path the OAM messages need to get the same ECMP (and LAG hashing) results as a given data flow. An encapsulator can choose between one of:

- Limit ECMP hashing to not look past the UDP header i.e. the entropy needs to be in the source/destination IP and UDP ports
- Make OAM packets look the same as data packets i.e. the initial part of the OAM payload has the inner Ethernet, IP, TCP/UDP headers as a payload. (This approach was taken in TRILL out of necessity since there is no UDP header.) Any OAM bit in the encapsulation header must in any case be excluded from the entropy.

There can be several ways to prevent OAM packets from accidentally being forwarded to the end station using:

- A bit in the frame (as in TRILL) indicating OAM
- A next-protocol indication with a designated value for "none" or "oam".

This assumes that the bit or next protocol, respectively, would not affect entropy/ECMP in the underlay. However, the next-protocol field might be used to provide differentiated treatment of packets based on their payload; for instance a TCP vs. IPsec ESP payload might be handled differently. Based on that observation it might be undesirable to overload the next protocol with the OAM drop behavior, resulting in a preference for having a bit to indicate that the packet should be forwarded to the end station after decapsulation.

There has been suggestions that one (or more) marker bits in the encaps header would be useful in order to delineate measurement epochs on the encapsulator and decapsulator and use that to compare counters to determine packet loss.

A result of the above is that OAM is likely to evolve and needs some degree of extensibility from the encapsulation format; a bit or two plus the ability to define additional larger extensions.

An open question is how to handle error messages or other reports relating to OAM. One can think if such reporting as being associated with the encapsulation the same way ICMP is associated with IP. Would it make sense for the IETF to develop a common Encapsulation Error Reporting Protocol as part of OAM, which can be used for different encapsulations? And if so, what are the technical challenges. For instance, how to avoid it being filtered as ICMP often is?

A potential additional consideration for OAM is the possible future existence of gateways that "stitch" together different dataplane encapsulations and might want to carry OAM end-to-end across the different encapsulations.

In summary:

- It makes sense to reserve a bit for "drop after decapsulation" for OAM out-of-band.
- An encapsulation needs sufficient extensibility for OAM (such as bits, timestamps, sequence numbers). That might be motivated by in-band OAM but it would make sense to leverage the same extensions for out-of band OAM.
- OAM places some constraints on use of entropy in forwarding devices.
- Should IETF look into error reporting that is independent of the specific encapsulation?

11. Security Considerations

Different encapsulation use cases will have different requirements around security. For instance, when encapsulation is used to build overlay networks for network virtualization, isolation between virtual networks may be paramount. BIER support of multicast may entail different security requirements than encapsulation for unicast.

In real deployment, the security of the underlying network may be considered for determining the level of security needed in the encapsulation layer. However for the purposes of this discussion, we

assume that network security is out of scope and that the underlying network does not itself provide adequate or as least uniform security mechanisms for encapsulation.

There are at least three considerations for security:

- Anti-spoofing/virtual network isolation
- Interaction with packet level security such as IPsec or DTLS
- Privacy (e.g., VNI ID confidentially for NVO3)

This section uses a VNI ID in NVO3 as an example. A SFC or BIER encapsulation is likely to have fields with similar security and privacy requirements.

11.1. Encapsulation-specific considerations

Some of these considerations appear for a new encapsulation, and others are more specific to network virtualization in datacenters.

- New attack vectors:
  - DDOS on specific queued/paths by attempting to reproduce the 5-tuple hash for targeted connections.
  - Entropy in outer 5-tuple may be too little or predictable.
  - Leakage of identifying information in the encapsulation header for an encrypted payload.
  - Vulnerabilities of using global values in fields like VNI ID.
- Trusted versus untrusted tenants in network virtualization:
  - The criticality of virtual network isolation depends on whether tenants are trusted or untrusted. In the most extreme cases, tenants might not only be untrusted but may be considered hostile.
  - For a trusted set of users (e.g. a private cloud) it may be sufficient to have just a virtual network identifier to provide isolation. Packets inadvertently crossing virtual networks should be dropped similar to a TCP packet with a corrupted port being received on the wrong connection.
  - In the presence of untrusted users (e.g. a public cloud) the virtual network identifier must be adequately protected against corruption and verified for integrity. This case may warrant keyed integrity.
- Different forms of isolation:
  - Isolation could be blocking all traffic between tenants (or except as allowed by some firewall)
  - Could also be about performance isolation i.e. one tenant can overload the network in a way that affects other tenants

    * Physical isolation of traffic for different tenants in network may be required, as well as required restrictions that tenants may have on where their packets may be routed.
- New attack vectors from untrusted tenants:
    * Third party VMs with untrusted tenants allows internally borne attacks within data centers
    * Hostile VMs inside the system may exist (e.g. public cloud)
    * Internally launched DDOS
    * Passive snooping for mis-delivered packets
    * Mitigate damage and detection in event that a VM is able to circumvent isolation mechanisms
- Tenant-provider relationship:
    * Tenant might not trust provider, hypervisors, network
    * Provider likely will need to provide SLA or a least a statement on security
    * Tenant may implement their own additional layers of security
    * Regulation and certification considerations
- Trend towards tighter security:
    * Tenants' data in network increases in volume and value, attacks become more sophisticated
    * Large DCs already encrypt everything on disk
    * DCs likely to encrypt inter-DC traffic at this point, use TLS to Internet.
    * Encryption within DC is becoming more commonplace, becomes ubiquitous when cost is low enough.
    * Cost/performance considerations. Cost of support for strong security has made strong network security in DCs prohibitive.
    * Are there lessons from MacSec?

11.2. Virtual network isolation

The first requirement is isolation between virtual networks. Packets sent in one virtual network should never be illegitimately received by a node in another virtual network. Isolation should be protected in the presence of malicious attacks or inadvertent packet corruption.

The second requirement is sender authentication. Sender identity is authenticated to prevent anti-spoofing. Even if an attacker has access to the packets in the network, they cannot send packets into a virtual network. This may have two possibilities:

- Pairwise sender authentication. Any two communicating hosts negotiate a shared key.

- o Group authentication. A group of hosts share a key (this may be more appropriate for multicast of encapsulation).

Possible security solutions:

- o Security cookie: This is similar to L2TP cookie mechanism [RFC3931]. A shared plain text cookie is shared between encapsulator and decapsulator. A receiver validates a packet by evaluating if the cookie is correct for the virtual network and address of a sender. Validation function is F(cookie, VNI ID, source address). If cookie matches, accept packet, else drop. Since cookie is plain text this method does not protect against an eavesdropping. Cookies are set and may be rotated out of band.
- o Secure hash: This is a stronger mechanism than simple cookies that borrows from IPsec and PPP authentication methods. In this model security field contains a secure hash of some fields in the packet using a shared key. Hash function may be something like H(key, VNI ID, address, salt). The salt ensures the hash is not the same for every packet, and if it includes a sequence number may also protect against replay attacks.

In any use of a shared key, periodic re-keying should be allowed. This could include use of techniques like generation numbers, key windows, etc. See [I-D.farrelll-mpls-opportunistic-encrypt] for an example application.

We might see firewalls that are aware of the encapsulation and can provide some defense in depth combined with the above example anti-spoofing approaches. An example would be an NVO3-aware firewall being able to check the VNI ID.

Separately and in addition to such filtering, there might be a desire to completely block an encapsulation protocol at certain places in the network, e.g., at the edge of a datacenter. Using a fixed standard UDP destination port number for each encapsulation protocol would facilitate such blocking.

11.3. Packet level security

An encapsulated packet may itself be encapsulated in IPsec (e.g. ESP). This should be straightforward and in fact is what would happen today in security gateways. In this case, there is no special consideration for the fact that packet is encapsulated, however since the encapsulation layer headers are included (part of encrypted data for instance) we lose visibility in the network of the encapsulation.

The more interesting case is when security is applied to the encapsulation payload. This will keep the encapsulation headers in

the outer header visible to the network (for instance in nvo3 we may way to firewall based on VNI ID even if the payload is encrypted). One possibility is to apply DTLS to the encapsulation payload. In this model the protocol stack may be something like IP|UDP|Encap|DTLS|encrypted_payload. The encapsulation and security should be done together at an encapsulator and resolved at the decapsulator. Since the encapsulation header is outside of the security coverage, this may itself require security (like described above).

In both of the above the security associations (SAs) may be between physical hosts, so for instance in nvo3 we can have packets of different virtual networks using the same SA-- this should not be an issue since it is the VNI ID that ensures isolation (which needs to be secured also).

11.4. In summary:

- Encapsulations need extensibility mechanisms to be able to add security features like cookies and secure hashes protecting the encapsulation header.
- NVO3 probably has specific higher requirements relating to isolation for network virtualization, which is in scope for the NVO3 WG.
- Our collective IETF experience is that successful protocols get deployed outside of the original intended context, hence the initial assumptions about the threat model might become invalid. That needs to be considered in the standardization of new encapsulations.

12. QoS

In the Internet architecture we support QoS using the Differentiated Services Code Points (DSCP) in the formerly named Type-of-Service field in the IPv4 header, and in the Traffic-Class field in the IPv6 header. The ToS and TC fields also contain the two ECN bits, which are discussed in Section 13.

We have existing specifications how to process those bits. See [RFC2983] for diffserv handling, which specifies how the received DSCP value is used to set the DSCP value in an outer IP header when encapsulating. (There are also existing specifications how DSCP can be mapped to layer2 priorities.)

Those specifications apply whether or not there is some intervening headers (e.g., for NVO3 or SFC) between the inner and outer IP headers. Thus the encapsulation considerations in this area are mainly about applying the framework in [RFC2983].

Note that the DSCP and ECN bits are not the only part of an inner packet that might potentially affect the outer packet. For example, [RFC2473] specifies handling of inner IPv6 hop-by-hop options that effectively result in copying some options to the outer header. It is simpler to not have future encapsulations depend on such copying behavior.

There are some other considerations specific to doing OAM for encapsulations. If OAM messages are used to measure latency, it would make sense to treat them the same as data payloads. Thus they need to have the same outer DSCP value as the data packets which they wish to measure.

Due to OAM there are constraints on middleboxes in general. If middleboxes inspect the packet past the outer IP+UDP and encapsulation header and look for inner IP and TCP/UDP headers, that might violate the assumption that OAM packets will be handled the same as regular data packets. That issue is broader than just QoS - applies to firewall filters etc.

In summary:

- Leverage the existing approach in [RFC2983] for DSCP handling.

13. Congestion Considerations

Additional encapsulation headers does not introduce anything new for Explicit Congestion Notification. It is just like IP-in-IP and IPsec tunnels which is specified in [RFC6040] in terms of how the ECN bits in the inner and outer header are handled when encapsulating and decapsulating packets. Thus new encapsulations can more or less include that by reference.

There are additional considerations around carrying non-congestion controlled traffic. These details have been worked out in [I-D.ietf-mpls-in-udp]. As specified in [RFC5405]: "IP-based traffic is generally assumed to be congestion-controlled, i.e., it is assumed that the transport protocols generating IP-based traffic at the sender already employ mechanisms that are sufficient to address congestion on the path. Consequently, a tunnel carrying IP-based traffic should already interact appropriately with other traffic sharing the path, and specific congestion control mechanisms for the tunnel are not necessary". Those considerations are being captured in [I-D.ietf-tsvwg-rfc5405bis].

For this reason, where an encapsulation method is used to carry IP traffic that is known to be congestion controlled, the UDP tunnels does not create an additional need for congestion control. Internet

IP traffic is generally assumed to be congestion-controlled. Similarly, in general Layer 3 VPNs are carrying IP traffic that is similarly assumed to be congestion controlled.

However, some of the encapsulations (at least NVO3) will be able to carry arbitrary Layer 2 packets to provide an L2 service, in which case one can not assume that the traffic is congestion controlled.

One could handle this by adding some congestion control support to the encapsulation header (one instance of which would end up looking like DCCP). However, if the underlay is well-provisioned and managed as opposed to being arbitrary Internet path, it might be sufficient to have a slower reaction to congestion induced by that traffic. There is work underway on a notion of "circuit breakers" for this purpose. See See [I-D.ietf-tsvwg-circuit-breaker]. Encapsulations which carry arbitrary Layer 2 packets want to consider that ongoing work.

If the underlay is provisioned in such a way that it can guarantee sufficient capacity for non-congestion controlled Layer 2 traffic, then such circuit breakers might not be needed.

Two other considerations appear in the context of these encapsulations as applied to overlay networks:

- Protect against malicious end stations
- Ensure fairness and/or measure resource usage across multiple tenants

Those issues are really orthogonal to the encapsulation, in that they are present even when no new encapsulation header is in use. However, the application of the new encapsulations are likely to be in environments where those issues are becoming more important. Hence it makes sense to consider them.

One could make the encapsulation header be extensible to that it can carry sufficient information to be able to measure resource usage, delays, and congestion. The suggestions in the OAM section about a single bit for counter synchronization, and optional timestamps and/or sequence numbers, could be part of such an approach. There might also be additional congestion-control extensions to be carried in the encapsulation. Overall this results in a consideration to support sufficient extensibility in the encapsulation to handle potential future developments in this space.

Coarse measurements are likely to suffice, at least for circuit-breaker-like purposes, see [I-D.wei-tsvwg-tunnel-congestion-feedback] and [I-D.briscoe-conex-data-centre] for examples on active work in

this area via use of ECN. [RFC6040] Appendix C is also relevant. The outer ECN bits seem sufficient (at least when everything uses ECN) to do this course measurements. Needs some more study for the case when there are also drops; might need to exchange counters between ingress and egress to handle drops.

Circuit breakers are not sufficient to make a network with different congestion control when the goal is to provide a predictable service to different tenants. The fallback would be to rate limit different traffic.

In summary:

- Leverage the existing approach in [RFC6040] for ECN handling.
- If the encapsulation can carry non-IP, hence non-congestion controlled traffic, then leverage the approach in [I-D.ietf-mpls-in-udp].
- "Watch this space" for circuit breakers.

14. Header Protection

Many UDP based encapsulations such as VXLAN [RFC7348] either discourage or explicitly disallow the use of UDP checksums. The reason is that the UDP checksum covers the entire payload of the packet and switching ASICs are typically optimized to look at only a small set of headers as the packet passes through the switch. In these case, computing a checksum over the packet is very expensive. (Software endpoints and the NICs used with them generally do not have the same issue as they need to look at the entire packet anyways.)

The lack a header checksum creates the possibility that bit errors can be introduced into any information carried by the new headers. Specifically, in the case of IPv6, the assumption is that a transport layer checksum - UDP in this case - will protect the IP addresses through the inclusion of a pseudo-header in the calculation. This is different from IPv4 on which many of these encapsulation protocols are initially deployed which contains its own header checksum. In addition to IP addresses, the encapsulation header often contains its own information which is used for addressing packets or other high value network functions. Without a checksum, this information is potentially vulnerable - an issue regardless of whether the packet is carried over IPv4 or IPv6.

Several protocols cite [RFC6935] and [RFC6936] as an exemption to the IPv6 checksum requirements. However, these are intended to be tailored to a fairly narrow set of circumstances - primarily relying on sparseness of the address space to detect invalid values and well managed networks - and are not a one size fits all solution. In

these cases, an analysis should be performed of the intended environment, including the probability of errors being introduced and the use of ECC memory in routing equipment.

Conceptually, the ideal solution to this problem is a checksum that covers only the newly added headers of interest. There is little value in the portion of the UDP checksum that covers the encapsulated packet because that would generally be protected by other checksums and this is the expensive portion to compute. In fact, this solution already exists in the form of UDP-Lite and UDP based encapsulations could be easily ported to run on top of it. Unfortunately, the main value in using UDP as part of the encapsulation header is that it is recognized by already deployed equipment for the purposes of ECMP, RSS, and middlebox operations. As UDP-Lite uses a different protocol number than UDP and it is not widely implemented in middleboxes, this value is lost. A possible solution is to incorporate the same partial-checksum concept as UDP-Lite or other header checksum protection into the encapsulation header and continue using UDP as the outer protocol. One potential challenge with this approach is the use of NAT or other form of translation on the outer header will result in an invalid checksum as the translator will not know to update the encapsulation header.

The method chosen to protect headers is often related to the security needs of the encapsulation mechanism. On one hand, the impact of a poorly protected header is not limited to only data corruption but can also introduce a security vulnerability in the form of misdirected packets to an unauthorized recipient. Conversely, high security protocols that already include a secure hash over the valuable portion of the header (such as by encrypting the entire IP packet using IPsec, or some secure hash of the encap header) do not require additional checksum protection as the hash provides stronger assurance than a simple checksum.

If the sender has included a checksum, then the receiver should verify that checksum or, if incapable, drop the packet. The assumption is that configuration and/or control-plane capability exchanges can be used when different receiver have different checksum validation capabilities.

In summary:

- Encapsulations need extensibility to be able to add checksum/CRC for the encapsulation header itself.
- When the encapsulation has a checksum/CRC, include the IPv6 pseudo-header in it.
- The checksum/CRC can potentially be avoided when cryptographic protection is applied to the encapsulation.

15. Extensibility Considerations

Protocol extensibility is the concept that a networking protocol may be extended to include new use cases or functionality that were not part of the original protocol specification. Extensibility may be used to add security, control, management, or performance features to a protocol. A solution may allow private extensions for customization or experimentation.

Extending a protocol often implies that a protocol header must carry new information. There are two usual methods to accomplish this:

1. Define or redefine the meaning of existing fields in a protocol header.
2. Add new (optional) fields to the protocol header.

It is also possible to create a new protocol version, but this is more associated with defining a protocol than extending it (IPv6 being a successor to IPv4 is an example of protocol versioning).

In some cases it might be more appropriate to define a new inner protocol which can carry the new functionality instead of extending the outer protocol. Examples where this works well is in the IP/ transport split, where the earlier architecture had a single NCP [RFC0033] protocol which carried both the hop-by-hop semantics which are now in IP, and the end-to-end semantics which are now in TCP. Such a split is effective when different nodes need to act upon the different information. Applying this for general protocol extensibility through nesting is not well understood, and does result in longer header chains. Furthermore, our experience with IPv6 extension headers [RFC2460] in middleboxes indicates that the header chaining approach does not help with middlebox traversal.

Many protocol definitions include some number of reserved fields or bits which can be used for future extension. VXLAN is an example of a protocol that includes reserved bits which are subsequently being allocated for new purposes. Another technique employed is to re-purpose existing header fields with new meanings. A classic example of this is the definition of DSCP code point which redefines the ToS field originally specified in IPv4. When a field is redefined, some mechanism may be needed to ensure that all interested parties agree on the meaning of the field. The techniques of defining meaning for reserved bits or redefining existing fields have the advantage that a protocol header can be kept a fixed length. The disadvantage is that the extensibility is limited. For instance, the number reserved bits in a fixed protocol header is limited. For standard protocols the decision to commit to a definition for a field can be wrenching since it is difficult to retract later. Also, it is difficult to predict a

priori how many reserved fields or bits to put into a protocol header to satisfy the extensions create over the lifetime of the protocol.

Extending a protocol header with new fields can be done in several ways.

- o TLVs are a very popular method used in such protocols as IP and TCP. Depending on the type field size and structure, TLVs can offer a virtually unlimited range of extensions. A disadvantage of TLVs is that processing them can be verbose, quite complicated, several validations must often be done for each TLV, and there is no deterministic ordering for a list of TLVs. TCP serves as an example of a protocol where TLVs have been successfully used (i.e. required for protocol operation). IP is an example of a protocol that allows TLVs but are rarely used in practice (router fast paths usually that assume no IP options). Note that TCP TLVs are implemented in software as well as (NIC) hardware handling various forms of TCP offload. Additional discussions about hardware implications for extensibility is captured in Section 18.
- o Extension headers are closely related to TLVs. These also carry type/value information, but instead of being a list of TLVs within a single protocol header, each one is in its own protocol header. IPv6 extension headers and SFC NSH are examples of this technique. Similar to TLVs these offer a wide range of extensibility, but have similarly complex processing. Another difference with TLVs is that each extension header is idempotent. This is beneficial in cases where a protocol implements a push/pop model for header elements like service chaining, but makes it more difficult group correlated information within one protocol header.
- o A particular form of extension headers are the tags used by IEEE 802 protocols. Those are similar to e.g., IPv6 extension headers but with the key difference that each tag is a fixed length header where the length is implicit in the tag value. Thus as long as a receiver can be programmed with a tag value to length map, it can skip those new tags.
- o Flag-fields are a non-TLV like method of extending a protocol header. The basic idea is that the header contains a set of flags, where each set flags corresponds to optional field that is present in the header. GRE is an example of a protocol that employs this mechanism. The fields are present in the header in the order of the flags, and the length of each field is fixed. Flag-fields are simpler to process compared to TLVs, having fewer validations and the order of the optional fields is deterministic. A disadvantage is that range of possible extensions with flag-fields is smaller than TLVs.

The requirements for receiving unknown or unimplemented extensible elements in an encapsulation protocol (flags, TLVs, optional fields)

need to be specified. There are two parties to consider, middle boxes and terminal endpoints of encapsulation (at the decapsulator).

A protocol may allow or expect nodes in a path to modify fields in an encapsulation (example use of this is BIER). In this case, the middleboxes should follow the same requirements as nodes terminating the encapsulation. In the case that middle boxes do not modify the encapsulation, we can assume that they may still inspect any fields of the encapsulation. Missing or unknown fields should be accepted per protocol specification, however it is permissible for a site to implement a local policy otherwise (e.g. a firewall may drop packets with unknown options).

For handling unknown options at terminal nodes, there are two possibilities: drop packet or accept while ignoring the unknown options. Many Internet protocols specify that reserved flags must be set to zero on transmission and ignored on reception. L2TP is example data protocol that has such flags. GRE is a notable exception to this rule, reserved flag bits 1-5 cannot be ignored [RFC2890]. For TCP and IPv4, implementations must ignore optional TLVs with unknown type; however in IPv6 if a packet contains an unknown extension header (unrecognized next header type) the packet must be dropped with an ICMP error message returned. The IPv6 options themselves (encoded inside the destinations options or hop-by-hop options extension header) have more flexibility. There are bits in the option code are used to instruct the receiver whether to ignore, silently drop, or drop and send error if the option is unknown. Some protocols define a "mandatory bit" that can is set with TLVs to indicate that an option must not be ignored. Conceptually, optional data elements can only be ignored if they are idempotent and do not alter how the rest of the packet is parsed or processed.

Depending on what type of protocol evolution one can predict, it might make sense to have a way for a sender to express that the packet should be dropped by a terminal node which does not understand the new information. In other cases it would make sense to have the receiver silently ignore the new info. The former can be expressed by having a version field in the encapsulation, or a notion of "mandatory bit" as discussed above.

A security mechanism which use some form secure hash over the encapsulation header would need to be able to know which extensions can be changed in flight.

In summary:

- o Encapsulations need the ability to be extended to handle e.g., the OAM or security aspects discussed in this document.
- o Practical experience seems to tell us that extensibility mechanisms which are not in use on day one might result in immediate ossification by lack of implementation support. In some cases that has occurred in routers and in other cases in middleboxes. Hence devising ways where the extensibility mechanisms are in use seems important.

16. Layering Considerations

One can envision that SFC might use NVO3 as a delivery/transport mechanism. With more imagination that in turn might be delivered using BIER. Thus it is useful to think about what things look like when we have BIER+NVO3+SFC+payload. Also, if NVO3 is widely deployed there might be cases of NVO3 nesting where a customer uses NVO3 to provide network virtualization e.g., across departments. That customer uses a service provider which happens to use NVO3 to provide transport for their customers.Thus NVO3 in NVO3 might happen.

A key question we set out to answer is what the packets might look like in such a case, and in particular whether we would end up with multiple UDP headers for entropy.

Based on the discussion in the Entropy section, the entropy is associated with the outer delivery IP header. Thus if there are multiple IP headers there would be a UDP header for each one of the IP headers. But SFC does not require its own IP header. So a case of NVO3+SFC would be IP+UDP+NVO3+SFC. A nested NVO3 encapsulation would have independent IP+UDP headers.

The layering also has some implications for middleboxes.

- o A device on the path between the ingress and egress is allowed to transparently inspect all layers of the protocol stack and drop or forward, but not transparently modify anything but the layer in which they operate. What this means is that an IP router is allowed modify the outer IP ttl and ECN bits, but not the encapsulation header or inner headers and payload. And a BIER router is allowed to modify the BIER header.
- o Alternatively such a device can become visible at a higher layer. E.g., a middlebox could a middlebox could first decapsulate, perform some function then encapsulate; which means it will generate a new encapsulation header.

The design team asked itself some additional questions:

- Would it make sense to have a common encapsulation base header (for OAM, security?, etc) and then followed by the specific information for NVO3, SFC, BIER? Given that there are separate proposals and the set of information needing to be carried differs, and the extensibility needs might be different, it would be difficult and not that useful to have a common base header.
- With a base header in place, one could view the different functions (NVO3, SFC, and BIER) as different extensions to that base header resulting in encodings which are more space optimal by not repeating the same base header. The base header would only be repeated when there is an additional IP (and hence UDP) header. That could mean a single length field (to skip to get to the payload after all the encapsulation headers). That might be technically feasible, but it would create a lot of dependencies between different WGs making it harder to make progress. Compare with the potential savings in packet size.

17. Service model

The IP service is lossy and subject to reordering. In order to avoid a performance impact on transports like TCP the handling of packets is designed to avoid reordering packets that are in the same transport flow (which is typically identified by the 5-tuple). But across such flows the receiver can see different ordering for a given sender. That is the case for a unicast vs. a multicast flow from the same sender.

There is a general tussle between the desire for high capacity utilization across a multipath network and the impact on packet ordering within the same flow (which results in lower transport protocol performance). That isn't affected by the introduction of an encapsulation. However, the encapsulation comes with some entropy, and there might be cases where folks want to change that in response to overload or failures. For instance, one might want to change UDP source port to try different ECMP route. Such changes can result in packet reordering within a flow, hence would need to be done infrequently and with care e.g., by identifying packet trains.

There might be some applications/services which are not able to handle reordering across flows. The IETF has defined pseudo-wires [RFC3985] which provides the ability to ensure ordering (implemented using sequence numbers and/or timestamps).

Architectural such services would make sense, but as a separate layer on top of an encapsulation protocol. They could be deployed between ingress and egress of a tunnel which uses some encaps. Potentially the tunnel control points at the ingress and egress could become a platform for fixing suboptimal behavior elsewhere in the network.

That would clearly be undesirable in the general case. However, handling encapsulation of non-IP traffic hence non-congestion-controlled traffic is likely to be required, which implies some fairness and/or QoS policing on the ingress and egress devices.

But the tunnels could potentially do more like increase reliability (retransmissions, FEC) or load spreading using e.g. MP-TCP between ingress and egress.

18. Hardware Friendly

Hosts, switches and routers often leverage capabilities in the hardware to accelerate packet encapsulation, decapsulation and forwarding.

Some design considerations in encapsulation that leverage these hardware capabilities may result in more efficiently packet processing and higher overall protocol throughput.

While "hardware friendliness" can be viewed as unnecessary considerations for a design, part of the motivation for considering this is ease of deployment; being able to leverage existing NIC and switch chips for at least a useful subset of the functionality that the new encapsulation provides. The other part is the ease of implementing new NICs and switch/router chips that support the encapsulation at ever increasing line rates.

[disclaimer] There are many different types of hardware in any given network, each maybe better at some tasks while worse at others. We would still recommend protocol designers to examine the specific hardware that are likely to be used in their networks and make decisions on a case by case basis.

Some considerations are:

- Keep the encap header small. Switches and routers usually only read the first small number of bytes into the fast memory for quick processing and easy manipulation. The bulk of the packets are usually stored in slow memory. A big encap header may not fit and additional read from the slow memory will hurt the overall performance and throughput.
- Put important information at the beginning of the encapsulation header. The reasoning is similar as explained in the previous point. If important information are located at the beginning of the encapsulation header, the packet may be processed with smaller number of bytes to be read into the fast memory and improve performance.

- o Avoid full packet checksums in the encapsulation if possible. Encapsulations should instead consider adding their own checksum which covers the encapsulation header and any IPv6 pseudo-header. The motivation is that most of the switch/router hardware make switching/forwarding decisions by reading and examining only the first certain number of bytes in the packet. Most of the body of the packet do not need to be processed normally. If we are concerned of preventing packet to be misdelivered due to memory errors, consider only perform header checksums. Note that NIC chips can typically already do full packet checksums for TCP/UDP, while adding a header checksum might require adding some hardware support.
- o Place important information at fixed offset in the encapsulation header. Packet processing hardware may be capable of parallel processing. If important information can be found at fixed offset, different part of the encapsulation header may be processed by different hardware units in parallel (for example multiple table lookups may be launched in parallel). It is easier for hardware to handle optional information when the information, if present, can be found in ideally one place, but in general, in as few places as possible. That facilitates parallel processing. TLV encoding with unconstrained order typically does not have that property.
- o Limit the number of header combinations. In many cases the hardware can explore different combinations of headers in parallel, however there is some added cost for this.

18.1. Considerations for NIC offload

This section provides guidelines to provide support of common offloads for encapsulation in Network Interface Cards (NICs). Offload mechanisms are techniques that are implemented separately from the normal protocol implementation of a host networking stack and are intended to optimize or speed up protocol processing. Hardware offload is performed within a NIC device on behalf of a host.

There are three basic offload techniques of interest:

- o Receive multi queue
- o Checksum offload
- o Segmentation offload

18.1.1. Receive multi-queue

Contemporary NICs support multiple receive descriptor queues (multi-queue). Multi-queue enables load balancing of network processing for a NIC across multiple CPUs. On packet reception, a NIC must select

the appropriate queue for host processing. Receive Side Scaling (RSS) is a common method which uses the flow hash for a packet to index an indirection table where each entry stores a queue number.

UDP encapsulation, where the source port is used for entropy, should be compatible with multi-queue NICs that support five-tuple hash calculation for UDP/IP packets as input to RSS. The source port ensures classification of the encapsulated flow even in the case that the outer source and destination addresses are the same for all flows (e.g. all flows are going over a single tunnel).

18.1.2. Checksum offload

Many NICs provide capabilities to calculate standard ones complement payload checksum for packets in transmit or receive. When using encapsulation over UDP there are at least two checksums that may be of interest: the encapsulated packet's transport checksum, and the UDP checksum in the outer header.

18.1.2.1. Transmit checksum offload

NICs may provide a protocol agnostic method to offload transmit checksum (NETIF_F_HW_CSUM in Linux parlance) that can be used with UDP encapsulation. In this method the host provides checksum related parameters in a transmit descriptor for a packet. These parameters include the starting offset of data to checksum, the length of data to checksum, and the offset in the packet where the computed checksum is to be written. The host initializes the checksum field to pseudo header checksum. In the case of encapsulated packet, the checksum for an encapsulated transport layer packet, a TCP packet for instance, can be offloaded by setting the appropriate checksum parameters.

NICs typically can offload only one transmit checksum per packet, so simultaneously offloading both an inner transport packet's checksum and the outer UDP checksum is likely not possible. In this case setting UDP checksum to zero (per above discussion) and offloading the inner transport packet checksum might be acceptable.

There is a proposal in [I-D.herbert-remotecsumoffload] to leverage NIC checksum offload when an encapsulator is co-resident with a host.

18.1.2.2. Receive checksum offload

Protocol encapsulation is compatible with NICs that perform a protocol agnostic receive checksum (CHECKSUM_COMPLETE in Linux parlance). In this technique, a NIC computes a ones complement checksum over all (or some predefined portion) of a packet. The

computed value is provided to the host stack in the packet's receive descriptor. The host driver can use this checksum to "patch up" and validate any inner packet transport checksum, as well as the outer UDP checksum if it is non-zero.

Many legacy NICs don't provide checksum-complete but instead provide an indication that a checksum has been verified (CHECKSUM_UNNECESSARY in Linux). Usually, such validation is only done for simple TCP/IP or UDP/IP packets. If a NIC indicates that a UDP checksum is valid, the checksum-complete value for the UDP packet is the "not" of the pseudo header checksum. In this way, checksum-unnecessary can be converted to checksum-complete. So if the NIC provides checksum-unnecessary for the outer UDP header in an encapsulation, checksum conversion can be done so that the checksum-complete value is derived and can be used by the stack to validate an checksums in the encapsulated packet.

18.1.3. Segmentation offload

Segmentation offload refers to techniques that attempt to reduce CPU utilization on hosts by having the transport layers of the stack operate on large packets. In transmit segmentation offload, a transport layer creates large packets greater than MTU size (Maximum Transmission Unit). It is only at much lower point in the stack, or possibly the NIC, that these large packets are broken up into MTU sized packet for transmission on the wire. Similarly, in receive segmentation offload, small packets are coalesced into large, greater than MTU size packets at a point low in the stack receive path or possibly in a device. The effect of segmentation offload is that the number of packets that need to be processed in various layers of the stack is reduced, and hence CPU utilization is reduced.

18.1.3.1. Transmit Segmentation Offload

Transmit Segmentation Offload (TSO) is a NIC feature where a host provides a large (larger than MTU size) TCP packet to the NIC, which in turn splits the packet into separate segments and transmits each one. This is useful to reduce CPU load on the host.

The process of TSO can be generalized as:

- Split the TCP payload into segments which allow packets with size less than or equal to MTU.
- For each created segment:

    1. Replicate the TCP header and all preceding headers of the original packet.

2. Set payload length fields in any headers to reflect the length of the segment.
3. Set TCP sequence number to correctly reflect the offset of the TCP data in the stream.
4. Recompute and set any checksums that either cover the payload of the packet or cover header which was changed by setting a payload length.

Following this general process, TSO can be extended to support TCP encapsulation UDP. For each segment the Ethernet, outer IP, UDP header, encapsulation header, inner IP header if tunneling, and TCP headers are replicated. Any packet length header fields need to be set properly (including the length in the outer UDP header), and checksums need to be set correctly (including the outer UDP checksum if being used).

To facilitate TSO with encapsulation it is recommended that optional fields should not contain values that must be updated on a per segment basis-- for example an encapsulation header should not include checksums, lengths, or sequence numbers that refer to the payload. If the encapsulation header does not contain such fields then the TSO engine only needs to copy the bits in the encapsulation header when creating each segment and does not need to parse the encapsulation header.

18.1.3.2. Large Receive Offload

Large Receive Offload (LRO) is a NIC feature where packets of a TCP connection are reassembled, or coalesced, in the NIC and delivered to the host as one large packet. This feature can reduce CPU utilization in the host.

LRO requires significant protocol awareness to be implemented correctly and is difficult to generalize. Packets in the same flow need to be unambiguously identified. In the presence of tunnels or network virtualization, this may require more than a five-tuple match (for instance packets for flows in two different virtual networks may have identical five-tuples). Additionally, a NIC needs to perform validation over packets that are being coalesced, and needs to fabricate a single meaningful header from all the coalesced packets.

The conservative approach to supporting LRO for encapsulation would be to assign packets to the same flow only if they have identical five-tuple and were encapsulated the same way. That is the outer IP addresses, the outer UDP ports, encapsulated protocol, encapsulation headers, and inner five tuple are all identical.

18.1.3.3. In summary:

In summary, for NIC offload:

- o The considerations for using full UDP checksums are different for NIC offload than for implementations in forwarding devices like routers and switches.
- o Be judicious about encapsulations that change fields on a per-packet basis, since such behavior might make it hard to use TSO.

19. Middlebox Considerations

This document has touched upon middleboxes in different section. The reason for this is as encapsulations get widely deployed one would expect different forms of middleboxes might become aware of the encapsulation protocol just as middleboxes have been made aware of other protocols where there are business and deployment opportunities. Such middleboxes are likely to do more than just drop packets based on the UDP port number used by an encapsulation protocol.

We note that various forms of encapsulation gateways that stitch one encapsulation protocol together with another form of protocol could have similar effects.

An example of a middlebox that could see some use would be an NVO3-aware firewall that would filter on the VNI IDs to provide some defense in depth inside or across NVO3 datacenters.

A question for the IETF is whether we should document what to do or what not to do in such middleboxes. This document touches on areas of OAM and ECMP as it relates to middleboxes and it might make sense to document how encapsulation-aware middleboxes should avoid unintended consequences in those (and perhaps other) areas.

In summary:

- o We are likely to see middleboxes that at least parse the headers for successful new encapsulations.
- o Should the IETF document considerations for what not to do in such middleboxes?

20. Related Work

The IETF and industry has defined encapsulations for a long time, with examples like GRE [RFC2890], VXLAN [RFC7348], and NVGRE [I-D.sridharan-virtualization-nvgre] being able to carry arbitrary Ethernet payloads, and various forms of IP-in-IP and IPsec

encapsulations that can carry IP packets. As part of NVO3 there has been additional proposals like Geneve [I-D.gross-geneve] and GUE [I-D.herbert-gue] which look at more extensibility. NSH [I-D.quinn-sfc-nsh] is an example of an encapsulation that tries to provide extensibility mechanisms which target both hardware and software implementations.

There is also a large body of work around MPLS encapsulations [RFC3032]. The MPLS-in-UDP work [I-D.ietf-mpls-in-udp] and GRE over UDP [I-D.ietf-tsvwg-gre-in-udp-encap] have worked on some of the common issues around checksum and congestion control. MPLS also introduced a entropy label [RFC6790]. There is also a proposal for MPLS encryption [I-D.farrelll-mpls-opportunistic-encrypt].

The idea to use a UDP encapsulation with a UDP source port for entropy for the underlay routers' ECMP dates back to LISP [RFC6830].

The pseudo-wire work [RFC3985] is interesting in the notion of layering additional services/characteristics such as ordered delivery or timely deliver on top of an encapsulation. That layering approach might be useful for the new encapsulations as well. For instance, the control word [RFC4385]. There is also material on congestion control for pseudo-wires in [I-D.ietf-pwe3-congcons].

Both MPLS and L2TP [RFC3931] rely on some control or signaling to establish state (for the path/labels in the case of MPLS, and for the session in the case of L2TP). The NVO3, SFC, and BIER encapsulations will also have some separation between the data plane and control plane, but the type of separation appears to be different.

IEEE 802.1 has defined encapsulations for L2 over L2, in the form of Provider backbone Bridging (PBB) [IEEE802.1Q-2014] and Equal Cost Multipath (ECMP) [IEEE802.1Q-2014]. The latter includes something very similar to the way the UDP source port is used as entropy: "The flow hash, carried in an F-TAG, serves to distinguish frames belonging to different flows and can be used in the forwarding process to distribute frames over equal cost paths"

TRILL, which is also a L2 over L2 encapsulation, took a different approach to entropy but preserved the ability for OAM frames [RFC7174] to use the same entropy hence ECMP path as data frames. In [I-D.ietf-trill-oam-fm] there 96 bytes of headers for entropy in the OAM frames, followed by the actual OAM content. This ensures that any headers, which fit in those 96 bytes except the OAM bit in the TRILL header, can be used for ECMP hashing.

As encapsulations evolve there might be a desire to fit multiple inner packets into one outer packet. The work in [I-D.saldana-tsvwg-simplemux] might be interesting for that purpose.

21. Acknowledgements

The authors acknowledge the comments from Alia Atlas, Fred Baker, David Black, Bob Briscoe, Stewart Bryant, Mike Cox, Andy Malis, Radia Perlman, Carlos Pignataro, Jamal Hadi Salim, Michael Smith, and Lucy Yong.

22. Open Issues

- Middleboxes:
  - Due to OAM there are constraints on middleboxes in general. If middleboxes inspect the packet past the outer IP+UDP and encapsulation header and look for inner IP and TCP/UDP headers, that might violate the assumption that OAM packets will be handled the same as regular data packets. That issue is broader than just QoS - applies to firewall filters etc.
  - Firewalls looking at inner payload? How does that work for OAM frames? Even if it only drops ... TRILL approach might be an option? Would that encourage more middleboxes making the network more fragile?
  - Editorially perhaps we should pull the above two into a separate section about middlebox considerations?
- Next-protocol indication - should it be common across different encapsulation headers? We will have different ways to indicate the presence of the first encapsulation header in a packet (could be a UDP destination port, an Ethernet type, etc depending on the outer delivery header). But for the next protocol past an encapsulation header one could envision creating or adoption a common scheme. Such a would also need to be able to identify following headers like Ethernet, IPv4/IPv6, ESP, etc.
- Common OAM error reporting protocol?
- There is discussion about timestamps, sequence numbers, etc in three different parts of the document. OAM, Congestion Considerations, and Service Model, where the latter argues that a pseudo-wire service should really be layered on top of the encapsulation using its own header. Those recommendations seem to be at odds with each other. Do we envision sequence numbers, timestamps, etc as potential extensions for OAM and CC? If so, those extensions could be used to provide a service which doesn’t reorder packets.

23. Change Log

The changes from draft-rtg-dt-encap-01 based on feedback at the Dallas IETF meeting:

- Setting the context that not all common issues might apply to all encapsulations, but that they should all be understood before being dismissed.
- Clarified that IPv6 flow label is useful for entropy in combination with a UDP source port.
- Editorially added a "summary" set of bullets to most sections.
- Editorial clarifications in the next protocol section to more clearly state the three areas.
- Folded the two next protocol sections into one.
- Mention the MPLS first nibble issue in the next protocol section.
- Mention that viewing the next protocol as an index to a table with processing instructions can provide additional flexibility in the protocol evolution.
- For the OAM "don’t forward to end stations" added that defining a bit seems better than using a special next-protocol value.
- Added mention of DTLS in addition to IPsec for security.
- Added some mention of IPv6 hob-by-hop options of other headers than potentially can be copied from inner to outer header.
- Added text on architectural considerations when it might make sense to define an additional header/protocol as opposed to using the extensibility mechanism in the existing encapsulation protocol.
- Clarified the "unconstrained TLVs" in the hardware friendly section.
- Clarified the text around checksum verification and full vs. header checksums.
- Added wording that the considerations might apply for encaps outside of the routing area.
- Added references to draft-ietf-pwe3-congcons, draft-ietf-tsvwg-rfc5405bis, RFC2473, and RFC7325
- Removed reference to RFC3948.
- Updated the acknowledgements section.
- Added this change log section.

24. References

24.1. Normative References

[I-D.ietf-tsvwg-rfc5405bis]
Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", draft-ietf-tsvwg-rfc5405bis-19 (work in progress), October 2016.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <http://www.rfc-editor.org/info/rfc2473>.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, DOI 10.17487/RFC2890, September 2000, <http://www.rfc-editor.org/info/rfc2890>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <http://www.rfc-editor.org/info/rfc2983>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <http://www.rfc-editor.org/info/rfc3032>.

[RFC3931] Lau, J., Ed., Townsley, M., Ed., and I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, DOI 10.17487/RFC3931, March 2005, <http://www.rfc-editor.org/info/rfc3931>.

[RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <http://www.rfc-editor.org/info/rfc3985>.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <http://www.rfc-editor.org/info/rfc4385>.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <http://www.rfc-editor.org/info/rfc6040>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <http://www.rfc-editor.org/info/rfc6438>.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <http://www.rfc-editor.org/info/rfc6790>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <http://www.rfc-editor.org/info/rfc6830>.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <http://www.rfc-editor.org/info/rfc6935>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <http://www.rfc-editor.org/info/rfc6936>.

[RFC7174] Salam, S., Senevirathne, T., Aldrin, S., and D. Eastlake 3rd, "Transparent Interconnection of Lots of Links (TRILL) Operations, Administration, and Maintenance (OAM) Framework", RFC 7174, DOI 10.17487/RFC7174, May 2014, <http://www.rfc-editor.org/info/rfc7174>.

[RFC7325] Villamizar, C., Ed., Kompella, K., Amante, S., Malis, A., and C. Pignataro, "MPLS Forwarding Compliance and Performance Requirements", RFC 7325, DOI 10.17487/RFC7325, August 2014, <http://www.rfc-editor.org/info/rfc7325>.

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

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

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

24.2. Informative References

[I-D.briscoe-conex-data-centre]
Briscoe, B. and M. Sridharan, "Network Performance Isolation in Data Centres using Congestion Policing", draft-briscoe-conex-data-centre-02 (work in progress), February 2014.

[I-D.farrelll-mpls-opportunistic-encrypt]
Farrel, A. and S. Farrell, "Opportunistic Security in MPLS Networks", draft-farrelll-mpls-opportunistic-encrypt-05 (work in progress), June 2015.

[I-D.gross-geneve]
Gross, J., Sridhar, T., Garg, P., Wright, C., Ganga, I., Agarwal, P., Duda, K., Dutt, D., and J. Hudson, "Geneve: Generic Network Virtualization Encapsulation", draft-gross-geneve-02 (work in progress), October 2014.

[I-D.herbert-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-herbert-gue-03 (work in progress), March 2015.

[I-D.herbert-remotecsumoffload]
Herbert, T., "Remote checksum offload for encapsulation", draft-herbert-remotecsumoffload-02 (work in progress), March 2016.

[I-D.ietf-mpls-in-udp]
Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", draft-ietf-mpls-in-udp-11 (work in progress), January 2015.

[I-D.ietf-nvo3-arch]
Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data Center Network Virtualization Overlays (NVO3)", draft-ietf-nvo3-arch-08 (work in progress), September 2016.

[I-D.ietf-pwe3-congcons]
Stein, Y., Black, D., and B. Briscoe, "Pseudowire Congestion Considerations", draft-ietf-pwe3-congcons-02 (work in progress), July 2014.

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-11 (work in progress), July 2015.

[I-D.ietf-sfc-problem-statement]
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-13 (work in progress), February 2015.

[I-D.ietf-trill-oam-fm]
Senevirathne, T., Finn, N., Salam, S., Kumar, D., Eastlake, D., Aldrin, S., and L. Yizhou, "TRILL Fault Management", draft-ietf-trill-oam-fm-11 (work in progress), October 2014.

[I-D.ietf-tsvwg-circuit-breaker]
Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-15 (work in progress), April 2016.

[I-D.ietf-tsvwg-gre-in-udp-encap]
Yong, L., Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", draft-ietf-tsvwg-gre-in-udp-encap-19 (work in progress), September 2016.

[I-D.ietf-tsvwg-port-use]
Touch, J., "Recommendations on Using Assigned Transport Port Numbers", draft-ietf-tsvwg-port-use-11 (work in progress), April 2015.

[I-D.quinn-sfc-nsh]
Quinn, P., Guichard, J., Surendra, S., Smith, M., Henderickx, W., Nadeau, T., Agarwal, P., Manur, R., Chauhan, A., Halpern, J., Majee, S., Elzur, U., Melman, D., Garg, P., McConnell, B., Wright, C., and K. Kevin, "Network Service Header", draft-quinn-sfc-nsh-07 (work in progress), February 2015.

[I-D.saldana-tsvwg-simplemux]
Saldana, J., "Simplemux. A generic multiplexing protocol", draft-saldana-tsvwg-simplemux-05 (work in progress), July 2016.

[I-D.shepherd-bier-problem-statement]
Shepherd, G., Dolganow, A., and a. arkadiy.gulko@thomsonreuters.com, "Bit Indexed Explicit Replication (BIER) Problem Statement", draft-shepherd-bier-problem-statement-02 (work in progress), February 2015.

[I-D.sridharan-virtualization-nvgre]
Garg, P. and Y. Wang, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-08 (work in progress), April 2015.

[I-D.wei-tsvwg-tunnel-congestion-feedback]
Wei, X., Zhu, L., Deng, L., and B. Briscoe, "Tunnel Congestion Feedback", draft-wei-tsvwg-tunnel-congestion-feedback-04 (work in progress), June 2015.

[I-D.wijnands-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-wijnands-bier-architecture-05 (work in progress), March 2015.

[I-D.wijnands-mpls-bier-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-wijnands-mpls-bier-encapsulation-02 (work in progress), December 2014.

[I-D.xu-bier-encapsulation]
Xu, X., somasundaram.s@alcatel-lucent.com, s., Jacquenet, C., Raszuk, R., and Z. Zhang, "A Transport-Independent Bit Index Explicit Replication (BIER) Encapsulation Header", draft-xu-bier-encapsulation-06 (work in progress), September 2016.

[IEEE802.1Q-2014]
IEEE, "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks", IEEE Std 802.1Q-2014, 2014, <http://www.ieee802.org/1/pages/802.1Q-2014.html>.

(Access Controlled link within page)

[RFC0033] Crocker, S., "New Host-Host Protocol", RFC 33, DOI 10.17487/RFC0033, February 1970, <http://www.rfc-editor.org/info/rfc33>.

Authors' Addresses

Erik Nordmark
Arista Networks
5453 Great America Parkway
Santa Clara, CA 95054
USA

Email: nordmark@arista.com

Albert Tian
Ericsson Inc.
300 Holger Way
San Jose, California 95134
USA

Email: albert.tian@ericsson.com

Jesse Gross
VMware
3401 Hillview Ave.
Palo Alto, CA 94304
USA

Email: jgross@vmware.com

Jon Hudson
Brocade Communications Systems, Inc.
130 Holger Way
San Jose, CA 95134
USA

Email: jon.hudson@gmail.com

Lawrence Kreeger
Cisco Systems, Inc.
170 W. Tasman Avenue
San Jose, CA 95134
USA

Email: kreeger@cisco.com

Pankaj Garg
Microsoft
1 Microsoft Way
Redmond, WA 98052
USA

Email: pankajg@microsoft.com

Patricia Thaler
Broadcom Corporation
3151 Zanker Road
San Jose, CA 95134
USA

Email: pthaler@broadcom.com

Tom Herbert
Facebook
1 Hacker Way
Menlo Park, CA 94052
USA

Email: tom@herbertland.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 20, 2018

L. Berger
LabN Consulting, L.L.C.
C. Hopps
Deutsche Telekom
A. Lindem
Cisco Systems
D. Bogdanovic

X. Liu
Jabil

March 19, 2018

YANG Model for Logical Network Elements
draft-ietf-rtgwg-lne-model-10

Abstract

This document defines a logical network element YANG module. This module can be used to manage the logical resource partitioning that may be present on a network device. Examples of common industry terms for logical resource partitioning are Logical Systems or Logical Routers. The YANG model in this document conforms to the Network Management Datastore Architecture as defined in RFCXXXX.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 20, 2018.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] module to support the creation of logical network elements on a network device. A logical network element (LNE) is an independently managed virtual device made up of resources allocated to it from the host or parent network device. An LNE running on a host network device conceptually parallels a virtual machine running on a host system. Using host-virtualization terminology one could refer to an LNE as a "Guest", and the containing network-device as the "Host". While LNEs may be implemented via host-virtualization technologies this is not a requirement. The YANG model in this document conforms to the Network

Management Datastore Architecture defined in the [I-D.ietf-netmod-revised-datastores].

This document also defines the necessary augmentations for allocating host resources to a given LNE. As the interface management model [I-D.ietf-netmod-rfc7223bis] is the only a module that currently defines host resources, this document currently defines only a single augmentation to cover the assignment of interfaces to an LNE. Future modules that define support for the control of host device resources are expected to, where appropriate, provide parallel support for the assignment of controlled resources to LNEs.

As each LNE is an independently managed device, each will have its own set of YANG modeled data that is independent of the host device and other LNEs. For example, multiple LNEs may all have their own "Tunnel0" interface defined which will not conflict with each other and will not exist in the host's interface model. An LNE will have its own management interfaces possibly including independent instances of netconf/restconf/etc servers to support configuration of their YANG models. As an example of this independence, an implementation may choose to completely rename assigned interfaces, so on the host the assigned interface might be called "Ethernet0/1" while within the LNE it might be called "eth1".

In addition to standard management interfaces, a host device implementation may support accessing LNE configuration and operational YANG models directly from the host system. When supported, such access is accomplished through a yang-schema-mount mount point [I-D.ietf-netmod-schema-mount] under which the root level LNE YANG models may be accessed.

Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric.

1.1. Terminology

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

Readers are expected to be familiar with terms and concepts of YANG [RFC7950] and YANG Schema Mount [I-D.ietf-netmod-schema-mount].

This document uses the graphical representation of data models defined in [I-D.ietf-netmod-yang-tree-diagrams].

2. Overview

In this document, we consider network devices that support protocols and functions defined within the IETF Routing Area, e.g, routers, firewalls, and hosts. Such devices may be physical or virtual, e.g., a classic router with custom hardware or one residing within a server-based virtual machine implementing a virtual network function (VNF). Each device may sub-divide their resources into logical network elements (LNEs), each of which provides a managed logical device. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric. Each LNE may also support virtual routing and forwarding (VRF) and virtual switching instance (VSI) functions, which are referred to below as a network instances (NIs). This breakdown is represented in Figure 1.

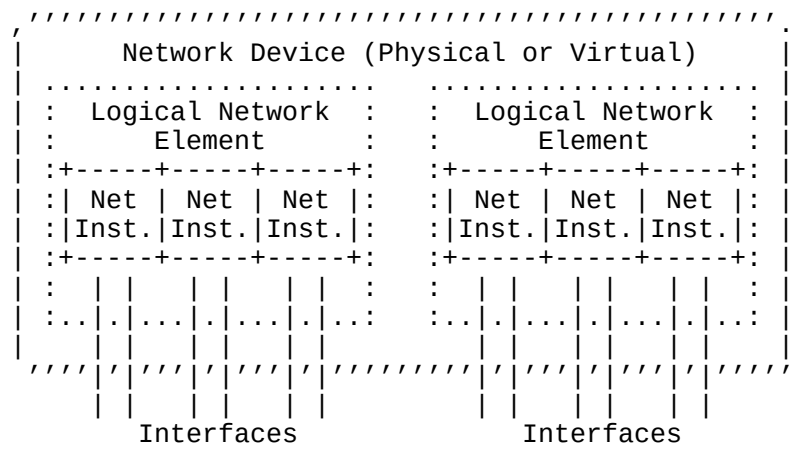


Figure 1: Module Element Relationships

A model for LNEs is described in Section 3 and the model for NIs is covered in [I-D.ietf-rtgwg-ni-model].

The interface management model [I-D.ietf-netmod-rfc7223bis] is an existing model that is impacted by the definition of LNEs and network instances. This document and [I-D.ietf-rtgwg-ni-model] define augmentations to the interface module to support LNEs and NIs. Similar elements, although perhaps only for LNEs, may also need to be included as part of the definition of the future hardware and QoS modules.

Interfaces are a crucial part of any network device’s configuration and operational state. They generally include a combination of raw

physical interfaces, link-layer interfaces, addressing configuration, and logical interfaces that may not be tied to any physical interface. Several system services, and layer 2 and layer 3 protocols may also associate configuration or operational state data with different types of interfaces (these relationships are not shown for simplicity). The interface management model is defined by [I-D.ietf-netmod-rfc7223bis]. The logical-network-element module augments existing interface management model by adding an identifier which is used on interfaces to identify an associated LNE.

The interface related augmentation is as follows:

```
    module: ietf-logical-network-element
        augment /if:interfaces/if:interface:
          +--rw bind-lne-name?   ->
                /logical-network-elements/logical-network-element/name
```

The interface model defined in [I-D.ietf-netmod-rfc7223bis] is structured to include all interfaces in a flat list, without regard to logical assignment of resources supported on the device. The bind-lne-name and leaf provides the association between an interface and its associated LNE. Note that as currently defined, to assign an interface to both an LNE and NI, the interface would first be assigned to the LNE and then within that LNE's interface module, the LNE's representation of that interface would be assigned to an NI using the mechanisms defined in [I-D.ietf-rtgwg-ni-model].

3. Logical Network Elements

Logical network elements support the ability of some devices to partition resources into independent logical routers and/or switches. Device support for multiple logical network elements is implementation specific. Systems without such capabilities need not include support for the logical-network-element module. In physical devices, some hardware features are shared across partitions, but control plane (e.g., routing) protocol instances, tables, and configuration are managed separately. For example, in logical routers or VNFs, this may correspond to establishing multiple logical instances using a single software installation. The model supports configuration of multiple instances on a single device by creating a list of logical network elements, each with their own configuration and operational state related to routing and switching protocols.

The LNE model can be represented as:

```
module: ietf-logical-network-element
    +--rw logical-network-elements
       +--rw logical-network-element* [name]
          +--rw name           string
          +--rw managed?       boolean
          +--rw description?   string
          +--mp root
  augment /if:interfaces/if:interface:
    +--rw bind-lne-name?
            -> /logical-network-elements/logical-network-element/name

  notifications:
    +---n bind-lne-name-failed
       +--ro name             -> /if:interfaces/interface/name
       +--ro bind-lne-name
       |       -> /if:interfaces/interface/lne:bind-lne-name
       +--ro error-info?      string
```

'name' identifies the logical network element. 'managed' indicates if the server providing the host network device will provide the client LNE information via the 'root' structure. The root of an LNE's specific data is the schema mount point 'root'. bind-lne-name is used to associated an interface with an LNE and bind-lne-name-failed is used in certain failure cases.

An LNE root MUST contain at least the YANG library [RFC7895] and Interfaces [I-D.ietf-netmod-rfc7223bis] modules.

3.1. LNE Instantiation and Resource Assignment

Logical network elements may be controlled by clients using existing list operations. When list entries are created, a new LNE is instantiated. The models mounted under an LNE root are expected to be dependent on the server implementation. When a list entry is deleted, an existing LNE is destroyed. For more information, see [RFC7950] Section 7.8.6.

Once instantiated, host network device resources can be associated with the new LNE. As previously mentioned, this document augments ietf-interfaces with the bind-lne-name leaf to support such associations for interfaces. When an bind-lne-name is set to a valid LNE name, an implementation MUST take whatever steps are internally necessary to assign the interface to the LNE or provide an error message (defined below) with an indication of why the assignment failed. It is possible for the assignment to fail while processing the set, or after asynchronous processing. Error notification in the latter case is supported via a notification.

On a successful interface assignment to an LNE, an implementation MUST also make the resource available to the LNE by providing a system created interface to the LNE. The name of the system created interface is a local matter and may be identical or completely different, and mapped from and to, the name used in the context of the host device. The system created interface SHOULD be exposed via the LNE-specific instance of the interfaces module [I-D.ietf-netmod-rfc7223bis].

3.2. LNE Management - LNE View

Each LNE instance is expected to support management functions from within the context of the LNE root, via a server that provides information with the LNE's root exposed as device root. Management functions operating within the context of an LNE are accessed through the LNE's standard management interfaces, e.g., NETCONF and SNMP. Initial configuration, much like the initial configuration of the host device, is a local implementation matter.

When accessing an LNE via the LNE's management interface, a network-device representation will be presented, but its scope will be limited to the specific LNE. Normal YANG/NETCONF mechanisms, together with the required YANG library [RFC7895] instance, can be used to identify the available modules. Each supported module will be presented as a top level module. Only LNE associated resources will be reflected in resource related modules, e.g., interfaces, hardware, and perhaps QoS. From the management perspective, there will be no difference between the available LNE view (information) and a physical network device.

3.3. LNE Management - Host Network Device View

There are multiple implementation approaches possible to enable a network device to support the logical-network-element module and multiple LNEs. Some approaches will allow the management functions operating at network device level to access LNE configuration and operational information, while others will not. Similarly, even when LNE management from the network device is supported by the implementation, it may be prohibited by user policy.

Independent of the method selected by an implementation, the 'managed' boolean mentioned above is used to indicate when LNE management from the network device context is possible. When the 'managed' boolean is 'false', the LNE cannot be managed by the host system and can only be managed from within the context of the LNE as described in the previous section, Section 3.2. Attempts to access information below a root node whose associated 'managed' boolean is set to 'false' MUST result in the error message indicated below. In

some implementations, it may not be possible to change this value. For example, when an LNE is implemented using virtual machine and traditional hypervisor technologies, it is likely that this value will be restricted to a 'false' value.

It is an implementation choice if the information can be accessed and modified from within the context of the LNE, or even the context of the host device. When the 'managed' boolean is 'true', LNE information SHALL be accessible from the context of the host device. When the associated schema-mount definition has the 'config' leaf set to 'true', then LNE information SHALL also be modifiable from the context of the host device. When LNE information is available from both the host device and from within the context of the LNE, the same information MUST be made available via the 'root' element, with paths modified as described in [I-D.ietf-netmod-schema-mount].

An implementation MAY represent an LNE's schema using either the 'inline' or 'shared-schema' approaches defined in [I-D.ietf-netmod-schema-mount]. The choice of which to use is completely an implementation choice. The inline case is anticipated to be generally used in the cases where the 'managed' will always be 'false'. The 'shared-schema' approach is expected to be be most useful in the case where all LNEs share the same schema. When 'shared-schema' is used with an LNE mount point, the YANG library rooted in the LNE's mount point MUST match the associated schema defined according to the ietf-yang-schema-mount module.

Beyond the two modules that will always be present for an LNE, as an LNE is a network device itself, all modules that may be present at the top level network device MAY also be present for the LNE. The list of available modules is expected to be implementation dependent. As is the method used by an implementation to support LNEs. Appendix B provide example uses of LNEs.

4. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

LNE information represents device and network configuration information. As such, the security of this information is important, but it is fundamentally no different than any other interface or device configuration information that has already been covered in other documents such as [I-D.ietf-netmod-rfc7223bis], [RFC7317] and [I-D.ietf-netmod-rfc8022bis].

The vulnerable "config true" parameters and subtrees are the following:

/logical-network-elements/logical-network-element: This list specifies the logical network element and the related logical device configuration.

/logical-network-elements/logical-network-element/managed: While this leaf is contained in the previous list, it is worth particular attention as it controls whether information under the LNE mount point is accessible by both the host device and within the LNE context. There may be extra sensitivity to this leaf in environments where an LNE is managed by a different party than the host device, and that party does not wish to share LNE information with the operator of the host device.

/if:interfaces/if:interface/bind-lne-name: This leaf indicates the LNE instance to which an interface is assigned. Implementations should pay particular attention to when changes to this leaf are permitted as removal of an interface from an LNE can have major impact on the LNEs operation as it is similar to physically removing an interface from the device. Implementations can reject an reassignment using the previously described error message generation.

Unauthorized access to any of these lists can adversely affect the security of both the local device and the network. This may lead to network malfunctions, delivery of packets to inappropriate destinations, and other problems.

5. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-logical-network-element

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
name:         ietf-logical-network-element
namespace:    urn:ietf:params:xml:ns:yang:ietf-logical-network-element
prefix:       lne
reference:    RFC XXXX
```

6. Logical Network Element Model

The structure of the model defined in this document is described by the YANG module below.

```
<CODE BEGINS> file "ietf-logical-network-element@2018-03-20.yang"
module ietf-logical-network-element {
  yang-version 1.1;

  // namespace

  namespace
     "urn:ietf:params:xml:ns:yang:ietf-logical-network-element";
  prefix lne;

  // import some basic types

  import ietf-interfaces {
    prefix if;
    reference "draft-ietf-netmod-rfc7223bis:
               A YANG Data Model for Interface Management";
  }
  import ietf-yang-schema-mount {
    prefix yangmnt;
    reference "draft-ietf-netmod-schema-mount: YANG Schema Mount";
    // RFC Ed.: Please replace this draft name with the corresponding
    // RFC number
  }

  organization
    "IETF Routing Area (rtgwg) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>

     Author:   Lou Berger
               <mailto:lberger@labn.net>
     Author:   Christan Hopps
               <mailto:chopps@chopps.org>
     Author:   Acee Lindem
```

```
                <mailto:acee@cisco.com>
       Author:  Dean Bogdanovic
                <mailto:ivandean@gmail.com>";
    description
      "This module is used to support multiple logical network
       elements on a single physical or virtual system.

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

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD
       License set forth in Section 4.c of the IETF Trust's Legal
       Provisions Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    // RFC Ed.: replace XXXX with actual RFC number and remove
    // this note
    // RFC Ed.: please update TBD

    revision 2018-03-20 {
      description
        "Initial revision.";
      reference "RFC XXXX";
    }

    // top level device definition statements

    container logical-network-elements {
      description
        "Allows a network device to support multiple logical
         network element (device) instances.";
      list logical-network-element {
        key "name";
        description
          "List of logical network elements.";
        leaf name {
          type string;
          description
            "Device-wide unique identifier for the
             logical network element.";
        }
        leaf managed {
          type boolean;
```

```
          default "true";
          description
            "True if the host can access LNE information
             using the root mount point.  This value
             my not be modifiable in all implementations.";
        }
        leaf description {
          type string;
          description
            "Description of the logical network element.";
        }
        container "root" {
          description
            "Container for mount point.";
          yangmnt:mount-point "root" {
            description
              "Root for models supported per logical
               network element.  This mount point may or may not
               be inline based on the server implementation.  It
               SHALL always contain a YANG library and interfaces
               instance.

               When the associated 'managed' leaf is 'false' any
               operation that attempts to access information below
               the root SHALL fail with an error-tag of
               'access-denied' and an error-app-tag of
               'lne-not-managed'.";
          }
        }
      }
    }

    // augment statements

    augment "/if:interfaces/if:interface" {
      description
        "Add a node for the identification of the logical network
         element associated with an interface. Applies to
         interfaces that can be assigned on a per logical network
         element basis.

         Note that a standard error will be returned if the
         identified leafref isn't present.  If an interfaces
         cannot be assigned for any other reason, the operation
         SHALL fail with an error-tag of 'operation-failed' and an
         error-app-tag of 'lne-assignment-failed'.  A meaningful
         error-info that indicates the source of the assignment
         failure SHOULD also be provided.";
```

```
      leaf bind-lne-name {
        type leafref {
          path
           "/logical-network-elements/logical-network-element/name";
        }
        description
          "Logical network element ID to which interface is bound.";
      }
    }

    // notification statements

    notification bind-lne-name-failed {
      description
        "Indicates an error in the association of an interface to an
         LNE. Only generated after success is initially returned when
         bind-lne-name is set.";
      leaf name {
        type leafref {
          path "/if:interfaces/if:interface/if:name";
        }
        mandatory true;
        description
          "Contains the interface name associated with the
           failure.";
      }
      leaf bind-lne-name {
        type leafref {
          path "/if:interfaces/if:interface/lne:bind-lne-name";
        }
        mandatory true;
        description
          "Contains the bind-lne-name associated with the
           failure.";
      }
      leaf error-info {
        type string;
        description
          "Optionally, indicates the source of the assignment
           failure.";
      }
    }
  }
  <CODE ENDS>
```

7. References

7.1. Normative References

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-10 (work in progress), January 2018.

[I-D.ietf-netmod-rfc7223bis]
Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-rfc7223bis-03 (work in progress), January 2018.

[I-D.ietf-netmod-schema-mount]
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-08 (work in progress), October 2017.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://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, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[I-D.ietf-netmod-entity]
Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", draft-ietf-netmod-entity-08 (work in progress), January 2018.

[I-D.ietf-netmod-rfc8022bis]
Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", draft-ietf-netmod-rfc8022bis-11 (work in progress), January 2018.

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-06 (work in progress), February 2018.

[I-D.ietf-rtgwg-device-model]
Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Logical Organization", draft-ietf-rtgwg-device-model-02 (work in progress), March 2017.

[I-D.ietf-rtgwg-ni-model]
Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Network Instances", draft-ietf-rtgwg-ni-model-11 (work in progress), March 2018.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

Appendix A. Acknowledgments

The Routing Area Yang Architecture design team members included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Lou Berger, Qin Wu, Rob Shakir, Stephane Litkowski, and Yan Gang. Useful review comments were also received by Martin Bjorklund, John Scudder, Dan Romascanu and Taylor Yu.

This document was motivated by, and derived from, [I-D.ietf-rtgwg-device-model].

The RFC text was produced using Marshall Rose’s xml2rfc tool.

Thanks to Alvaro Retana for IESG review.

Appendix B. Examples

The following subsections provide example uses of LNEs.

B.1. Example: Host Device Managed LNE

This section describes an example of the LNE model using schema mount to achieve the parent management. An example device supports multiple instances of LNEs (logical routers), each of which supports features of layer 2 and layer 3 interfaces [I-D.ietf-netmod-rfc7223bis], routing information base [I-D.ietf-netmod-rfc8022bis], and OSPF protocol. Each of these features is specified by a YANG model, and they are combined using YANG Schema Mount as shown below. Not all possible mounted modules are shown. For example implementations could also mount the model defined in [I-D.ietf-netmod-entity].

```
module: ietf-logical-network-element
   +--rw logical-network-elements
      +--rw logical-network-element* [name]
         +--rw name           string
         +--mp root
            +--ro yanglib:modules-state/
            |  +--ro module-set-id    string
            |  +--ro module* [name revision]
            |     +--ro name                yang:yang-identifier
            +--rw sys:system/
            |  +--rw contact?          string
            |  +--rw hostname?         inet:domain-name
            |  +--rw authentication {authentication}?
            |     +--rw user-authentication-order*   identityref
            |     +--rw user* [name] {local-users}?
            |        +--rw name              string
            |        +--rw password?         ianach:crypt-hash
            |        +--rw authorized-key* [name]
            |           +--rw name         string
            |           +--rw algorithm    string
            |           +--rw key-data     binary
            +--ro sys:system-state/
            |     ...
            +--rw rt:routing/
            |  +--rw router-id?                yang:dotted-quad
            |  +--rw control-plane-protocols
            |     +--rw control-plane-protocol* [type name]
            |        +--rw ospf:ospf/
            |          +--rw areas
```

```
               |             +--rw area* [area-id]
               |                +--rw interfaces
               |                   +--rw interface* [name]
               |                      +--rw name if:interface-ref
               |                      +--rw cost?   uint16
               +--rw if:interfaces/
                  +--rw interface* [name]
                     +--rw name            string
                     +--rw ip:ipv4!/
                     |  +--rw address* [ip]
                     |  ...

   module: ietf-interfaces
      +--rw interfaces
         +--rw interface* [name]
            +--rw name                          string
            +--rw lne:bind-lne-name?            string
            +--ro oper-status        enumeration

   module: ietf-yang-library
      +--ro modules-state
         +--ro module-set-id    string
         +--ro module* [name revision]
            +--ro name                yang:yang-identifier

   module: ietf-system
      +--rw system
      |  +--rw contact?          string
      |  +--rw hostname?         inet:domain-name
      |  +--rw authentication {authentication}?
      |     +--rw user-authentication-order*   identityref
      |     +--rw user* [name] {local-users}?
      |        +--rw name              string
      |        +--rw password?         ianach:crypt-hash
      |        +--rw authorized-key* [name]
      |           +--rw name         string
      |           +--rw algorithm    string
      |           +--rw key-data     binary
      +--ro system-state
         +--ro platform
         |  +--ro os-name?      string
         |  +--ro os-release?   string
```

To realize the above schema, the example device implements the following schema mount instance:

```
"ietf-yang-schema-mount:schema-mounts": {
  "mount-point": [
    {
      "module": "ietf-logical-network-element",
      "label": "root",
      "shared-schema":  {}
    }
  ]
}
```

By using the implementation of the YANG schema mount, an operator can create instances of logical routers. An interface can be assigned to a logical router, so that the logical router has the permission to access this interface. The OSPF protocol can then be enabled on this assigned interface.

For this implementation, a parent management session has access to the schemas of both the parent hosting system and the child logical routers. In addition, each child logical router can grant its own management sessions, which have the following schema:

```
   module: ietf-yang-library
      +--ro modules-state
         +--ro module-set-id    string
         +--ro module* [name revision]
            +--ro name                yang:yang-identifier

   module: ietf-system
      +--rw system
      |  +--rw contact?          string
      |  +--rw hostname?         inet:domain-name
      |  +--rw authentication {authentication}?
      |     +--rw user-authentication-order*   identityref
      |     +--rw user* [name] {local-users}?
      |        +--rw name              string
      |        +--rw password?         ianach:crypt-hash
      |        +--rw authorized-key* [name]
      |           +--rw name         string
      |           +--rw algorithm    string
      |           +--rw key-data     binary
      +--ro system-state
         +--ro platform
            +--ro os-name?      string
            +--ro os-release?   string

   module: ietf-routing
      rw-- routing
         +--rw router-id?                 yang:dotted-quad
         +--rw control-plane-protocols
            +--rw control-plane-protocol* [type name]
               +--rw ospf:ospf/
                 +--rw areas
                    +--rw area* [area-id]
                       +--rw interfaces
                          +--rw interface* [name]
                             +--rw name    if:interface-ref
                             +--rw cost?   uint16

   module: ietf-interfaces
      +--rw interfaces
         +--rw interface* [name]
            +--rw name                        string
            +--ro oper-status        enumeration
```

B.1.1. Configuration Data

The following shows an example where two customer specific LNEs are configured:

```
{
  "ietf-logical-network-element:logical-network-elements": {
    "logical-network-element": [
      {
        "name": "cust1",
        "root": {
          "ietf-system:system": {
            "authentication": {
              "user": [
                {
                  "name": "john",
                  "password": "$0$password"
                }
              ]
            }
          },
          "ietf-routing:routing": {
            "router-id": "192.0.2.1",
            "control-plane-protocols": {
              "control-plane-protocol": [
                {
                  "type": "ietf-routing:ospf",
                  "name": "1",
                  "ietf-ospf:ospf": {
                    "af": "ipv4",
                    "areas": {
                      "area": [
                        {
                          "area-id": "203.0.113.1",
                          "interfaces": {
                            "interface": [
                              {
                                "name": "eth1",
                                "cost": 10
                              }
                            ]
                          }
                        }
                      ]
                    }
                  }
                }
              ]
            }
          },
          "ietf-interfaces:interfaces": {
            "interfaces": {
              "interface": [
```

```
                {
                  "name": "eth1",
                  "ip:ipv4": {
                    "address": [
                      {
                        "ip": "192.0.2.11",
                        "prefix-length": 24,
                      }
                    ]
                  },
                  "ip:ipv6": {
                    "address": [
                      {
                        "ip": "2001:db8:0:2::11",
                        "prefix-length": 64,
                      }
                    ]
                  }
                }
              ]
            }
          }
        }
      },
      {
        "name": "cust2",
        "root": {
          "ietf-system:system": {
            "authentication": {
              "user": [
                {
                  "name": "john",
                  "password": "$0$password"
                }
              ]
            }
          }
          "ietf-routing:routing": {
            "router-id": "192.0.2.2",
            "control-plane-protocols": {
              "control-plane-protocol": [
                {
                  "type": "ietf-routing:ospf",
                  "name": "1",
                  "ietf-ospf:ospf": {
                    "af": "ipv4",
                    "areas": {
                      "area": [
```

```
                          {
                            "area-id": "203.0.113.1",
                            "interfaces": {
                              "interface": [
                                {
                                  "name": "eth1",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            },
            "ietf-interfaces:interfaces": {
              "interfaces": {
                {
                  "name": "eth1",
                  "ip:ipv4": {
                    "address": [
                      {
                        "ip": "192.0.2.11",
                        "prefix-length": 24,
                      }
                    ]
                  }
                }
              ]
            }
          }
        }
      ]
    },

    "ietf-interfaces:interfaces": {
      "interfaces": {
        "interface": [
          {
            "name": "eth0",
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.10",
                  "prefix-length": 24,
```

```
                }
              ]
            },
            "ip:ipv6": {
              "address": [
                {
                  "ip": "2001:db8:0:2::10",
                  "prefix-length": 64,
                }
              ]
            }
          },
          {
            "name": "cust1:eth1",
            "lne:bind-lne-name": "cust1"
          },
          {
            "name": "cust2:eth1",
            "lne:bind-lne-name": "cust2"
          }
        ]
      }
    },

    "ietf-system:system": {
      "authentication": {
        "user": [
          {
            "name": "root",
            "password": "$0$password"
          }
        ]
      }
    }
  }
```

B.1.2. State Data

The following shows possible state data associated the above configuration data:

```
  {
    "ietf-logical-network-element:logical-network-elements": {
      "logical-network-element": [
        {
          "name": "cust1",
          "root": {
            "ietf-yang-library:modules-state": {
```

```
              "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
              "module": [
                {
                  "name": "iana-if-type",
                  "revision": "2014-05-08",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:iana-if-type",
                  "conformance-type": "import"
                },
                {
                  "name": "ietf-inet-types",
                  "revision": "2013-07-15",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-inet-types",
                  "conformance-type": "import"
                },
                {
                  "name": "ietf-interfaces",
                  "revision": "2014-05-08",
                  "feature": [
                    "arbitrary-names",
                    "pre-provisioning"
                  ],
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-interfaces",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-ip",
                  "revision": "2014-06-16",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-ip",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-ospf",
                  "revision": "2018-03-03",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-ospf",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-routing",
                  "revision": "2018-03-13",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-routing",
                  "conformance-type": "implement"
                },
```

```
          {
            "name": "ietf-system",
            "revision": "2014-08-06",
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-system",
            "conformance-type": "implement"
          },
          {
            "name": "ietf-yang-library",
            "revision": "2016-06-21",
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-yang-library",
            "conformance-type": "implement"
          },
          {
            "name": "ietf-yang-types",
            "revision": "2013-07-15",
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-yang-types",
            "conformance-type": "import"
          }
        ]
      },
      "ietf-system:system-state": {
        "ietf-system:system-state": {
          "platform": {
            "os-name": "NetworkOS"
          }
        }
      },
      "ietf-routing:routing": {
        "router-id": "192.0.2.1",
        "control-plane-protocols": {
          "control-plane-protocol": [
            {
              "type": "ietf-routing:ospf",
              "name": "1",
              "ietf-ospf:ospf": {
                "af": "ipv4",
                "areas": {
                  "area": [
                    {
                      "area-id": "203.0.113.1",
                      "interfaces": {
                        "interface": [
                          {
                            "name": "eth1",
                            "cost": 10
```

```
                                  }
                                ]
                              }
                            }
                          ]
                        }
                      }
                    }
                  ]
                }
              },
              "ietf-interfaces:interfaces": {
                "interfaces": {
                  "interface": [
                    {
                      "name": "eth1",
                      "type": "iana-if-type:ethernetCsmacd",
                      "oper-status": "up",
                      "phys-address": "00:01:02:A1:B1:C1",
                      "statistics": {
                        "discontinuity-time": "2017-06-26T12:34:56-05:00"
                      },
                      "ip:ipv4": {
                        "address": [
                          {
                            "ip": "192.0.2.11",
                            "prefix-length": 24,
                          }
                        ]
                      }
                    }
                  ]
                }
              }
            },
            {
              "name": "cust2",
              "root": {
                "ietf-yang-library:modules-state": {
                  "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
                  "module": [
                    {
                      "name": "iana-if-type",
                      "revision": "2014-05-08",
                      "namespace":
                      "urn:ietf:params:xml:ns:yang:iana-if-type",
                      "conformance-type": "import"
```

```
                },
                {
                  "name": "ietf-inet-types",
                  "revision": "2013-07-15",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-inet-types",
                  "conformance-type": "import"
                },
                {
                  "name": "ietf-interfaces",
                  "revision": "2014-05-08",
                  "feature": [
                    "arbitrary-names",
                    "pre-provisioning"
                  ],
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-interfaces",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-ip",
                  "revision": "2014-06-16",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-ip",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-ospf",
                  "revision": "2018-03-03",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-ospf",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-routing",
                  "revision": "2018-03-13",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-routing",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-system",
                  "revision": "2014-08-06",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-system",
                  "conformance-type": "implement"
                },
                {
```

```
                "name": "ietf-yang-library",
                "revision": "2016-06-21",
                "namespace":
                "urn:ietf:params:xml:ns:yang:ietf-yang-library",
                "conformance-type": "implement"
              },
              {
                "name": "ietf-yang-types",
                "revision": "2013-07-15",
                "namespace":
                "urn:ietf:params:xml:ns:yang:ietf-yang-types",
                "conformance-type": "import"
              }
            ]
          }
          "ietf-system:system-state": {
            "ietf-system:system-state": {
              "platform": {
                "os-name": "NetworkOS"
              }
            }
          },
          "ietf-routing:routing": {
            "router-id": "192.0.2.2",
            "control-plane-protocols": {
              "control-plane-protocol": [
                {
                  "type": "ietf-routing:ospf",
                  "name": "1",
                  "ietf-ospf:ospf": {
                    "af": "ipv4",
                    "areas": {
                      "area": [
                        {
                          "area-id": "203.0.113.1",
                          "interfaces": {
                            "interface": [
                              {
                                "name": "eth1",
                                "cost": 10
                              }
                            ]
                          }
                        }
                      ]
                    }
                  }
                }
```

```
              ]
            }
          }
          "ietf-interfaces:interfaces": {
            "interfaces": {
              {
                "name": "eth1",
                "type": "iana-if-type:ethernetCsmacd",
                "oper-status": "up",
                "phys-address": "00:01:02:A1:B1:C2",
                "statistics": {
                  "discontinuity-time": "2017-06-26T12:34:56-05:00"
                },
                "ip:ipv4": {
                  "address": [
                    {
                      "ip": "192.0.2.11",
                      "prefix-length": 24,
                    }
                  ]
                }
              }
            ]
          }
        }
      }
    ]
  },

  "ietf-interfaces:interfaces": {
    "interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "phys-address": "00:01:02:A1:B1:C0",
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.10",
                "prefix-length": 24,
              }
            ]
          }
```

```
          },
          {
            "name": "cust1:eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C1",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            }
          },
          {
            "name": "cust2:eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C2",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            }
          }
        ]
      }
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    },

    "ietf-yang-library:modules-state": {
      "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
      "module": [
        {
          "name": "iana-if-type",
          "revision": "2014-05-08",
          "namespace":
          "urn:ietf:params:xml:ns:yang:iana-if-type",
          "conformance-type": "import"
        },
        {
          "name": "ietf-inet-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-inet-types",
          "conformance-type": "import"
        },
        {
          "name": "ietf-interfaces",
```

```
          "revision": "2014-05-08",
          "feature": [
            "arbitrary-names",
            "pre-provisioning"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-interfaces",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ip",
          "revision": "2014-06-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-ip",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-logical-network-element",
          "revision": "2017-03-13",
          "feature": [
            "bind-lne-name"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-logical-network-element",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ospf",
          "revision": "2018-03-03",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-ospf",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-routing",
          "revision": "2018-03-13",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-routing",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement"
        },
        {
```

```
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-schema-mount",
          "revision": "2017-05-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-yang-schema-mount:schema-mounts": {
      "mount-point": [
        {
          "module": "ietf-logical-network-element",
          "label": "root",
          "shared-schema": {}
        }
      ]
    }
  }
```

B.2. Example: Self Managed LNE

This section describes an example of the LNE model using schema mount to achieve child independent management. An example device supports multiple instances of LNEs (logical routers), each of them has the features of layer 2 and layer 3 interfaces [I-D.ietf-netmod-rfc7223bis], routing information base [I-D.ietf-netmod-rfc8022bis], and the OSPF protocol. Each of these features is specified by a YANG model, and they are put together by the YANG Schema Mount as following:

```
         module: ietf-logical-network-element
      +--rw logical-network-elements
         +--rw logical-network-element* [name]
            +--rw name           string
            +--mp root
               // The internal modules of the LNE are not visible to
               // the parament management.
               // The child manages its modules, including ietf-routing
               // and ietf-interfaces

   module: ietf-interfaces
      +--rw interfaces
         +--rw interface* [name]
            +--rw name                          string
            +--rw lne:bind-lne-name?            string
            +--ro oper-status         enumeration

   module: ietf-yang-library
      +--ro modules-state
         +--ro module-set-id    string
         +--ro module* [name revision]
            +--ro name                yang:yang-identifier

   module: ietf-system
      +--rw system
      |  +--rw contact?          string
      |  +--rw hostname?         inet:domain-name
      |  +--rw authentication {authentication}?
      |     +--rw user-authentication-order*   identityref
      |     +--rw user* [name] {local-users}?
      |        +--rw name              string
      |        +--rw password?         ianach:crypt-hash
      |        +--rw authorized-key* [name]
      |           +--rw name          string
      |           +--rw algorithm     string
      |           +--rw key-data      binary
      +--ro system-state
         +--ro platform
         |  +--ro os-name?      string
         |  +--ro os-release?   string
```

To realize the above schema, the device implements the following schema mount instance:

```
"ietf-yang-schema-mount:schema-mounts": {
  "mount-point": [
    {
      "module": "ietf-logical-network-element",
      "label": "root",
      "inline": {}
    }
  ]
}
```

By using the implementation of the YANG schema mount, an operator can create instances of logical routers, each with their logical router specific in-line modules. An interface can be assigned to a logical router, so that the logical router has the permission to access this interface. The OSPF protocol can then be enabled on this assigned interface. Each logical router independently manages its own set of modules, which may or may not be the same as other logical routers. The following is an example of schema set implemented on one particular logical router:

```
   module: ietf-yang-library
      +--ro modules-state
         +--ro module-set-id    string
         +--ro module* [name revision]
            +--ro name                yang:yang-identifier

   module: ietf-system
      +--rw system
      |  +--rw contact?          string
      |  +--rw hostname?         inet:domain-name
      |  +--rw authentication {authentication}?
      |     +--rw user-authentication-order*   identityref
      |     +--rw user* [name] {local-users}?
      |        +--rw name              string
      |        +--rw password?         ianach:crypt-hash
      |        +--rw authorized-key* [name]
      |           +--rw name         string
      |           +--rw algorithm    string
      |           +--rw key-data     binary
      +--ro system-state
         +--ro platform
         |  +--ro os-name?      string
         |  +--ro os-release?   string

   module: ietf-routing
      +--rw routing
         +--rw router-id?                 yang:dotted-quad
         +--rw control-plane-protocols
            +--rw control-plane-protocol* [type name]
               +--rw ospf:ospf/
                  +--rw areas
                    +--rw area* [area-id]
                       +--rw interfaces
                          +--rw interface* [name]
                             +--rw name    if:interface-ref
                             +--rw cost?   uint16

   module: ietf-interfaces
      +--rw interfaces
         +--rw interface* [name]
            +--rw name                        string
            +--ro oper-status        enumeration
```

B.2.1. Configuration Data

Each of the child virtual routers is managed through its own sessions and configuration data.

B.2.1.1. Logical Network Element 'vnf1'

The following shows an example configuration data for the LNE name "vnf1":

```
{
  "ietf-system:system": {
    "authentication": {
      "user": [
        {
          "name": "john",
          "password": "$0$password"
        }
      ]
    }
  },
  "ietf-routing:routing": {
    "router-id": "192.0.2.1",
    "control-plane-protocols": {
      "control-plane-protocol": [
        {
          "type": "ietf-routing:ospf",
          "name": "1",
          "ietf-ospf:ospf": {
            "af": "ipv4",
            "areas": {
              "area": [
                {
                  "area-id": "203.0.113.1",
                  "interfaces": {
                    "interface": [
                      {
                        "name": "eth1",
                        "cost": 10
                      }
                    ]
                  }
                }
              ]
            }
          }
        }
      ]
    }
  },
  "ietf-interfaces:interfaces": {
    "interfaces": {
      "interface": [
```

```
          {
            "name": "eth1",
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            }
          }
        ]
      }
    }
  }
```

B.2.1.2. Logical Network Element 'vnf2'

The following shows an example configuration data for the LNE name "vnf2":

```
{
  "ietf-system:system": {
    "authentication": {
      "user": [
        {
          "name": "john",
          "password": "$0$password"
        }
      ]
    }
  },
  "ietf-routing:routing": {
    "router-id": "192.0.2.2",
    "control-plane-protocols": {
      "control-plane-protocol": [
        {
          "type": "ietf-routing:ospf",
          "name": "1",
          "ietf-ospf:ospf": {
            "af": "ipv4",
            "areas": {
              "area": [
                {
                  "area-id": "203.0.113.1",
                  "interfaces": {
                    "interface": [
                      {
```

```
                          "name": "eth1",
                          "cost": 10
                        }
                      ]
                    }
                  }
                ]
              }
            }
          ]
        }
      },
      "ietf-interfaces:interfaces": {
    "interfaces": {
          "interface": [
            {
              "name": "eth1",
              "ip:ipv4": {
                "address": [
                  {
                    "ip": "192.0.2.11",
                    "prefix-length": 24,
                  }
                ]
              }
            }
          ]
        }
      }
    }
```

B.2.2. State Data

The following sections shows possible state data associated the above configuration data. Note that there are three views: the host device's, and each LNE's.

B.2.2.1. Host Device

The following shows state data for the device hosting the example LNEs:

```
{
  "ietf-logical-network-element:logical-network-elements": {
    "logical-network-element": [
      {
        "name": "vnf1",
```

```
          "root": {
          }
        },
        {
          "name": "vnf2",
          "root": {
          }
        }
      ]
    },

    "ietf-interfaces:interfaces": {
      "interfaces": {
        "interface": [
          {
            "name": "eth0",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C0",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.10",
                  "prefix-length": 24,
                }
              ]
            }
          },
          {
            "name": "vnf1:eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C1",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            }
          },
          {
            "name": "vnf2:eth2",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C2",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            }
```

```
          }
        ]
      }
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    },

    "ietf-yang-library:modules-state": {
      "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
      "module": [
        {
          "name": "iana-if-type",
          "revision": "2014-05-08",
          "namespace":
          "urn:ietf:params:xml:ns:yang:iana-if-type",
          "conformance-type": "import"
        },
        {
          "name": "ietf-inet-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-inet-types",
          "conformance-type": "import"
        },
        {
          "name": "ietf-interfaces",
          "revision": "2014-05-08",
          "feature": [
            "arbitrary-names",
            "pre-provisioning"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-interfaces",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ip",
          "revision": "2014-06-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-ip",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-logical-network-element",
```

```
          "revision": "2017-03-13",
          "feature": [
            "bind-lne-name"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-logical-network-element",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-schema-mount",
          "revision": "2017-05-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-yang-schema-mount:schema-mounts": {
      "mount-point": [
        {
          "module": "ietf-logical-network-element",
          "label": "root",
          "inline": {}
        }
      ]
    }
```

```
   }
```

B.2.2.2. Logical Network Element 'vnf1'

The following shows state data for the example LNE with name "vnf1":

```
   {
     "ietf-yang-library:modules-state": {
       "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
       "module": [
         {
           "name": "iana-if-type",
           "revision": "2014-05-08",
           "namespace":
           "urn:ietf:params:xml:ns:yang:iana-if-type",
           "conformance-type": "import"
         },
         {
           "name": "ietf-inet-types",
           "revision": "2013-07-15",
           "namespace":
           "urn:ietf:params:xml:ns:yang:ietf-inet-types",
           "conformance-type": "import"
         },
         {
           "name": "ietf-interfaces",
           "revision": "2014-05-08",
           "feature": [
             "arbitrary-names",
             "pre-provisioning"
           ],
           "namespace":
           "urn:ietf:params:xml:ns:yang:ietf-interfaces",
           "conformance-type": "implement"
         },
         {
           "name": "ietf-ip",
           "revision": "2014-06-16",
           "namespace":
           "urn:ietf:params:xml:ns:yang:ietf-ip",
           "conformance-type": "implement"
         },
         {
           "name": "ietf-ospf",
           "revision": "2018-03-03",
           "namespace":
           "urn:ietf:params:xml:ns:yang:ietf-ospf",
           "conformance-type": "implement"
```

```
        },
        {
          "name": "ietf-routing",
          "revision": "2018-03-13",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-routing",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    },

    "ietf-routing:routing": {
      "router-id": "192.0.2.1",
      "control-plane-protocols": {
        "control-plane-protocol": [
          {
            "type": "ietf-routing:ospf",
            "name": "1",
            "ietf-ospf:ospf": {
              "af": "ipv4",
              "areas": {
```

```
                "area": [
                  {
                    "area-id": "203.0.113.1",
                    "interfaces": {
                      "interface": [
                        {
                          "name": "eth1",
                          "cost": 10
                        }
                      ]
                    }
                  }
                ]
              }
            }
          }
        ]
      }
    },

    "ietf-interfaces:interfaces": {
      "interfaces": {
        "interface": [
          {
            "name": "eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C1",
            "statistics": {
              "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            }
          }
        ]
      }
    }
  }
```

B.2.2.3. Logical Network Element 'vnf2'

The following shows state data for the example LNE with name "vnf2":

```
{
  "ietf-yang-library:modules-state": {
    "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
    "module": [
      {
        "name": "iana-if-type",
        "revision": "2014-05-08",
        "namespace":
        "urn:ietf:params:xml:ns:yang:iana-if-type",
        "conformance-type": "import"
      },
      {
        "name": "ietf-inet-types",
        "revision": "2013-07-15",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-inet-types",
        "conformance-type": "import"
      },
      {
        "name": "ietf-interfaces",
        "revision": "2014-05-08",
        "feature": [
          "arbitrary-names",
          "pre-provisioning"
        ],
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-interfaces",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-ip",
        "revision": "2014-06-16",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-ip",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-ospf",
        "revision": "2018-03-03",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-ospf",
        "conformance-type": "implement"
      },
      {
```

```
          "name": "ietf-routing",
          "revision": "2018-03-13",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-routing",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    },

    "ietf-routing:routing": {
      "router-id": "192.0.2.2",
      "control-plane-protocols": {
        "control-plane-protocol": [
          {
            "type": "ietf-routing:ospf",
            "name": "1",
            "ietf-ospf:ospf": {
              "af": "ipv4",
              "areas": {
                "area": [
                  {
```

```
                    "area-id": "203.0.113.1",
                    "interfaces": {
                      "interface": [
                        {
                          "name": "eth1",
                          "cost": 10
                        }
                      ]
                    }
                  }
                ]
              }
            }
          }
        ]
      }
    },

    "ietf-interfaces:interfaces": {
      "interfaces": {
        "interface": [
          {
            "name": "eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C2",
            "statistics": {
              "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            }
          }
        ]
      }
    }
  }
```

Authors' Addresses

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Dean Bogdanovic

Email: ivandean@gmail.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 20, 2018

L. Berger
LabN Consulting, L.L.C.
C. Hopps
Deutsche Telekom
A. Lindem
Cisco Systems
D. Bogdanovic

X. Liu
Jabil

March 19, 2018

YANG Model for Network Instances
draft-ietf-rtgwg-ni-model-12

Abstract

This document defines a network instance module. This module can be used to manage the virtual resource partitioning that may be present on a network device. Examples of common industry terms for virtual resource partitioning are Virtual Routing and Forwarding (VRF) instances and Virtual Switch Instances (VSIs).

The YANG model in this document conforms to the Network Management Datastore Architecture defined in I-D.ietf-netmod-revised-datastores.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 20, 2018.

Table of Contents

1. Introduction

This document defines the second of two new modules that are defined to support the configuration and operation of network-devices that allow for the partitioning of resources from both, or either, management and networking perspectives. Both leverage the YANG functionality enabled by YANG Schema Mount [I-D.ietf-netmod-schema-mount].

The YANG model in this document conforms to the Network Management Datastore Architecture defined in the [I-D.ietf-netmod-revised-datastores].

The first form of resource partitioning provides a logical partitioning of a network device where each partition is separately managed as essentially an independent network element which is 'hosted' by the base network device. These hosted network elements are referred to as logical network elements, or LNEs, and are supported by the logical-network-element module defined in [I-D.ietf-rtgwg-lne-model]. That module is used to identify LNEs and associate resources from the network-device with each LNE. LNEs themselves are represented in YANG as independent network devices; each accessed independently. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric.

The second form, which is defined in this document, provides support for what is commonly referred to as Virtual Routing and Forwarding (VRF) instances as well as Virtual Switch Instances (VSI), see [RFC4026] and [RFC4664]. In this form of resource partitioning, multiple control plane and forwarding/bridging instances are provided by and managed via a single (physical or logical) network device. This form of resource partitioning is referred to as a Network Instance and is supported by the network-instance module defined below. Configuration and operation of each network-instance is always via the network device and the network-instance module.

One notable difference between the LNE model and the NI model is that the NI model provides a framework for VRF and VSI management. This document envisions the separate definition of VRF and VSI, i.e., L3 and L2 VPN, technology specific models. An example of such can be found in the emerging L3VPN model defined in [I-D.ietf-bess-l3vpn-yang] and the examples discussed below.

1.1. Terminology

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

Readers are expected to be familiar with terms and concepts of YANG [RFC7950] and YANG Schema Mount [I-D.ietf-netmod-schema-mount].

This document uses the graphical representation of data models defined in [I-D.ietf-netmod-yang-tree-diagrams].

2. Overview

In this document, we consider network devices that support protocols and functions defined within the IETF, e.g, routers, firewalls, and hosts. Such devices may be physical or virtual, e.g., a classic router with custom hardware or one residing within a server-based virtual machine implementing a virtual network function (VNF). Each device may sub-divide their resources into logical network elements (LNEs) each of which provides a managed logical device. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric. Each LNE may also support virtual routing and forwarding (VRF) and virtual switching instance (VSI) functions, which are referred to below as a network instances (NIs). This breakdown is represented in Figure 1.

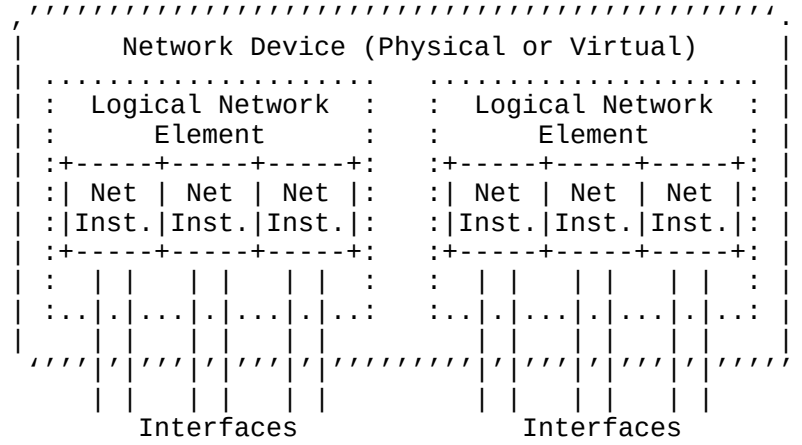


Figure 1: Module Element Relationships

A model for LNEs is described in [I-D.ietf-rtgwg-lne-model] and the model for NIs is covered in this document in Section 3.

The current interface management model [I-D.ietf-netmod-rfc7223bis] is impacted by the definition of LNEs and NIs. This document and [I-D.ietf-rtgwg-lne-model] define augmentations to the interface module to support LNEs and NIs.

The network instance model supports the configuration of VRFs and VSIs. Each instance is supported by information that relates to the device, for example the route target used when advertising VRF routes via the mechanisms defined in [RFC4364], and information that relates to the internal operation of the NI, for example for routing protocols [I-D.ietf-netmod-rfc8022bis] and OSPF [I-D.ietf-ospf-yang].

This document defines the network-instance module that provides a basis for the management of both types of information.

NI information that relates to the device, including the assignment of interfaces to NIs, is defined as part of this document. The defined module also provides a placeholder for the definition of NI-technology specific information both at the device level and for NI internal operation. Information related to NI internal operation is supported via schema mount [I-D.ietf-netmod-schema-mount] and mounting appropriate modules under the mount point. Well known mount points are defined for L3VPN, L2VPN, and L2+L3VPN NI types.

3. Network Instances

The network instance container is used to represent virtual routing and forwarding instances (VRFs) and virtual switching instances (VSIs). VRFs and VSIs are commonly used to isolate routing and switching domains, for example to create virtual private networks, each with their own active protocols and routing/switching policies. The model supports both core/provider and virtual instances. Core/provider instance information is accessible at the top level of the server, while virtual instance information is accessible under the root schema mount points.

```
module: ietf-network-instance
  +--rw network-instances
     +--rw network-instance* [name]
        +--rw name           string
        +--rw enabled?       boolean
        +--rw description?   string
        +--rw (ni-type)?
        +--rw (root-type)
           +--:(vrf-root)
           |  +--mp vrf-root
           +--:(vsi-root)
           |  +--mp vsi-root
           +--:(vv-root)
              +--mp vv-root
augment /if:interfaces/if:interface:
  +--rw bind-ni-name?   -> /network-instances/network-instance/name
augment /if:interfaces/if:interface/ip:ipv4:
  +--rw bind-ni-name?   -> /network-instances/network-instance/name
augment /if:interfaces/if:interface/ip:ipv6:
  +--rw bind-ni-name?   -> /network-instances/network-instance/name

notifications:
  +---n bind-ni-name-failed
     +--ro name          -> /if:interfaces/interface/name
     +--ro interface
     |  +--ro bind-ni-name?
     |                   -> /if:interfaces/interface/ni:bind-ni-name
     +--ro ipv4
     |  +--ro bind-ni-name?
     |          -> /if:interfaces/interface/ip:ipv4/ni:bind-ni-name
     +--ro ipv6
     |  +--ro bind-ni-name?
     |          -> /if:interfaces/interface/ip:ipv6/ni:bind-ni-name
     +--ro error-info?   string
```

A network instance is identified by a 'name' string. This string is used both as an index within the network-instance module and to associate resources with a network instance as shown above in the interface augmentation. The ni-type and root-type choice statements are used to support different types of L2 and L3 VPN technologies. The bind-ni-name-failed notification is used in certain failure cases.

3.1. NI Types and Mount Points

The network-instance module is structured to facilitate the definition of information models for specific types of VRFs and VSIs using augmentations. For example, the information needed to support

VPLS, VxLAN and EVPN based L2VPNs are likely to be quite different. Example models under development that could be restructured to take advantage on NIs include, for L3VPNs [I-D.ietf-bess-l3vpn-yang] and for L2VPNs [I-D.ietf-bess-l2vpn-yang].

Documents defining new YANG models for the support of specific types of network instances should augment the network instance module. The basic structure that should be used for such augmentations include a case statement, with containers for configuration and state data and finally, when needed, a type specific mount point. Generally ni types, are expected to not need to define type specific mount points, but rather reuse one of the well known mount point, as defined in the next section. The following is an example type specific augmentation:

```
 augment "/ni:network-instances/ni:network-instance/ni:ni-type" {
   case l3vpn {
     container l3vpn {
         ...
     }
     container l3vpn-state {
         ...
     }
   }
 }
```

3.1.1. Well Known Mount Points

YANG Schema Mount, [I-D.ietf-netmod-schema-mount], identifies mount points by name within a module. This definition allows for the definition of mount points whose schema can be shared across ni-types. As discussed above, ni-types largely differ in the configuration information needed in the core/top level instance to support the NI, rather than in the information represented within an NI. This allows the use of shared mount points across certain NI types.

The expectation is that there are actually very few different schema that need to be defined to support NIs on an implementation. In particular, it is expected that the following three forms of NI schema are needed, and each can be defined with a well known mount point that can be reused by future modules defining ni-types.

The three well known mount points are:

vrf-root
: vrf-root is intended for use with L3VPN type ni-types.

vsi-root
vsi-root is intended for use with L2VPN type ni-types.

vv-root
vv-root is intended for use with ni-types that simultaneously support L2VPN bridging and L3VPN routing capabilities.

Future model definitions should use the above mount points whenever possible. When a well known mount point isn't appropriate, a model may define a type specific mount point via augmentation.

3.1.2. NI Type Example

The following is an example of an L3VPN VRF using a hypothetical augmentation to the networking instance schema defined in [I-D.ietf-bess-l3vpn-yang]. More detailed examples can be found in Appendix B.

```
module: ietf-network-instance
  +--rw network-instances
     +--rw network-instance* [name]
        +--rw name           string
        +--rw enabled?       boolean
        +--rw description?   string
        +--rw (ni-type)?
        |  +--:(l3vpn)
        |     +--rw l3vpn:l3vpn
        |     |  ... // config data
        |     +--ro l3vpn:l3vpn-state
        |     |  ... // state data
        +--rw (root-type)
           +--:(vrf-root)
              +--mp vrf-root
                 +--rw rt:routing/
                 |  +--rw router-id?                 yang:dotted-quad
                 |  +--rw control-plane-protocols
                 |     +--rw control-plane-protocol* [type name]
                 |     +--rw ospf:ospf/
                 |          +--rw area* [area-id]
                 |             +--rw interfaces
                 |                +--rw interface* [name]
                 |                   +--rw name if:interface-ref
                 |                   +--rw cost?   uint16
                 +--ro if:interfaces@
                 |  ...
```

This shows YANG Routing Management [I-D.ietf-netmod-rfc8022bis] and YANG OSPF [I-D.ietf-ospf-yang] as mounted modules. The mounted

modules can reference interface information via a parent-reference to the containers defined in [I-D.ietf-netmod-rfc7223bis].

3.2. NIs and Interfaces

Interfaces are a crucial part of any network device's configuration and operational state. They generally include a combination of raw physical interfaces, link-layer interfaces, addressing configuration, and logical interfaces that may not be tied to any physical interface. Several system services, and layer 2 and layer 3 protocols may also associate configuration or operational state data with different types of interfaces (these relationships are not shown for simplicity). The interface management model is defined by [I-D.ietf-netmod-rfc7223bis].

As shown below, the network-instance module augments the existing interface management model by adding a name which is used on interface or sub-interface types to identify an associated network instance. Similarly, this name is also added for IPv4 and IPv6 types, as defined in [I-D.ietf-netmod-rfc7277bis].

The following is an example of envisioned usage. The interfaces container includes a number of commonly used components as examples:

```
module: ietf-interfaces
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name                          string
   |     +--rw ip:ipv4!
   |     |  +--rw ip:enabled?                      boolean
   |     |  +--rw ip:forwarding?                   boolean
   |     |  +--rw ip:mtu?                          uint16
   |     |  +--rw ip:address* [ip]
   |     |  |  +--rw ip:ip              inet:ipv4-address-no-zone
   |     |  |  +--rw (ip:subnet)
   |     |  |     +--:(ip:prefix-length)
   |     |  |     |  +--rw ip:prefix-length?   uint8
   |     |  |     +--:(ip:netmask)
   |     |  |        +--rw ip:netmask?         yang:dotted-quad
   |     |  +--rw ip:neighbor* [ip]
   |     |  |  +--rw ip:ip                   inet:ipv4-address-no-zone
   |     |  |  +--rw ip:link-layer-address yang:phys-address
   |     |  +--rw ni:bind-network-instance-name?   string
   |     +--rw ni:bind-network-instance-name?   string
```

The [I-D.ietf-netmod-rfc7223bis] defined interface model is structured to include all interfaces in a flat list, without regard to virtual instances (e.g., VRFs) supported on the device. The bind-

network-instance-name leaf provides the association between an interface and its associated NI (e.g., VRF or VSI). Note that as currently defined, to assign an interface to both an LNE and NI, the interface would first be assigned to the LNE using the mechanisms defined in [I-D.ietf-rtgwg-lne-model] and then within that LNE's interface module, the LNE's representation of that interface would be assigned to an NI.

3.3. Network Instance Management

Modules that may be used to represent network instance information will be available under the ni-type specific 'root' mount point. The "shared-schema" method defined in the "ietf-yang-schema-mount" module [I-D.ietf-netmod-schema-mount] MUST be used to identify accessible modules. A future version of this document could relax this requirement. Mounted modules SHOULD be defined with access, via the appropriate schema mount parent-references [I-D.ietf-netmod-schema-mount], to device resources such as interfaces. An implementation MAY choose to restrict parent referenced information to information related to a specific instance, e.g., only allowing references to interfaces that have a "bind-network-instance-name" which is identical to the instance's "name".

All modules that represent control-plane and data-plane information may be present at the 'root' mount point, and be accessible via paths modified per [I-D.ietf-netmod-schema-mount]. The list of available modules is expected to be implementation dependent, as is the method used by an implementation to support NIs.

For example, the following could be used to define the data organization of the example NI shown in Section 3.1.2:

```
  "ietf-yang-schema-mount:schema-mounts": {
    "mount-point": [
      {
        "module": "ietf-network-instance",
        "label": "vrf-root",
        "shared-schema": {
          "parent-reference": [
            "/*[namespace-uri() = 'urn:ietf:...:ietf-interfaces']"
          ]
        }
      }
    ]
  }
```

Module data identified according to the ietf-yang-schema-mount module will be instantiated under the mount point identified under "mount-

point". These modules will be able to reference information for nodes belonging to top-level modules that are identified under "parent-reference". Parent referenced information is available to clients via their top level paths only, and not under the associated mount point.

To allow a client to understand the previously mentioned instance restrictions on parent referenced information, an implementation MAY represent such restrictions in the "parent-reference" leaf-list. For example:

```
  "namespace": [
    {
      "prefix": "if",
      "uri": "urn:ietf:params:xml:ns:yang:ietf-interfaces"
    },
    {
      "prefix": "ni",
      "uri": "urn:ietf:params:xml:ns:yang:ietf-network-instance"
    }
  ],
  "mount-point": [
    {
      "module": "ietf-network-instance",
      "label": "vrf-root",
      "shared-schema": {
        "parent-reference": [
          "/if:interfaces/if:interface
             [ni:bind-network-instance-name = current()/../ni:name]",
          "/if:interfaces/if:interface/ip:ipv4
             [ni:bind-network-instance-name = current()/../ni:name]",
          "/if:interfaces/if:interface/ip:ipv6
             [ni:bind-network-instance-name = current()/../ni:name]"
        ]
      }
    }
  ],
```

The same such "parent-reference" restrictions for non-NMDA implementations can be represented based on the [RFC7223] and [RFC7277] as:

```
    "namespace": [
      {
        "prefix": "if",
        "uri": "urn:ietf:params:xml:ns:yang:ietf-interfaces"
      },
      {
        "prefix": "ni",
        "uri": "urn:ietf:params:xml:ns:yang:ietf-network-instance"
      }
    ],
    "mount-point": [
      {
        "module": "ietf-network-instance",
        "label": "vrf-root",
        "shared-schema": {
          "parent-reference": [
            "/if:interfaces/if:interface
               [ni:bind-network-instance-name = current()/../ni:name]",
            "/if:interfaces-state/if:interface
               [if:name = /if:interfaces/if:interface
                 [ni:bind-ni-name = current()/../ni:name]/if:name]",
            "/if:interfaces/if:interface/ip:ipv4
               [ni:bind-network-instance-name = current()/../ni:name]",
            "/if:interfaces-state/if:interface/ip:ipv4
               [if:name = /if:interfaces/if:interface/ip:ipv4
                 [ni:bind-ni-name = current()/../ni:name]/if:name]",
            "/if:interfaces/if:interface/ip:ipv6
               [ni:bind-network-instance-name = current()/../ni:name]",
            "/if:interfaces-state/if:interface/ip:ipv6
               [if:name = /if:interfaces/if:interface/ip:ipv4
                 [ni:bind-ni-name = current()/../ni:name]/if:name]"
          ]
        }
      }
    ],
```

3.4. Network Instance Instantiation

Network instances may be controlled by clients using existing list operations. When a list entry is created, a new instance is instantiated. The models mounted under an NI root are expected to be dependent on the server implementation. When a list entry is deleted, an existing network instance is destroyed. For more information, see [RFC7950] Section 7.8.6.

Once instantiated, host network device resources can be associated with the new NI. As previously mentioned, this document augments ietf-interfaces with the bind-ni-name leaf to support such

associations for interfaces. When a bind-ni-name is set to a valid NI name, an implementation MUST take whatever steps are internally necessary to assign the interface to the NI or provide an error message (defined below) with an indication of why the assignment failed. It is possible for the assignment to fail while processing the set operation, or after asynchronous processing. Error notification in the latter case is supported via a notification.

4. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are two different sets of security considerations to consider in the context of this document. One set is security related to information contained within mounted modules. The security considerations for mounted modules are not substantively changed based on the information being accessible within the context of an NI. For example, when considering the modules defined in [I-D.ietf-netmod-rfc8022bis], the security considerations identified in that document are equally applicable, whether those modules are accessed at a server’s root or under an NI instance’s root node.

The second area for consideration is information contained in the NI module itself. NI information represents network configuration and route distribution policy information. As such, the security of this information is important, but it is fundamentally no different than any other interface or routing configuration information that has already been covered in [I-D.ietf-netmod-rfc7223bis] and [I-D.ietf-netmod-rfc8022bis].

The vulnerable "config true" parameters and subtrees are the following:

/network-instances/network-instance: This list specifies the network instances and the related control plane protocols configured on a device.

/if:interfaces/if:interface/*/bind-network-instance-name: This leaf indicates the NI instance to which an interface is assigned.

Unauthorized access to any of these lists can adversely affect the routing subsystem of both the local device and the network. This may lead to network malfunctions, delivery of packets to inappropriate destinations and other problems.

5. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-network-instance

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
name:         ietf-network-instance
namespace:    urn:ietf:params:xml:ns:yang:ietf-network-instance
prefix:       ni
reference:    RFC XXXX
```

6. Network Instance Model

The structure of the model defined in this document is described by the YANG module below.

```
<CODE BEGINS> file "ietf-network-instance@2018-03-20.yang"
module ietf-network-instance {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-instance";
  prefix ni;

  // import some basic types

  import ietf-interfaces {
    prefix if;
    reference "draft-ietf-netmod-rfc7223bis: A YANG Data Model
               for Interface Management";
  }
  import ietf-ip {
    prefix ip;
```

```
      reference "draft-ietf-netmod-rfc7277bis: A YANG Data Model
                 for IP Management";
    }
    import ietf-yang-schema-mount {
      prefix yangmnt;
      reference "draft-ietf-netmod-schema-mount: YANG Schema Mount";
      // RFC Ed.: Please replace this draft name with the
      // corresponding RFC number
    }

    organization
      "IETF Routing Area (rtgwg) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
       WG List:  <mailto:rtgwg@ietf.org>

       Author:   Lou Berger
                 <mailto:lberger@labn.net>
       Author:   Christan Hopps
                 <mailto:chopps@chopps.org>
       Author:   Acee Lindem
                 <mailto:acee@cisco.com>
       Author:   Dean Bogdanovic
                 <mailto:ivandean@gmail.com>";
    description
      "This module is used to support multiple network instances
       within a single physical or virtual device.  Network
       instances are commonly known as VRFs (virtual routing
       and forwarding) and VSIs (virtual switching instances).

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

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    // RFC Ed.: replace XXXX with actual RFC number and remove
    // this note
    // RFC Ed.: please update TBD

    revision 2018-03-20 {
```

```
      description
        "Initial revision.";
      reference "RFC TBD";
    }

    // top level device definition statements

    container network-instances {
      description
        "Network instances each of which consists of a
         VRFs (virtual routing and forwarding) and/or
         VSIs (virtual switching instances).";
      reference "draft-ietf-rtgwg-rfc8022bis - A YANG Data Model
                 for Routing Management";
      list network-instance {
        key "name";
        description
          "List of network-instances.";
        leaf name {
          type string;
          mandatory true;
          description
            "device scoped identifier for the network
             instance.";
        }
        leaf enabled {
          type boolean;
          default "true";
          description
            "Flag indicating whether or not the network
             instance is enabled.";
        }
        leaf description {
          type string;
          description
            "Description of the network instance
             and its intended purpose.";
        }
        choice ni-type {
          description
            "This node serves as an anchor point for different types
             of network instances.  Each 'case' is expected to
             differ in terms of the information needed in the
             parent/core to support the NI, and may differ in their
             mounted schema definition. When the mounted schema is
             not expected to be the same for a specific type of NI
             a mount point should be defined.";
        }
```

```
        choice root-type {
          mandatory true;
          description
            "Well known mount points.";
          container vrf-root {
            description
              "Container for mount point.";
            yangmnt:mount-point "vrf-root" {
              description
                "Root for L3VPN type models. This will typically
                 not be an inline type mount point.";
            }
          }
          container vsi-root {
            description
              "Container for mount point.";
            yangmnt:mount-point "vsi-root" {
              description
                "Root for L2VPN type models. This will typically
                 not be an inline type mount point.";
            }
          }
          container vv-root {
            description
              "Container for mount point.";
            yangmnt:mount-point "vv-root" {
              description
                "Root models that support both L2VPN type bridging
                 and L3VPN type routing. This will typically
                 not be an inline type mount point.";
            }
          }
        }
      }
    }

    // augment statements

    augment "/if:interfaces/if:interface" {
      description
        "Add a node for the identification of the network
         instance associated with the information configured
         on a interface.

         Note that a standard error will be returned if the
         identified leafref isn't present.  If an interfaces cannot
         be assigned for any other reason, the operation SHALL fail
         with an error-tag of 'operation-failed' and an
```

```
          error-app-tag of 'ni-assignment-failed'.  A meaningful
          error-info that indicates the source of the assignment
          failure SHOULD also be provided.";
        leaf bind-ni-name {
          type leafref {
            path "/network-instances/network-instance/name";
          }
          description
            "Network Instance to which an interface is bound.";
        }
      }
      augment "/if:interfaces/if:interface/ip:ipv4" {
        description
          "Add a node for the identification of the network
           instance associated with the information configured
           on an IPv4 interface.

           Note that a standard error will be returned if the
           identified leafref isn't present.  If an interfaces cannot
           be assigned for any other reason, the operation SHALL fail
           with an error-tag of 'operation-failed' and an
           error-app-tag of 'ni-assignment-failed'.  A meaningful
           error-info that indicates the source of the assignment
           failure SHOULD also be provided.";
        leaf bind-ni-name {
          type leafref {
            path "/network-instances/network-instance/name";
          }
          description
            "Network Instance to which IPv4 interface is bound.";
        }
      }
      augment "/if:interfaces/if:interface/ip:ipv6" {
        description
          "Add a node for the identification of the network
           instance associated with the information configured
           on an IPv6 interface.

           Note that a standard error will be returned if the
           identified leafref isn't present.  If an interfaces cannot
           be assigned for any other reason, the operation SHALL fail
           with an error-tag of 'operation-failed' and an
           error-app-tag of 'ni-assignment-failed'.  A meaningful
           error-info that indicates the source of the assignment
           failure SHOULD also be provided.";
        leaf bind-ni-name {
          type leafref {
            path "/network-instances/network-instance/name";
```

```
        }
        description
          "Network Instance to which IPv6 interface is bound.";
      }
    }

    // notification statements

    notification bind-ni-name-failed {
      description
        "Indicates an error in the association of an interface to an
         NI. Only generated after success is initially returned when
         bind-ni-name is set.

         Note: some errors may need to be reported for multiple
         associations, e.g., a single error may need to be reported
         for an IPv4 and an IPv6 bind-ni-name.

         At least one container with a bind-ni-name leaf MUST be
         included in this notification.";
      leaf name {
        type leafref {
          path "/if:interfaces/if:interface/if:name";
        }
        mandatory true;
        description
          "Contains the interface name associated with the
          failure.";
      }
      container interface {
        description
          "Generic interface type.";
        leaf bind-ni-name {
          type leafref {
            path "/if:interfaces/if:interface/ni:bind-ni-name";
          }
          description
            "Contains the bind-ni-name associated with the
             failure.";
        }
      }
      container ipv4 {
        description
          "IPv4 interface type.";
        leaf bind-ni-name {
          type leafref {
            path "/if:interfaces/if:interface"
              + "/ip:ipv4/ni:bind-ni-name";
```

```
          }
          description
            "Contains the bind-ni-name associated with the
             failure.";
        }
      }
      container ipv6 {
        description
          "IPv6 interface type.";
        leaf bind-ni-name {
          type leafref {
            path "/if:interfaces/if:interface"
              + "/ip:ipv6/ni:bind-ni-name";
          }
          description
            "Contains the bind-ni-name associated with the
             failure.";
        }
      }
      leaf error-info {
        type string;
        description
          "Optionally, indicates the source of the assignment
           failure.";
      }
    }
  }
  <CODE ENDS>
```

7. References

7.1. Normative References

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-10 (work in progress), January 2018.

[I-D.ietf-netmod-rfc7223bis]
Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-rfc7223bis-03 (work in progress), January 2018.

[I-D.ietf-netmod-rfc7277bis]
Bjorklund, M., "A YANG Data Model for IP Management", draft-ietf-netmod-rfc7277bis-03 (work in progress), January 2018.

[I-D.ietf-netmod-schema-mount]
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-08 (work in progress), October 2017.

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-06 (work in progress), February 2018.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://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, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[I-D.ietf-bess-l2vpn-yang]
Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-08 (work in progress), February 2018.

[I-D.ietf-bess-l3vpn-yang]
Jain, D., Patel, K., Brissette, P., Li, Z., Zhuang, S., Liu, X., Haas, J., Esale, S., and B. Wen, "Yang Data Model for BGP/MPLS L3 VPNs", draft-ietf-bess-l3vpn-yang-02 (work in progress), October 2017.

[I-D.ietf-netmod-rfc8022bis]
Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", draft-ietf-netmod-rfc8022bis-11 (work in progress), January 2018.

[I-D.ietf-ospf-yang]
Yeung, D., Qu, Y., Zhang, Z., Chen, I., and A. Lindem, "Yang Data Model for OSPF Protocol", draft-ietf-ospf-yang-10 (work in progress), March 2018.

[I-D.ietf-rtgwg-device-model]
Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Logical Organization", draft-ietf-rtgwg-device-model-02 (work in progress), March 2017.

[I-D.ietf-rtgwg-lne-model]
Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Logical Network Elements", draft-ietf-rtgwg-lne-model-09 (work in progress), March 2018.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <https://www.rfc-editor.org/info/rfc4026>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <https://www.rfc-editor.org/info/rfc4664>.

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

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <https://www.rfc-editor.org/info/rfc7277>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <https://www.rfc-editor.org/info/rfc8022>.

Appendix A. Acknowledgments

The Routing Area Yang Architecture design team members included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Lou Berger, Qin Wu, Rob Shakir, Stephane Litkowski, and Yan Gang. Useful review comments were also received by Martin Bjorklund and John Scudder.

This document was motivated by, and derived from, [I-D.ietf-rtgwg-device-model].

Thanks for AD and IETF last call comments from Alia Atlas, Liang Xia, Benoit Claise, and Adam Roach.

The RFC text was produced using Marshall Rose’s xml2rfc tool.

Appendix B. Example NI usage

The following subsections provide example uses of NIs.

B.1. Configuration Data

The following shows an example where two customer specific network instances are configured:

```
{
  "ietf-network-instance:network-instances": {
    "network-instance": [
```

```
        {
          "name": "vrf-red",
          "vrf-root": {
            "ietf-routing:routing": {
              "router-id": "192.0.2.1",
              "control-plane-protocols": {
                "control-plane-protocol": [
                  {
                    "type": "ietf-routing:ospf",
                    "name": "1",
                    "ietf-ospf:ospf": {
                      "af": "ipv4",
                      "areas": {
                        "area": [
                          {
                            "area-id": "203.0.113.1",
                            "interfaces": {
                              "interface": [
                                {
                                  "name": "eth1",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            }
          }
        },
        {
          "name": "vrf-blue",
          "vrf-root": {
            "ietf-routing:routing": {
              "router-id": "192.0.2.2",
              "control-plane-protocols": {
                "control-plane-protocol": [
                  {
                    "type": "ietf-routing:ospf",
                    "name": "1",
                    "ietf-ospf:ospf": {
                      "af": "ipv4",
                      "areas": {
                        "area": [
```

```
                          {
                            "area-id": "203.0.113.1",
                            "interfaces": {
                              "interface": [
                                {
                                  "name": "eth2",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            }
          }
        ]
      },

      "ietf-interfaces:interfaces": {
        "interfaces": {
          "interface": [
            {
              "name": "eth0",
              "ip:ipv4": {
                "address": [
                  {
                    "ip": "192.0.2.10",
                    "prefix-length": 24,
                  }
                ]
              },
              "ip:ipv6": {
                "address": [
                  {
                    "ip": "2001:db8:0:2::10",
                    "prefix-length": 64,
                  }
                ]
              }
            },
            {
              "name": "eth1",
              "ip:ipv4": {
```

```
            "address": [
              {
                "ip": "192.0.2.11",
                "prefix-length": 24,
              }
            ]
          },
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::11",
                "prefix-length": 64,
              }
            ]
          },
          "ni:bind-network-instance-name": "vrf-red"
        },
        {
          "name": "eth2",
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.11",
                "prefix-length": 24,
              }
            ]
          },
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::11",
                "prefix-length": 64,
              }
            ]
          },
          "ni:bind-network-instance-name": "vrf-blue"
        }
      ]
    }
  },

  "ietf-system:system": {
    "authentication": {
      "user": [
        {
          "name": "john",
          "password": "$0$password"
        }
```

```
        ]
      }
    }
  }
```

B.2. State Data - Non-NMDA Version

The following shows state data for the configuration example above based on [RFC7223], [RFC7277], and [RFC8022].

```
{
  "ietf-network-instance:network-instances": {
    "network-instance": [
      {
        "name": "vrf-red",
        "vrf-root": {
          "ietf-yang-library:modules-state": {
            "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
            "module": [
              {
                "name": "ietf-yang-library",
                "revision": "2016-06-21",
                "namespace":
                "urn:ietf:params:xml:ns:yang:ietf-yang-library",
                "conformance-type": "implement"
              },
              {
                "name": "ietf-ospf",
                "revision": "2018-03-03",
                "namespace":
                "urn:ietf:params:xml:ns:yang:ietf-ospf",
                "conformance-type": "implement"
              },
              {
                "name": "ietf-routing",
                "revision": "2018-03-13",
                "namespace":
                "urn:ietf:params:xml:ns:yang:ietf-routing",
                "conformance-type": "implement"
              }
            ]
          },
          "ietf-routing:routing-state": {
            "router-id": "192.0.2.1",
            "control-plane-protocols": {
              "control-plane-protocol": [
                {
                  "type": "ietf-routing:ospf",
```

```
                    "name": "1",
                    "ietf-ospf:ospf": {
                      "af": "ipv4",
                      "areas": {
                        "area": [
                          {
                            "area-id": "203.0.113.1",
                            "interfaces": {
                              "interface": [
                                {
                                  "name": "eth1",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            }
          }
        },
        {
          "name": "vrf-blue",
          "vrf-root": {
            "ietf-yang-library:modules-state": {
              "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
              "module": [
                {
                  "name": "ietf-yang-library",
                  "revision": "2016-06-21",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-yang-library",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-ospf",
                  "revision": "2018-03-03",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-ospf",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-routing",
                  "revision": "2018-03-13",
```

```
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-routing",
                  "conformance-type": "implement"
                }
              ]
            },
            "ietf-routing:routing-state": {
              "router-id": "192.0.2.2",
              "control-plane-protocols": {
                "control-plane-protocol": [
                  {
                    "type": "ietf-routing:ospf",
                    "name": "1",
                    "ietf-ospf:ospf": {
                      "af": "ipv4",
                      "areas": {
                        "area": [
                          {
                            "area-id": "203.0.113.1",
                            "interfaces": {
                              "interface": [
                                {
                                  "name": "eth2",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            }
          }
        }
      ]
    },

    "ietf-interfaces:interfaces-state": {
      "interfaces": {
        "interface": [
          {
            "name": "eth0",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C0",
```

```
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.10",
                "prefix-length": 24,
              }
            ]
          }
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::10",
                "prefix-length": 64,
              }
            ]
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "phys-address": "00:01:02:A1:B1:C1",
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.11",
                "prefix-length": 24,
              }
            ]
          }
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::11",
                "prefix-length": 64,
              }
            ]
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
```

```
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C2",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            }
            "ip:ipv6": {
              "address": [
                {
                  "ip": "2001:db8:0:2::11",
                  "prefix-length": 64,
                }
              ]
            }
          }
        ]
      }
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    }

    "ietf-yang-library:modules-state": {
      "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
      "module": [
        {
          "name": "iana-if-type",
          "revision": "2014-05-08",
          "namespace":
          "urn:ietf:params:xml:ns:yang:iana-if-type",
          "conformance-type": "import"
        },
        {
          "name": "ietf-inet-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-inet-types",
          "conformance-type": "import"
```

```
        },
        {
          "name": "ietf-interfaces",
          "revision": "2014-05-08",
          "feature": [
            "arbitrary-names",
            "pre-provisioning"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-interfaces",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ip",
          "revision": "2014-06-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-ip",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-network-instance",
          "revision": "2018-02-03",
          "feature": [
            "bind-network-instance-name"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-network-instance",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ospf",
          "revision": "2018-03-03",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-ospf",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-routing",
          "revision": "2018-03-13",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-routing",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement"
```

```
        },
        {
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-schema-mount",
          "revision": "2017-05-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-yang-schema-mount:schema-mounts": {
      "mount-point": [
        {
          "module": "ietf-network-instance",
          "label": "vrf-root",
          "shared-schema": {
            "parent-reference": [
              "/*[namespace-uri() = 'urn:ietf:...:ietf-interfaces']"
            ]
          }
        }
      ]
    }
  }
```

B.3. State Data - NMDA Version

The following shows state data for the configuration example above based on [I-D.ietf-netmod-rfc7223bis], [I-D.ietf-netmod-rfc7277bis], and [I-D.ietf-netmod-rfc8022bis].

```
{
  "ietf-network-instance:network-instances": {
```

```
    "network-instance": [
      {
        "name": "vrf-red",
        "vrf-root": {
          "ietf-yang-library:yang-library": {
            "checksum": "41e2ab5dc325f6d86f743e8da3de323f1a61a801",
            "module-set": [
              {
                "name": "ni-modules",
                "module": [
                  {
                    "name": "ietf-yang-library",
                    "revision": "2016-06-21",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-yang-library",
                    "conformance-type": "implement"
                  },
                  {
                    "name": "ietf-ospf",
                    "revision": "2018-03-03",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-ospf",
                    "conformance-type": "implement"
                  },
                  {
                    "name": "ietf-routing",
                    "revision": "2018-03-13",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-routing",
                    "conformance-type": "implement"
                  }
                ],
                "import-only-module": [
                  {
                    "name": "ietf-inet-types",
                    "revision": "2013-07-15",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types"
                  },
                  {
                    "name": "ietf-yang-types",
                    "revision": "2013-07-15",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types"
                  },
                  {
                    "name": "ietf-datastores",
                    "revision": "2018-02-14",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-datastores"
                  }
```

```
            ]
          }
        ],
        "schema": [
          {
            "name": "ni-schema",
            "module-set": [ "ni-modules" ]
          }
        ],
        "datastore": [
          {
            "name": "ietf-datastores:running",
            "schema": "ni-schema"
          },
          {
            "name": "ietf-datastores:operational",
            "schema": "ni-schema"
          }
        ]
      },
      "ietf-routing:routing": {
        "router-id": "192.0.2.1",
        "control-plane-protocols": {
          "control-plane-protocol": [
            {
              "type": "ietf-routing:ospf",
              "name": "1",
              "ietf-ospf:ospf": {
                "af": "ipv4",
                "areas": {
                  "area": [
                    {
                      "area-id": "203.0.113.1",
                      "interfaces": {
                        "interface": [
                          {
                            "name": "eth1",
                            "cost": 10
                          }
                        ]
                      }
                    }
                  ]
                }
              }
            }
          ]
        }
```

```
          }
        }
      },
      {
        "name": "vrf-blue",
        "vrf-root": {
          "ietf-yang-library:yang-library": {
            "checksum": "41e2ab5dc325f6d86f743e8da3de323f1a61a801",
            "module-set": [
              {
                "name": "ni-modules",
                "module": [
                  {
                    "name": "ietf-yang-library",
                    "revision": "2016-06-21",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-yang-library",
                    "conformance-type": "implement"
                  },
                  {
                    "name": "ietf-ospf",
                    "revision": "2018-03-03",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-ospf",
                    "conformance-type": "implement"
                  },
                  {
                    "name": "ietf-routing",
                    "revision": "2018-03-13",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-routing",
                    "conformance-type": "implement"
                  }
                ],
                "import-only-module": [
                  {
                    "name": "ietf-inet-types",
                    "revision": "2013-07-15",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types"
                  },
                  {
                    "name": "ietf-yang-types",
                    "revision": "2013-07-15",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types"
                  },
                  {
                    "name": "ietf-datastores",
                    "revision": "2018-02-14",
```

```
                  "namespace": "urn:ietf:params:xml:ns:yang:ietf-datastores"
                }
              ]
            }
          ],
          "schema": [
            {
              "name": "ni-schema",
              "module-set": [ "ni-modules" ]
            }
          ],
          "datastore": [
            {
              "name": "ietf-datastores:running",
              "schema": "ni-schema"
            },
            {
              "name": "ietf-datastores:operational",
              "schema": "ni-schema"
            }
          ]
        },
        "ietf-routing:routing": {
          "router-id": "192.0.2.2",
          "control-plane-protocols": {
            "control-plane-protocol": [
              {
                "type": "ietf-routing:ospf",
                "name": "1",
                "ietf-ospf:ospf": {
                  "af": "ipv4",
                  "areas": {
                    "area": [
                      {
                        "area-id": "203.0.113.1",
                        "interfaces": {
                          "interface": [
                            {
                              "name": "eth2",
                              "cost": 10
                            }
                          ]
                        }
                      }
                    ]
                  }
                }
              }
```

```
              ]
            }
          }
        }
      }
    ]
  },

  "ietf-interfaces:interfaces": {
    "interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "phys-address": "00:01:02:A1:B1:C0",
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.10",
                "prefix-length": 24,
              }
            ]
          }
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::10",
                "prefix-length": 64,
              }
            ]
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "phys-address": "00:01:02:A1:B1:C1",
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.11",
```

```
                "prefix-length": 24,
              }
            ]
          }
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::11",
                "prefix-length": 64,
              }
            ]
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "phys-address": "00:01:02:A1:B1:C2",
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.11",
                "prefix-length": 24,
              }
            ]
          }
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::11",
                "prefix-length": 64,
              }
            ]
          }
        }
      ]
    }
  },

  "ietf-system:system-state": {
    "platform": {
      "os-name": "NetworkOS"
    }
  }
```

```
  "ietf-yang-library:modules-state": {
    "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
    "module": [
      {
        "name": "iana-if-type",
        "revision": "2014-05-08",
        "namespace":
        "urn:ietf:params:xml:ns:yang:iana-if-type",
        "conformance-type": "import"
      },
      {
        "name": "ietf-inet-types",
        "revision": "2013-07-15",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-inet-types",
        "conformance-type": "import"
      },
      {
        "name": "ietf-interfaces",
        "revision": "2018-01-09",
        "feature": [
          "arbitrary-names",
          "pre-provisioning"
        ],
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-interfaces",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-ip",
        "revision": "2018-01-09",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-ip",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-network-instance",
        "revision": "2018-02-03",
        "feature": [
          "bind-network-instance-name"
        ],
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-network-instance",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-ospf",
        "revision": "2017-10-30",
```

```
        "namespace": "urn:ietf:params:xml:ns:yang:ietf-ospf",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-routing",
        "revision": "2018-01-25",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-routing",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-system",
        "revision": "2014-08-06",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-system",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-yang-library",
        "revision": "2016-06-21",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-yang-library",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-yang-schema-mount",
        "revision": "2017-05-16",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-yang-types",
        "revision": "2013-07-15",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-yang-types",
        "conformance-type": "import"
      }
    ]
  },

  "ietf-yang-schema-mount:schema-mounts": {
    "mount-point": [
      {
        "module": "ietf-network-instance",
        "label": "vrf-root",
        "shared-schema": {
          "parent-reference": [
```

```
            "/*[namespace-uri() = 'urn:ietf:...:ietf-interfaces']"
          ]
        }
      }
    ]
  }
}
```

Authors' Addresses

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Dean Bogdanovic

Email: ivandean@gmail.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 31, 2017

A. Lindem, Ed.
Cisco Systems
Y. Qu
Huawei
D. Yeung
Arrcus, Inc
I. Chen
Jabil
J. Zhang
Juniper Networks

April 29, 2017

Routing Key Chain YANG Data Model
draft-ietf-rtgwg-yang-key-chain-24.txt

Abstract

This document describes the key chain YANG data model. Key chains are commonly used for routing protocol authentication and other applications requiring symmetric keys. A key chain is a list of elements each containing a key string, send lifetime, accept lifetime, and algorithm (authentication or encryption). By properly overlapping the send and accept lifetimes of multiple key chain elements, key strings and algorithms may be gracefully updated. By representing them in a YANG data model, key distribution can be automated.

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 October 31, 2017.

Table of Contents

1. Introduction

This document describes the key chain YANG [YANG-1.1] data model. Key chains are commonly used for routing protocol authentication and other applications requiring symmetric keys. A key chain is a list of elements each containing a key string, send lifetime, accept lifetime, and algorithm (authentication or encryption). By properly

overlapping the send and accept lifetimes of multiple key chain elements, key strings and algorithms may be gracefully updated. By representing them in a YANG data model, key distribution can be automated.

In some applications, the protocols do not use the key chain element key directly, but rather a key derivation function is used to derive a short-lived key from the key chain element key (e.g., the Master Keys used in [TCP-AO]).

1.1. Requirements Notation

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

1.2. Tree Diagrams

A simplified graphical representation of the complete data tree is presented in Section 3.3. The following tree notation is used.

- Brackets "[" and "]" enclose YANG list keys. These YANG list keys should not be confused with the key-chain keys.
- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), "ro" state data (read-only), "-x" RPC operations, and "-n" notifications.
- Symbols after data node names: "?" means an optional node, "!" a container with presence, and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Problem Statement

This document describes a YANG [YANG-1.1] data model for key chains. Key chains have been implemented and deployed by a large percentage of network equipment vendors. Providing a standard YANG model will facilitate automated key distribution and non-disruptive key

rollover. This will aid in tightening the security of the core routing infrastructure as recommended in [IAB-REPORT].

A key chain is a list containing one or more elements containing a Key ID, key string, send/accept lifetimes, and the associated authentication or encryption algorithm. A key chain can be used by any service or application requiring authentication or encryption using symmetric keys. In essence, the key-chain is a reusable key policy that can be referenced wherever it is required. The key-chain construct has been implemented by most networking vendors and deployed in many networks.

A conceptual representation of a crypto key table is described in [CRYPTO-KEYTABLE]. The crypto key table also includes keys as well as their corresponding lifetimes and algorithms. Additionally, the key table includes key selection criteria and envisions a deployment model where the details of the applications or services requiring authentication or encryption permeate into the key database. The YANG key-chain model described herein doesn’t include key selection criteria or support this deployment model. At the same time, it does not preclude it. The draft [YANG-CRYPTO-KEYTABLE] describes augmentations to the key chain YANG model in support of key selection criteria.

2.1. Applicability

Other YANG modules may reference ietf-key-chain YANG module key-chain names for authentication and encryption applications. A YANG type has been provided to facilitate reference to the key-chain name without having to specify the complete YANG XML Path Language (XPath) selector.

2.2. Graceful Key Rollover using Key Chains

Key chains may be used to gracefully update the key string and/or algorithm used by an application for authentication or encryption. To achieve graceful key rollover, the receiver MAY accept all the keys that have a valid accept lifetime and the sender MAY send the key with the most recent send lifetime. One scenario for facilitating key rollover is to:

1. Distribute a key chain with a new key to all the routers or other network devices in the domain of that key chain. The new key’s accept lifetime should be such that it is accepted during the key rollover period. The send lifetime should be a time in the future when it can be assured that all the routers in the domain of that key are upgraded. This will have no immediate impact on the keys used for transmission.

2. Assure that all the network devices have been updated with the updated key chain and that their system times are roughly synchronized. The system times of devices within an administrative domain are commonly synchronized (e.g., using Network Time Protocol (NTP) [NTP-PROTO]). This also may be automated.

3. When the send lifetime of the new key becomes valid, the network devices within the domain of key chain will using the new key for transmissions.

4. At some point in the future, a new key chain with the old key removed may be distributed to the network devices within the domain of the key chain. However, this may be deferred until the next key rollover. If this is done, the key chain will always include two keys; either the current and future key (during key rollovers) or the current and previous keys (between key rollovers).

Since the most recent send lifetime is defined as the one with the latest start-time, specification of "always" will prevent using the graceful key rollover technique described above. Other key configuration and usage scenarios are possible but these are beyond the scope of this document.

3. Design of the Key Chain Model

The ietf-key-chain module contains a list of one or more keys indexed by a Key ID. For some applications (e.g., OSPFv3 [OSPFV3-AUTH]), the Key ID is used to identify the key chain key to be used. In addition to the Key ID, each key chain key includes a key-string and a cryptographic algorithm. Optionally, the key chain keys include send/accept lifetimes. If the send/accept lifetime is unspecified, the key is always considered valid.

Note that different key values for transmission versus acceptance may be supported with multiple key chain elements. The key used for transmission will have a valid send-lifetime and invalid accept-lifetime (e.g., has an end-time equal to the start-time). The key used for acceptance will have a valid accept-lifetime and invalid send-lifetime.

Due to the differences in key chain implementations across various vendors, some of the data elements are optional. Finally, the crypto algorithm identities are provided for reuse when configuring legacy authentication and encryption not using key-chains.

A key-chain is identified by a unique name within the scope of the network device. The "key-chain-ref" typedef SHOULD be used by other YANG modules when they need to reference a configured key-chain.

3.1. Key Chain Operational State

The key chain operational state is included in the same tree as key chain configuration consistent with Network Management Datastore Architecture [NMDA]. The timestamp of the last key chain modification is also maintained in the operational state. Additionally, the operational state includes an indication of whether or not a key chain key is valid for sending or acceptance.

3.2. Key Chain Model Features

Features are used to handle differences between vendor implementations. For example, not all vendors support configuration of an acceptance tolerance or configuration of key strings in hexadecimal. They are also used to support of security requirements (e.g., TCP-AO Algorithms [TCP-AO-ALGORITHMS]) not yet implemented by vendors or only a single vendor.

It is common for an entity with sufficient permissions to read and store a device’s configuration which would include the contents of this model. To avoid unnecessarily seeing and storing the keys in clear-text, this model provides the aes-key-wrap feature. More details are described in Security Considerations Section 5.

3.3. Key Chain Model Tree

```
+--rw key-chains
   +--rw key-chain* [name]
   |  +--rw name                        string
   |  +--rw description?                string
   |  +--rw accept-tolerance {accept-tolerance}?
   |  |  +--rw duration?   uint32
   |  +--ro last-modified-timestamp?   yang:date-and-time
   |  +--rw key* [key-id]
   |     +--rw key-id                     uint64
   |     +--rw lifetime
   |     |  +--rw (lifetime)?
   |     |     +--:(send-and-accept-lifetime)
   |     |     |  +--rw send-accept-lifetime
   |     |     |     +--rw (lifetime)?
   |     |     |        +--:(always)
   |     |     |        |  +--rw always?            empty
   |     |     |        +--:(start-end-time)
   |     |     |           +--rw start-date-time?
```

```
|     |     |           |       yang:date-and-time
|     |     |           +--rw (end-time)?
|     |     |              +--:(infinite)
|     |     |              |  +--rw no-end-time?       empty
|     |     |              +--:(duration)
|     |     |              |  +--rw duration?          uint32
|     |     |              +--:(end-date-time)
|     |     |                 +--rw end-date-time?
|     |     |                            yang:date-and-time
|     |     +--:(independent-send-accept-lifetime)
|     |     |   {independent-send-accept-lifetime}?
|     |     +--rw send-lifetime
|     |     |  +--rw (lifetime)?
|     |     |     +--:(always)
|     |     |     |  +--rw always?            empty
|     |     |     +--:(start-end-time)
|     |     |        +--rw start-date-time?
|     |     |        |       yang:date-and-time
|     |     |        +--rw (end-time)?
|     |     |           +--:(infinite)
|     |     |           |  +--rw no-end-time?       empty
|     |     |           +--:(duration)
|     |     |           |  +--rw duration?          uint32
|     |     |           +--:(end-date-time)
|     |     |              +--rw end-date-time?
|     |     |                         yang:date-and-time
|     |     |        +--rw accept-lifetime
|     |     |           +--rw (lifetime)?
|     |     |              +--:(always)
|     |     |              |  +--rw always?            empty
|     |     |              +--:(start-end-time)
|     |     |                 +--rw start-date-time?
|     |     |                 |       yang:date-and-time
|     |     |                 +--rw (end-time)?
|     |     |                    +--:(infinite)
|     |     |                    |  +--rw no-end-time?       empty
|     |     |                    +--:(duration)
|     |     |                    |  +--rw duration?          uint32
|     |     |                    +--:(end-date-time)
|     |     |                       +--rw end-date-time?
|     |     |                                  yang:date-and-time
|     |     +--rw crypto-algorithm identityref
|     |     +--rw key-string
|     |     |  +--rw (key-string-style)?
|     |     |     +--:(keystring)
|     |     |     |  +--rw keystring?            string
|     |     |     +--:(hexadecimal) {hex-key-string}?
|     |     |        +--rw hexadecimal-string?   yang:hex-string
```

```
      |     +--ro send-lifetime-active?     boolean
      |     +--ro accept-lifetime-active?   boolean
      +--rw aes-key-wrap {aes-key-wrap}?
         +--rw enable?   boolean
```

4. Key Chain YANG Model

```
<CODE BEGINS> file "ietf-key-chain@2017-04-18.yang"
module ietf-key-chain {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-key-chain";
  prefix key-chain;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-netconf-acm {
    prefix nacm;
  }

  organization
    "IETF RTG (Routing) Working Group";
  contact
    "Acee Lindem - acee@cisco.com";
  description
    "This YANG module defines the generic configuration
     data for key-chain. It is intended that the module
     will be extended by vendors to define vendor-specific
     key-chain configuration parameters.

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

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).
     This version of this YANG module is part of RFC XXXX; see
     the RFC itself for full legal notices.";

  revision 2017-04-18 {
    description
      "Initial RFC Revision";
    reference "RFC XXXX: A YANG Data Model for key-chain";
  }
```

```
     feature hex-key-string {
       description
         "Support hexadecimal key string.";
     }

     feature accept-tolerance {
       description
         "Support the tolerance or acceptance limit.";
     }

     feature independent-send-accept-lifetime {
       description
         "Support for independent send and accept key lifetimes.";
     }

     feature crypto-hmac-sha-1-12 {
       description
         "Support for TCP HMAC-SHA-1 12 byte digest hack.";
     }

     feature clear-text {
       description
         "Support for clear-text algorithm. Usage is
          NOT RECOMMENDED.";
     }

     feature aes-cmac-prf-128 {
       description
         "Support for AES Cipher based Message Authentication
          Code Pseudo Random Function.";
     }

     feature aes-key-wrap {
       description
         "Support for Advanced Encryption Standard (AES) Key Wrap.";
     }

     feature replay-protection-only {
       description
         "Provide replay-protection without any authentication
          as required by protocols such as Bidirectional
          Forwarding Detection (BFD).";
     }
     identity crypto-algorithm {
       description
         "Base identity of cryptographic algorithm options.";
     }
```

```
     identity hmac-sha-1-12 {
       base crypto-algorithm;
       if-feature "crypto-hmac-sha-1-12";
       description
         "The HMAC-SHA1-12 algorithm.";
     }

     identity aes-cmac-prf-128 {
       base crypto-algorithm;
       if-feature "aes-cmac-prf-128";
       description
         "The AES-CMAC-PRF-128 algorithm - required by
          RFC 5926 for TCP-AO key derivation functions.";
     }

     identity md5 {
       base crypto-algorithm;
       description
         "The MD5 algorithm.";
     }

     identity sha-1 {
       base crypto-algorithm;
       description
         "The SHA-1 algorithm.";
     }

     identity hmac-sha-1 {
       base crypto-algorithm;
       description
         "HMAC-SHA-1 authentication algorithm.";
     }

     identity hmac-sha-256 {
       base crypto-algorithm;
       description
         "HMAC-SHA-256 authentication algorithm.";
     }

     identity hmac-sha-384 {
       base crypto-algorithm;
       description
         "HMAC-SHA-384 authentication algorithm.";
     }

     identity hmac-sha-512 {
       base crypto-algorithm;
       description
```

```
        "HMAC-SHA-512 authentication algorithm.";
    }

    identity clear-text {
      base crypto-algorithm;
      if-feature "clear-text";
      description
        "Clear text.";
    }

    identity replay-protection-only {
      base crypto-algorithm;
      if-feature "replay-protection-only";
      description
        "Provide replay-protection without any authentication as
         required by protocols such as Bidirectional Forwarding
         Detection (BFD).";
    }

    typedef key-chain-ref {
      type leafref {
        path
        "/key-chain:key-chains/key-chain:key-chain/key-chain:name";
      }
      description
        "This type is used by data models that need to reference
         configured key-chains.";
    }

    grouping lifetime {
      description
        "Key lifetime specification.";
      choice lifetime {
        default "always";
        description
          "Options for specifying key accept or send lifetimes";
        case always {
          leaf always {
            type empty;
            description
              "Indicates key lifetime is always valid.";
          }
        }
        case start-end-time {
          leaf start-date-time {
            type yang:date-and-time;
            description
              "Start time.";
```

```
          }
          choice end-time {
            default "infinite";
            description
              "End-time setting.";
            case infinite {
              leaf no-end-time {
                type empty;
                description
                  "Indicates key lifetime end-time is infinite.";
              }
            }
            case duration {
              leaf duration {
                type uint32 {
                  range "1..2147483646";
                }
                units "seconds";
                description
                  "Key lifetime duration, in seconds";
              }
            }
            case end-date-time {
              leaf end-date-time {
                type yang:date-and-time;
                description
                  "End time.";
              }
            }
          }
        }
      }
    }
    grouping key-common {
      description
        "Key-chain key data nodes common to
         configuration and state.";
    }

    container key-chains {
      description
        "All configured key-chains on the device.";
      list key-chain {
        key "name";
        description
          "List of key-chains.";
        leaf name {
          type string;
```

```
          description
            "Name of the key-chain.";
        }
        leaf description {
          type string;
          description
            "A description of the key-chain";
        }
        container accept-tolerance {
          if-feature "accept-tolerance";
          description
            "Tolerance for key lifetime acceptance (seconds).";
          leaf duration {
            type uint32;
            units "seconds";
            default "0";
            description
              "Tolerance range, in seconds.";
          }
        }
        leaf last-modified-timestamp {
          type yang:date-and-time;
          config false;
          description
            "Timestamp of the most recent update to the key-chain";
        }
        list key {
          key "key-id";
          description
            "Single key in key chain.";
          leaf key-id {
            type uint64;
            description
              "Numeric value uniquely identifying the key";
          }
          container lifetime {
            description
              "Specify a key's lifetime.";
            choice lifetime {
              description
                "Options for specification of send and accept
                 lifetimes.";
              case send-and-accept-lifetime {
                description
                  "Send and accept key have the same lifetime.";
                container send-accept-lifetime {
                  description
                    "Single lifetime specification for both
```

```
                    send and accept lifetimes.";
                uses lifetime;
              }
            }
            case independent-send-accept-lifetime {
              if-feature "independent-send-accept-lifetime";
              description
                "Independent send and accept key lifetimes.";
              container send-lifetime {
                description
                  "Separate lifetime specification for send
                   lifetime.";
                uses lifetime;
              }
              container accept-lifetime {
                description
                  "Separate lifetime specification for accept
                   lifetime.";
                uses lifetime;
              }
            }
          }
        }
        leaf crypto-algorithm {
          type identityref {
            base crypto-algorithm;
          }
          mandatory true;
          description
            "Cryptographic algorithm associated with key.";
        }
        container key-string {
          description
            "The key string.";
          nacm:default-deny-all;
          choice key-string-style {
            description
              "Key string styles";
             case keystring {
               leaf keystring {
                type string;
                description
                  "Key string in ASCII format.";
              }
            }
            case hexadecimal {
              if-feature "hex-key-string";
              leaf hexadecimal-string {
```

```
                    type yang:hex-string;
                    description
                      "Key in hexadecimal string format. When compared
                       to ASCII, specification in hexadecimal affords
                       greater key entropy with the same number of
                       internal key-string octets. Additionally, it
                       discourages usage of well-known words or
                       numbers.";
                  }
                }
              }
            }
            leaf send-lifetime-active {
              type boolean;
              config false;
              description
                "Indicates if the send lifetime of the
                 key-chain key is currently active.";
               }
            leaf accept-lifetime-active {
              type boolean;
              config false;
              description
                "Indicates if the accept lifetime of the
                 key-chain key is currently active.";
            }
          }
        }
        container aes-key-wrap {
          if-feature "aes-key-wrap";
          description
            "AES Key Wrap encryption for key-chain key-strings. The
             encrypted key-strings are encoded as hexadecimal key
             strings using the hex-key-string leaf.";
          leaf enable {
            type boolean;
            default "false";
            description
              "Enable AES Key Wrap encryption.";
          }
        }
      }
    }
    <CODE ENDS>
```

5. Security Considerations

The YANG module defined in this document is designed to be accessed via network management protocols such as NETCONF [NETCONF] or RESTCONF [RESTCONF]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [NETCONF-SSH]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [TLS].

The NETCONF access control model [NETCONF-ACM] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content. The key strings are not accessible by default and NETCONF Access Control Mode [NETCONF-ACM] rules are required to configure or retrieve them.

When configured, the key-strings can be encrypted using the AES Key Wrap algorithm [AES-KEY-WRAP]. The AES key-encryption key (KEK) is not included in the YANG model and must be set or derived independent of key-chain configuration. When AES key-encryption is used, the hex-key-string feature is also required since the encrypted keys will contain characters that are not representable in the YANG string built-in type [YANG-1.1]. It is RECOMMENDED that key-strings be encrypted using AES key-encryption to prevent key-chains from being retrieved and stored with the key-strings in clear text. This recommendation is independent of the access protection that is availed from the NETCONF Access Control Model (NACM) [NETCONF-ACM].

The clear-text algorithm is included as a YANG feature. Usage is NOT RECOMMENDED except in cases where the application and device have no other alternative (e.g., a legacy network device that must authenticate packets at intervals of 10 milliseconds or less for many peers using Bidirectional Forwarding Detection [BFD]). Keys used with the clear-text algorithm are considered insecure and SHOULD NOT be reused with more secure algorithms.

Similarly, the MD5 and SHA-1 algorithms have been proven to be insecure ([Dobb96a], [Dobb96b], and [SHA-SEC-CON]) and usage is NOT RECOMMENDED. Usage should be confined to deployments where it is required for backward compatibility.

Implementations with keys provided via this model should store them using best current security practices.

6. IANA Considerations

This document registers a URI in the IETF XML registry [XML-REGISTRY]. Following the format in [XML-REGISTRY], the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-key-chain
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [YANG-1.0].

name: ietf-key-chain
namespace: urn:ietf:params:xml:ns:yang:ietf-key-chain
prefix: key-chain
reference: RFC XXXX

7. Contributors

Contributors' Addresses

Yi Yang
SockRate

Email: yi.yang@sockrate.com

8. References

8.1. Normative References

[NETCONF] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[NETCONF-ACM] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

[RFC-KEYWORDS] Bradner, S., "Key words for use in RFC's to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[XML-REGISTRY] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[YANG-1.0]
Bjorklund, M., "YANG - A Data Modeling Language for Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[YANG-1.1]
Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.

8.2. Informative References

[AES-KEY-WRAP]
Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 5649, August 2009.

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

[CRYPTO-KEYTABLE]
Housley, R., Polk, T., Hartman, S., and D. Zhang, "Table of Cryptographic Keys", RFC 7210, April 2014.

[Dobb96a] Dobbertin, H., "Cryptanalysis of MD5 Compress", Technical Report (Presented at the RUMP Session of EuroCrypt 1996), 2 May 1996.

[Dobb96b] Dobbertin, H., "The Status of MD5 After a Recent Attack", CryptoBytes Vol. 2, No. 2, Summer 1996.

[IAB-REPORT]
Andersson, L., Davies, E., and L. Zhang, "Report from the IAB workshop on Unwanted Traffic March 9-10, 2006", RFC 4948, August 2007.

[NETCONF-SSH]
Wasserman, M., "NETCONF over SSH", RFC 6242, June 2011.

[NMDA] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watson, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-01.txt (work in progress), March 2017.

[NTP-PROTO]
Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[OSPFV3-AUTH]
Bhatia, M., Manral, V., and A. Lindem, "Supporting Authentication Trailer for OSPFv3", RFC 7166, March 2014.

[RESTCONF]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.

[SHA-SEC-CON]
Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, February 2011.

[TCP-AO] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

[TCP-AO-ALGORITHMS]
Lebovitz, G. and E. Rescorla, "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)", RFC 5926, June 2010.

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol", RFC 5246, August 2008.

[YANG-CRYPTO-KEYTABLE]
Chen, I., "YANG Data Model for RFC 7210 Key Table", draft-chen-rtg-key-table-yang-00.txt (work in progress), November 2015.

Appendix A. Examples

A.1. Simple Key Chain with Always Valid Single Key

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <key-chains xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain">
    <key-chain>
      <name>keychain-no-end-time</name>
      <description>
        A key chain with a single key that is always valid for
        transmission and reception.
      </description>
      <key>
        <key-id>100</key-id>
        <lifetime>
          <send-accept-lifetime>
            <always/>
          </send-accept-lifetime>
        </lifetime>
        <crypto-algorithm>hmac-sha-256</crypto-algorithm>
        <key-string>
          <keystring>keystring_in_ascii_100</keystring>
        </key-string>
      </key>
    </key-chain>
  </key-chains>
</data>
```

A.2. Key Chain with Keys having Different Lifetimes

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <key-chains xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain">
    <key-chain>
      <name>keychain2</name>
      <description>
        A key chain where each key contains different send time
        and accept time and a different algorithm illustrating
        algorithm agility.
      </description>
      <key>
        <key-id>35</key-id>
        <lifetime>
          <send-lifetime>
            <start-date-time>2017-01-01T00:00:00Z</start-date-time>
            <end-date-time>2017-02-01T00:00:00Z</end-date-time>
          </send-lifetime>
          <accept-lifetime>
            <start-date-time>2016-12-31T23:59:55Z</start-date-time>
            <end-date-time>2017-02-01T00:00:05Z</end-date-time>
          </accept-lifetime>
        </lifetime>
        <crypto-algorithm>hmac-sha-256</crypto-algorithm>
        <key-string>
          <keystring>keystring_in_ascii_35</keystring>
        </key-string>
      </key>
      <key>
        <key-id>36</key-id>
        <lifetime>
          <send-lifetime>
            <start-date-time>2017-02-01T00:00:00Z</start-date-time>
            <end-date-time>2017-03-01T00:00:00Z</end-date-time>
          </send-lifetime>
          <accept-lifetime>
            <start-date-time>2017-01-31T23:59:55Z</start-date-time>
            <end-date-time>2017-03-01T00:00:05Z</end-date-time>
          </accept-lifetime>
        </lifetime>
        <crypto-algorithm>hmac-sha-512</crypto-algorithm>
        <key-string>
          <hexadecimal-string>fe:ed:be:af:36</hexadecimal-string>
        </key-string>
      </key>
    </key-chain>
  </key-chains>
</data>
```

A.3. Key Chain with Independent Send and Accept Lifetimes

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <key-chains xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain">
    <key-chain>
      <name>keychain2</name>
      <description>
        A key chain where each key contains different send time
        and accept times.
      </description>
      <key>
        <key-id>35</key-id>
        <lifetime>
          <send-lifetime>
            <start-date-time>2017-01-01T00:00:00Z</start-date-time>
            <end-date-time>2017-02-01T00:00:00Z</end-date-time>
          </send-lifetime>
          <accept-lifetime>
            <start-date-time>2016-12-31T23:59:55Z</start-date-time>
            <end-date-time>2017-02-01T00:00:05Z</end-date-time>
          </accept-lifetime>
        </lifetime>
        <crypto-algorithm>hmac-sha-256</crypto-algorithm>
        <key-string>
          <keystring>keystring_in_ascii_35</keystring>
        </key-string>
      </key>
      <key>
        <key-id>36</key-id>
        <lifetime>
          <send-lifetime>
            <start-date-time>2017-02-01T00:00:00Z</start-date-time>
            <end-date-time>2017-03-01T00:00:00Z</end-date-time>
          </send-lifetime>
          <accept-lifetime>
            <start-date-time>2017-01-31T23:59:55Z</start-date-time>
            <end-date-time>2017-03-01T00:00:05Z</end-date-time>
          </accept-lifetime>
        </lifetime>
        <crypto-algorithm>hmac-sha-256</crypto-algorithm>
        <key-string>
          <hexadecimal-string>fe:ed:be:af:36</hexadecimal-string>
        </key-string>
      </key>
    </key-chain>
  </key-chains>
</data>
```

Appendix B. Acknowledgments

The RFC text was produced using Marshall Rose's xml2rfc tool.

Thanks to Brian Weis for fruitful discussions on security requirements.

Thanks to Ines Robles for Routing Directorate QA review comments.

Thanks to Ladislav Lhotka for YANG Doctor comments.

Thanks to Martin Bjorklund for additional YANG Doctor comments.

Thanks to Tom Petch for comments during IETF last call.

Thanks to Matthew Miller for comments made during the Gen-ART review.

Thanks to Vincent Roca for comments made during the Security Directorate review.

Thanks to Warren Kumari, Ben Campbell, Adam Roach, and Benoit Claise for comments received during the IESG review.

Authors' Addresses

Acee Lindem (editor)
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Yingzhen Qu
Huawei

Email: yingzhen.qu@huawei.com

Derek Yeung
Arrcus, Inc

Email: derek@arrcus.com

Ing-Wher Chen
Jabil

Email: ing-wher_chen@jabil.com

Jeffrey Zhang
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: zzhang@juniper.net

Network Working Group   Z. Li
Internet-Draft   J. Zhang
Intended status: Informational   Huawei Technologies
Expires: May 4, 2017   October 31, 2016

An Architecture of Network Artificial Intelligence(NAI)
draft-li-rtgwg-network-ai-arch-00

Abstract

Artificial intelligence is an important technical trend in the industry. With the development of network, it is necessary to introduce artificial intelligence technology to achieve self-adjustment, self- optimization, self-recovery of the network through collection of huge data of network state and machine learning. This draft defines the architecture of Network Artificial Intelligence (NAI), including the key components and the key protocol extension requirements.

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 inRFC 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 May 4, 2017.

Copyright Notice

Table of Contents

1. Introduction

Artificial Intelligence is an important technical trend in the industry. The two key aspects of Artificial Intelligence are perception and cognition. Artificial Intelligence has evolved from an early non-learning expert system to a learning-capable machine learning era. In recent years, the rapid development of the deep learning branch based on the neural network and the maturity of the big data technology and software distributed architecture make the Artificial Intelligence in many fields (such as transportation, medical treatment, education, etc.) have been applied. With the development of network, it is necessary to introduce artificial intelligence technology to achieve self-adjustment, self-optimization, self-recovery of the network through collection of huge data of network state and machine learning. The areas of machine learning which are easier to be used in the network field may include: troubleshooting of network problems, network traffic prediction, traffic optimization adjustment, security defense, security auditing, etc., to implement network perception and cognition.

This draft defines the architecture of Network Artificial Intelligence (NAI), including the key components and the key protocol extension requirements.

2. Terminology

AI: Artificial Intelligence

NAI: Network Artificial Intelligence

3. Architecture

3.1. Reference Model

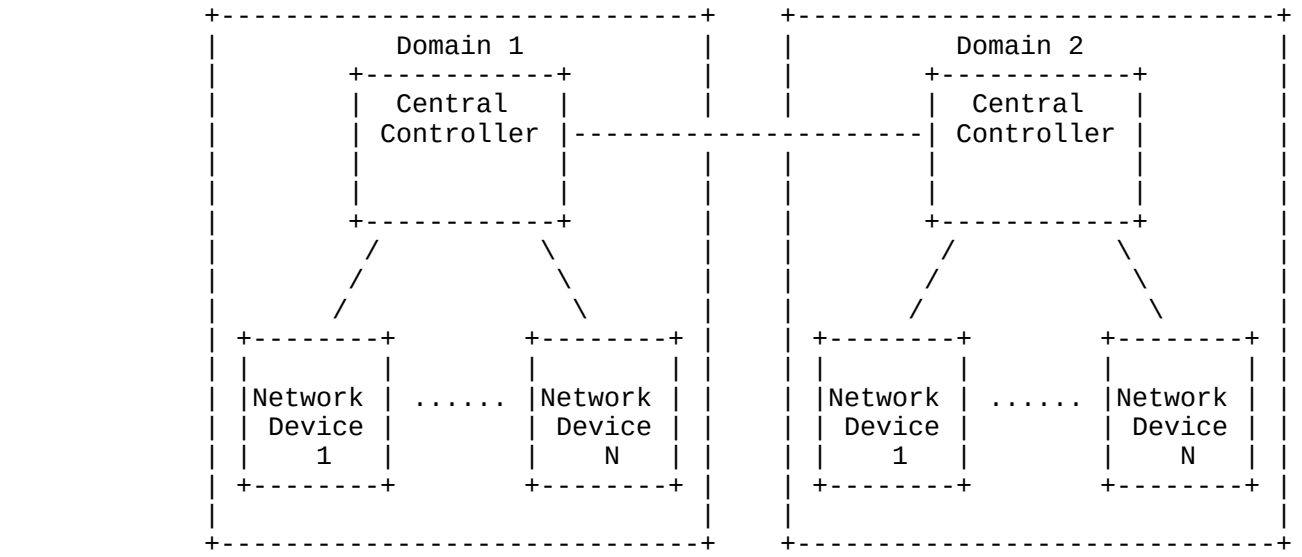


Figure 1: An Architecture of Network Artificial Intelligence(NAI)

The architecture of Network artificial intelligence includes following key component:

1. Central Controller: Centralized controller is the core component of Network Artificial Intelligence which can be called as 'Network Brain'. It man collect huge data of network states, store the data based on the big data platform, and carry on the machine learning, to achieve network perception and cognition, including network self-optimization, self- adjustment, self-recovery, intelligent fault location and a series of network artificial intelligence goals.

2. Network Device: IP network operation and maintenance are always a big challenge since the network can only provide limited state information. The network states includes but are not limited to topology, traffic engineering, operation and maintenance information, network failure information and related information to locate the

network failure.. In order to provide these information, the network must be able to support more OAM mechanisms to acquire more state information and report to the controller. Then the controller can get the complete state information of the network which is the base of Network Artificial Intelligence(NAI).

3. Southbound Protocol and Models of Controller: As network devices provide huge network state information, it proposes a number of new requirements for protocols and models between controllers and network devices. The traditional southbound protocol such as Netconf and SNMP can not meet the performance requirements. It is necessary to introduce some new high-performance protocols to collect network state data. At the same time, the models of network data should be completed. Moreover with the introduction of new OAM mechanisms of network devices, new models of network data should be introduced.

4. Northbound Model of Controller: The goal of the Network Artificial Intelligence is to reduce the technical requirements on the network administrators and release them from the heavy network management, control, maintenance work. The abstract northbound model of the controller for different network services should be simple and easy to be understood.

3.2. Requirement of Protocol Extensions

REQ 01: The new southbound protocol of the controller should be introduced to meet the performance requirements of collecting huge data of network states.

REQ 02: The models of network elements should be completed to collect the network states based on the new southbound protocol of the controller.

REQ 03: New OAM mechanisms should be introduced for the network devices in order to acquire more types of network state data.

REQ 04: New models of network elements should be introduced as the new OAM mechanisms are introduced.

REQ 05: The operation models of network elements should be completed based on the new southbound protocol to carry on the corresponding network operation as the result of Network Artificial Intelligence.

REQ 06: The abstract network-based service models should be provided by the controller as the northbound models to satisfy the requirements of different services.

4. IANA Considerations

This document makes no request of IANA.

5. Security Considerations

TBD.

6. 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>.

Authors' Addresses

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Jinhui Zhang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: jason.zhangjinhui@huawei.com

NVO3 Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2017

G. Mirsky
ZTE Corp.
N. Kumar
D. Kumar
Cisco Systems, Inc.
M. Chen
Y. Li
Huawei Technologies
D. Dolson
Sandvine

March 10, 2017

Echo Request and Echo Reply for Overlay Networks
draft-ooamdt-rtgwg-demand-cc-cv-03

Abstract

This document defines Overlay Echo Request and Echo Reply that enable on-demand Continuity Check, Connectivity Verification among other operations in overlay networks.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2017.

Copyright Notice

Table of Contents

1. Introduction

Operations, Administration, and Maintenance (OAM) toolset provides methods for fault management and performance monitoring in each layer of the network, in order to improve their ability to support services with guaranteed and strict Service Level Agreements (SLAs) while reducing operational costs.

1.1. Conventions used in this document

1.1.1. Terminology

Term "Overlay OAM" used in this document interchangeably with longer version "set of OAM protocols, methods and tools for Overlay networks". And "Overlay ping" is used interchangeably with longer version Overlay Echo Request/Reply.

CC Continuity Check

CV Connectivity Verification

ECMP Equal Cost Multipath

FM Fault Management

Geneve Generic Network Virtualization Encapsulation

GUE Generic UDP Encapsulation

MPLS Multiprotocol Label Switching

NVO3 Network Virtualization Overlays

OAM Operations, Administration, and Maintenance

SFC Service Function Chaining

SFP Service Function Path

VXLAN Virtual eXtensible Local Area Network

VXLAN-GPE Generic Protocol Extension for VXLAN

1.1.2. Requirements Language

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

2. On-demand Continuity Check and Connectivity Verification

2.1. Requirements Towards On-demand CC/CV OAM

Availability, not as performance metric, is understood as ability to reach the node, i.e. the fact that path between ingress and egress does exist. Such OAM mechanism also referred as Continuity Check (CC). Connectivity Verification (CV) extends Continuity Check functionality in order to provide confirmation that the desired source is connected to the desired sink.

Echo Request/Reply OAM mechanism enables detection of the loss of continuity defect, its localization and collection information in order to discover root cause. These are requirements considered:

REQ#1: MUST support fault localization of Loss of Continuity check at Overlay layer.

REQ#2: MAY support fault localization of Loss of Continuity check at transport layer.

REQ#3: MUST support tracing path in overlay network through the overlay nodes.

REQ#4: MAY support tracing path in underlay network connecting overlay border nodes.

REQ#5: MAY support verification of the mapping between its data plane state and client layer services.

REQ#6: MUST have the ability to discover and exercise equal cost multipath (ECMP) paths in its underlay network.

REQ#7: MUST be able to trigger on-demand FM with responses being directed towards initiator of such proxy request.

2.2. Proposed Solution

The format of the Echo Request/Echo Reply control packet is to support ping and traceroute functionality in overlay networks Figure 1 resembles the format of MPLS LSP Ping [RFC4379] with some exceptions.

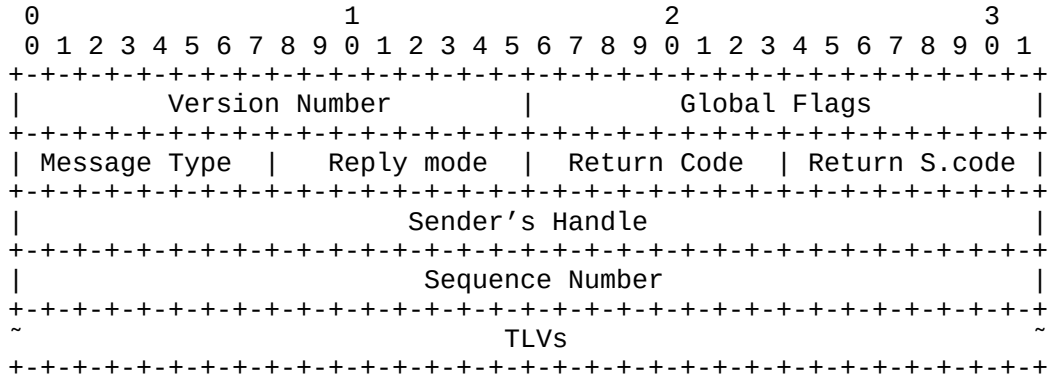


Figure 1: Overlay OAM Ping format

The interpretation of the fields is as following:

The Version reflects the current version. The version number is to be incremented whenever a change is made that affects the

ability of an implementation to correctly parse or process control packet.

The Global Flags is a bit vector field

The Message Type filed reflects the type of the packet. Value TBA2 identifies Echo Request and TBA3 - Echo Reply

The Reply Mode defines the type of the return path requested by the sender of the Echo Request.

Return Codes and Subcodes can be used to inform the sender about result of processing its request.

The Sender's Handle is filled in by the sender, and returned unchanged by the receiver in the Echo Reply.

The Sequence Number is assigned by the sender and can be (for example) used to detect missed replies.

TLVs (Type-Length-Value tuples) have the two octets long Type field, two octets long Length field that is length of the Value field in octets.

2.3. Overlay Echo Request Transmission

Overlay Echo Request control packet MUST use the appropriate encapsulation of the monitored overlay network. Overlay network encapsulation MUST identify Echo Request as OAM packet. Overlay encapsulation uses different methods to identify OAM payload [I-D.ietf-nvo3-vxlan-gpe], [I-D.ietf-nvo3-gue], [I-D.ietf-nvo3-geneve], [I-D.ietf-sfc-nsh],[I-D.ietf-bier-mpls-encapsulation]. Overlay network's header MUST be immediately followed by the Overlay OAM Header [I-D.ooamdt-rtgwg-ooam-header]. Message Type field in the Overlay OAM Header MUST be set to Overlay Echo Request value (TBA2).

Value of the Reply Mode field MAY be set to:

- Do Not Reply (TBA4) if one-way monitoring is desired. If Echo Request is used to measure synthetic packet loss, the receiver MAY report loss measurement results to a remote node.
- Reply via an IPv4/IPv6 UDP Packet (TBA5) value likely will be the most used.
- Reply via Application Level Control Channel (TBA6) value if the overlay network MAY have bi-directional paths.

- o Reply via Specified Path (TBA7) value in order to enforce use of the particular return path specified in the included TLV to verify bi-directional continuity and also increase robustness of the monitoring by selecting more stable path.

2.4. Overlay Echo Request Reception

2.5. Overlay Echo Reply Transmission

The Reply Mode field directs whether and how the Echo Reply message should be sent. The sender of the Echo Request MAY use TLVs to request that corresponding Echo Reply be sent using the specified path. Value TBA3 is referred as "Do not reply" mode and suppresses transmission of Echo Reply packet. Default value (TBA5) for the Reply mode field requests the responder to send the Echo Reply packet out-of-band as IPv4 or IPv6 UDP packet. [Selection of destination and source IP addresses and UDP port numbers to be provided in the next update.]

2.6. Overlay Echo Reply Reception

3. IANA Considerations

3.1. Overlay Echo Request/Echo Reply Type

IANA is requested to assign new type from the Overlay OAM Protocol Types registry as follows:

Value	Description	Reference
TBA1	Overlay Echo Request/Echo Reply	This document

Table 1: Overlay Echo Request/Echo Reply Type

3.2. Overlay Ping Parameters

IANA is requested to create new Overlay Echo Request/Echo Reply Parameters registry.

3.3. Overlay Echo Request/Echo Reply Message Types

IANA is requested to create in the Overlay Echo Request/Echo Reply Parameters registry the new sub-registry Message Types. All code points in the range 1 through 191 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC5226] and assign values as follows:

Value	Description	Reference
0	Reserved	
TBA2	Overlay Echo Request	This document
TBA3	Overlay Echo Reply	This document
TBA3+1-191	Unassigned	IETF Review
192-251	Unassigned	First Come First Served
252-254	Unassigned	Private Use
255	Reserved	

Table 2: Overlay Echo Request/Echo Reply Message Types

3.4. Overlay Echo Reply Modes

IANA is requested to create in the Overlay Echo Request/Echo Reply Parameters registry the new sub-registry Reply Modes All code points in the range 1 through 191 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC5226] and assign values as follows:

Value	Description	Reference
0	Reserved	
TBA4	Do Not Reply	This document
TBA5	Reply via an IPv4/IPv6 UDP Packet	This document
TBA6	Reply via Application Level Control Channel	This document
TBA7	Reply via Specified Path	This document
TBA7+1-191	Unassigned	IETF Review
192-251	Unassigned	First Come First Served
252-254	Unassigned	Private Use
255	Reserved	

Table 3: Overlay Echo Reply Modes

4. Security Considerations

Overlay Echo Request/Reply operates within the domain of the overlay network and thus inherits any security considerations that apply to the use of that overlay technology and, consequently, underlay data plane. Also, the security needs for Overlay Echo Request/Reply are

similar to those of ICMP ping [RFC0792], [RFC4443] and MPLS LSP ping [I-D.ietf-mpls-rfc4379bis].

There are at least three approaches of attacking a node in the overlay network using the mechanisms defined in the document. One is a Denial-of-Service attack, by sending Overlay ping to overload a node in the overlay network. The second may use spoofing, hijacking, replying, or otherwise tampering with Overlay Echo Requests and/or Replies to misrepresent, alter operator's view of the state of the overlay network. The third is an unauthorized source using an Overlay Echo Request/Reply to obtain information about the overlay and/or underlay network.

To mitigate potential Denial-of-Service attacks, it is RECOMMENDED that implementations throttle the Overlay ping traffic going to the control plane.

Reply and spoofing attacks involving faking or replying Overlay Echo Reply messages would have to match the Sender's Handle and Sequence Number of an outstanding Overlay Echo Request message which is highly unlikely. Thus the non-matching reply would be discarded. But since "even a broken clock is right twice a day" implementations MAY use Timestamp control block [I-D.ooamdt-rtgwg-ooam-header] to validate the TimeStamp Sent by requiring an exact match on this field.

To protect against unauthorized sources trying to obtain information about the overlay and/or underlay an implementation MAY check that the source of the Echo Request is indeed part of the overlay domain.

5. Contributors

Work on this documented started by Overlay OAM Design Team with contributions from:

Carlos Pignataro

Cisco Systems, Inc.

cpignata@cisco.com

Santosh Pallagatti

santosh.pallagatti@gmail.com

Erik Nordmark

Arista Networks

nordmark@acm.org

Ignas Bagdonas

ibagdona@gmail.com

David Mozes

Mellanox Technologies Ltd.

davidm@mellanox.com

6. Acknowledgment

TBD

7. References

7.1. Normative References

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication in MPLS and non-MPLS Networks", draft-ietf-bier-mpls-encapsulation-06 (work in progress), December 2016.

[I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-03 (work in progress), September 2016.

[I-D.ietf-nvo3-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-nvo3-gue-05 (work in progress), October 2016.

[I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-03 (work in progress), October 2016.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-12 (work in progress), February 2017.

[I-D.ooamdt-rtgwg-ooam-header]
Mirsky, G., Kumar, N., Kumar, D., Chen, M., Yizhou, L., Mozes, D., and D. Dolson, "OAM Header for use in Overlay Networks", draft-ooamdt-rtgwg-ooam-header-02 (work in progress), February 2017.

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

7.2. Informative References

[I-D.ietf-mpls-rfc4379bis]
Kompella, K., Swallow, G., Pignataro, C., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", draft-ietf-mpls-rfc4379bis-09 (work in progress), October 2016.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <http://www.rfc-editor.org/info/rfc792>.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, DOI 10.17487/RFC4379, February 2006, <http://www.rfc-editor.org/info/rfc4379>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Nagendra Kumar
Cisco Systems, Inc.

Email: naikumar@cisco.com

Deepak Kumar
Cisco Systems, Inc.

Email: dekumar@cisco.com

Mach Chen
Huawei Technologies

Email: mach.chen@huawei.com

Yizhou Li
Huawei Technologies

Email: liyizhou@huawei.com

David Dolson
Sandvine

Email: ddolson@sandvine.com

NVO3 Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2018

G. Mirsky
ZTE Corp.
N. Kumar
D. Kumar
Cisco Systems, Inc.
M. Chen
Y. Li
Huawei Technologies
D. Dolson
Sandvine

March 18, 2018

OAM Header for use in Overlay Networks
draft-ooamdt-rtgwg-ooam-header-04

Abstract

This document introduces Overlay Operations, Administration, and Maintenance (OOAM) Header to be used in overlay networks to create Overlay Associated Channel (OAC) to ensure that OOAM control packets are in-band with user traffic and de-multiplex OOAM protocols.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 19, 2018.

Copyright Notice

Table of Contents

1. Introduction

New protocols that support overlay networks like VxLAN-GPE [I-D.ietf-nvo3-vxlan-gpe], GUE [I-D.ietf-nvo3-gue], Geneve [I-D.ietf-nvo3-geneve], BIER [RFC8296], and NSH [RFC8300] support multi-protocol payload, e.g. Ethernet, IPv4/IPv6, and recognize Operations, Administration, and Maintenance (OAM) as one of distinct types. That ensures that Overlay OAM (OOAM)packets are sharing fate with Overlay data packet traversing the underlay.

This document introduces generic requirements to OAM protocols used in overlay networks and defines OOAM Header to be used in overlay networks to de-multiplex OOAM protocols.

1.1. Conventions used in this document

1.1.1. Terminology

Term "Overlay OAM" used in this document interchangeably with longer version "set of OAM protocols, methods and tools for Overlay networks".

NTP Network Time Protocol

OAC Overlay Associated Channel

OAM Operations, Administration, and Maintenance

OOAM Overlay OAM

PTP Precision Time Protocol

1.1.2. Requirements Language

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

2. General Requirements to OAM Protocols in Overlay Networks

OAM protocols, whether it is part of fault management or performance monitoring, intended to provide reliable information that can be used to identify defect, localize it and apply corrective actions. One of the main challenges that network operators may encounter is interpretations of reports of the defect or service degradation and correlation to affected services. In order to improve reliability of the correlation process we set forth the following requirements:

REQ#1: Overlay OAM packets SHOULD be fate sharing with data traffic, i.e. in-band with the monitored traffic, i.e. follow exactly the same overlay and transport path as data plane traffic, in forward direction, i.e. from ingress toward egress end point(s) of the OAM test.

REQ#2: Encapsulation of OAM control message and data packets in underlay network MUST be indistinguishable from underlay network forwarding point of view.

REQ#3: Presence of OAM control message in overlay packet MUST be unambiguously identifiable.

REQ#4: It MUST be possible to express entropy for underlay Equal Cost Multipath in overlay encapsulation in order to avoid using data packet content by underlay transient nodes.

3. Associated Channel in Overlay Networks

Associated channel in the overlay network is the channel that, by using the same encapsulation as user traffic, follows the same path through the underlay network as user traffic. In other words, the associated channel is in-band with user traffic. Creating notion of the overlay associated channel (OAC) in the overlay network ensures that control packets of active OAM protocols carried in the OAC are in-band with user traffic. Additionally, OAC allows development of OAM tools that, from operational point of view, function in essentially the same manner in any type of overlay.

4. Overlay OAM Header

OOAM Header immediately follows the header of the overlay and identifies OAC. The format of the OOAM Header is:

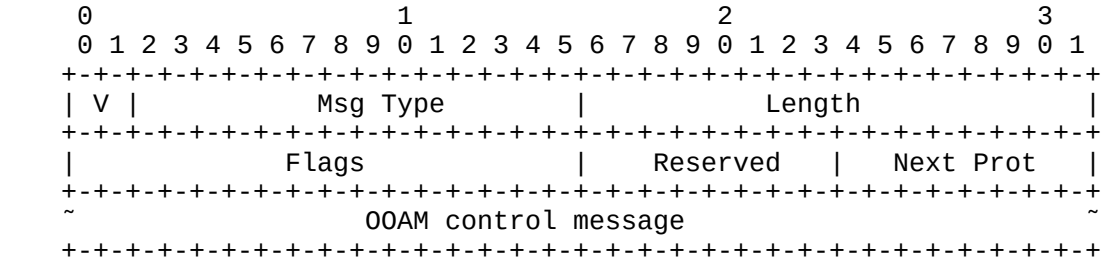


Figure 1: Overlay OAM Header format

The OAM Header consists of the following fields:

- V - two bits long field indicates the current version of the Overlay OAM Header. The current value is 0;
- Msg Type - 14 bits long field identifies OAM protocol, e.g. Echo Request/Reply, BFD, Performance Measurement;
- Length - two octets long field that is length of the OOAM control packet in octets;
- Flags -two octets long field carries bit flags that define optional capability and thus processing of the OOAM control packet;

- Reserved - one octet field that MUST be zeroed on transmit and ignored on receipt;

- Next Prot - one octet long field that defines optional payload that is present after the OOAM Control Packet.

The format of the Flags field is:

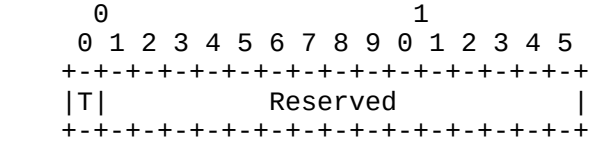


Figure 2: Flags field format

where:

- T - Timestap block flag.

- Reserved - must be set to all zeroes on transmission and ignored on receipt.

The OOAM header may be followed by the Timestamp control block Figure 3 and then by OOAM Control Packet identified by the Msg Type field.

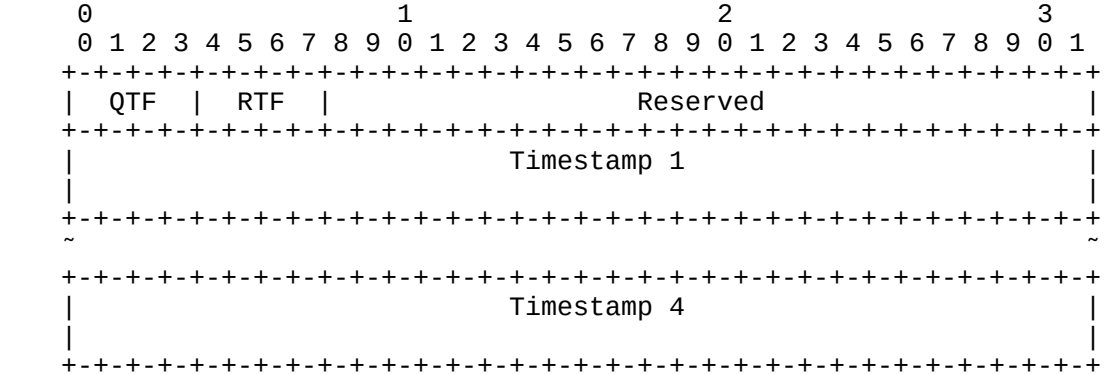


Figure 3: Timestamp block format

where:

QTF - Querier timestamp format

RTF - Responder timestamp format

Timestamp 1-4 - 64-bit timestamp values

Network Time Protocol (NTP), described in [RFC5905], is widely used and has long history of deployment. But it is the IEEE 1588 Precision Time Protocol (PTP) [IEEE.1588.2008] that is being broadly used to achieve high-quality clock synchronization. Converging between NTP and PTP time formats is possible but is not trivial and does come with cost, particularly when it is required to be performed in real time without loss of accuracy. And recently protocols that supported only NTP time format, like One-Way Active Measurement Protocol [RFC4656] and Two-Way Active Measurement Protocol [RFC5357], have been enhanced to support the PTP time format as well [RFC8186]. This document proposes to select PTP time format as default time format for Overlay OAM performance measurement. Hence QTF, RTF fields MUST be set to 0 if querier or responder use PTP time format respectively. If the querier or responder use the NTP time format, then QTF and/or RTF MUST be set to 1. Use of other values MUST be considered as error and MAY be reported.

4.1. Use of OOAM Header in Active OAM

Active OAM methods, whether used for fault management or performance monitoring, generate dedicated test packets [RFC7799]. Format of an OAM test packet in overlay network presented in Figure 4.

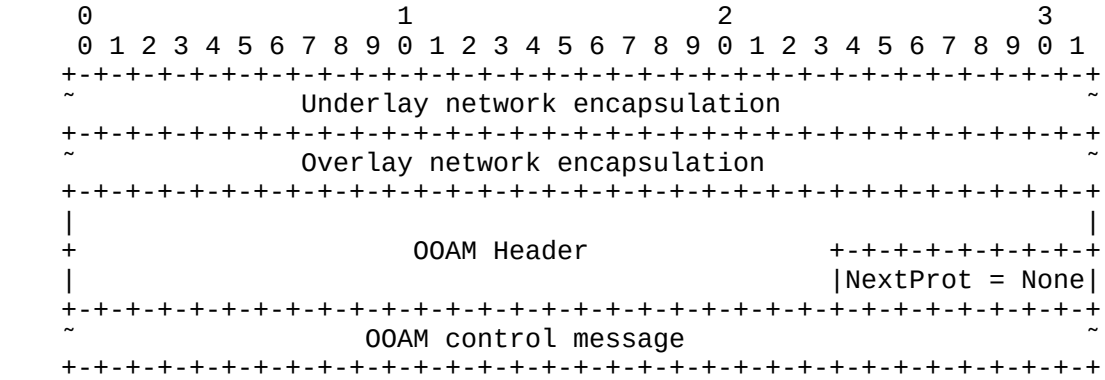


Figure 4: Overlay OAM Header in Active OAM Control Packet

Because active OAM method uses only OAM protocol value of Next Prot field in the OOAM header is set to None indicating that there’s no content from other protocol immediately after OOAM control message in the packet.

4.2. Use of OOAM Header in Hybrid OAM

Hybrid OAM Type I methods, whether used for fault management or performance monitoring, modify user data packets [RFC7799]. Format of such modified packet in overlay network presented in Figure 5.

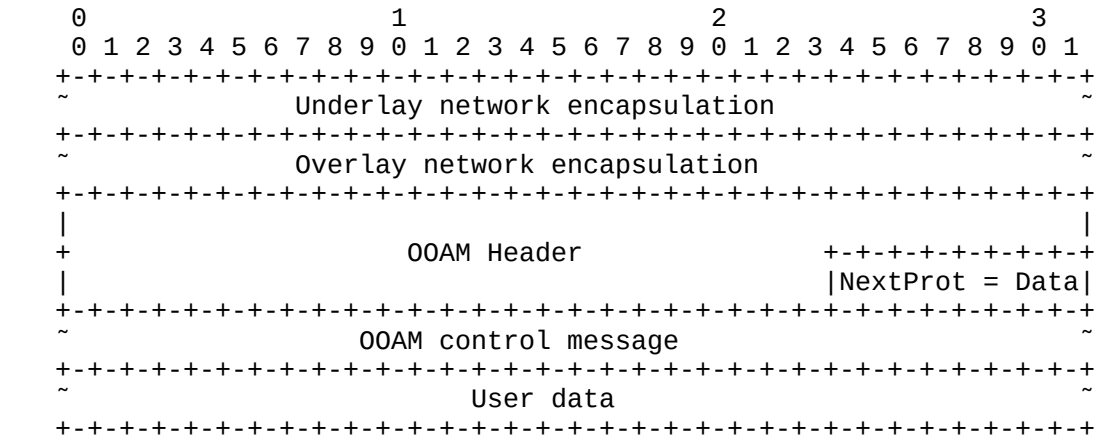


Figure 5: Overlay OAM Header in Hybrid OAM Control Packet

In case when OOAM header used for Hybrid Type I OAM method value of the Next Prot field is set to the value associated with the protocol of the user data.

5. IANA Considerations

IANA is requested to create new registry called "Overlay OAM".

5.1. OOAM Message Types

IANA is requested to create new sub-registry called "Overlay OAM Protocol Types" in the "Overlay OAM" registry. All code points in the range 1 through 15615 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC8126] . Remaining code points are allocated according to the Table 1:

Value	Description	Reference
0	Reserved	
1 - 15615	Unassigned	IETF Review
15616 - 16127	Unassigned	First Come First Served
16128 - 16143	Experimental	This document
16144 - 16382	Private Use	This document
16383	Reserved	This document

Table 1: Overlay OAM Protocol type

5.2. OOAM Header Flags

IANA is requested to create sub-registry "Overlay OAM Header Flags" in "Overlay OAM" registry. Two flags are defined in this document. New values are assigned via Standards Action [RFC8126].

Flags bit	Description	Reference
Bit 0	Timestamp field	This document
Bit 1-15	Unassigned	

Table 2: Overlay OAM Flags

6. Security Considerations

TBD

7. Contributors

Work on this documented started by Overlay OAM Design Team with contributions from:

Carlos Pignataro

Cisco Systems, Inc.

cpignata@cisco.com

Erik Nordmark

Arista Networks

nordmark@acm.org

Ignas Bagdonas

ibagdona@gmail.com

David Mozes

Mellanox Technologies Ltd.

davidm@mellanox.com

8. Acknowledgement

TBD

9. References

9.1. Normative References

[IEEE.1588.2008]
"Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Standard 1588, July 2008.

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

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

9.2. Informative References

[I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-06 (work in progress), March 2018.

[I-D.ietf-nvo3-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-nvo3-gue-05 (work in progress), October 2016.

[I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-05 (work in progress), October 2017.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <https://www.rfc-editor.org/info/rfc4656>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <https://www.rfc-editor.org/info/rfc5357>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8186] Mirsky, G. and I. Meilik, "Support of the IEEE 1588 Timestamp Format in a Two-Way Active Measurement Protocol (TWAMP)", RFC 8186, DOI 10.17487/RFC8186, June 2017, <https://www.rfc-editor.org/info/rfc8186>.

[RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <https://www.rfc-editor.org/info/rfc8296>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Nagendra Kumar
Cisco Systems, Inc.

Email: naikumar@cisco.com

Deepak Kumar
Cisco Systems, Inc.

Email: dekumar@cisco.com

Mach Chen
Huawei Technologies

Email: mach.chen@huawei.com

Yizhou Li
Huawei Technologies

Email: liyizhou@huawei.com

David Dolson
Sandvine

Email: ddolson@sandvine.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2017

X. Liu
Kuatro Technologies
Y. Qu
A. Lindem
Cisco Systems
C. Hopps
Deutsche Telekom
L. Berger
LabN Consulting, L.L.C.

October 28, 2016

Routing Area Common YANG Data Types
draft-rtgyangdt-rtgwg-routing-types-00

Abstract

This document defines a collection of common data types using YANG data modeling language. These derived common types are designed to be imported by other modules defined in the routing area.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 1, 2017.

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC6020] [RFC7950] is a data modeling language used to model configuration data, state data, Remote Procedure Calls, and notifications for network management protocols. The YANG language supports a small set of built-in data types and provides mechanisms to derive other types from the built-in types.

This document introduces a collection of common data types derived from the built-in YANG data types. The derived types are designed to be the common types applicable for modeling in the routing area.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

1.2. Terminology

The terminology for describing YANG data models is found in [RFC7950].

2. Overview

This document defines the following data types:

router-id
: Router Identifiers are commonly used to identify a nodes in routing and other control plane protocols. An example usage of router-id can be found in [I-D.ietf-ospf-yang].

address-family
: This type defines values for use in address family identifiers. The values are based on the IANA Address Family Numbers Registry [1]. An example usage can be found in [I-D.ietf-idr-bgp-model].

route-target
: Route Targets (RTs) are commonly used to control the distribution of virtual routing and forwarding (VRF) information, see [RFC4364], in support of virtual private networks (VPNs). An example usage can be found in [I-D.ietf-idr-bgp-model] and

route-distinguisher
: Route Distinguishers (RDs) are commonly used to identify separate routes in support of virtual private networks (VPNs). For example, in [RFC4364], RDs are commonly used to identify independent VPNs and VRFs, and more generally, to identify multiple routes to the same prefix. An example usage can be found in [I-D.ietf-idr-bgp-model].

ieee-bandwidth
: Bandwidth in IEEE 754 floating point 32-bit binary format [IEEE754]. Commonly used in Traffic Engineering control plane protocols. An example of where this type may/will be used is [I-D.ietf-ospf-yang].

link-access-type
: This type identifies the IGP link type. An example of where this type may/will be used is [I-D.ietf-ospf-yang].

multicast-source-ipv4-addr-type
: IPv4 source address type for use in multicast control protocols. This type also allows the indication of wildcard sources, i.e., "*". An example of where this type may/will be used is [I-D.ietf-pim-yang].

multicast-source-ipv6-addr-type
: IPv6 source address type for use in multicast control protocols. This type also allows the indication of wildcard sources, i.e.,

"*". An example of where this type may/will be used is [I-D.ietf-pim-yang].

timer-multiplier
This type is used in conjunction with a timer-value type. It is generally used to indicate define the number of timer-value intervals that may expire before a specific event must occur. Examples of this include the arrival of any BFD packets, see [RFC5880] Section 6.8.4, or hello_interval in [RFC3209]. Example of where this type may/will be used is [I-D.ietf-idr-bgp-model] and [I-D.ietf-teas-yang-rsvp].

timer-value-seconds16
This type covers timers which can be set in seconds, not set, or set to infinity. This type supports a range of values that can be represented in a uint16 (2 octets). An example of where this type may/will be used is [I-D.ietf-ospf-yang].

timer-value-seconds32
This type covers timers which can be set in seconds, not set, or set to infinity. This type supports a range of values that can be represented in a uint32 (4 octets). An example of where this type may/will be used is [I-D.ietf-teas-yang-rsvp].

timer-value-milliseconds
This type covers timers which can be set in milliseconds, not set, or set to infinity. This type supports a range of values that can be represented in a uint32 (4 octets). Examples of where this type may/will be used include [I-D.ietf-teas-yang-rsvp] and [I-D.ietf-bfd-yang].

3. YANG Module

<CODE BEGINS> file "ietf-routing-types@2016-10-28.yang"

```
module ietf-routing-types {

  namespace "urn:ietf:params:xml:ns:yang:ietf-routing-types";
  prefix "rt-types";

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

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

  organization "IETF Routing Area Working Group (rtgwg)";
```

```
     contact
         "Routing Area Working Group - <rtgwg@ietf.org>";

     description
       "This module contains a collection of YANG data types
        considered generally useful for routing protocols.";

     revision 2016-10-28 {
       description
         "Initial revision.";
       reference
         "RFC TBD: Routing YANG Data Types";
     }

     /*** collection of types related to routing ***/
     typedef router-id {
       type yang:dotted-quad;
       description
         "A 32-bit number in the dotted quad format assigned to each
          router. This number uniquely identifies the router within an
          Autonomous System.";
     }

     // address-family
     identity address-family {
       description
         "Base identity from which identities describing address
          families are derived.";
     }

     identity ipv4 {
       base address-family;
       description
         "This identity represents IPv4 address family.";
     }

     identity ipv6 {
       base address-family;
       description
         "This identity represents IPv6 address family.";
     }

     //The rest of the values deinfed in the IANA registry

     identity nsap {
       base address-family;
       description
         "Address family from IANA registry.";
```

```
  }
  identity hdlc {
    base address-family;
    description
      "(8-bit multidrop)
       Address family from IANA registry.";
  }
  identity bbn1822 {
    base address-family;
    description
      "AHIP (BBN report #1822)
       Address family from IANA registry.";
  }
  identity ieee802 {
    base address-family;
    description
      "(includes all 802 media plus Ethernet canonical format)
       Address family from IANA registry.";
  }
  identity e163 {
    base address-family;
    description
      "Address family from IANA registry.";
  }
  identity e164 {
    base address-family;
    description
      "SMDS, Frame Relay, ATM
       Address family from IANA registry.";
  }
  identity f69 {
    base address-family;
    description
      "(Telex)
       Address family from IANA registry.";
  }
  identity x121 {
    base address-family;
    description
      "(X.25, Frame Relay)
       Address family from IANA registry.";
  }
  identity ipx {
    base address-family;
    description
      "Address family from IANA registry.";
  }
  identity appletalk {
```

```
      base address-family;
      description
        "Address family from IANA registry.";
    }
    identity decnet-iv {
      base address-family;
      description
        "Decnet IV
         Address family from IANA registry.";
    }
    identity vines {
      base address-family;
      description
        "Banyan Vines
         Address family from IANA registry.";
    }
    identity e164-nsap {
      base address-family;
      description
        "E.164 with NSAP format subaddress
         Address family from IANA registry.";
    }
    identity dns {
      base address-family;
      description
        "Domain Name System
         Address family from IANA registry.";
    }
    identity dn {
      base address-family;
      description
        "Distinguished Name
         Address family from IANA registry.";
    }
    identity as-num {
      base address-family;
      description
        "AS Number
         Address family from IANA registry.";
    }
    identity xtp-v4 {
      base address-family;
      description
        "XTP over IPv4
         Address family from IANA registry.";
    }
    identity xtp-v6 {
      base address-family;
```

```
      description
        "XTP over IPv6
         Address family from IANA registry.";
    }
    identity xtp {
      base address-family;
      description
        "XTP native mode XTP
         Address family from IANA registry.";
    }
    identity fc-port {
      base address-family;
      description
        "Fibre Channel World-Wide Port Name
         Address family from IANA registry.";
    }
    identity fc-node {
      base address-family;
      description
        "Fibre Channel World-Wide Node Name
         Address family from IANA registry.";
    }
    identity gwid {
      base address-family;
      description
        "Address family from IANA registry.";
    }
    identity l2vpn {
      base address-family;
      description
        "Address family from IANA registry.";
    }
    identity mpls-tp-section-eid {
      base address-family;
      description
        "MPLS-TP Section Endpoint Identifier
         Address family from IANA registry.";
    }
    identity mpls-tp-lsp-eid {
      base address-family;
      description
        "MPLS-TP LSP Endpoint Identifier
         Address family from IANA registry.";
    }
    identity mpls-tp-pwe-eid {
      base address-family;
      description
        "MPLS-TP Pseudowire Endpoint Identifier
```

```
          Address family from IANA registry.";
      }
      identity mt-v4 {
        base address-family;
        description
          "Multi-Topology IPv4.
           Address family from IANA registry.";
      }
      identity mt-v6 {
        base address-family;
        description
          "Multi-Topology IPv6.
           Address family from IANA registry.";
      }

      /*** collection of types related to VPN ***/
      typedef route-target {
        type string {
          pattern
            '(0:(6553[0-5]|655[0-2]\d|65[0-4]\d{2}|6[0-4]\d{3}|'
          + '[0-5]?\d{0,3}\d):(429496729[0-5]|42949672[0-8]\d|'
          + '4294967[01]\d{2}|429496[0-6]\d{3}|42949[0-5]\d{4}|'
          + '4294[0-8]\d{5}|429[0-3]\d{6}|42[0-8]\d{7}|4[01]\d{8}|'
          + '[0-3]?\d{0,8}\d))|'
          + '(1:(((\d|[1-9]\d|1\d{2}|2[0-4]\d|25[0-5])\.){3}(\d|[1-9]\d|'
          + '1\d{2}|2[0-4]\d|25[0-5])):(6553[0-5]|655[0-2]\d|'
          + '65[0-4]\d{2}|6[0-4]\d{3}|[0-5]?\d{0,3}\d))|'
          + '(2:(429496729[0-5]|42949672[0-8]\d|4294967[01]\d{2}|'
          + '429496[0-6]\d{3}|42949[0-5]\d{4}|4294[0-8]\d{5}|'
          + '429[0-3]\d{6}|42[0-8]\d{7}|4[01]\d{8}|[0-3]?\d{0,8}\d):'
          + '(6553[0-5]|655[0-2]\d|65[0-4]\d{2}|6[0-4]\d{3}|'
          + '[0-5]?\d{0,3}\d))';
        }
        description
          "Route target has a similar format to route distinguisher.
           A route target consists of three fields:
           a 2-byte type field, an administrator field,
           and an assigned number field.
           According to the data formats for type 0, 1, and 2 defined in
           RFC4360, the encoding pattern is defined as:

           0:2-byte-asn:4-byte-number
           1:4-byte-ipv4addr:2-byte-number
           2:4-byte-asn:2-byte-number.

           Some valid examples are: 0:100:100, 1:1.1.1.1:100, and
           2:1234567890:203.";
        reference
```

```
        "RFC4360: BGP Extended Communities Attribute.";
      }

      typedef route-distinguisher {
        type string {
          pattern
            '(0:(6553[0-5]|655[0-2]\d|65[0-4]\d{2}|6[0-4]\d{3}|'
          + '[0-5]?\d{0,3}\d):(429496729[0-5]|42949672[0-8]\d|'
          + '4294967[01]\d{2}|429496[0-6]\d{3}|42949[0-5]\d{4}|'
          + '4294[0-8]\d{5}|429[0-3]\d{6}|42[0-8]\d{7}|4[01]\d{8}|'
          + '[0-3]?\d{0,8}\d))|'
          + '(1:(((\d|[1-9]\d|1\d{2}|2[0-4]\d|25[0-5])\.){3}(\d|[1-9]\d|'
          + '1\d{2}|2[0-4]\d|25[0-5])):(6553[0-5]|655[0-2]\d|'
          + '65[0-4]\d{2}|6[0-4]\d{3}|[0-5]?\d{0,3}\d))|'
          + '(2:(429496729[0-5]|42949672[0-8]\d|4294967[01]\d{2}|'
          + '429496[0-6]\d{3}|42949[0-5]\d{4}|4294[0-8]\d{5}|'
          + '429[0-3]\d{6}|42[0-8]\d{7}|4[01]\d{8}|[0-3]?\d{0,8}\d):'
          + '(6553[0-5]|655[0-2]\d|65[0-4]\d{2}|6[0-4]\d{3}|'
          + '[0-5]?\d{0,3}\d))|'
          + '(([3-9a-fA-F]|[1-9a-fA-F][\da-fA-F]{1,3}):'
          + '[\da-fA-F]{1,12})';
        }
        description
          "Route distinguisher has a similar format to route target.
           An route distinguisher consists of three fields:
           a 2-byte type field, an administrator field,
           and an assigned number field.
           According to the data formats for type 0, 1, and 2 defined in
           RFC4364, the encoding pattern is defined as:

           0:2-byte-asn:4-byte-number
           1:4-byte-ipv4addr:2-byte-number
           2:4-byte-asn:2-byte-number.
           2-byte-other-hex-number:6-byte-hex-number

           Some valid examples are: 0:100:100, 1:1.1.1.1:100, and
           2:1234567890:203.";
        reference
          "RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs).";
      }

      /*** collection of types common to protocols ***/
      typedef ieee-bandwidth {
        type string {
          pattern
            '0[xX](0((\.0?)?[pP](\+)?0?|(\.0?))|'
          + '1(\.([\da-fA-F]{0,5}[02468aAcCeE]?)?)?[pP](\+)?(12[0-7]|'
          + '1[01]\d|0?\d?\d)?)|0[xX][\da-fA-F]{1,8}';
```

```
      }
      description
        "Bandwidth in IEEE 754 floating point 32-bit binary format:
         (-1)**(S) * 2**(Exponent-127) * (1 + Fraction),
         where Exponent uses 8 bits, and Fraction uses 23 bits.
         The units are bytes per second.
         The encoding format is the external hexadecimal-significand
         character sequences specified in IEEE 754 and C99,
         restricted to be normalized, non-negative, and non-fraction:
         0x1.hhhhhhp{+}d or 0X1.HHHHHHP{+}D
         where 'h' and 'H' are hexadecimal digits, 'd' and 'D' are
         integers in the range of [0..127].
         When six hexadecimal digits are used for 'hhhhhh' or 'HHHHHH',
         the least significant digit must be an even number.
         'x' and 'X' indicate hexadecimal; 'p' and 'P' indicate power
         of two.
         Some examples are: 0x0p0, 0x1p10, and 0x1.abcde2p+20";
      reference
        "IEEE Std 754-2008: IEEE Standard for Floating-Point
         Arithmetic.";
    }

    typedef link-access-type {
      type enumeration {
        enum "broadcast" {
          description
          "Specify broadcast multi-access network.";
        }
        enum "non-broadcast" {
          description
          "Specify Non-Broadcast Multi-Access (NBMA) network.";
        }
        enum "point-to-multipoint" {
          description
          "Specify point-to-multipoint network.";
        }
        enum "point-to-point" {
          description
          "Specify point-to-point network.";
        }
      }
      description
        "Link access type.";
    }

    typedef multicast-source-ipv4-addr-type {
      type union {
        type enumeration {
```

```
          enum '*' {
            description
            "Any source address.";
          }
        }
        type inet:ipv4-address;
      }
      description
        "Multicast source IP address type.";
    }

    typedef multicast-source-ipv6-addr-type {
      type union {
        type enumeration {
          enum '*' {
            description
            "Any source address.";
          }
        }
        type inet:ipv6-address;
      }
      description
        "Multicast source IP address type.";
    }

    typedef timer-multiplier {
      type uint8;
      description
        "The number of timer value intervals that should be
         interpreted as a failure.";
    }

    typedef timer-value-seconds16 {
      type union {
        type uint16 {
          range "1..65535";
          }
        type enumeration {
          enum "infinity" {
            description "The timer is set to infinity.";
          }
          enum "no-expiry" {
            description "The timer is not set.";
          }
        }
      }
      units seconds;
      description "Timer value type, in seconds (16 bit range).";
```

```
    }

    typedef timer-value-seconds32 {
      type union {
        type uint32 {
          range "1..4294967295";
          }
        type enumeration {
          enum "infinity" {
            description "The timer is set to infinity.";
          }
          enum "no-expiry" {
            description "The timer is not set.";
          }
        }
      }
      units seconds;
      description "Timer value type, in seconds (32 bit range).";
    }

    typedef timer-value-milliseconds {
      type union {
        type uint32{
          range "1..4294967295";
          }
        type enumeration {
          enum "infinity" {
            description "The timer is set to infinity.";
          }
          enum "no-expiry" {
            description "The timer is not set.";
          }
        }
      }
      units milliseconds;
      description "Timer value type, in milliseconds.";
    }
  }
  <CODE ENDS>
```

4. IANA Considerations

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

```
---------------------------------------------------------------------
URI: urn:ietf:params:xml:ns:yang:ietf-routing-types
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
---------------------------------------------------------------------
```

This document registers the following YANG modules in the YANG Module Names registry [RFC6020]:

```
---------------------------------------------------------------------
name:         ietf-routing-types
namespace:    urn:ietf:params:xml:ns:yang:ietf-routing-types
prefix:       rt-types
reference:    RFC XXXX
---------------------------------------------------------------------
```

5. Security Considerations

This document defines common data types using the YANG data modeling language. The definitions themselves have no security impact on the Internet, but the usage of these definitions in concrete YANG modules might have. The security considerations spelled out in the YANG specification [RFC7950] apply for this document as well.

6. Acknowledgements

The Routing Area Yang Architecture design team members included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Ebben Aries, Lou Berger, Qin Wu, Rob Shakir, Xufeng Liu, and Yingzhen Qu.

7. References

7.1. Normative References

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

[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>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

7.2. Informative References

[IEEE754] IEEE, "IEEE Standard for Floating-Point Arithmetic", IEEE Std 754-2008, August 2008.

[I-D.ietf-bfd-yang] Zheng, L., Rahman, R., Networks, J., Jethanandani, M., and G. Mirsky, "Yang Data Model for Bidirectional Forwarding Detection (BFD)", draft-ietf-bfd-yang-03 (work in progress), July 2016.

[I-D.ietf-idr-bgp-model] Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Zhdankin, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-ietf-idr-bgp-model-02 (work in progress), July 2016.

[I-D.ietf-ospf-yang] Yeung, D., Qu, Y., Zhang, Z., Bogdanovic, D., and K. Koushik, "Yang Data Model for OSPF Protocol", draft-ietf-ospf-yang-05 (work in progress), July 2016.

[I-D.ietf-pim-yang] Liu, X., McAllister, P., Peter, A., Sivakumar, M., Liu, Y., and f. hu, "A YANG data model for Protocol-Independent Multicast (PIM)", draft-ietf-pim-yang-03 (work in progress), October 2016.

[I-D.ietf-teas-yang-rsvp] Beeram, V., Saad, T., Gandhi, R., Liu, X., Shah, H., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Resource Reservation Protocol (RSVP)", draft-ietf-teas-yang-rsvp-04 (work in progress), October 2016.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <http://www.rfc-editor.org/info/rfc3209>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <http://www.rfc-editor.org/info/rfc4364>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <http://www.rfc-editor.org/info/rfc5880>.

7.3. URIs

[1] http://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml

Authors' Addresses

Xufeng Liu
Kuatro Technologies
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA

EMail: xliu@kuatrotech.com

Yingzhen Qu
Cisco Systems
170 West Tasman Drive
San Jose CA 95134
USA

EMail: yiqu@cisco.com

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

EMail: acee@cisco.com

Christian Hopps
Deutsche Telekom

EMail: chopps@chopps.org

Lou Berger
LabN Consulting, L.L.C.

EMail: lberger@labn.net