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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 29 September 2024

Y. Luo
L. Qu
China Telcom Co., Ltd.
X. Huang
Tencent
G. Mishra
Verizon Inc.
H. Chen
Futurewei
S. Zhuang
Z. Li
Huawei

28 March 2024

Architecture for Use of BGP as Central Controller
draft-cth-rtgwg-bgp-control-13

Abstract

BGP is a core part of a network including Software-Defined Networking (SDN) system. It has the traffic engineering information on the network topology and can compute optimal paths for a given traffic flow across the network.

This document describes some reference architectures for BGP as a central controller. A BGP-based central controller can simplify the operations on the network and use network resources efficiently for providing services with high quality.

Requirements Language

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

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

Border Gateway Protocol (BGP) [RFC1771] is an exterior gateway protocol (EGP). It is developed to exchange routing information among routers in different autonomous systems (ASes). Along its developments, BGP has been extended to provide numerous new functions. It collects the link states including traffic engineering (TE) information from other protocols such as IGP and distributes them among routers in different ASes [RFC7752]. It also controls the redirection of traffic flows [RFC5575]. Furthermore, it distributes MPLS labels [RFC3107]. For scalability, BGP is extended to have Route Reflector (RR) [RFC4456].

For segment routing (SR), BGP is extended to advertise SR policies with candidate paths to the policy headend routers, which are typically ingress routers [I-D.ietf-idr-segment-routing-te-policy]. The SR specific PCEP extensions are defined in [I-D.ietf-pce-segment-routing]. A stateful PCE can compute an SR traffic engineering (SR-TE) path satisfying a set of constraints, and initiate an SR-TE path on a headend router using the extensions.

An SDN controller (or controller for short) is the core of an SDN system or network. It is between network elements (NEs) such as routers or switches at one end and applications such as Operational Support System (OSS) or Network Management System (NMS) at the other end. The essential function of a controller is to steer traffic flows across the network for providing more services with higher quality. It manages network resources such as link bandwidth, computes expected paths for carrying traffic flows based on available network resources, programs the network elements for the creation of tunnels along the paths, and redirects traffic flows into corresponding tunnels.

Based on the current BGP, it is natural, beneficial and relatively simple to extend BGP to become a controller. Using BGP as a controller for a network will greatly simplify the operations on the network. It avoids deploying, operating and maintaining a new extra component or protocol such as PCE as a controller in the network.

This document describes some reference architectures for BGP as a central controller and introduces some scenarios to which the BGP controller can be applied.

2. Terminology

* SR: Segment Routing
* RR: Route Reflector
* SID: Segment Identifier
* SR-Path: Segment Routing Path
* SR-Tunnel: Segment Routing Tunnel
* TEDB: Traffic Engineering Database
* LSDB: Link State Database
* SLDB: SID/Label Database
* TPDB: Tunnel and Path Database
* CSPF: Constrained Shortest Path First
* TM: Tunnel Manager
* NMS: Network Management System
* SRLB: SR Local Block
* NE: Network Element
* PCE: Path Computation Element
* AS: Autonomous System
* QoS: Quality of Service
* ISP: Internet Service Provider
* MAN: Metropolitan Area Network
* OTT: Over the Top
* OTTSP: Over the Top Service Provider, or Content Operator
* AR: Access Router

3. Architectures

An architecture for the use of BGP as a central controller is based on the essential function of a controller. It is constructed from some building blocks or components. After introduction to building blocks, a few of reference architectures are described in this section.

3.1. Building Blocks

Some critical building blocks are briefed. They are Traffic Engineering Database (TEDB or TED for short), SID/Label Database (SLDB), Tunnel and Path Database (TPDB), Constrained Shortest Path First (CSPF), and Tunnel Manager (TM).

3.1.1. TEDB

The Traffic Engineering Database (TEDB) stores the Traffic Engineering (TE) information about the network. It includes the unreserved bandwidth at each of eight priority levels for every link in the network.

TEDB can be an individual block, which is constructed from the link state information received. It may be embedded into the link state database (LSDB) in the BGP when the BGP creates/updates the LSDB from the link state information it receives.

3.1.2. SLDB

The SID/Label Database (SLDB) records and maintains the status of every Segment Identifier (SID) and label for every node, interface/ link and/or prefix in the network, which the controller controls. The status of SID/label indicates whether the SID/Label is assigned. If it is assigned, then the object such as the node, link or prefix, to which it is assigned, is recorded.

SLDB can be an individual block, which is constructed from the link state information such as SR Local Block (SRLB) that the BGP receives. It may be embedded into the link state database (LSDB) in the BGP when the BGP creates the LSDB from the link state information it receives.

3.1.3. TPDB

The Tunnel and Path Database (TPDB) stores the information for every tunnel, which includes:

- the parameters received for the tunnel from a user/application,

- the path computed for the tunnel,
- the resources such as link bandwidth reserved along the path for the tunnel,
- the SID/labels assigned along the path for the tunnel, and
- the status of the tunnel.

3.1.4. CSPF

The Constrained Shortest Path First (CSPF) computes a path for a tunnel such as SR tunnel or LSP tunnel that satisfies a set of given constraints using the information in TEDB.

3.1.5. TM

The Tunnel Manager (TM) receives a request for an operation on a tunnel from a user or an application such as Network Management System (NMS). The operation may be a creation of a new tunnel, a deletion of an existing tunnel, or a change to an existing tunnel.

When receiving a request for creating a new tunnel, the TM asks the CSPF to compute a path for the tunnel that satisfies the constraints given for the tunnel.

After obtaining the path for the tunnel from the CSPF, the TM requests the SLDB to assign SID/labels along the path for the tunnel and asks the TEDB to reserve the resources such as link bandwidth along the path for the tunnel.

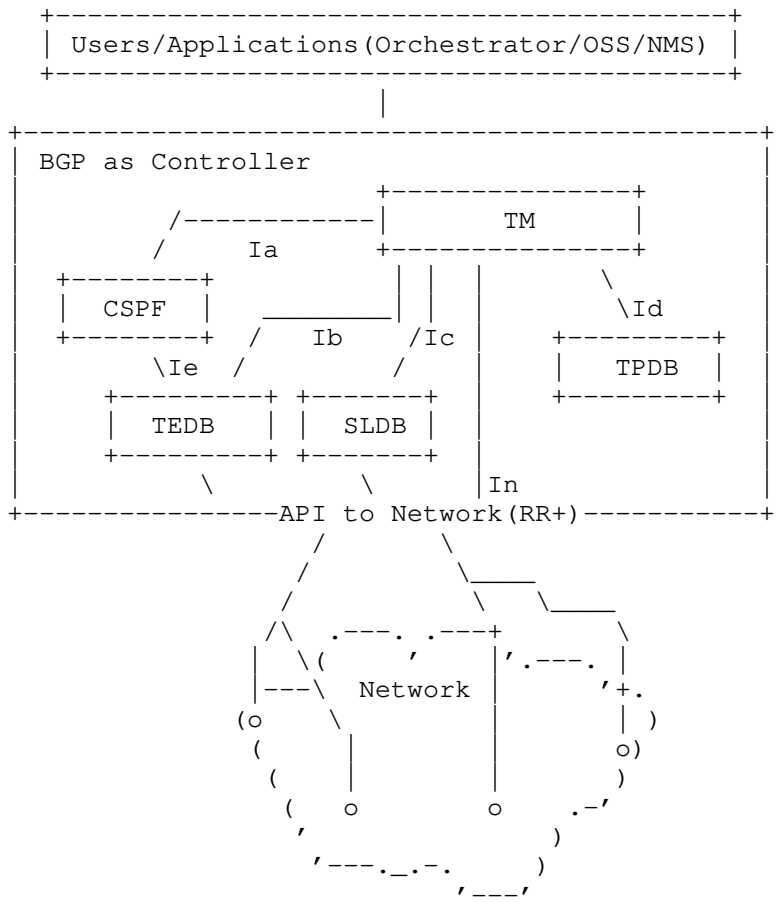
The TM in a central controller may set up the tunnel along the path in the network by programming each of the NEs along the path through the API to the network. In a SR network, the TM initiates a SR tunnel in the network by sending a sequence of SID/labels to the source NE of the tunnel.

The TM records the information for the tunnel in the Tunnel and Path Database (TPDB). The information includes the path computed for the tunnel, the resources such as bandwidth reserved along the path, the SID/labels assigned along the path for the tunnel, and the status of the tunnel.

3.2. One Controller

Figure below illustrates a reference architecture for using the BGP as a central controller, which controls a network. The BGP as a controller in the reference architecture controls a network through an API to the network such as BGP+/RR+ (extensions to BGP for central controller). The BGP controller is responsible for creating and maintaining every tunnel in the network. It also controls the redirection of traffic flow to each tunnel.

The BGP controller comprises a number of modules, including a TM, a CSPF, a TEDB, a SLDB and a TPDB. The interfaces among these modules are listed as follows:



* Interface Ia between the TM and the CSPF. Through this interface, the TM requests the CSPF to compute a path for a tunnel with a set of constraints, and the CSPF responses the TM with the path computed that satisfies the constraints.

* Interface Ib between the TM and the TEDB. When a tunnel is to be created, through this interface, the TM reserves in the TEDB the TE resources such as link bandwidths on every link along the path computed for the tunnel. When a tunnel is deleted, the TM releases the TE resources such as link bandwidths on every link along the path for the tunnel.

* Interface Ic between the TM and the SLDB. When a tunnel is to be created, through this interface, the TM reserves in the SLDB a SID/label for every link or some links along the path computed for the tunnel. When a tunnel is deleted, the TM releases the SID/ label for every link or some links along the path for the tunnel.

* Interface Id between the TM and the TPDB. the TM updates the information for every tunnel in the TPDB through this interface.

* Interface Ie between the CSPF and the TEDB. Through this interface, the CSPF accesses the traffic engineering information such as link bandwidths when it computes a path for a tunnel.

There is an interface In between the BGP controller and the network. In fact, there is a control channel (or interface) between the BGP controller and every (edge) node in the network.

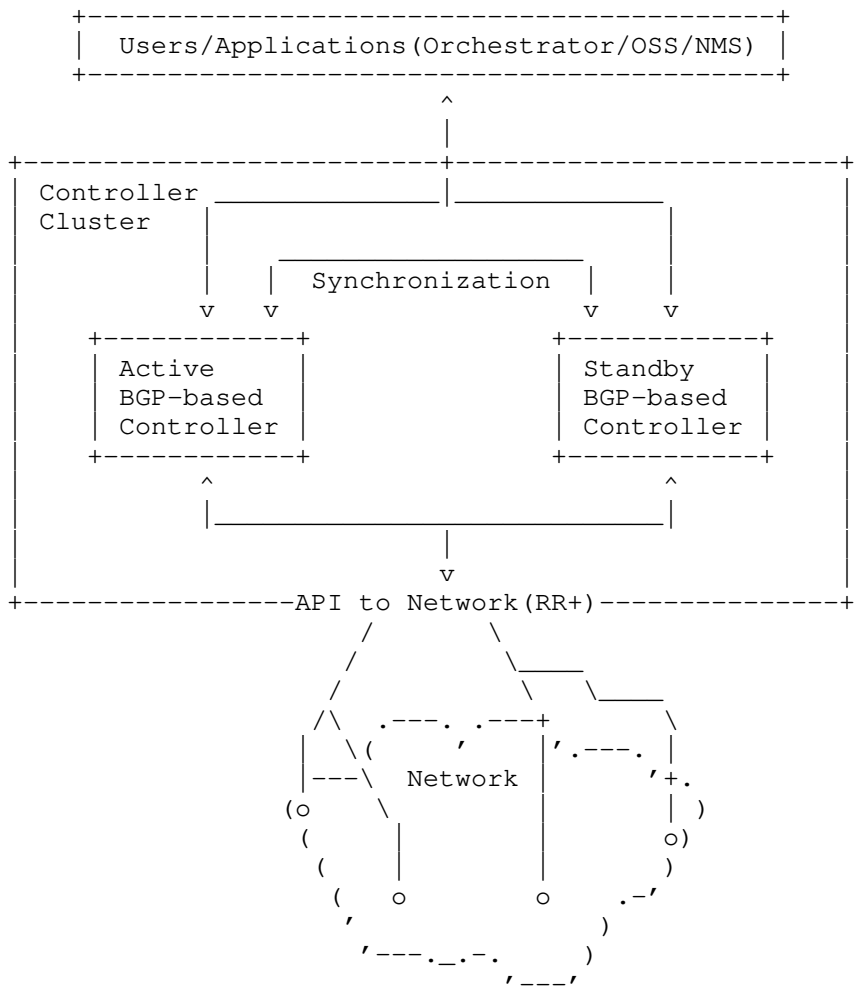
Initially, the TEDB obtains the original traffic engineering (TE) information such as link bandwidths from the network through the interface In (i.e., API to network) for every link in the network. The SLDB gets the original SID/label resources from the network through the interface for every node, link and prefix in the network.

3.3. Controller Cluster

A critical issue in a network with a central controller is the failure of the controller, which is a single point of failure (SPOF). If the controller fails, the entire network may not work.

A controller cluster (i.e., a group of controllers) works as a single controller from user's point of view. A simple controller cluster consists of two controllers. One works as a active (or say primary) controller, and the other as a standby (or say secondary) controller. In normal operations, the active controller is responsible for the network it controls. It also synchronizes with the standby controller. When the active controller fails, the standby controller becomes a new active controller, which controls the network.

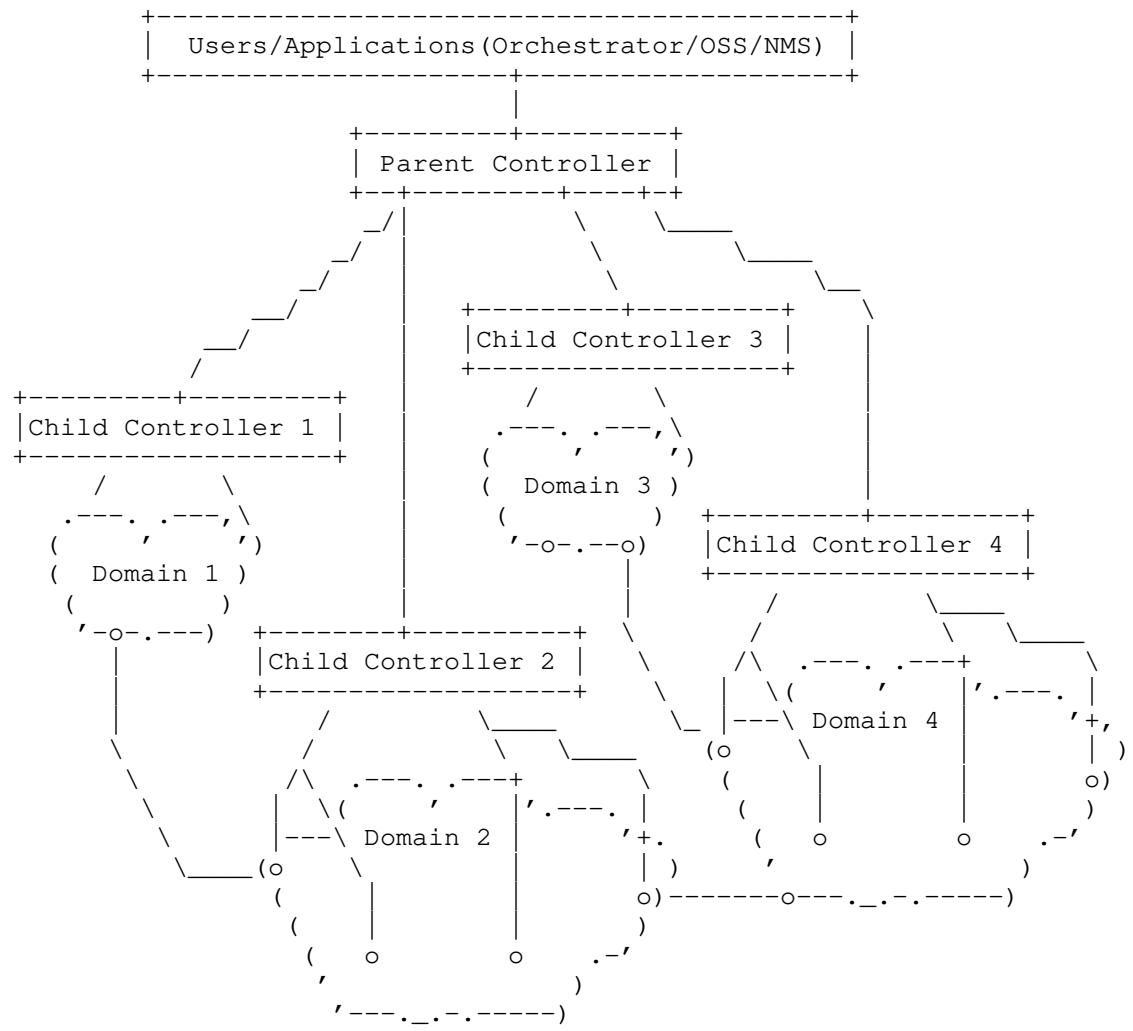
The Figure below illustrates a simple controller cluster containing two BGP-based controllers: Active BGP-based Controller and Standby BGP-based Controller. In normal operations, the active controller interacts with users and/or applications. For example, it receives configurations for tunnels and the traffic flows to tunnels from users. The active controller instructs the network elements in the network to provide the services requested by users and/or applications. For example, after receiving the configurations for a tunnel and a traffic flow to the tunnel, the active controller computes a path for the tunnel, programs (or say instructs) the network elements along the path for creating the tunnel, and instructs the ingress of the tunnel to direct the traffic flow into the tunnel.



During this process, the status information about the network is updated in the active controller. The information includes: the traffic engineering information in their TEDBs, the SID/label information in their SLDBs, and the configurations, paths, resources and status for tunnels in their TPDBs. The active controller synchronizes this information with the standby controller. Thus these two controllers have the same status information about the network. When the active controller fails, the standby controller takes over the role of the active controller smoothly and becomes active controller.

3.4. Hierarchical Controllers

The Figure below illustrates a system with hierarchical controllers. There is one Parent Controller and four Child Controllers: Child Controller 1, Child Controller 2, Child Controller 3 and Child Controller 4.



The parent controller communicates with these four child controllers and controls them, each of which controls (or is responsible for) a domain. Child controller 1 controls domain 1, Child controller 2 controls domain 2, Child controller 3 controls domain 3, and Child controller 4 controls domain 4.

One level of hierarchy of controllers is illustrated in the figure above. There is one parent controller at top level, which is not a child controller. Under the parent controller, there are four child controllers, which are not parent controllers.

In a general case, at top level there is one parent controller that is not a child controller, there are some controllers that are both parent controllers and child controllers, and there are a number of child controllers that are not parent controllers. This is a system of multiple levels of hierarchies, in which one parent controller controls or communicates with a first number of child controllers, some of which are also parent controllers, each of which controls or communicates with a second number of child controllers, and so on.

The parent controller receives requests for creating end to end tunnels from users or applications. For each request, the parent controller is responsible for obtaining a path for the tunnel and creating the tunnel along the path through sending instructions to the corresponding child controllers.

4. Application Scenarios

This section introduces a set of scenarios to which the controller can be applied.

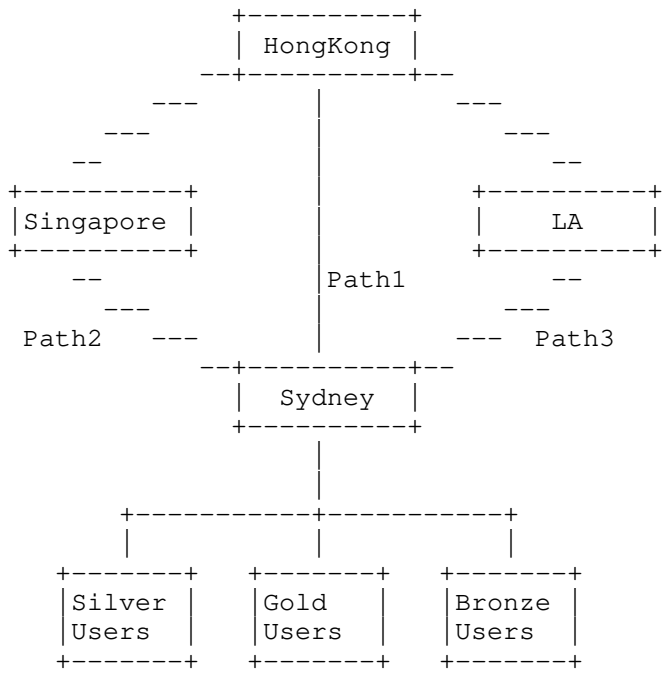
4.1. Business-oriented Traffic Steering

It is reasonable in commercial sense to provide multiple paths to the same destination with differentiated experiences for preferential users/services. This is an efficient approach to maximize providers' network resource usage as well as their profit and offer more choices to network users.

4.1.1. Preferential Users

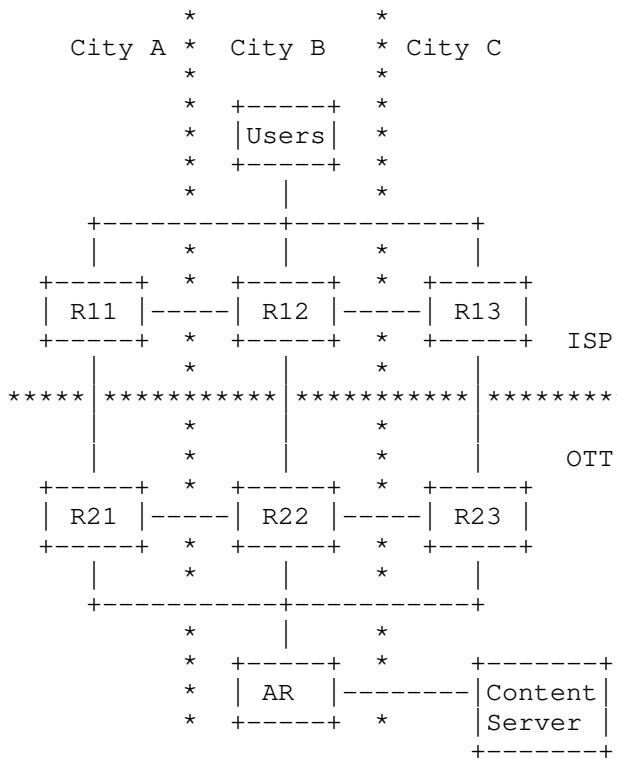
In the Figure below for an ISP network, there are three kinds of users in Sydney, saying Gold, Silver and Bronze, and they wish to visit website located in HongKong. The ISP provides three different paths with different experiences according to users' priority. The Gold Users may use Path1 with less latency and loss. The Silver Users may use the Path2 through Singapore with less latency but maybe some congestion there. The Bronze Users may use Path3 through LA

with some latency and loss.



4.1.2. Preferential Services

As depicted in the Figure below, the OTTSP has 3 exits with one ISP, which are located in City A, City B and City C. The content is obtained from Content Server and send to the exits through AR. An OTTSP may make its steering strategy based on different services. For example, the OTTSP in the Figure may choose exit R21 for video service and exit R22 for web service, which REQUIREs a mechanism/ system exists to identify different services from traffic flow.



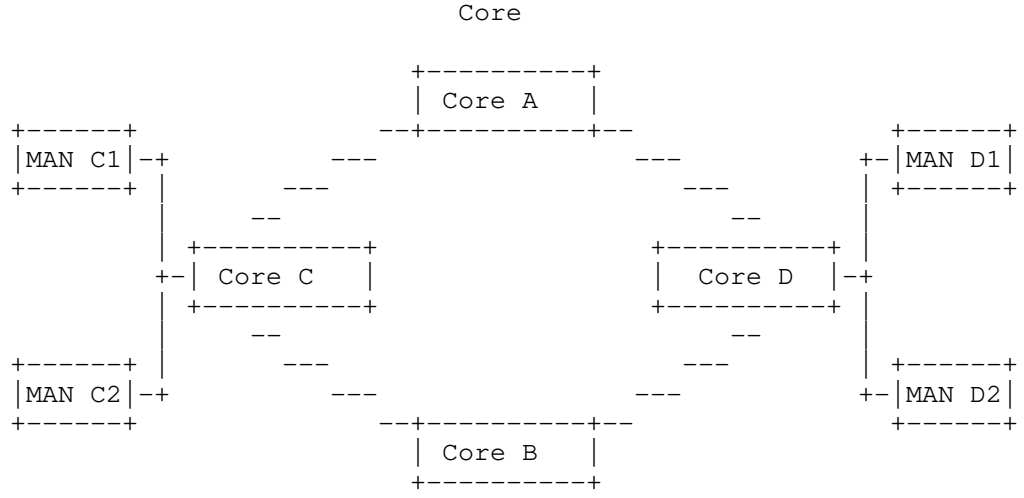
4.2. Traffic Congestion Mitigation

It is a persistent goal for providers to increase the utilization ratio of their current network resources, and to mitigate the traffic congestion. Traffic congestion is possible to happen anywhere in the ISP network(MAN, IDC, core and the links between them), because internet traffic is hard to predict. For example, there might be some local online events that the network operators didn't know beforehand, or some sudden attack just happened. Even for the big events that can be predicted, such as annual online discount of e-commerce company, or IOS update of Apple Inc, we could not guarantee there is no congestion. Since the network capacity expansion is usually an annual operation, there could be delay on any links of the engineering. As a result, the temporary traffic steering is always needed. The same thing happens to the OTT networks as well.

It should be noted that, the traffic steering is absolutely not a global behavior. It just acts on part of the network, and it's temporary.

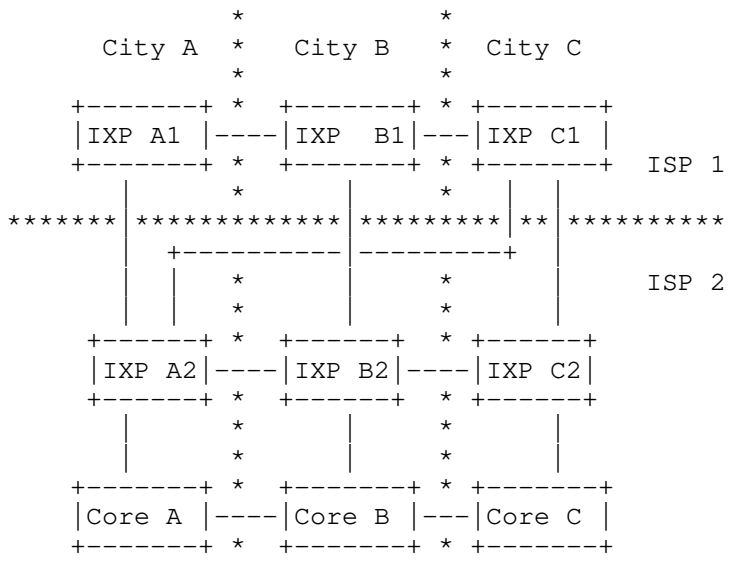
4.2.1. Congestion Mitigation in Core

As depicted in the Figure below, traffic from MAN C1 to MAN D2 follows the path Core C->Core B->Core D as the primary path, but somehow the load ratio becomes too much. It is reasonable to transfer some traffic load to less utilized path Core C->Core A->Core D when the primary path has congestion.



4.2.2. Congestion Mitigation among ISPs

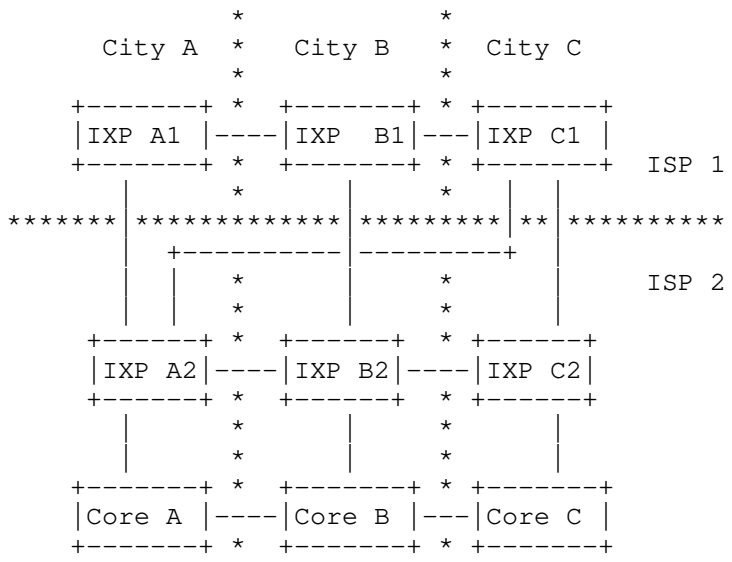
As depicted in the Figure below, ISP1 and ISP2 are interconnect by 3 exits which are located in 3 cities respectively. The links between ISP1 and ISP2 in the same city are called local links, and the rest are long distance links. Traffic from IXP C1 to Core A in ISP 2 usually passes through link IXP C1->IXP A2->Core A. This is a long distant route, directly connecting city C and city A. Part of traffic could be transferred to link IXP.



4.2.3. Congestion Mitigation at International Edge

An ISP usually interconnects with more than 2 transit networks at the international edge, so it is quite common that multiple paths may exist for the same foreign destination. Usually those paths with better QoS properties such as latency, loss, jitter and etc are often preferred. Since these properties keep changing from time to time, the decision of path selection has to be made dynamically.

As depicted in the Figure below, the traffic to the foreign destination H from IP core network (AS C1) has two choices on transit network, saying Transit A and Transit B. Under normal conditions, Transit B is the primary choice, but Transit A will be preferred when the QoS of Transit B gets worse. As a result, the same traffic will go through Transit A instead.



5. Security Considerations

The interactions with a BGP-based controller are similar to those with any other SDN controller. The security implications of SDN controller have not been fully discussed or described. Therefore, protocol and applicability for solutions around this architecture must take proper account of these concerns.

6. IANA Considerations

This document does not require any IANA actions.

7. Acknowledgements

The authors would like to thank Chris Bowers, Jeff Tantsura for their valuable suggestions and comments on this draft.

8. Contributors

Nan Wu
Huawei
Email: eric.wu@huawei.com

9. References

9.1. Normative References

[RFC1771] Rekhter, Y. and T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, DOI 10.17487/RFC1771, March 1995, <https://www.rfc-editor.org/info/rfc1771>.

[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>.

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, DOI 10.17487/RFC3107, May 2001, <https://www.rfc-editor.org/info/rfc3107>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <https://www.rfc-editor.org/info/rfc4456>.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <https://www.rfc-editor.org/info/rfc5575>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

9.2. Informative References

[I-D.ietf-idr-bgpls-segment-routing-epe] Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "Border Gateway Protocol - Link State (BGP-LS) Extensions for Segment Routing BGP Egress Peer Engineering", Work in Progress, Internet-Draft, draft-ietf-idr-bgpls-segment-routing-epe-19, 16 May 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgpls-segment-routing-epe-19>.

[I-D.ietf-idr-flowspec-path-redirect] Van de Velde, G., Patel, K., and Z. Li, "Flowspec Indirection-id Redirect", Work in Progress, Internet-

Draft, draft-ietf-idr-flowspec-path-redirect-12, 24 November 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-flowspec-path-redirect-12>.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., and D. Jain, "Advertising Segment Routing Policies in BGP", Work in Progress, Internet-Draft, draft-ietf-idr-segment-routing-te-policy-26, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-segment-routing-te-policy-26>.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", Work in Progress, Internet-Draft, draft-ietf-isis-segment-routing-extensions-25, 19 May 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-isis-segment-routing-extensions-25>.

[I-D.ietf-pce-segment-routing]
Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing", Work in Progress, Internet-Draft, draft-ietf-pce-segment-routing-16, 4 March 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-pce-segment-routing-16>.

[I-D.ietf-rtgwg-bgp-routing-large-dc]
Lapukhov, P., Premji, A., and J. Mitchell, "Use of BGP for Routing in Large-Scale Data Centers", Work in Progress, Internet-Draft, draft-ietf-rtgwg-bgp-routing-large-dc-11, 4 June 2016, <https://datatracker.ietf.org/doc/html/draft-ietf-rtgwg-bgp-routing-large-dc-11>.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", Work in Progress, Internet-Draft, draft-ietf-spring-segment-routing-15, 25 January 2018, <https://datatracker.ietf.org/doc/html/draft-ietf-spring-segment-routing-15>.

Authors' Addresses

Yujia
China Telcom Co., Ltd.
109 West Zhongshan Ave,Tianhe District

Guangzhou
510630
China
Email: luoyuj@sdu.edu.cn

Liang
China Telcom Co., Ltd.
109 West Zhongshan Ave,Tianhe District
Guangzhou
510630
China
Email: ouliang@chinatelecom.cn

Xiang
Tencent
Email: terranhuang@tencent.com

Gyan S. Mishra
Verizon Inc.
13101 Columbia Pike
Silver Spring, MD 20904
United States of America
Phone: 301 502-1347
Email: gyan.s.mishra@verizon.com

Huaimo Chen
Futurewei
Boston, MA,
United States of America
Email: hchen.ietf@gmail.com

Shunwan Zhuang
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing
100095
China
Email: zhuangshunwan@huawei.com

Zhenbin Li
Huawei
Huawei Bld., No.156 Beiqing Rd.

Beijing
100095
China
Email: lizhenbin@huawei.com

INTERNET-DRAFT
Intended status: Informational

S. Hu
F. Qin
Z. Li
China Mobile
T. Chua
Singapore Telecommunications Ltd
V. Lopez
Telefonica
D. Eastlake
Z. Wang
J. Song
Huawei

Expires: September 10, 2019 March 11, 2019

Architecture for Control Plane and User Plane Separated BNG
draft-cuspdt-rtgwg-cu-separation-bng-architecture-04.txt

Abstract

This document defines an architecture for Broadband Network Gateway (BNG) devices with control plane (CP) and user plane (UP) separation. A BNG-CP is a user control management component while a BNG-UP takes responsibility as the network edge and user policy implementation component. Both BNG-CP and BNG-UP are core components for fixed broadband services and are deployed separately at different network layers.

Table of Contents

1. Introduction

A Broadband Network Gateway (BNG) device is defined as an Ethernet-centric IP edge router, and the aggregation point for user traffic. It performs Ethernet aggregation and packet forwarding via IP/MPLS, and supports user management, access protocols termination, QoS, policy management, etc.

This document describes an architecture for BNG devices with control plane (CP) and user plane (UP) separation. A BNG-CP is a user control management component while a BNG-UP takes responsibility as the network edge and user policy implementation components. Both BNG-CP and BNG- UP are core components for fixed broadband services and are deployed separately at different network layers in the network.

1.1 Motivation

The rapid development of new services, such as 4K TV, IoT, etc., and increasing numbers of home broadband service users present some new challenges for BNGs such as:

Low resource utilization: The traditional BNG acts as both a gateway for user access authentication and accounting and an IP network's Layer 3 edge. The mutually affecting nature of the tightly coupled control plane and forwarding plane makes it difficult to achieve the maximum performance of either plane.

Complex management and maintenance: Due to the large numbers of traditional BNGs, configuring each device in a network is very tedious when deploying global service policies. As the network expands and new services are introduced, this deployment mode will cease to be feasible as it is unable to manage services effectively and rectify faults rapidly.

Slow service provisioning: The coupling of control plane and forwarding plane, in addition to a distributed network control mechanism, means that any new technology has to rely heavily on the existing network devices.

To address these challenges for fixed networks, the framework for a cloud-based BNG with CU separation conception is defined in [TR-384]. The main idea of Control-Plane and User-Plane separation is to extract and centralize the user management functions of multiple BNG devices, forming a unified and centralized control plane (CP). And the traditional router's Control Plane and Forwarding Plane are both preserved on BNG devices in the form of a user plane (UP). Note that the CU separation concept has also been introduced in the 3GPP 5G architecture [3GPP.23.501].

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.

The following acronyms are used as specified below:

AAA: Authentication Authorization Accounting.

BNG: Broadband Network Gateway. A broadband remote access server (BRAS (Broadband Access Server), B-RAS or BBRAS) that routes traffic to and from broadband remote access devices such as digital subscriber line access multiplexers (DSLAM) on an Internet service provider's (ISP) network. BRAS can also be referred to as a Broadband Network Gateway (BNG).

CP: Control Plane. The CP is a user control management component which manages the UP's resources such as the user entry and user's QoS policy

DHCP: Dynamic Host Configuration Protocol.

EMS: Element Management System.

IPoE: IP over Ethernet.

MANO: Management and Orchestration.

NFV: Network Function Virtualization.

NFVI: NFV Infrastructure.

PPPoE: Point-to-Point Protocol over Ethernet.

UP: User Plane. UP is a network edge and user policy implementation component. The traditional router's Control Plane and forwarding plane are both preserved on BNG devices in the form of a user plane.

3. CU Separated BNG Architecture

The functions in a traditional BNG can be divided into two parts: one is the user access management function, the other is the router function. In a cloud-based BNG, we find that tearing these two functions apart can make a difference. The user management function can be centralized and deployed as a concentrated module or device, called the BNG-CP (Control Plane). The other functions, such as the router function and forwarding engine, can be deployed in the form of the BNG User Plane. Thus, the Cloud-based BNG architecture is made up of control plane and user plane.

The following figure describes the architecture of CU separated BNG:

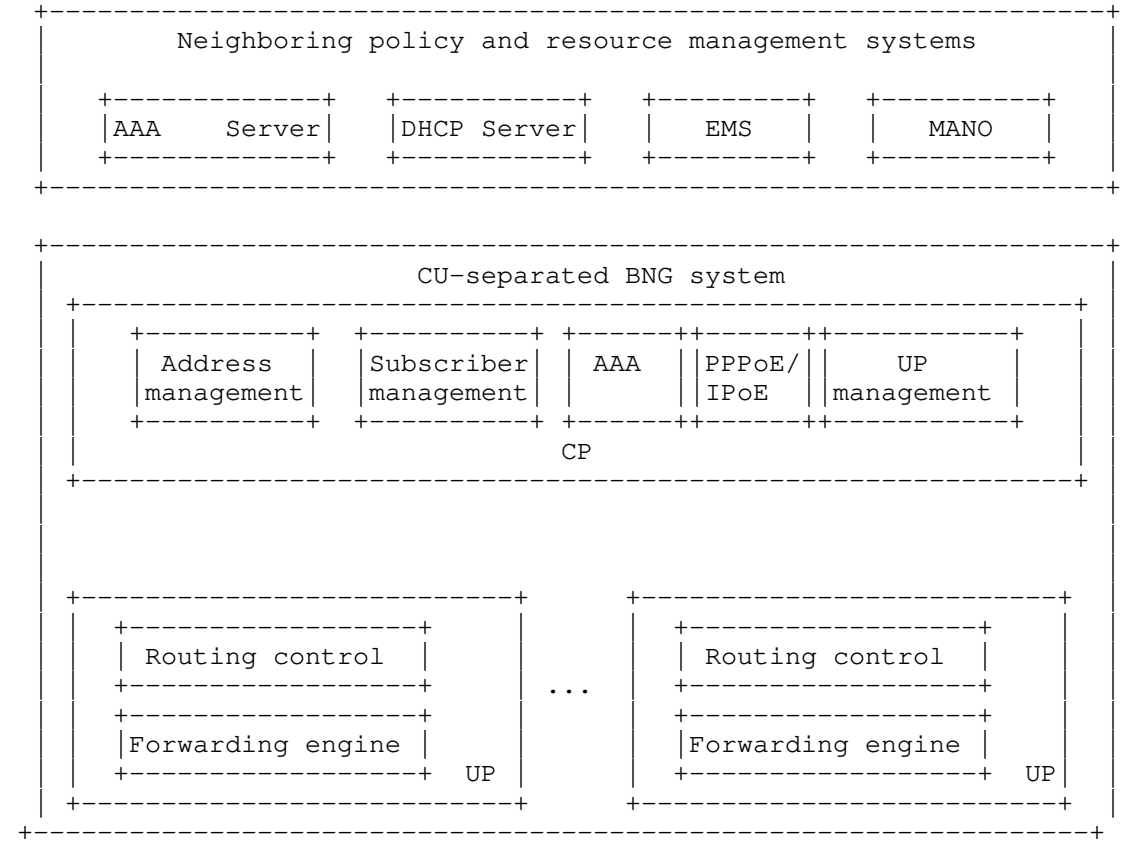


Figure 1. Architecture of CU Separated BNG

As in Figure 1, the BNG Control Plane could be virtualized and centralized, which provides significant benefits such as centralized session management, flexible address allocation, high scalability for subscriber management capacity, and cost-efficient redundancy, etc.

The functional components inside the BNG Service Control Plane can be implemented as Virtual Network Functions (VNFs) and hosted in a Network Function Virtualization Infrastructure (NFVI).

The User Plane Management module in the BNG control plane centrally manages the distributed BNG User Planes (e.g. load balancing), as well as the setup, deletion, and maintenance of channels between Control Planes and User Planes. Other modules in the BNG control plane, such as address management, AAA, etc., are responsible for the connection with outside subsystems in order to fulfill those services. Note that the User Plane SHOULD support both physical and virtual network functions. For example, BNG user plane L3 forwarding related network functions can be disaggregated and distributed across the physical infrastructure. And the other control plane and management plane functions in the CU Separation BNG can be moved into the NFVI for virtualization [TR-384].

The details of CU separated BNG's function components are as following:

The Control Plane should support:

(1) Address management: unified address pool management.

(2) AAA: This component performs Authentication, Authorization and Accounting, together with RADIUS/DIAMETER. The BNG communicates with the AAA server to check whether the subscriber who sent an Access-Request has network access authority. Once the subscriber goes online, this component together with the Service Control component implement accounting, data capacity limitation, and QoS enforcement policies.

(3) Subscriber management: user entry management and forwarding policy management.

(4) PPPoE/IPoE: process user dialup packets via PPPoE/IPoE.

(5) UP management: management of UP interface status, and the setup, deletion, and maintenance of channels between CP and UP.

The User Plane should support:

(1) Control plane functions including routing, multicast, and MPLS.

(2) Forwarding plane functions including traffic forwarding, QoS and traffic statistics collection.

3.1 Internal Interfaces Between the CP and UP

To support the communication between the Control Plane and User Plane, several interfaces are involved. Figure 2 illustrates the internal interfaces of CU Separated BNG.

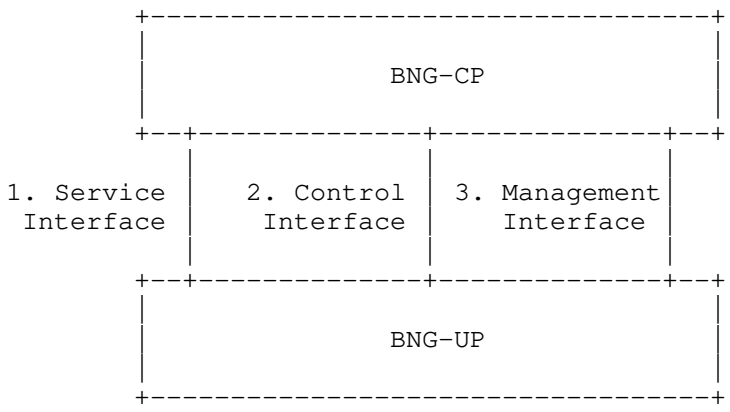


Figure 2. Internal Interfaces Between the CP and UP of the BNG

Service Interface: The CP and UP use this interface to establish tunnels with each other and transmit PPPoE and IPoE packets over those tunnels. VXLAN is commonly used for such tunnels as discussed in [hu-nvo3-vxlan-gpe-extension-for-vbng].

Control Interface: The CP uses this interface to deliver service entries, and the UP uses this interface to report service events to the CP. The requirements of this interface are introduced in [cuspdt-rtgwg-cusp-requirements], and the carrying protocol is presented in [cuspdt-rtgwg-cu-separation-bng-protocol] which specifies the Simple Control and User Plane Separation protocol (S-CUSP). The information model of this interface is presented in [cuspdt-rtgwg-cu-separation-infor-model].

Management Interface: The CP uses this interface to deliver configurations to the UP. This interface uses NETCONF [cuspdt-rtgwg-cu-separation-yang-model].

4. Usage of the CU Separation BNG

In the CU separated BNG scenario, there are several processes when a home user accesses the Internet:

(1) User dialup packets via PPPoE or IPoE from the BNG-UP are sent to the BNG-CP through the BNG-UP's Service Interface.

(2) BNG-CP processes the dialup packet. Confirming the user's authorization with the outside neighboring systems in the management network, the BNG-CP makes the decision to permit or deny the user access.

(3) After that, the BNG-CP tells the UP to do perform authorized forwarding actions with appropriate QoS policies.

(4) If the user is certificated and permitted, the UP forwards the traffic into the Internet with appropriate QoS policies such as limited bandwidth, etc. Otherwise, the user is denied to access the Internet.

In actual deployments, a CU separated BNG device is composed of a CP and one or more UPs. The CP is usually centrally deployed and takes responsibility as a user control management component managing UP's resources such as the user entry and forwarding policy. The UPs are distributed and act as a network edge and user policy implementation component.

In order to fulfill a service, neighboring policy and resource management systems are deployed outside the BNG. In the neighboring systems, different service systems such as RADIUS/DIAMETER server, DHCP server and EMS are included. If a BNG-CP is virtualized as a NFV, the NFVI management system MANO is also included here. A BNG-CP has connections with the outside neighboring systems to transmit management traffic.

The deployment scenario is shown in the following figure:

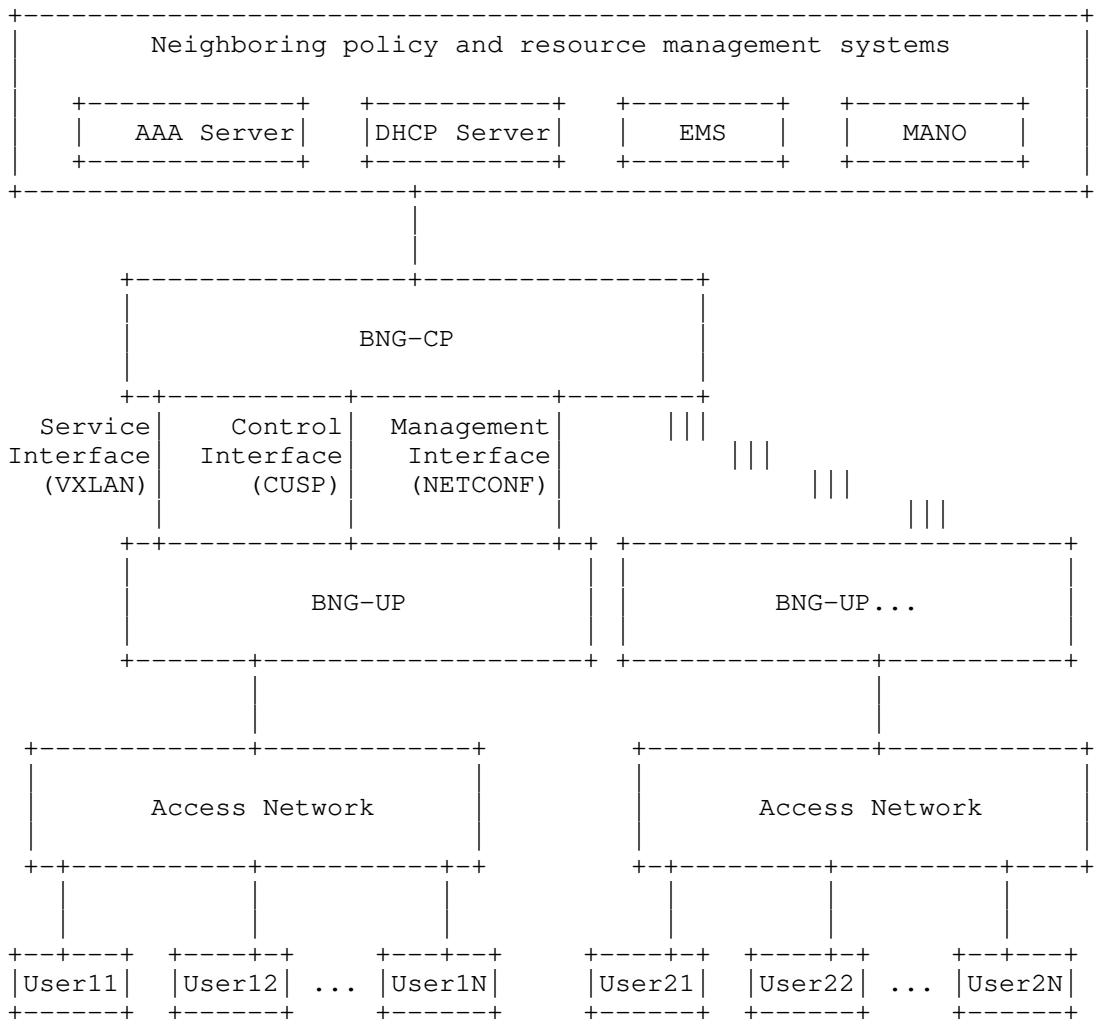


Figure 3. Deployment Example

5. Security Considerations

The Service, Control, and Management Interfaces between the CP and UP might be across the general Internet or other hostile environment. Thus, appropriate protections MUST be implemented to provide integrity, authenticity, and secrecy of traffic over those interfaces. For example, the implementation of IPSEC, DTLS, or TLS as appropriate. However, such security protocols need not always be used and lesser security precautions might be appropriate because, in some cases, the communication between the CP and UP might be in a more benign environment.

6. IANA Considerations

This document requires no IANA actions.

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>.

[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>.

Informative References

[_3GPP.23.501] "System Architecture for the 5G System", 3GPP GPP TS 23.501 15.0.0, 2018.

[cuspdt-rtgwg-cu-separation-bng-deployment] Gu, R., "Deployment Model of Control Plane and User Plane Separated BNG", draft-cuspdt-rtgwg-cu-separation-bng-deployment, work in progress, 2018.

[cuspdt-rtgwg-cu-separation-bng-protocol] Wang, Z., "Control-Plane and User-Plane separation BNG control channel Protocol", draft-cuspdt-rtgwg-cu-separation-bng-protocol, work in progress, 2018.

[cuspdt-rtgwg-cu-separation-infor-model] Wang, Z., "Information Model of Control-Plane and User- Plane separation BNG", draft-cuspdt-rtgwg-cu-separation-infor-model, work in progress, 2018.

[cuspdt-rtgwg-cusp-requirements] Hu, S., "Requirements for Control Plane and User Plane Separated BNG Protocol", draft-cuspdt-rtgwg-cusp-requirements, work in progress, 2018.

[cuspdt-rtgwg-cu-separation-yang-model] Hu, F., "YANG Data Model for Configuration Interface of Control-Plane and User-Plane separation BNG", draft-cuspdt-rtgwg-cu-separation-yang-model, work in progress, 2018.

[hu-nov3-vxlan-gpe-extension-for-vbng] Huang, L., "VXLAN GPE Extension for Packets Exchange Between Control and User Plane of vBNG", draft-hu-nvo3-vxlan-gpe-extension-for-vbrg, work in progress, 2017.

[TR-384] Broadband Forum, "Cloud Central Office Reference Architectural Framework", BBF TR-384, 2018.

Authors' Addresses

Shujun Hu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: hushujun@chinamobile.com

Fengwei Qin
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: qinfengwei@chinamobile.com

Zhenqiang Li
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: lizhenqiang@chinamobile.com

Tee Mong Chua
Singapore Telecommunications Limited
31 Exeter Road, #05-04 Comcentre Podium Block
Singapore City 239732
Singapore

Email: teemong@singtel.com

Victor Lopez
Telefonica
Spain

Email: victor.lopezalvarez@telefonica.com

Donald Eastlake, 3rd
Huawei Technologies
1424 Pro Shop Court
Davenport, FL 33896
USA

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

Zitao Wang
Huawei Technologies
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: wangzitao@huawei.com

Jun Song
Huawei Technologies
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: song.jun@huawei.com

rtgwg
Internet-Draft
Intended status: Informational
Expires: April 25, 2019

S. Hu
China Mobile
Donald. Eastlake
M. Wang, Ed.
Huawei
V. Lopez
Telefonica
F. Qin
Z. Li
China Mobile
T. Chua
Singapore Telecommunications Limited

October 22, 2018

Information Model of Control-Plane and User-Plane Separation BNG
draft-cuspdt-rtgwg-cu-separation-infor-model-03

Abstract

To improve network resource utilization and reduce operational expense, the Control-Plane and User-Plane separation concept is defined in Broadband Forum TR-384. This document describes the information model for the interface between the Control-Plane (CP) and the User-Plane (UP) in the CP/UP separation BNG. This information model may involve both the control channel interface and the configuration channel interface. The interface for the control channel allows the Control-Plane to send flow tables to the User-Plane, such as user's information table, user's interface table, and user's QoS table. And it also allows the User-Plane to report resource and statistics information to the Control-Plane. The interface for the configuration channel is in charge of the protocol version negotiation between the Control-Plane and User-Plane, the configuration for devices of the Control-Plane and User-Plane, and the reports of User-Plane's capabilities, etc. The information model defined in this document supports defining a standardized data model. Such a data model can be used to specify an interface to the CU separation BNG.

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/.

Copyright Notice

Table of Contents

1. Introduction

To improve network resource utilization and reduce operational expense, the Control-Plane and User-Plane separation concept is defined in Broadband Forum [TR-384]. The motivation for and architecture of the Control-Plane and User-Plane separation BNG is discussed in [I.D.cuspdt-rtgwg-cu-separation-bng-architecture].

This document describes an information model for the interface between the Control-Plane (CP) and the User-Plane (UP) separation in the CP / UP Separated BNG. This information model may involve both the control channel interface and the configuration channel interface. The interface for control channel allows the Control-Plane to send several flow tables to the User-Plane, such as user's information table, user's interface table, and user's QoS table, etc. And it also allows the User-Plane to report the resources and statistics information to the Control-Plane. The interface for configuration channel is in charge of the protocol version negotiation of protocols between the Control-Plane and User-Plane, the configuration for the devices of Control-Plane and User-Plane, and the report of User-Plane's capabilities, etc. The information model defined in this document enables supports defining a standardized data model. Such a data model can be used to define an interface to the CU separation BNG.

2. Concept and Terminology

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

2.1. Terminology

BNG: Broadband Network Gateway. A broadband remote access server (BRAS, B-RAS or BBRAS) routes traffic to and from broadband remote access devices such as digital subscriber line access multiplexers (DSLAM) on an Internet service provider's (ISP) network. BRAS can also be referred to as a Broadband Network Gateway (BNG).

CP: Control Plane. CP is a user control management component which supports the management of UP's resources such as the user entry and forwarding policy

UP: User Plane. UP is a network edge and user policy implementation component. The traditional router's Control Plane and Forwarding Plane are both preserved on BNG devices in the form of a user plane.

3. Control Plane and User Plane Separation BNG Information Model Overview

Briefly, a CU separation BNG is made up of a centralized CP and a set of UPs. The CP is a user control management component that manages UP's resources such as the user entry and forwarding policy, for example, the access bandwidth and priority management. The UP is a network edge and user policy implementation component. It can support the forwarding plane functions on traditional BNG devices, such as traffic forwarding, QoS, and traffic statistics collection, and it can also support the control plane functions on traditional BNG devices, such as routing, multicast, etc.

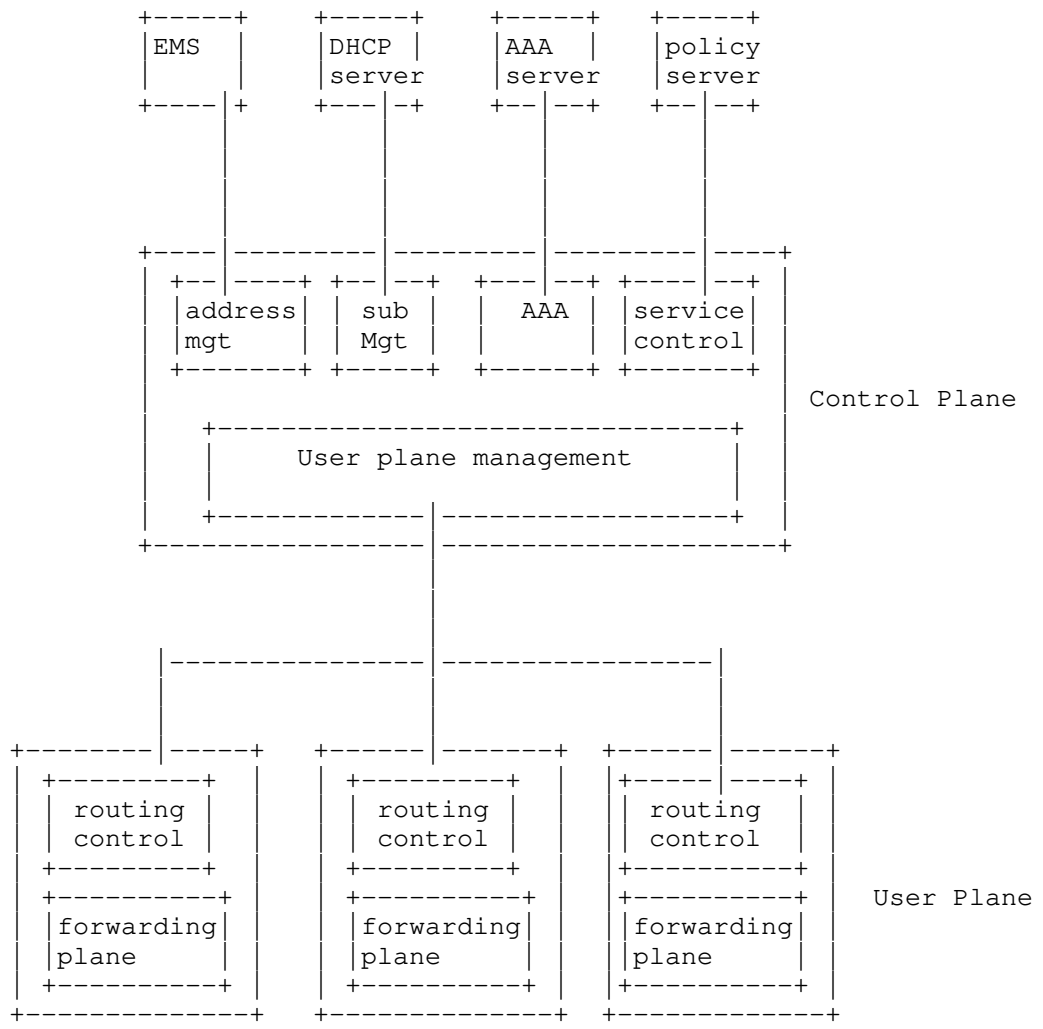


Figure 1. CU Separated BNG

The CU separated BNG is shown in Figure 1. The BNG Control Plane could be virtualized and centralized, which provides significant benefits such as centralized session management, flexible address allocation, high scalability for subscriber management capacity, and cost-efficient redundancy, etc. The functional components inside the BNG Service Control Plane can be implemented as Virtual Network Functions (VNFs) and hosted in a Network Function Virtualization Infrastructure (NFVI).

The User Plane Management module in the BNG control plane centrally manages the distributed BNG User Planes (e.g. load balancing), as

well as the setup, deletion, maintenance of channels between Control Planes and User Planes. Other modules in the BNG control plane, such as address management, AAA, and etc., are responsible for the connection with outside subsystems in order to provide the service. The routing control and forwarding Plane in the BNG User Plane (local) could be distributed across the infrastructure.

3.1. Service Data Model Usage

The idea of this information model is to propose a set of generic and abstract information models to be used in both Control Plane and User Planes. A typical scenario would be that this model can be used as a compendium to realize the communication between the Control Plane and User Planes of the CU separation BNG.

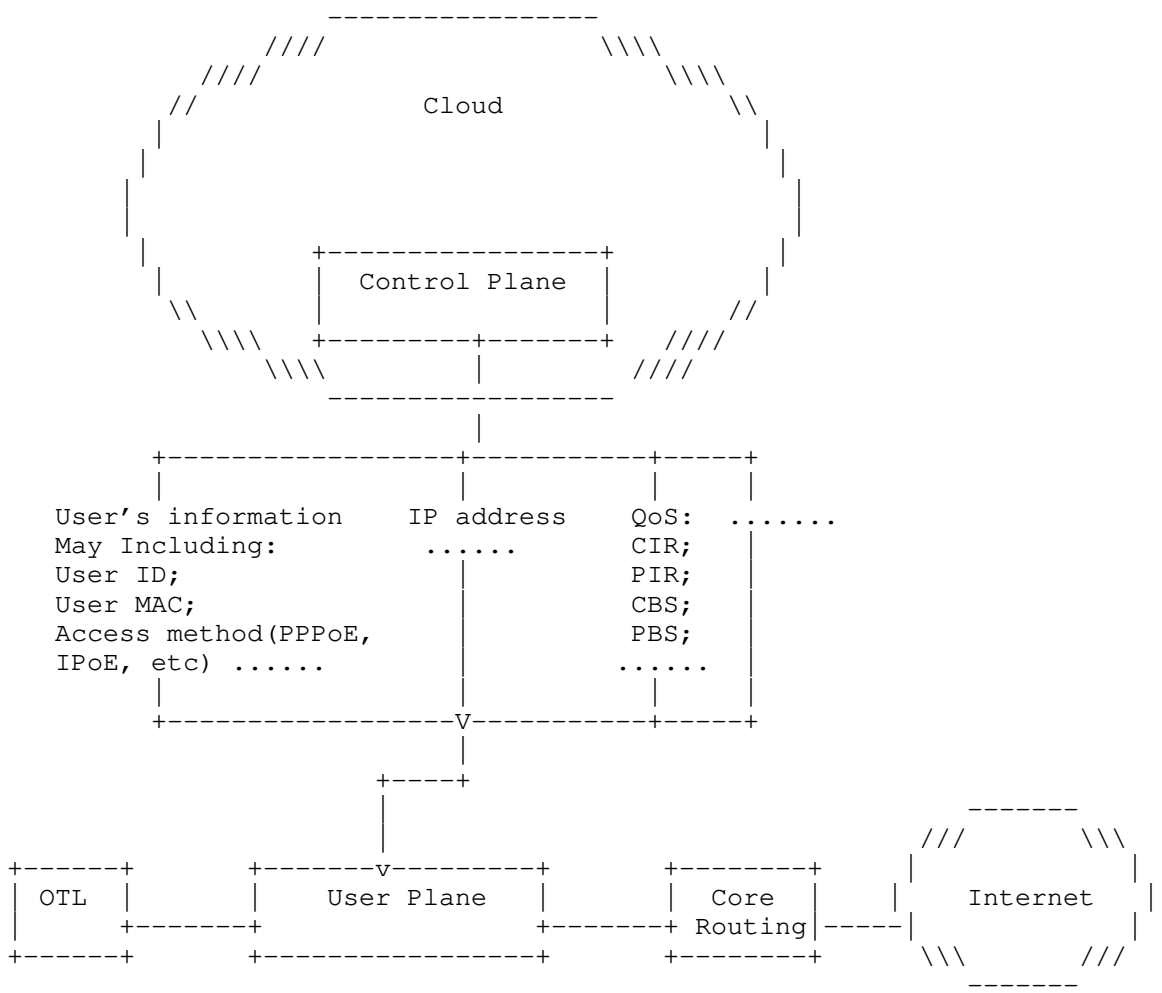


Figure 2. CU Separation BNG

As shown in Figure 2, when users access the BNG network, the control plane solicits these users' information (such as user's ID, user's MAC, user's access methods, for example via PPPoE/IPoE), associates them with available bandwidth which is reported by User planes, and based on the service's requirements generates a set of tables, which may include user's information, user's IP address, and QoS. Then the control plane can transmit these tables to the User planes. User planes receive these tables, parse them, matches these rules, and then performs corresponding actions.

4. Information Model

This section specifies the information model in Routing Backus-Naur Form [RFC5511]. This grammar intends to help readers better understand the English text description in order to derive a data model. However it may not provide all the details provided by the English text. When there is a lack of clarity, the English text will take precedence.

This section describes the information model that represents the interface of the CU separation BNG that is language and protocol neutral.

The following Routing BNF grammar describes the Overview of Information Model for CU separation BNG.

```
<cu-separation-bng-infor-model>::=<control-plane-information-model>
                                  <user-plane-information-model>

<control-plane-information-model>::=<user-related-infor-model>
                                    <interface-related-infor-model>
                                    <device-related-infor-model>

<user-related-infor-model>::= <user-basic-information>
                              [<ipv4-informatiom>]|[<ipv6-information>]
                              [<qos-information>]

<user-basic-information> :: = <USER_ID> <MAC_ADDRESS>
                              [<ACCESS_TYPE>][<SESSION_ID>]
                              [<INNER_VLAN-ID>][<OUTER_VLAN_ID>]
                              <USER_INTERFACE>

<ipv4-informatiom>::=<USER_ID><USER_IPV4>
                      <MASK_LENGTH><GATEWAY>
                      <VRF>

<ipv6-information>::=<USER_ID>(<USER_IPV6>
                      <PREFIX_LEN>)|(<PD_ADDRESS><PD_PREFIX_LEN>)
                      <VRF>

<qos-information>::=<USER_ID>
                    (<CIR><PIR><CBS><PBS>)
                    [<QOS_PROFILE>]

<interface-related-infor-model>::=<interface-information>

<interface-information>::=<IFINDEX><BAS_ENABLE>
                           <service-type>
```

```
<service-type>::=<PPP_Only><IPV4_TRIG>
                 <IPV6_TRIG><ND-TRIG>
                 <ARP_PROXY>

<device-related-infor-model>::=<address-field-distribute>

<address-field-distribute>::=<ADDRESS_SEGMENT><ADDRESS_SEGMENT_MASK>
                             <ADDRESS_SEGMENT_VRF><NEXT_HOP>
                             <IF_INDEX><MASK_LENGTH>

<user-plane-information-model>::=<port-resources-infor-model>
                                 <traffic-statistics>

<port-resource-information>::=<IF_INDEX><IF_NAME>
                              <IF_TYPE><LINK_TYPE>
                              <MAC_ADDRESS><IF_PHY_STATE>
                              <MTU>
<traffic-statistics-information>::=<USER_ID><STATISTICS_TYPE>
                                   <INGRESS_STATIISTICS_PACKETS>
                                   <INGRESS STATISTICS_BYTES>
                                   <EGRESS_STATISTICS_PACKETS>
                                   <EGRESS_STATISTICS_BYTES>
```

4.1. Information Model for Control-Plane

This section describes information model for the Control-Plane (CP). As mentioned in Section 3, the Control Plane is a user control management component which manages the user's information, User-Plane's resources and forwarding policy, etc. The control plane can generate several tables which contain a set of rules based on the resources and specific requirements of user's service. After that, the control plane sends the tables to User Planes, and User planes receive the tables, parse them, match the rules, and then perform corresponding actions.

The Routing Backus-Naur Form grammar below specifies the Information model for Control-Plane:

```
<control-plane-information-model>::=<user-related-infor-model>
                                    <interface-related-infor-model>
                                    <device-related-infor-model>

<user-related-infor-model>::= <user-basic-information>
                              [<ipv4-informatiom>]|[<ipv6-information>]
                              [<qos-information>]

<user-basic-information> :: = <USER_ID> <MAC_ADDRESS>
                              [<ACCESS_TYPE>][<SESSION_ID>]
                              [<INNER_VLAN-ID>][<OUTER_VLAN_ID>]
                              <USER_INTERFACE>

<ipv4-informatiom>::=<USER_ID><USER_IPV4>
                      <MASK_LENGTH><GATEWAY>
                      <VRF>

<ipv6-information>::=<USER_ID>(<USER_IPV6>
                      <PREFIX_LEN>)|(<PD_ADDRESS><PD_PREFIX_LEN>)
                      <VRF>

<qos-information>::=<USER_ID>
                    (<CIR><PIR><CBS><PBS>)
                    [<QOS_PROFILE>]

<interface-related-infor-model>::=<interface-information>

<interface-information>::=<IFINDEX><BAS_ENABLE>
                           <service-type>

<service-type>::=<PPP_Only><IPV4_TRIG>
                 <IPV6_TRIG><ND-TRIG>
                 <ARP_PROXY>

<device-related-infor-model>::=<address-field-distribute>

<address-field-distribute>::=<ADDRESS_SEGMENT><ADDRESS_SEGMENT_MASK>
                             <ADDRESS_SEGMENT_VRF><NEXT_HOP>
                             <IF_INDEX><MASK_LENGTH>
```

user-related-infor-model: presents the attributes that can describe the user's profile, such as user's basic information, qos, and IP address.

interface-related-infor-model: presents the attributes that relate to some physical/virtual interface. This model can be used to indicate which kinds of service can be supported by interfaces.

device-related-infor-model: presents the attributes which relate to a specific device. For example the control plane can manage and distribute the users, which belong to same subnet, to some specific devices. And the user plane's devices provide corresponding service for these users.

4.1.1. User-Related Information

The user related information is a collection of attributes bound to specific users. For example, the control plane can use a unified ID to distinguish different users and distribute the IP address and QoS rules to a specific user. In this section, the user related information models are presented. The user related information models include the user information model, IPv4/IPv6 information model, QoS information model, etc.

The Routing Backus-Naur Form grammar below specifies the user related information model:

```
<user-related-infor-model>::= <user-basic-information>
                              [<ipv4-informatiom>]|[<ipv6-information>]
                              [<qos-information>]

<user-basic-information> :: = <USER_ID> <MAC_ADDRESS>
                              [<ACCESS_TYPE>][<SESSION_ID>]
                              [<INNER_VLAN-ID>][<OUTER_VLAN_ID>]
                              <USER_INTERFACE>

<ipv4-informatiom>::=<USER_ID><USER_IPV4>
                      <MASK_LENGTH><GATEWAY>
                      <VRF>

<ipv6-information>::=<USER_ID>(<USER_IPV6>
                      <PREFIX_LEN>)|(<PD_ADDRESS><PD_PREFIX_LEN>)
                      <VRF>

<qos-information>::=<USER_ID>
                    (<CIR><PIR><CBS><PBS>)
                    [<QOS_PROFILE>]
```

4.1.1.1. User Basic Information Model

The User Basic Information model contains a set of attributes to describe the basic information of a specific user, such as the user's MAC address, access type (via PPPoE, IPoE, etc), inner VLAN ID, outer VLAN ID, etc.

The Routing Backus-Naur Form grammar below specifies the user basic information model:

```
<user-basic-information> :: = <USER_ID> <MAC_ADDRESS>
                              [<ACCESS_TYPE>][<SESSION_ID>]
                              [<INNER_VLAN-ID>][<OUTER_VLAN_ID>]
                              <USER_INTERFACE>
```

USER_ID (4 bytes): is the identifier for a user. This parameter is unique and mandatory. It can be used to distinguish different users.

MAC_ADDRESS (6 bytes): is the MAC address of the user.

ACCESS_TYPE (2 bytes): This attribute is an optional parameter. It can be used to indicate the protocol being used for the user's access, such as PPPoE, IPoE, etc.

SESSION_ID (4 bytes): This attribute is an optional parameter. It can be used as the identifier of PPPoE session.

INNER_VLAN-ID (2 bytes): The 12-bit identifier of user's inner VLAN in network byte order. The unused high-order 4 bits MUST be sent as zero and ignored on receipt.

OUTER_VLAN_ID (2 bytes): The 12-bit identifier of user's outer VLAN in network byte order. The unused high-order 4 bits MUST be sent as zero and ignored on receipt.

USER_INTERFACE (4 bytes): This attribute specifies the binding interface of a specific user. The IfIndex of the interface MAY be included. This is the 32-bit IfIndex assigned to the interface by the device as specified by the Interfaces Group MIB [RFC2863]. The IfIndex can be utilized within a management domain to map to an actual interface, but it is also valuable in public applications [RFC5837]. The IfIndex can be used as an opaque token to discern which interface of the User-Plane is providing corresponding service for specific user.

4.1.1.2. IPv4 Information Model

The IPv4 information model presents the user's IPv4 parameters. It is an optional constructs. The Routing Backus-Naur Form grammar below sepcifies the user's IPv4 information model:

```
<ipv4-informatiom>::=<USER_ID><USER_IPV4>
                     <MASK_LENGTH><GATEWAY>
                     <VRF>
```

USER_ID (4 bytes): is the identifier of user. This parameter is unique and mandatory. This attribute is used to distinguish different users. In conjunction with other IPv4 parameters it links the user to the user's IPv4 information.

USER_IPV4 (4 bytes): This attribute specifies the user's IPv4 address, It is usually used in user plane discovery and ARP reply message.

MASK_LENGTH (4 bytes): This attribute specifies the user's subnet mask length which can identify a range of IP addresses that are on the same network.

GATEWAY (4 bytes): This attribute specifies the user's gateway, and is usually used in User Plane discovery and ARP reply message.

VRF (4 bytes): is the identifier of VRF instance.

4.1.1.3. IPv6 Information Model

The IPv6 information model presents the user's IPv6 parameters. It is an optional constructs. The Routing Backus-Naur Form grammar below specifies the user's IPv6 information model:

```
<ipv6-information>::=<USER_ID>(<USER_IPV6>
                      <PREFIX_LEN>)|(<PD_ADDRESS><PD_PREFIX_LEN>)
                      <VRF>
```

USER_ID (4 bytes): is the identifier of user. This parameter is unique and mandatory. This attribute is used to distinguish different users. in conjunction with other IPv6 parameters, I tlink the user to the user's IPv6 information.

USER_IPV6 (2 bytes): This attribute specifies the user's IPv6 address. It is usually used in neighbor discovery (ND discovery).

PREFIX_LEN (4 bytes): This attribute specifies the user's subnet prefix lengths which can identify a range of IP addresses that are on the same network.

PD_ADDRESS (4 bytes): In IPv6 networking, DHCPv6 prefix delegation is used to assign a network address prefix and automate configuration and provisioning of the public routable addresses for the network. This attribute specifies the user's DHCPv6 prefix delegation address, and is usually used in neighbor discovery (ND discovery).

PD_PREFIX_LEN (4 bytes): This attribute specifies the user's DHCPv6 delegation prefix length, and it's usually used in neighbor discovery (ND discovery).

VRF (4 bytes): is the identifier of a VRF instance

4.1.1.4. QoS Information Model

In the CU separation BNG, the Control-Plane (CP) generates the QoS table base based on the UP's bandwidth resources and the specific QoS requirements ofof the user's services. This table contains a set of QoS matching rules such as user's committed information rate, peak information rate, committed burst size, etc. And itIs is an optional constructs. The Routing Backus-Naur Form grammar below illustratesspecifies the user's qos information model:

```
<qos-information>::=<USER_ID>
                    (<CIR><PIR><CBS><PBS>)
                    [<QOS_PROFILE>]
```

USER_ID (4 bytes): is the identifier of user. This parameter is unique and mandatory. This attribute is used to distinguish different users. within conjunction with other qos parameters it links the user to the user's qos information.

CIR (4 bytes): In a BNG network, the Committed Information Rate (CIR) is the bandwidth available for a user guaranteed by an internet service provider under normal conditions. This attribute is used to indicate the user's committed information rate, and it usually appears with other qos attributes (such as PIR, CBS, PBS, etc) to give the user's QoS profile.

PIR (4 bytes): Peak Information Rate (PIR) is a burstable rate set on routers and/or switches that allow throughput bursts. This attribute is used to indicate the user's peak information rate. In conjunction with with other QoS attributes (such as CIR, CBS, PBS, etc) it is used to give the user's QoS profile.

CBS (4 bytes): The Committed Burst Size (CBS) specifies the relative amount of reserved buffers for a specific ingress network's forwarding class queue or egress network's forwarding class queue. This attribute is used to indicate the user's committed burst size. In conjunction with other qos attributes (such as CIR, PIR, PBS, etc) it is used to give the user's QoS profile.

PBS (4 bytes): The Peak Burst Size (PBS) specifies the maximum size of the first token bucket. This attribute is used to indicate the

user's peak burst size. In conjunction with other qos attributes (such as CIR, PIR, CBS, etc) it is used to give the user's QoS profile.

QOS_PROFILE (4 bytes): This attribute specifies the standard profile provided by the operator. It can be used as a QoS template that is defined as a list of classes of services and associated properties. The properties may include:

- o Rate-limit: used to rate-limit the class of service. The value is expressed as a percentage of the global service bandwidth.

- o latency: used to define the latency constraint of the class. The latency constraint can be expressed as the lowest possible latency or a latency boundary expressed in milliseconds.

- o jitter: used to define the jitter constraint of the class. The jitter constraint can be expressed as the lowest possible jitter or a jitter boundary expressed in microseconds.

- o bandwidth: used to define a guaranteed amount of bandwidth for the class of service. It is expressed as a percentage.

4.1.2. Interface Related Information

This model contains the necessary information for an interface. It is used to indicate which kind of service can be supported by this interface. The Routing Backus-Naur Form grammar below specifies the interface related information model:

```
 <interface-related-infor-model>::=<interface-information>

 <interface-information>::=<IFINDEX><BAS_ENABLE>
                           <service-type>

 <service-type>::=<PPP_Only><IPV4_TRIG>
                  <IPV6_TRIG><ND-TRIG>
                  <ARP_PROXY>
```

4.1.2.1. Interface Information Model

The interface model mentioned here is a logical construct that identifies a specific process or a type of network service. In CU separation BNG network, the Control-Plane (CP) generates the Interface-Infor table based on the available resources, which are received from the User-Plane (UP), and the specific requirements of user's services.

The Routing Backus-Naur Form grammar below specifies the interface information model:

```
<interface-information>::=<IFINDEX><BAS_ENABLE>
                          <service-type>

<service-type>::=<PPP_Only><IPV4_TRIG>
                 <IPV6_TRIG><ND-TRIG>
                 <ARP_PROXY>
```

IFINDEX (4 bytes): The IfIndex is the 32-bit index assigned to the interface by the device as specified by the Interfaces Group MIB [RFC2863]. The IfIndex can be utilized within a management domain to map to an actual interface, but it is also valuable in public applications. The IfIndex can be used as an opaque token to discern which interface of the User-Plane is providing the corresponding service for specific user.

BAS_ENABLE (2 bytes): This is a flag, and if it is TRUE, the BRAS is enabled on this interface.

PPP_Only (2 bytes): This is a flag, and if it is TRUE, the interface only supports PPP user.

IPV4_TRIG (2 bytes): This is a flag, and if it is TRUE, the interface supports the user being triggered to connect to the internet by using an IPv4 message.

IPV6_TRIG (2 bytes): This is a flag, and if it is TRUE, the interface supports that the user being triggered to connect to the internet by using an IPv6 message.

ND-TRIG (2 bytes): This is a flag, and if it is TRUE, the interface supports the user being triggered to connect to the internet by using a neighbor discovery message.

ARP_PROXY (2 bytes): This is a flag, and if it is TRUE, ARP PROXY is enabled on this interface.

4.1.3. Device Related Information

The device related information model presents the attributes which relate to a specific device. For example the control plane can manage and distribute the users, who belong to the same subnet, to some specific devices. And then the user plane’s devices can provide the corresponding service for these users. The Routing Backus-Naur Form grammar below specifies the device related information model:

```
<device-related-infor-model>::=<address-field-distribute>

<address-field-distribute>::=<ADDRESS_SEGMENT><ADDRESS_SEGMENT_MASK>
                             <ADDRESS_SEGMENT_VRF><NEXT_HOP>
                             <IF_INDEX><MASK_LENGTH>
```

4.1.3.1. Address field distribute Table

In the CU separation BNG information model, the Control-Plane (CP) generates and sends this Address field distribute table to UP. Based on this table, the user-plane's devices can be divided into several blocks, and each block is in charge of working for users with the same subnet. The Routing Backus-Naur Form grammar below illustrates the address field sepcifies information model:

```
<address-field-distribute>::=<ADDRESS_SEGMENT><ADDRESS_SEGMENT_MASK>
                             <ADDRESS_SEGMENT_VRF><NEXT_HOP>
                             <IF_INDEX><MASK_LENGTH>
```

4.2. Information Model for User Plane

This section describes the information model for the interface of to the User Plane (UP). As mentioned in section Section 3, the UP is a network edge and user policy implementation component. It supports the following: Forwarding plane functions on traditional BNG devices, including traffic forwarding, QoS, and traffic statistics collection and Control plane functions on traditional BNG devices, including routing, multicast, and MPLS.

In CU separation BNG information model, the CP generates tables and provides the entries. The UP plays two roles:

1. It receives these tables, parses them, then performs corresponding actions.

2. It also generates several tables to report the available resources (such as usable interfaces, etc.) and statistical information to CP.

The Routing Backus-Naur Form grammar below specifies the User Plane information model:

```
<user-plane-information-model>::=<port-resources-infor-model>
                                  <traffic-statistics>

port-resource-information>::=<IF_INDEX><IF_NAME>
                             <IF_TYPE><LINK_TYPE>
                             <MAC_ADDRESS><IF_PHY_STATE>
                             <MTU>
<traffic-statistics-information>::=<USER_ID><STATISTICS_TYPE>
                                    <INGRESS_STATIISTICS_PACKETS>
                                    <INGRESS STATISTICS_BYTES>
                                    <EGRESS_STATISTICS_PACKETS>
                                    <EGRESS_STATISTICS_BYTES>
```

4.2.1. Port Resources of UP

The User Plane can generate the network resource table, which contains a bunch of attributes to present the available network resources, for example the usable interfaces.

The Figure Routing BNF grammar below illustratesspecifies the Port Resources Information Table of User-Plane:

```
<port-resource-information>::<IF_INDEX><IF_NAME>
                             <IF_TYPE><LINK_TYPE>
                             <MAC_ADDRESS><IF_PHY_STATE>
                             <MTU>
```

IFINDEX (4 bytes): IfIndex is the 32-bit index assigned to the interface by the device as specified by the Interfaces Group MIB [RFC2863]. The IfIndex can be utilized within a management domain to map to an actual interface, but it is also valuable in public applications. The IfIndex can be used as an opaque token to discern which interface of the User-Plane is available.

IF_NAME (64 bytes): the textual name of the interface. The value of this object should be the name of the interface as assigned by the local device and should be suitable for use in commands entered at the device's "console". This might be a text name, such as "le0" or a simple port number, such as "1", depending on the interface naming syntax of the device. If several entries in the ifTable together represent a single interface as named by the device, then each will have the same value of ifName.

IF_TYPE (4 bytes): the type of interface, such as Ethernet, GE, Eth-Trunk, etc.

LINK_TYPE (4 bytes): This attribute specifies the type of link, such as point-to-point, broadcast, multipoint, point-to-multipoint, private and public (accessibility and ownership), etc.

MAC_ADDRESS (6 bytes): This attribute specifies the available interface's MAC address.

IF_PHY_STATE (1 byte): The current operational state of the interface. This is an enumeration type node:

1- Up: ready to pass packets;

2- Down

3- Testing: in some test mode;

4- Unknow: status cannot be determined for some reason;

5- Dormant;

6- Not present: some component is missing.

MTU: This attribute specifies the available interface's MTU (Maximum Transmission Unit).

4.2.2. Traffic Statistics Infor

The user-plane also generates the traffic statistics table to report the current traffic statistics.

The Figure below specifies the Traffic Statistics Infor model of User-Plane:

```
<traffic-statistics-information>::=<USER_ID><STATISTICS_TYPE>
                                    <INGRESS_STATIISTICS_PACKETS>
                                    <INGRESS STATISTICS_BYTES>
                                    <EGRESS_STATISTICS_PACKETS>
                                    <EGRESS_STATISTICS_BYTES>
```

USER_ID (4 bytes): is the identifier of user. This parameter is unique and mandatory. This attribute is used to distinguish different users. In conjunction with other statistics parameters such as ingress packets, egress packets, etc, it is used to report the user's status profile.

STATISTICS_TYPE (4 bytes): This attributes specifies the traffic type such as IPv4, IPv6, etc.

INGRESS_STATIISTICS_PACKETS (8 bytes): This attribute specifies the Ingress Statistics Packets of specific user.

INGRESS STATISTICS_BYTES (8 bytes): This attribute specifies the Ingress Statistics Bytes of specific user.

EGRESS_STATISTICS_PACKETS (8 bytes): This attribute specifies the Egress Statistics Packets of specific user.

EGRESS_STATISTICS_BYTES (8 bytes): This attribute specifies the Egress Statistics Bytes of specific user.

5. Security Considerations

None.

6. IANA Considerations

None.

7. References

7.1. Normative References

[I-D.cuspdt-rtgwg-cu-separation-bng-architecture]
Hu, S., Qin, F., Li, Z., Chua, T., Wang, Z., and J. Song, "Architecture for Control Plane and User Plane Separated BNG", draft-cuspdt-rtgwg-cu-separation-bng-architecture-01 (work in progress), July 2018.

[I-D.cuspdt-rtgwg-cu-separation-bng-deployment]
Gu, R., Hu, S., and Z. Wang, "Deployment Model of Control Plane and User Plane Separation BNG", draft-cuspdt-rtgwg-cu-separation-bng-deployment-00 (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>.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <https://www.rfc-editor.org/info/rfc2863>.

[RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, DOI 10.17487/RFC5511, April 2009, <https://www.rfc-editor.org/info/rfc5511>.

[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, <https://www.rfc-editor.org/info/rfc5837>.

[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

[TR-384] Broadband Forum, ""Cloud Central Office Reference Architectural Framework",", BBF TR-384, January. 2018.

Authors' Addresses

Shujun Hu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: hushujun@chinamobile.com

Donald Eastlake, 3rd
Huawei
1424 Pro Shop Court
Davenport, FL 33896
USA

Email: d3e3e3@gmail.com

Michael Wang (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: wangzitao@huawei.com

Victor Lopez
Telefonica
Sur 3 building, 3rd floor, Ronda de la Comunicacion s/n
Madrid 28050
Spain

Email: victor.lopezalvarez@telefonica.com

Fengwei Qin
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: qinfengwei@chinamobile.com

Zhenqiang Li
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: lizhenqiang@chinamobile.com

Tee Mong Chua
Singapore Telecommunications Limited
31 Exeter Road, #05-04 Comcentre Podium Block
Singapore City 239732
Singapore

Email: teemong@singtel.com

IETF RTGWG
Internet-Draft
Intended status: Standards Track
Expires: March 12, 2020

Guangping Huang
ZTE Corporation
Shujun Hu
Fengwei Qin
China Mobile

Sep 9, 2019

YANG Data Model for Configuration Interface of Control-Plane and User-Plane separation BNG
draft-cuspdt-rtgwg-cu-separation-yang-model-04

Abstract

This document defines the YANG data model for management of Control-Plane and User-Plane separation of BNGs (Broadband Network Gateways).

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 12, 2020.

Table of Contents

1. Introduction

The main idea of Broadband Network Gateway (BNG) Control-Plane and User-Plane separation is to extract and centralize the user management functions of multiple BNG devices, forming a unified and centralized control plane (CP), while the traditional router's control and forwarding information are both preserved on BNG devices in the form of a user plane (UP). We call the Control-Plane and User-plane separation BNG a vBNG (virtual BNG).

The architecture of Control-plane and User-plane separated BNG is shown as the following figure.

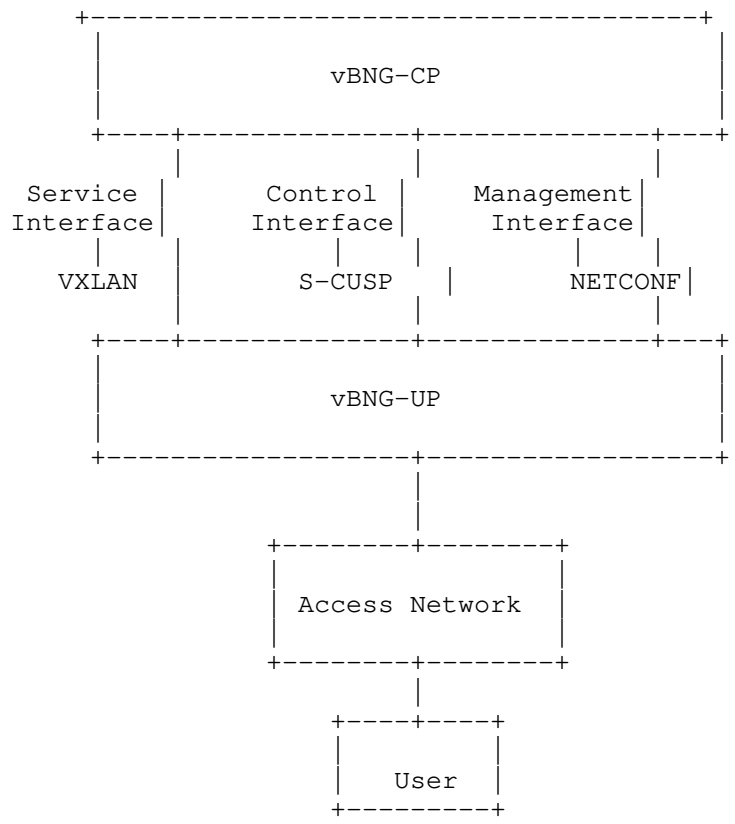


Figure 1: Architecture of C/U separated BNG

There are three interfaces between vBNG-CP(vBNG Control Plane) and vBNG-UP(vBNG User Plane): Service interface, control interface and management interface. The service interface is used to carry PPPoE/ IPoE dialup packets between user plane and control plane. The requirements and possible solution are defined in the [I-D.hu-nvo3-vxlan-gpe-extension-for-vbng]. The control interface is used for setting forwarding entries of the user plane using Simple CUSP (S-CUSP) [I-D.cuspdt-rtgwg-cu-separation-bng-protocol] or other protocols. The management interface is used by vBNG-CP to carry out related configurations of vBNG-UP through NETCONF protocol [RFC6241].

This document defines the YANG data model for vBNG(vBNG-CP and vBNG-UP). There are three types of YANG data model for vBNG in this document: The YANG data models for vBNG-CP, the YANG data models for direct network management of vBNG-UP, and the YANG data models for BNG-UP through the management interfaces among the vBNG-UP and vBNG-CP.

2. Conventions used in this document

2.1. Terminology

BNG: Broadband Network Gateway. A broadband remote access server routes traffic to and from broadband remote access devices such as digital subscriber line access multiplexers (DSLAM) on an Internet service provider's (ISP) network.

CUSP: Control-plane and User-plane Separation Protocol.

S-CUSP: Simple CUSP.

vBNG: Virtualization Broadband Network Gateway. An vBNG is to extract and centralize the user management functions of multiple BNG devices, and to form an unified and centralized control plane (CP). The vBNG devices include vBNG-UP and vBNG-CP.

vBNG-CP: vBNG Control Plane. The vBNG-CP is a user control management component which support to manage UP's resources such as the user entry and forwarding policy.

vBNG-UP: vBNG User Plane. vBNG-UP is a network edge and user policy implementation component.

2.2. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Design Tree

3.1. Yang Data model through Management Interfaces

The vBNG-UP or vBNG-CP part can be a physical or virtualized network element. The LNE model [I-D.ietf-rtgwg-lne-model] is augmented to define the YANG data models for vBNG-UP and vBNG-CP in this document.

The YANG data model for vBNG through the management interface includes vBNG-UP interface configuration, control channel and service channel configuration, ACL and QoS.

The vBNG-UP interface configuration is to configure the basic interface informations of a vBNG-UP element, such as interface name, the VLAN parameters for the sub-interface.

The control channel is to configure the S-CUSP parameters. The control channel parameters include: name, id, port, S-CUSP version, hello interval, dead time, and keepalive time.

The VXLAN tunnel is the suggested service interface protocol between vBNG-CP and vBNG-UP. The VXLAN tunnel parameters include: tunnel-source-ip,tunnel-destination-ip, vxlan-id, vxlan-tunnel-id, vxlan-tunnel-name, etc.

The ACL information includes ipv4-acl,ipv6-acl,link-acl,etc. The YANG data model for ACL refers to [I-D.ietf-netmod-acl-model]

The QoS information includes IP-DSCP, MPLS,VPLS, VPWS etc. The YANG data model for QoS refers to [I-D.asechoud-rtgwg-qos-model]

```
module: ietf-vbng
      augment /lne:logical-network-elements/lne:logical-network-element:
       +--rw ietf-vbng
           +--rw interfaces
           |  +--rw interface* [name]
           |     +--rw name      if:interface-ref
           |     +--rw ethernet
           |     |  +--rw lacp?   boolean
           |     +--rw mac-offset?   uint32
           |     +--rw vlans
           |        +--rw tag* [index]
           |           +--rw index    uint8
           |           +--rw tag
           |              +--rw tag-type?   string
           |              +--rw vlan-id?    vlan-id
           +--rw control-channel
           |  +--rw name?            string
           |  +--rw id?              uint32
           |  +--rw port?            uint32
           |  +--rw version          uint8
           |  +--rw hellointerval    uint32
           |  +--rw deadtime         uint32
           |  +--rw keepalivetime    uint32
           +--rw service-channel* [vxlan-tunnel-id]
           |  +--rw vxlan-tunnel-id      uint32
           |  +--rw vxlan-tunnel-name?   string
           |  +--rw address-family* [af]
           |     +--rw af                        address-family-type
           |     +--rw tunnel-source-ip?         inet:ip-address
           |     +--rw tunnel-destination-ip?    inet:ip-address
           |     +--rw bind-vxlan-id* [vxlan-id]
           |        +--rw vxlan-id    vxlan-id
           +--rw acl
           ... ...
           +--rw qos
           ... ...
```

3.2. YANG Data Model for vBNG-CP

The ietf-vbng-cp module is to configure vBNG-CP. The YANG data model includes: vbng-cp-name, netconf-server and PPPoE parameters, etc.

```
module: ietf-vbng-cp
  augment /lne:logical-network-elements/lne:logical-network-element:
    +--rw ietf-vbng-cp
       +--rw vbng-cp-name?      string
       +--rw enable?           boolean
       +--rw netconf-server!
       |  +--rw address-family* [af]
       |  |  +--rw af    address-family-type
       |  |  +--rw ip    inet:ip-address
       |  +--rw user-name?        string
       |  +--rw password?         string
       |  +--rw port?             uint32
       +--rw vbng-pppoe
          +--rw pppoe-switch
          |  +--rw delay-time?             uint16
          |  +--rw keepalive-timer?        enumeration
          |  +--rw ppp-max-payload?        enumeration
          |  +--rw service?                enumeration
          |  +--rw ppp-mru-verify?         enumeration
          |  +--rw keepalive-fast-reply?   enumeration
          +--rw pppoe-cfg* [template]
             +--rw template                     uint32
             +--rw ppp-authentication?          enumeration
             +--rw ppp-check-magic-num?         enumeration
             +--rw ppp-mru?                     uint32
             +--rw pppoe-ac-name?               string
             +--rw pppoe-service-name-omit?     enumeration
             +--rw pppoe-ac-cookie-check?       enumeration
             +--rw pppoe-password-string?       string
             +--rw pppoe-username-string?       string
             +--rw (ppp-quick-redial)?
             |  +--:(quick-redial-disable)
             |  |  +--rw ppp-quick-redial-disable?   enumeration
             |  +--:(fast-response)
             |     +--rw ppp-fast-response?          enumeration
             |     +--rw ppp-quick-redial-enable?    enumeration
             +--rw ppp-keepalive
             |  +--rw ppp-keepalive-timer?   uint32
             |  +--rw ppp-keepalive-count?   uint16
             +--rw ppp-timeout
                +--rw ppp-timeout-negtimeoutsec?    uint8
                +--rw ppp-timeout-authentication?   uint8
```

3.3. YANG Data Model for vBNG-UP

The ietf-vbng-up module is to configure the vBNG-UP. The YANG data model includes: shelf-number, vbng-up-name, netconf-client and keepalive-sink , etc.

```
   module: ietf-vbng-up
     augment /lne:logical-network-elements/lne:logical-network-element:
       +--rw ietf-vbng-up
          +--rw vbng-up* [shelf-no]
             +--rw shelf-no          uint8
             +--rw vbng-up-name?     string
             +--rw netconf-client!
             |  +--rw address-family* [af]
             |  |  +--rw af    address-family-type
             |  |  +--rw ip    inet:ip-address
             |  +--rw user-name?        string
             |  +--rw password?         string
             |  +--rw port?             uint32
             +--rw keepalive-sink?   enumeration
```

4. vBNG YANG Data Model

4.1. ietf-vbng module

```
<CODE BEGINS> file "ietf-vbng@2019-03-08.yang"
    module ietf-vbng{
    namespace "urn:ietf:params:xml:ns:yang:ietf-vbng";
    prefix "vbng";

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

    import ietf-logical-network-element {
      prefix lne;
    }
    organization
      "IETF NETCONF Working Group";

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

         Editor:   Guangping Huang
                   <mailto:huang.Guangping@zte.com.cn>
        ";

    description
       "The YANG module defines a generic configuration
         model for vbng";
```

```
    revision 2019-03-08{
      description "Initial a new vbng control and user plane separation
          yang data model, it includes ietf-vbng, ietf-vbng-cp,and ietf-vbng-up,
          this module is ietf-vbng";
      reference
         "draft-cuspdt-rtgwg-cu-separation-yang-model-02";
    }

   /* Typedefs  */

     typedef vlan-id {
      type uint16 {
        range "0..4094";
      }
      description
        "Typedef for VLAN ID.";
    }

    typedef vxlan-id {
      type uint32;
      description
        "Typedef for VxLAN ID.";
    }

     typedef address-family-type {
       type enumeration {
         enum ipv4 {
           description
             "IPv4";
         }
         enum ipv6 {
           description
             "IPv6";
         }
       }
       description
         "Typedef for address family type.";
     }

    /* Configuration Data */
    augment /lne:logical-network-elements/lne:logical-network-element {
    container ietf-vbng{
        container interfaces {
          list interface {
            key name;
            leaf name {
               type if:interface-ref;
               description "interface name";
```

```
            }
          container ethernet {
            leaf lacp {
             type boolean;
             description "enable lacp function";
             }
             description "configure ethernet interface";
            }
          leaf mac-offset {
             type uint32;
             description "configure mac offset";
            }
          container vlans {
              list tag {
                  key index;
                  max-elements 2;
                  leaf index {
                     type uint8 {
                       range "0..1";
                      }
                      must ". = 0 or
                        count(../../tag[index = 0]/index) > 0" {
                        error-message "An inner tag can only be specified
                                       if anouter tag has also been specifie
d";
                       description "Ensure that an inner tag cannot be
                               specified without an outer tag'";
                       }

                      description "The index into the tag stack, outermost
                                   tag assigned index 0";
                  }

                  container tag{
                    leaf tag-type {
                       type string;
                       description "tag type";
                       }
                    leaf vlan-id {
                      type vlan-id;
                      description "vlan id value";
                     }

                 description "tag";
                 }
               description "tag list";
             }
            description "vlans";
          }
```

```
           description "interfaces list";
          }
         description "interface container";
        }

        container control-channel {
          leaf name {
            type string;
            description "control channel protocol logical name";
          }
          leaf id {
            type uint32;
            description "the s-cusp session id";
          }
          leaf port {
            type uint32;
            description "s-cusp tcp connection port number";
          }
          leaf version {
            type uint8;
            description "s-cusp version number";
          }
          leaf hellointerval {
            type uint32;
            description "s-cusp hello interval";
          }
          leaf deadtime {
            type uint32;
            description "s-cusp dead time";
          }
          leaf keepalivetime {
            type uint32;
            description "s-cusp keepalive time";
          }

          description "configure s-cusp parameters";
        }

        list service-channel{
            key vxlan-tunnel-id;
            leaf vxlan-tunnel-id {
               type uint32;
               description
                     "Static VxLAN tunnel ID.";
            }

            leaf vxlan-tunnel-name {
               type string;
```

```
               description
               "Name of the static VxLAN tunnel.";
            }

            list address-family {
               key "af";
               leaf af {
                 type address-family-type;
                 description
                 "Address family type value.";
               }

               leaf tunnel-source-ip {
                 type inet:ip-address;
                 description
                 "Source IP address for the static VxLAN tunnel";
               }

               leaf tunnel-destination-ip {
                  type inet:ip-address;
                  description
                   "Destination IP address for the static VxLAN tunnel";
               }

               list bind-vxlan-id {
                  key vxlan-id;
                  leaf vxlan-id {
                    type vxlan-id;
                    description
                       "VxLAN ID.";
                  }
                  description
                     "VxLAN ID list for the VTEP.";
               }

                description
                   "Per-af params.";
             }
            description
              "Configure VxLAN channel";
        }
       description "ietf-bng configuration!";
    }
   description "augment lne model";
  }
}
<CODE ENDS>
```

4.2. ietf-vbng-cp module

```
<CODE BEGINS> file "ietf-vbng-cp@2019-03-08.yang"
    module ietf-vbng-cp{
    namespace "urn:ietf:params:xml:ns:yang:ietf-vbng-cp";
    prefix "vbng-cp";

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

    import ietf-interfaces {
      prefix if;
     }

    import ietf-logical-network-element {
      prefix lne;
    }

    organization
      "IETF NETCONF Working Group";

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

         Editor:   Guangping Huang
                   <mailto:huang.guangping@zte.com.cn>
        ";

    description
       "The YANG module defines a generic configuration
        model for vbng-cp";

    revision 2019-03-08{
      description "Initial a new vbng control and user plane separation
      yang data model, it includes ietf-vbng, ietf-vbng-cp,and ietf-vbng-up, thi
s
      is ietf-vbng-cp";
      reference
         "draft-cuspdt-rtgwg-cu-separation-yang-model-02";
    }

   /* Typedefs  */

     typedef address-family-type {
       type enumeration {
         enum ipv4 {
           description
```

```
            "IPv4";
          }
          enum ipv6 {
            description
              "IPv6";
          }
        }
        description
          "Typedef for address family type.";
      }

    /* Configuration Data */

    augment /lne:logical-network-elements/lne:logical-network-element {
     container ietf-vbng-cp{
       leaf bng-cp-name  {
       type string;
       description "configure vbng-cp name";
     }
      leaf enable  {
       type boolean;
       description "'true' to support vbng separation";
      }
     container netconf-server {
        presence netconf-server ;
        list address-family {
           key "af";
           leaf af {
              type address-family-type;
              description
                "Address family type value.";
           }
           leaf ip {
            type inet:ip-address;
            mandatory true ;
            description 'Configure ip address of netconf server.';
           }
            description "address family list";
        }
        leaf user-name {
          type string {
            length 1..65 ;
          }
         description 'configure user name, default: "who".';
        }
        leaf password {
          type string {
           length 3..32 ;
```

```
          }
        description 'configure password, default: "who".';
        }

        leaf port {
          type uint32;
          description 'Configure port.';
        }
         description 'Configure netconf server.';
      }
      container vbng-pppoe {
       container pppoe-switch {
        leaf delay-time {
          type uint16 {
            range 1..300 ;
          }
          description 'Trigger user offline when VCC phys-interface down';
        }
        leaf keepalive-timer {
          type enumeration {
            enum start {
              value 1 ;
              description "start keepalive timer";
            }
            enum stop {
              value 0 ;
              description "stop keepalive timer";
            }
          }
          default start ;
          description 'Start or stop send keepalive packet';
        }
        leaf ppp-max-payload {
          type enumeration {
            enum disable {
              value 0 ;
              description "disable ppp max payload";
            }
            enum enable {
              value 1 ;
              description "enable ppp max payload";
            }
          }
          default disable ;
          description 'Enable or disable pppoe ppp-max-payload';
        }
        leaf service {
          type enumeration {
```

```
            enum advertise{
              value 1 ;
              description "enable ppp service!";
            }
            enum disable {
              value 0 ;
              description "disable ppp service!";
            }
          }
          default advertise ;
           description 'Open or close pppoe service';
        }
        leaf ppp-mru-verify {
          type enumeration {
            enum open {
              value 1 ;
              description "enable ppp mru verify!";
            }
            enum close {
              value 0 ;
              description "disable ppp mru!";
            }
          }
          default close ;
          description 'set ppp lcp mru verify when mru over 1492';
        }

        leaf keepalive-fast-reply {
          type enumeration {
            enum enable {
              value 1 ;
              description 'Enable keepalive fast reply!';
            }
            enum disable {
              value 0 ;
              description 'Disable keepalive fast reply!';
            }
          }
          description 'Set keepalive fast reply flag.';
        }
        description 'Configuration about pppoe switch.';
      }
      list pppoe-cfg {
        key template ;
        leaf template {
          type uint32 {
            range 1..1000 ;
          }
```

```
          description 'PPPoX template number';
        }
        leaf ppp-authentication {
          type enumeration {
            enum pap {
              value 1 ;
              description "configure pap authentication!";
            }
            enum chap {
              value 2 ;
              description "configure chap authentication!";
            }
            enum mschapv1 {
              value 6 ;
              description "configure mschapv1 authentication!";
            }
            enum mschapv2 {
              value 7 ;
              description "configure mschapv2 tication!";
            }
            enum pap-chap {
              value 21 ;
              description "configure pap-chap authentication!";
            }
          }
          default pap-chap ;
          description 'Set ppp authentication';
        }
        leaf ppp-check-magic-num {
          type enumeration {
            enum disable {
              value 0 ;
              description 'disable ppp magic check';
            }
            enum enable {
              value 1 ;
              description 'enable ppp magic check';
            }
          }
          default enable ;
          description 'Check magic number or not';
        }
        leaf ppp-mru {

          type uint32 {
            range 320..9000 ;
          }
          default 1492 ;
```

```
          description 'Set mru value';
        }
        leaf pppoe-ac-name {
          type string ;
          description 'Set ac-name';
        }
        leaf pppoe-service-name-omit {
          type enumeration {
            enum disable {
              value 0 ;
              description "disable pppoe service name omit";
            }
            enum enable {
              value 1 ;
              description "enable pppoe service name omit";
            }
          }
          default disable ;
          description 'Check service-name value';
        }
        leaf pppoe-ac-cookie-check {
          type enumeration {
            enum disable {
              value 0 ;
              description "disable pppoe ac cookie check";
            }
            enum enable {
              value 1 ;
              description "enable pppoe ac cookie check";
            }
          }
          default enable ;
          description 'Check options';
        }
        leaf pppoe-password-string {
          type string ;
          description 'Set authentication failure password string';
        }
        leaf pppoe-username-string {
          type string ;
          description 'Set authentication failure username error string';
        }

        choice ppp-quick-redial {
          case quick-redial-disable {
            leaf ppp-quick-redial-disable {
              type enumeration {
                enum disable {
```

```
                value 0 ;
                description "disable ppp quick redial";
              }
            }
            default disable ;
            description 'disable quick-redial';
          }
          description 'disable quick-redial';
        }
        case fast-response {
          leaf ppp-fast-response {
            type enumeration {
              enum diable {
                value 0 ;
                 description "disable ppp fast response";
              }
              enum enable {
                value 1 ;
                description "enable ppp fast response";
              }
            }
            description 'set Response the access request immediately';
          }
          leaf ppp-quick-redial-enable {
            type enumeration {
              enum enable {
                value 1 ;
                description "enable ppp quick redial";
              }
            }
            default enable ;
            description 'Enable quick-redial';
          }
          description 'set quick-redial or Response the access request immedia
tely';
        }
        default quick-redial-disable ;
        description 'Enable or disable quick-redial';
      }
      container ppp-keepalive {
        leaf ppp-keepalive-timer {
          type uint32 {
            range 10..14400 ;
          }
          default 60 ;
          description 'Set keepalive time(unit:seconds)';
        }
        leaf ppp-keepalive-count {
          type uint16 {
```

```
              range 1..10 ;
            }
            default 3 ;
            description 'Set keepalive counter';
          }
         description 'Set keepalive time and counter';
        }
        container ppp-timeout {
          leaf ppp-timeout-negtimeoutsec {
            type uint8 {
              range 1..10 ;
            }
            default 3 ;
            description 'Set ppp negtimeoutsec timeout(unit:seconds)';
          }
          leaf ppp-timeout-authentication {
            type uint8 {
              range 1..10 ;
            }
            default 3 ;
            description 'Set ppp authentication timeout(unit:seconds)';
          }
         description 'Set ppp negtimeoutsec and authentication timeout';
        }
        description 'Configuration pppoe template';
      }
      description 'Configuration vBRAS PPPoE.';
    }
     description "configure bng-cp";
  }
  description "augment lne model";
 }
}
<CODE ENDS>
```

4.3. ietf-vbng-up module

```
<CODE BEGINS> file "ietf-vbng-up@2019-03-08.yang"
    module ietf-vbng-up{
    namespace "urn:ietf:params:xml:ns:yang:ietf-vbng-up";
    prefix "vbng-up";

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

    import ietf-logical-network-element {
      prefix lne;
```

```
    }

    organization
      "IETF NETCONF Working Group";

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

         Editor:   Guangping Huang
                   <mailto:huang.Guangping@zte.com.cn>
        ";

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

    revision 2019-03-08{
      description "Initial a new vbng control and user plane separation
      yang data model, it includes ietf-vbng, ietf-vbng-cp,and ietf-vbng-up, thi
s
      is ietf-vbng-up";
      reference
         "draft-cuspdt-rtgwg-cu-separation-yang-model-02";
    }

   /* Typedefs  */

     typedef address-family-type {
       type enumeration {
         enum ipv4 {
           description
             "IPv4";
         }
         enum ipv6 {
           description
             "IPv6";
         }
       }
       description
         "Typedef for address family type.";
     }

    /* Configuration Data */

    augment /lne:logical-network-elements/lne:logical-network-element {
    container ietf-vbng-up{
      list vbng-up {
        key shelf-no ;
```

```
        leaf shelf-no {
          type uint8 {
            range 1..127 ;
          }
          description 'Configure shelf-no of forwarder,1-127.';
        }

        leaf vbng-up-name {
          type string {
            length 1..31 ;
          }
          description 'Configure bng up name.' ;
        }
        container netconf-client {
          presence netconf-client ;
          list address-family {
           key "af";
           leaf af {
              type address-family-type;
              description
                "Address family type value.";
           }
           leaf ip {
            type inet:ip-address;
            mandatory true ;
            description 'Configure ip address of netconf server.';
           }
           description "address family list";
          }
          leaf user-name {
            type string {
              length 1..65 ;
            }
            description 'configure user name, default: "who".';
          }

          leaf password {
            type string {
              length 3..32 ;
            }
            description 'configure password, default: "who".';
          }

          leaf port {
            type uint32;
            description 'Configure port.';
          }
         description 'Configure netconf server.';
```

```
        }

        leaf keepalive-sink {
          type enumeration {
            enum enable {
              value 1 ;
              description 'enable the keepalive-sink function';
            }
            enum disable {
              value 0 ;
              description 'disable keepalive-sink function';
            }
          }
          description "configure keepalive-sink";
         }
         description "configure vbng-up list";
        }
      description "vbng-up configuration!";
    }
    description "augment lne model";
   }
  }
<CODE ENDS>
```

5. Security Considerations

The YANG module specified in this document defines 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.

vBNG(vBNG-CP, vBNG-UP) represents device and network configuration information based on the LNE. 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-rtgwg-lne-model].

The vulnerable "config true" parameters and subtree are the following:

lne:logical-network-elements/lne:logical-network-element/ietf-vbng/interfaces: this subtree specifies vBNG-UP interface parameters configuration. Modify the configuration can cause the vBNG-UP interfaces disable.

lne:logical-network-elements/lne:logical-network-element/ietf-vbng/control-channel: this subtree specifies control channel parameters configuration. Modify the configuration can cause the S-CUSP protocol sessions interrupted among the vBNG-CPs and vBNG-UPs.

lne:logical-network-elements/lne:logical-network-element/ietf-vbng/service-channel: this subtree specifies the service channel parameters configuration among vbng user planes and control plane. Modify the configuration can cause the VxLAN session interrupted among vBGN-UPs and vBNG-CPs.

lne:logical-network-elements/lne:logical-network-element/ietf-vbng-cp/netconf-server: this subtree specifies netconf parameters of vBNG-CP. Modify the configuration can cause the netconf session among vBNG-CPs and vBNG-UPs interrupted.

lne:logical-network-elements/lne:logical-network-element/ietf-vbng-cp/vbng-pppoe: this subtree specifies PPPoE parameters of vBNG-CP. Modify the configuration can cause the PPPoE session interrupted.

lne:logical-network-elements/lne:logical-network-element/ietf-vbng-cp/netconf-client: this subtree specifies netconf parameters of vBNG-UP. Modify the configuration can cause the netconf session among vBNG-CP and vBNG-UP interrupted.

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.

6. Acknowledgements

7. IANA Considerations

This document registers three URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-vbng.

Registrant Contact: The IESG.

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

URI: urn:ietf:params:xml:ns:yang:ietf-vbng-cp.

Registrant Contact: The IESG.

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

URI: urn:ietf:params:xml:ns:yang:ietf-vbng-up.

Registrant Contact: The IESG.

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

This document registers three YANG modules in the YANG Module Names registry [RFC6020].

```
name:        ietf-vbng
namespace:   urn:ietf:params:xml:ns:yang:ietf-vbng
prefix:      vbng
reference:   RFC XXXX

name:        ietf-vbng
namespace:   urn:ietf:params:xml:ns:yang:ietf-vbng-cp
prefix:      vbng-cp
reference:   RFC XXXX

name:        ietf-vbng
namespace:   urn:ietf:params:xml:ns:yang:ietf-vbng-up
prefix:      vbng-up
reference:   RFC XXXX
```

8. References

8.1. Normative References

[I-D.asechoud-rtgwg-qos-model]
Choudhary, A., Jethanandani, M., Strahle, N., Aries, E., and I. Chen, "YANG Model for QoS", draft-asechoud-rtgwg-qos-model-10 (work in progress), July 2019.

[I-D.ietf-netmod-acl-model]
Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-21 (work in progress), November 2018.

[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-10 (work in progress), March 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>.

8.2. Informative References

[I-D.cuspdt-rtgwg-cu-separation-bng-protocol]
Hu, S., Eastlake, D., Chen, M., Qin, F., Li, Z., Chua, T., and D. Huang, "Control-Plane and User-Plane Separation BNG Simple Control Channel Protocol (S-CUSP)", draft-cuspdt-rtgwg-cu-separation-bng-protocol-06 (work in progress), July 2019.

[I-D.hu-nvo3-vxlan-gpe-extension-for-vbng]
Hu, S., Qin, F., Wang, Z., and D. Huang, "VXLAN GPE Extension for Packets Exchange Between Control and User Plane of vBNG", draft-hu-nvo3-vxlan-gpe-extension-for-vbng-00 (work in progress), June 2019.

Authors' Addresses

Guangping Huang
ZTE Corporation
No.50, Software Avenue
Nanjing, Jiangsu 210012
China

Email: huang.guangping@zte.com.cn

Shujun Hu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: shujun_hu@outlook.com

Fengwei Qin
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: qinfengwei@chinamobile.com

rtgwg
Internet-Draft
Intended status: Informational
Expires: April 25, 2019

S. Hu
China Mobile
V. Lopez
Telefonica
F. Qin
Z. Li
China Mobile
T. Chua
Singapore Telecommunications Limited
Donald. Eastlake
M. Wang
J. Song
Huawei

October 22, 2018

Requirements for Control Plane and User Plane Separated BNG Protocol draft-cuspdt-rtgwg-cusp-requirements-03

Abstract

This document introduces the Control Plane and User Plane separated BNG (Broadband Network Gateway) architecture and defines a set of associated terminology. It also specifies a set of protocol requirements for communication between the BNG-CP and the BNG-UPs in the Control Plane and User Plane Separated BNG.

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 April 25, 2019.

Table of Contents

1. Introduction

A Broadband Network Gateway (BNG) is an Ethernet-centric IP edge router and the aggregation point for user traffic. To provide centralized session management, flexible address allocation, high scalability for subscriber management capacity, and cost-efficient redundancy, the CU separated BNG is introduced [TR-384]. The CU separated Service Control Plane could be virtualized and centralized;

it is responsible for user access authentication and sending forwarding entries to user planes. The routing control and forwarding plane, i.e. BNG user plane (local), could be distributed across the infrastructure.

This document introduces the Control Plane and User Plane separated BNG architecture and modeling. This document also defines the protocol requirements for Control Plane and User Plane Separated BNG (CUSP).

2. Concept and 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.

2.1. Terminology

BNG: Broadband Network Gateway. A broadband remote access server (BRAS, B-RAS or BBRAS) that routes traffic to and from broadband remote access devices such as digital subscriber line access multiplexers (DSLAM) on an Internet service provider's (ISP) network. BRAS can also be referred to as a Broadband Network Gateway (BNG).

CP: Control Plane. The CP is a user control management component which manages UP's resources such as the user entry and user's QoS policy.

CUSP: Control Plane and User Plane Separated BNG Protocol.

UP: User Plane. UP is a network edge and user policy implementation component. The traditional router's Control Plane and forwarding plane are both preserved on BNG devices in the form of a user plane.

3. CU Separated BNG Model

Figure 1 shows the architecture of CU separated BNG

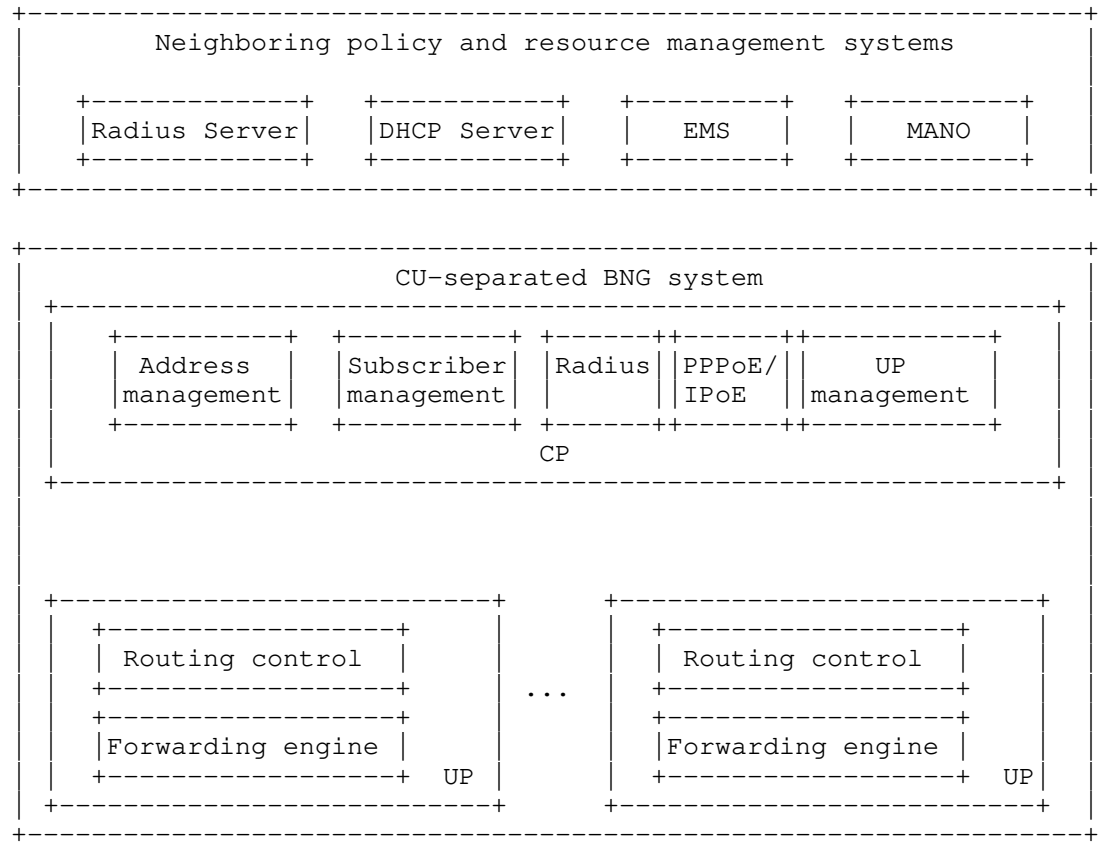


Figure 1. Architecture of CU Separated BNG

Briefly, a CU separated BNG is made up of a Control Plane (CP) and a set of User Planes (UPs) [TR-384], [I-D.cuspdt-rtgwg-cu-separation-bng-deployment]. The Control Plane is a user control management component which manages UP's resources such as the user entry and user's Quality of Service (QoS) policy, for example, the access bandwidth and priority management. This Control Plane could be virtualized and centralized. The functional modules inside the BNG Service Control Plane can be implemented as Virutl Network Functions (VNFs) and hosted in a Network Function Virtualization Infrastructure (NFVI). The User Plane Management module in the BNG control plane centrally manages the distributed BNG user planes (e.g. load balancing), as well as the setup, deletion, update, and maintenance of channels between control planes and user planes [TR-384], [I-D.cuspdt-rtgwg-cu-separation-bng-deployment]. The User Plane (UP) is a network edge and user policy implementation component. It can support the forwarding plane functions on traditional BNG devices,

such as traffic forwarding, QoS, and traffic statistics collection, and it can also support the control plane functions on traditional BNG devices, such as routing, multicast, etc [TR-384], [I-D.cuspdt-rtgwg-cu-separation-bng-deployment].

3.1. Internal interfaces between the CP and UP

To support communication between the Control Plane and User Plane, several interfaces are involved. Figure 2 illustrates the three internal interfaces of CU Separated BNG.

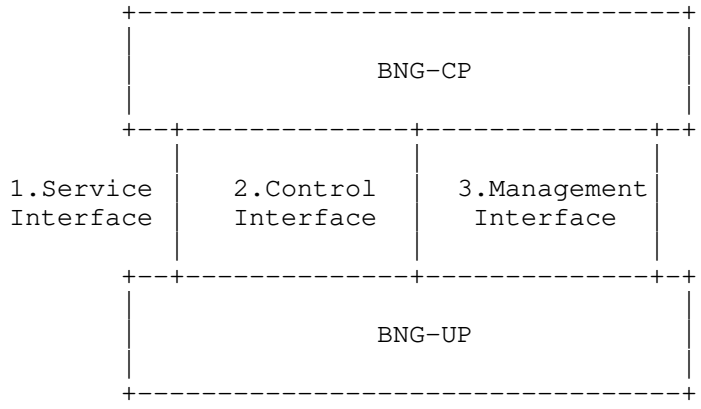


Figure 2. Interfaces between the BNG-CP and the BNG-UP

Service interface: The CP and UP use this interface to establish VXLAN tunnels with each other and transmit PPPoE and IPoE packets over the VXLAN tunnels.

Control interface: The CP uses this interface to deliver service entries, and the UP uses this interface to report service events to the CP.

Management interface: The CP uses this interface to deliver configurations to the UP. This interface uses NETCONF.

The CUSP (Control plane and User plane Separated BNG protocol) defines the control interface, and specifies the communication between the centralized control plane and user planes. This protocol should be designed to support establishing and maintaining a conversation between CP and UPs, and transporting the tables that are specified in [draft-cuspdt-rtgwg-cu-separation-infor-model].

4. The usage of CU separation BNG protocol

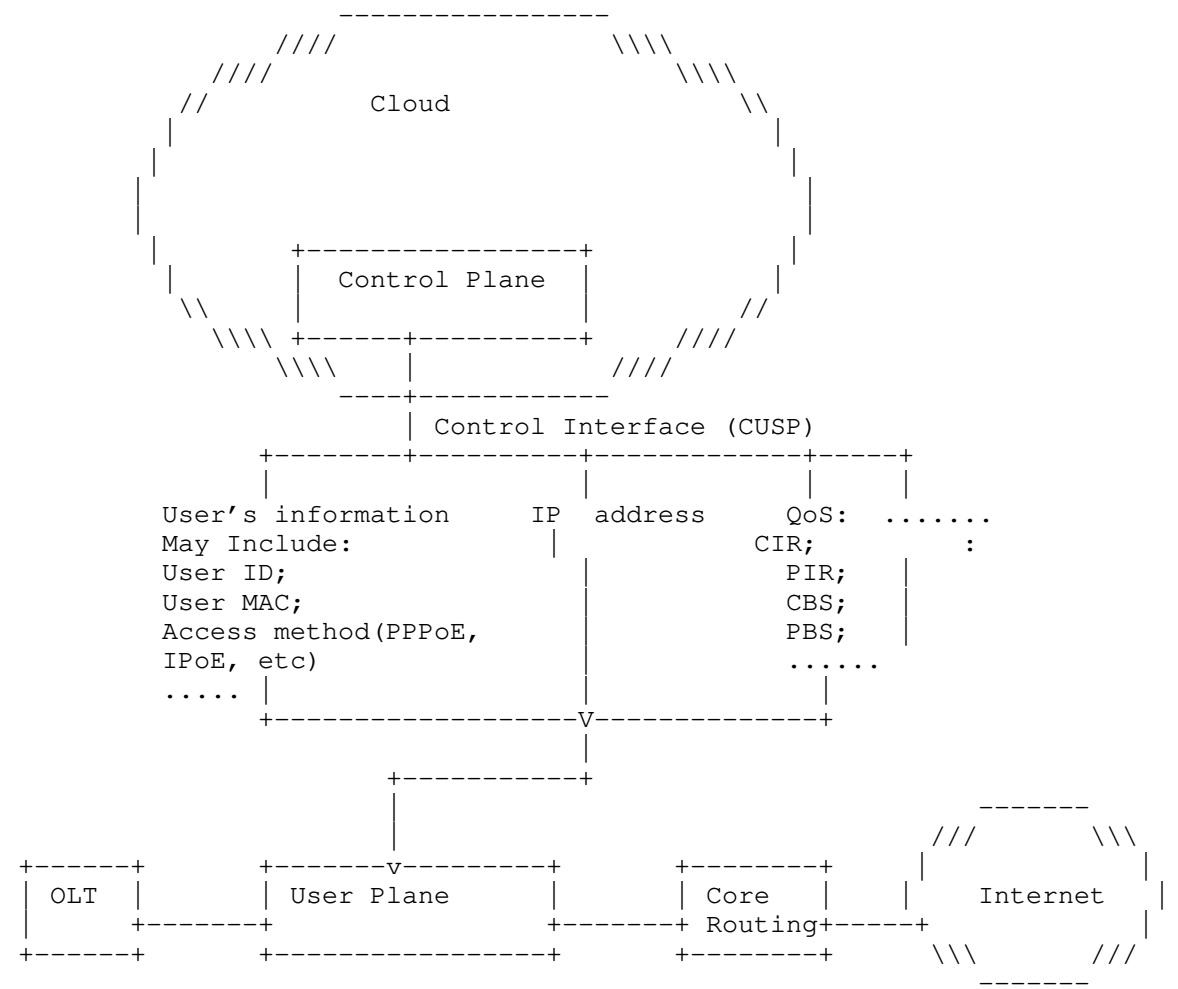


Figure 3. CU Separation BNG protocol usage

As shown in Figure 3, when users access the BNG network, the control plane solicits user information (such as user's ID, user's MAC, user's access methods, for example via PPPoE/IPoE), associates users with available bandwidth which is reported by User planes, and, based on the service's requirement, generates a set of tables, which may include user's information, UP's IP segment, and QoS, etc. Then the control plane can transmit these tables to the User planes. User

planes receive these tables, parse them, and then perform corresponding actions.

5. Control Plane and User Plane Separation Protocol Requirements

This section specifies the requirements for the CU separation protocol.

5.1. Transmit information tables

The Control Plane and User Plane Separation Protocol MUST allow the CP to send tables to each User Plane device.

a) The current BNG service requires that the UP should support at least 2000 users being accessed every second. And every user requires at least 2000 bytes. To achieve high performance, the CU Separation protocol SHOULD be lightweight.

b) CU separation protocol should support data encoded as either XML or binary. It allows user information data to be read, saved, and manipulated with tools specific to XML or binary.

c) In order to provide centralized session management, high scalability for subscriber management capacity, and cost-efficient redundancy, batching ability should be provided. The CU Separation protocol should be able to group an ordered set of commands to a UP device. Each such group of commands SHOULD be sent to the UP in as few messages as practical. Furthermore, the protocol MUST support the ability to specify if a command group MUST have all-or-nothing semantics.

d) The CU Separation protocol SHOULD be able to support at least hundreds of UP devices and tens of thousands of ports. For example, the protocol field sizes corresponding to UP or port numbers SHALL be large enough to support the minimum required numbers. This requirement does not relate to the performance of the system as the number of UPs or ports in the system grows.

5.2. Message Priority

The CU Separation protocol MUST provide a means to express the protocol message priorities.

5.3. Reliability

Heartbeat is a periodic signal generated by hardware or software to test for some aspects of normal operation or to synchronize other parts of network system.

In the CU separation BNG, a heartbeat is sent between CP and UPs at a regular interval on the order of seconds. If the CP/UP does not receive a heartbeat for a time--usually a few heartbeat intervals-- the CP/UP that should have sent the heartbeat is assumed to have failed.

The CU separation protocol should support some kind of heartbeat monitoring mechanism. And this mechanism should have ability to distinguish whether the interruption is an actual failure. For example, in some scenarios (i.e. CP/UP update, etc), the connection between the UP and CP need to be interrupted. In this case, the interruption should not be reported.

5.4. Support for Secure Communication

As mentioned above, CP may send some information tables to the UP which may be critical to the network function (e.g, User Information, IPv4/IPv6 information) and may reflect the business information (e.g, QoS, service level agreements, etc). Therefore, supporting the integrity of all CU Separation protocol messages and protecting against man-in-the-middle attacks MUST be supported.

The CP Separation protocol should support security in a variety of scenarios. For example, the connections between the CP and UPs could be dedicated lines, VPNs within one domain, or could cross several domains, that is, cross third party networks. Thus it is likely that more than one security mechanism SHOULD be supported. TLS and IPsec are good candidates for such mechanisms.

5.5. Version negotiation

The CU separated BNG may consist of different vendors' devices implementing different versions of protocol. Threfore, the CU separation protocol MUST provide some mechanisms to perform the version negotiation.

Version negotiation is the process that the CU separated BNG's Control-Plane uses to evaluate the protocol versions supported by both the control-plane and the user-plane devices. Then a suitable protocol version is selected for communication in CUSP. The process is a "negotiation" because it requires identifying the most recent protocol version that is supported by both the control-plane and the user-plane devices or determining that they have no version in common.

5.6. Capability Exchange

The UP Capability Report displays the device's profile, service capability, and other assigned capabilities within the CU separated BNG. The CU separation protocol should MUST provide some mechanism to exchange the UP device's capabilities.

5.7. CP primary/backup capability

A backup CP for failure recovery is required for the CU separated BNG network. And the CUSP should provide some mechanism to implement the backup CP:

a) In some scenarios, there may be two CP devices both declaring the primary CP. Thus the CUSP should support or associate with some mechanisms to determine which CP is the primary device.

b) In the scenario of the primary CP down, the CUSP should support switching between primary and backup CP.

5.8. Event Notification

The CUSP protocol SHOULD be able to asynchronously notify the CP of events on the UP such as failures and changes in available resources and capabilities. Some scenarios that may initiate event notifications are listed below.

a) Sending response message: As mentioned above, the control plane solicits users' information, associates them with available bandwidth, and generates a set of tables based on the service's requirement. Then the control plane transmits these tables to the conresponding User plane. The UP should respond with an event notification to inform the CP that the tables are received.

b) User trace: The user trace mechanism can support the Control Plane tracing and monitoring the network status for users (for example the real-time bandwidth, etc), to help debug the user's application. Therefore, the UPs SHOULD be able to notify the CP with the User trace message.

c) Sending statistics parameters: In CU separation BNG, the User-plane will report the traffic statistics parameters to the Control-plane, such as the ingress packets, ingress bytes, egress packets, egress bytes, etc. These parameters can help measure the BNG network performance. Available network resources can be allocated basing on the statistics parameters by the BNG-CP. Therefore, the UPs SHOULD be able to notify the CP with statistics parameters.

d) Report the result of User Detect: "User Detect" message will be send periodically to detect user dial-up and disconnect. The UPs SHOULD be able to notify the CP with the result of User Detect.

5.9. Query Statistics

The CUSP protocol MUST provide a means for the CP to be able to query statistics (performance monitoring) from the UP.

6. Security Considerations

As this is an Informational requirements document, detailed technical Security Considerations are not included. However, Section 5.4 covers general security requirements and Section 5.7 covers backup requirements relevant to some denial of service scenarios.

7. IANA Considerations

This document requires no IANA actions.

8. References

8.1. Normative References

[I-D.cuspdt-rtgwg-cu-separation-bng-deployment]
Gu, R., Hu, S., and Z. Wang, "Deployment Model of Control Plane and User Plane Separation BNG", draft-cuspdt-rtgwg-cu-separation-bng-deployment-00 (work in progress), October 2017.

[I-D.cuspdt-rtgwg-cu-separation-infor-model]
Wang, Z., Gu, R., Lopezalvarez, V., and S. Hu, "Information Model of Control-Plane and User-Plane separation BNG", draft-cuspdt-rtgwg-cu-separation-infor-model-00 (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>.

[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>.

8.2. Informative References

[TR-384] Broadband Forum, ""Cloud Central Office Reference Architectural Framework",", BBF TR-384, January. 2018.

Authors' Addresses

Shujun Hu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: shujun_hu@outlook.com

Victor Lopez
Telefonica
Sur 3 building, 3rd floor, Ronda de la Comunicacion s/n
Madrid 28050
Spain

Email: victor.lopezalvarez@telefonica.com

Fengwei Qin
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: qinfengwei@chinamobile.com

Zhenqiang Li
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: lizhenqiang@chinamobile.com

Tee Mong Chua
Singapore Telecommunications Limited
31 Exeter Road, #05-04 Comcentre Podium Block
Singapore City 239732
Singapore

Email: teemong@singtel.com

Donald Eastlake, 3rd
Huawei
1424 Pro Shop Court
Davenport, FL 33896
USA

Email: d3e3e3@gmail.com

Michael Wang
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: wangzitao@huawei.com

Jun Song
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: song.jun@huawei.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: August 6, 2019

L. Dunbar
A. Malis
Huawei
C. Jacquenet
Orange

February 6, 2019

Gap Analysis of Interconnecting Underlay with Cloud Overlay
draft-dm-net2cloud-gap-analysis-04

Abstract

This document analyzes the technological gaps when using SD-WAN to interconnect workloads & apps hosted in various locations, especially cloud data centers when the network service providers do not have or have limited physical infrastructure to reach the locations [Net2Cloud-problem].

Table of Contents

1. Introduction

[Net2Cloud-Problem] describes the problems that enterprises face today in transitioning their IT infrastructure to support digital

economy, such as connecting enterprises' branch offices to dynamic workloads in different Cloud DCs.

This document analyzes the technological gaps to interconnect dynamic workloads & apps hosted in various locations and in Cloud DCs that the enterprise existing VPN service provider might not have or have limited the physical infrastructure to reach. When enterprise' VPN service providers do not have or have insufficient bandwidth to reach a location, SD-WAN is emerged as way to aggregate bandwidth of multiple networks, such as MPLS VPN, Public Internet, etc. This document primarily focuses on the technological gaps of SD-WAN.

For ease of description, SD-WAN edge, SD-WAN end points, C-PE, or CPE are used interchangeably throughout this document.

2. Conventions used in this document

Cloud DC: Third party Data Centers that usually host applications and workload owned by different organizations or tenants.

Controller: Used interchangeably with SD-WAN controller to manage SD-WAN overlay path creation/deletion and monitor the path conditions between sites.

CPE-Based VPN: Virtual Private Secure network formed among CPEs. This is to differentiate from most commonly used PE-based VPNs a la RFC 4364.

OnPrem: On Premises data centers and branch offices

SD-WAN: Software Defined Wide Area Network. In this document, "SD-WAN" refers to the solutions specified by ONUG (Open Network User Group), https://www.onug.net/software-defined-wide-area-network-sd-wan/, which is about pooling WAN bandwidth from multiple underlay networks to get better WAN bandwidth management, visibility & control. When the underlay networks are private

networks, traffic can traverse without additional encryption; when the underlay networks are public, such as Internet, some traffic needs to be encrypted when traversing through (depending on user provided policies).

3. Gap Analysis of C-PEs Registration Protocol

SD-WAN, conceived in ONUG (Open Network User Group) a few years ago as a means to aggregate multiple connections between any two points, has emerged as an on-demand technology to securely interconnect the OnPrem branches with the workloads instantiated in Cloud DCs that do not connect to BGP/MPLS VPN PEs or have very limited bandwidth.

Some SD-WAN networks use the NHRP protocol [RFC2332] to register SD-WAN edges with a "Controller" (or NHRP server), which then has the ability to map a private VPN address to a public IP address of the destination node. DSVPN [DSVPN] or DMVPN [DMVPN] are used to establish tunnels among SD-WAN edge nodes.

NHRP was originally intended for ATM address resolution, and as a result, it misses many attributes that are necessary for dynamic endpoint C-PE registration to controller, such as:

- Interworking with MPLS VPN control plane. A SD-WAN edge can have some ports facing MPLS VPN network and some ports facing public Internet that requires encryption for some sensitive data to traverse.
- Scalability. NHRP/DSVPN/DMVPN works fine with small number of edge nodes. When a network has more than 100 nodes, the protocol does not work well.
- NHRP does not have the IPsec attributes, which are needed for peers to build Security Associations over public internet.
- NHRP does not have field to indicate C-PE supported encapsulation types, such as IPsec-GRE, IPsec-VxLAN, or others.
- NHRP does not have field to indicate C-PE Location identifier, such as Site Identifier, System ID, and/or Port ID.
- NHRP does not have field to describe the gateway to which the C-PE is attached. When a C-PE is instantiated in a Cloud DC, to

establish connection to the C-PE, it is necessary to know the Cloud DC operator's Gateway to which the CPE is attached.
- NHRP does not have field to describe C-PE's NAT properties if the C-PE is using private addresses, such as the NAT type, Private address, Public address, Private port, Public port, etc.

[BGP-SDWAN-EXT] describes how to use BGP for SD-WAN edge nodes to register its properties to SD-WAN controller, which then disseminates the information to other SD-WAN edge nodes that are authenticated to communicate.

4. Gap Analysis in aggregating VPN paths and Internet paths

Most likely, enterprises especially large ones already have their CPEs interconnected by providers' VPNs, such as EVPN, L2VPN, or L3VPN. The VPN can be PE based or CPE based as shown in the following diagram. The commonly used CPE-based VPNs have CPE directly attached to PEs, therefore the communication is considered as secure. BGP are used to distribute routes among CPEs, even though sometimes routes among CPEs are statically configured.

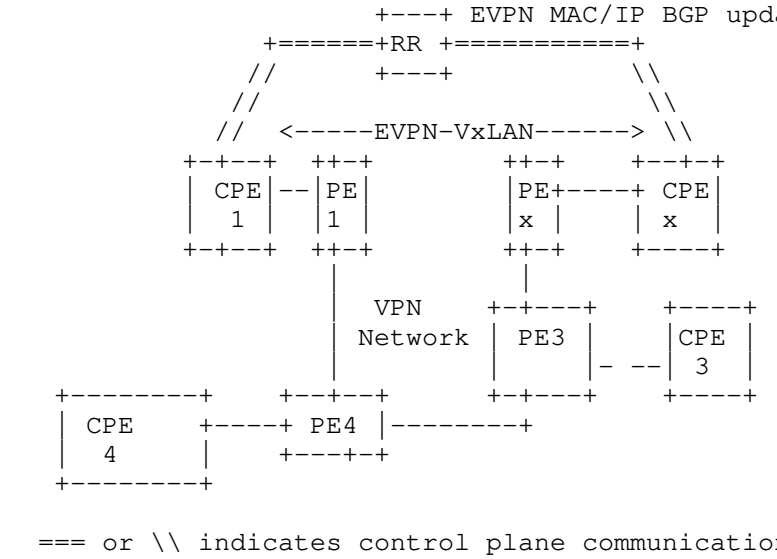


Figure 1: L2 or L3 VPNs over IP WAN

To use SD-WAN to aggregate Internet routes with the VPN routes, the C-PEs need to have some ports connected to PEs and other ports connected to the Internet. It is necessary to have a registration protocol for C-PEs to register with their SD-WAN Controllers to establish secure tunnels among relevant C-PEs.

If using NHRP for registration, C-PEs need to participate in two separate control planes: EVPN&BGP for CPE-based VPNs via links directly attached to PEs and NHRP & DSVPN/DMVPN for ports connected to internet. Two separate control planes not only add complexity to C-PEs, but also increase operational cost.

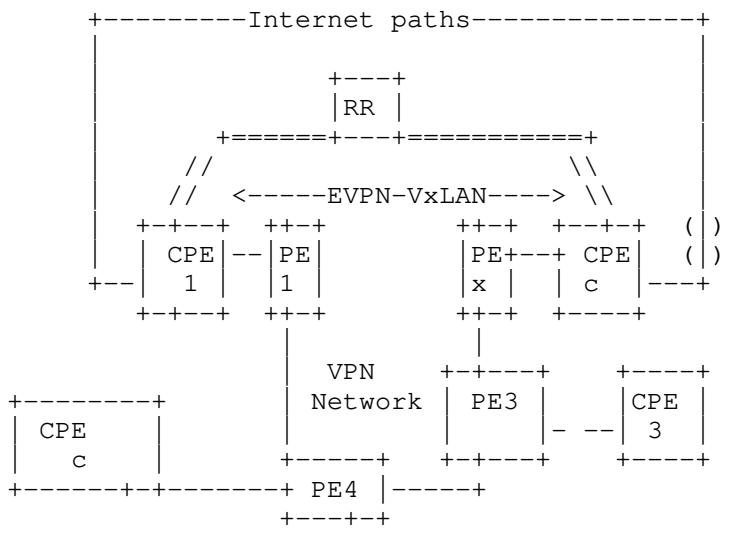


Figure 2: CPEs interconnected by VPN paths and Internet Paths

4.1. Gap analysis of Using BGP to cover SD-WAN paths

Since C-PE already supports BGP, it is desirable to consider using BGP to control the SD-WAN instead of two separate control planes. This section analyzes the gaps of using BGP to control SD-WAN.

As described [BGP-SDWAN-Usage], SD-WAN Overlay Control Plane has three distinct functional tiers:

- SD-WAN node's private address and WAN Ports/Addresses registration to the SD-WAN Controller.

  - It is for informing the SD-WAN controller and potential peers of the underlay networks to which the C-PE is connected.

- Controller Facilitated IPsec SA management and NAT information distribution
  - It is for SD-WAN controller to facilitate or manage the IPsec configurations and peer authentications for all IPsec tunnels terminated at the SDWAN nodes. For some scenarios where the WAN ports are private addresses, this step is for informing the type of NAT translating the private addresses to public ones.

- Establishing and Managing the topology and reachability for services attached to the client ports of SD-WAN nodes.
  - This is for the overlay layer's routes distribution, so that a C-PE can establish the overlay routing table that identifies the next hop for reaching a specific route/service attached to remote nodes. [SECURE-EVPN] describes EVPN and other options.

RFC5512 and [Tunnel-Encap] describe methods for endpoints to advertise tunnel information and to trigger tunnel establishment. RFC5512 & [Tunnel-Encap] have the Endpoint Address to indicate IPv4 or IPv6 address format, the Tunnel Encapsulation attribute to indicate different encapsulation formats, such as L2TPv3, GRE, VxLAN, IP-in-IP, etc. There are sub-TLVs to describe the detailed tunnel information for each of the encapsulations.

[Tunnel-Encap] removed SAFI =7 (which was specified by RFC5512) for distributing encapsulation tunnel information. [Tunnel-Encap] require Tunnels being associated with routes.

There is also the Color sub-TLV to describe customer-specified information about the tunnels (which can be creatively used for SD-WAN)

Here are some of the gaps using [Tunnel-Encap] to control SD-WAN:

- Lacking C-PE Registration functionality
- Lacking IPsec Tunnel type

- [Tunnel-Encap] has Remote Address SubTLV, but does not have any field to indicate the Tunnel originating interface, which was in RFC5512.
- The mechanisms described by [Tunnel-Encap] cannot be effectively used for SD-WAN overlay network because a SD-WAN Tunnel can be between Internet facing WAN ports of two C-PEs and needs to be established before data arrival because the tunnel establishment can fail, e.g. two end points supporting different encryption algorithms.
- Client traffic (e.g. an EVPN route) can have option of going through MPLS network natively without encryption, or going through the IPsec tunnels between the internet facing WAN ports of two C-PEs.
- There is no routes to be associated with the SD-WAN Tunnel between two C-PE's internet facing WAN ports, unless consider using the interface facing WAN Port addresses assigned by ISP (Internet Service Providers) as the route for the Tunnel.
  There is a suggestion on using a "Fake Route" for a SD-WAN node to use [Tunnel-Encap] to advertise its SD-WAN tunnel end-points properties. However, using "Fake Route" can create deployment complexity for large SD-WAN networks with many tunnels. For example, for a SD-WAN network with hundreds of nodes, with each node having many ports & many end-points to establish SD-WAN tunnels to their corresponding peers, the node would need many "fake addresses". For large SD-WAN networks (such as has more than 10000 nodes), each node might need 10's thousands of "fake addresses", which is very difficult to manage and needs lots of configuration to get the nodes provisioned.
- Does not have fields to carry detailed information of the remote C-PE: such as Site-ID, System-ID, Port-ID
- Does not have the proper field to express IPsec attributes among the SD-WAN edge nodes to establish proper IPsec Security Associations.
- Does not have proper way for two peer CPEs to negotiate IPSec keys, based on the configuration sent by the Controller.
- Does not have field to indicate the UDP NAT private address <-> public address mapping
- C-PEs tend to communicate with a few other CPEs, not all the C-PEs need to form mesh connections. Without any BGP extension, many

nodes can get dumped with too much information of other nodes that they never need to communicate with.

[VPN-over-Internet] describes a way to securely interconnect C-PEs via IPsec using BGP. This method is useful, however, it still misses some aspects to aggregate CPE-based VPN routes with internet routes that interconnect the CPEs. In addition:

- The draft does not have options of C-PE having both MPLS ports and Internet ports.
- The draft assumes that CPE "registers" with the RR. However, it does not say how. It assumes that the remote CPEs are pre-configured with the IPsec SA manually. In SD-WAN, Zero Touch Provisioning is expected. It is not acceptable to require manual configuration.
- For RR communication with CPE, this draft only mentioned IPSec. Missing TLS/DTLS.
- The draft assumes that CPEs and RR are connected with an IPsec tunnel. With zero touch provisioning, we need an automatic way to synchronize the IPsec SA between CPE and RR. The draft assumes:
  A CPE must also be provisioned with whatever additional information is needed in order to set up an IPsec SA with each of the red RRs
- IPsec requires periodic refreshment of the keys. The draft hasn't addressed how to synchronize the refreshment among multiple nodes.
- IPsec usually only sends configuration parameters to two endpoints and let the two endpoints negotiate the KEY. Now we assume that RR is responsible for creating the KEY for all endpoints. When one endpoint is compromised, all other connections are impacted.

4.2. Gaps in preventing attacks from Internet facing ports

When C-PEs have ports facing Internet, it brings in the security risks of potential DDoS attacks to the C-PEs from the ports facing internet.

To mitigate security risks, in addition to requring Anti-DDoS features on C-PEs to prevent major DDoS attacks, it is necessary to have ways for C-PEs to validate traffic from remote peers to prevent spoofed traffic.

5. Gap analysis of CPEs not directly connected to VPN PEs

Because of the ephemeral property of the selected Cloud DCs, an enterprise or its network service provider may not have the direct links to the Cloud DCs that are optimal for hosting the enterprise's specific workloads/Apps. Under those circumstances, SD-WAN is a very flexible choice to interconnect the enterprise on-premises data centers & branch offices to its desired Cloud DCs.

However, SD-WAN paths over public Internet can have unpredictable performance, especially over long distances and across domains. Therefore, it is highly desirable to place as much as possible the portion of SD-WAN paths over service provider VPN (e.g., enterprise's existing VPN) that have guaranteed SLA to minimize the distance/segments over public Internet.

MEF Cloud Service Architecture [MEF-Cloud] also describes a use case of network operators needing to use SD-WAN over LTE or public Internet for last mile accesses where they are not present.

Under those scenarios, one or both of the SD-WAN endpoints may not directly be attached to the PEs of a VPN Domain.

Using SD-WAN to connect the enterprise existing sites with the workloads in Cloud DC, the enterprise existing sites' CPEs have to be upgraded to support SD-WAN. If the workloads in Cloud DC need to be connected to many sites, the upgrade process can be very expensive.

[Net2Cloud-Problem] describes a hybrid network approach that integrates SD-WAN with traditional MPLS-based VPNs, to extend the existing MPLS-based VPNs to the Cloud DC Workloads over the access paths that are not under the VPN provider control. To make it work properly, a small number of the PEs of the MPLS VPN can be designated to connect to the remote workloads via SD-WAN secure IPsec tunnels. Those designated PEs are shown as fPE (floating PE or smart PE) in Figure 3. Once the secure IPsec tunnels are established, the workloads in Cloud DC can be reached by the enterprise's VPN without upgrading all of the enterprise's existing CPEs. The only CPE that needs to support SD-WAN would be a virtualized CPE instantiated within the cloud DC.

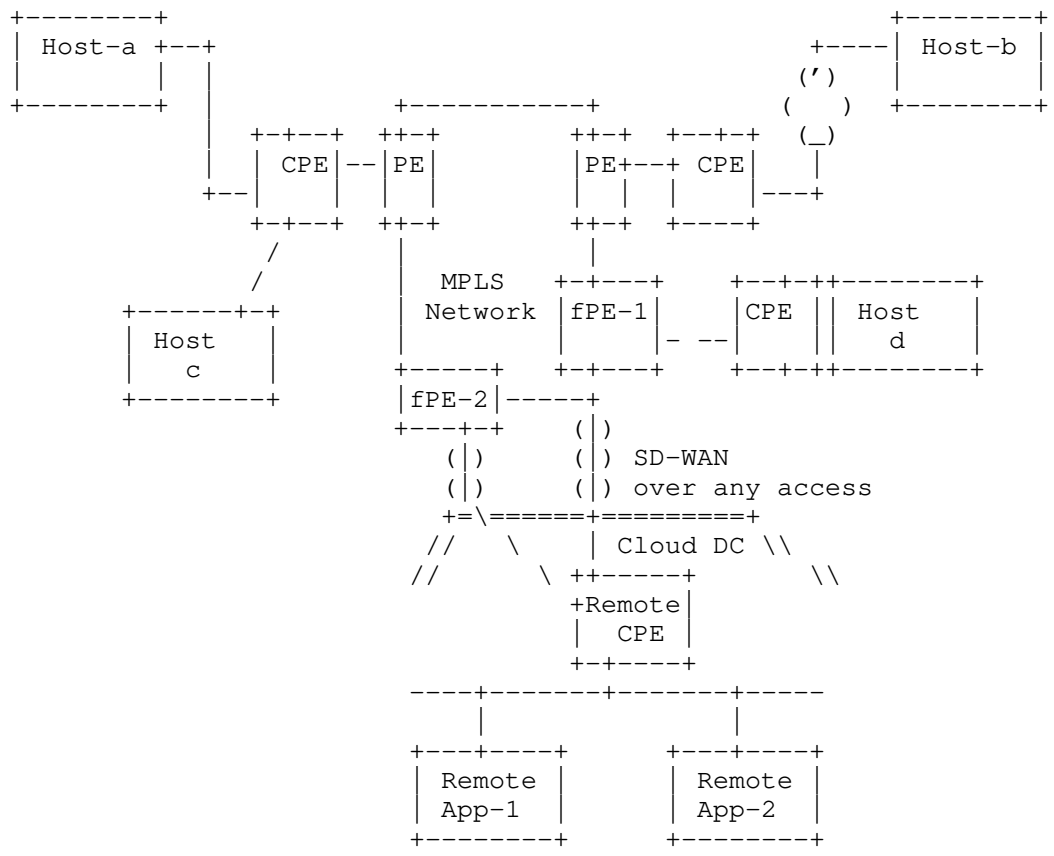


Figure 3: VPN Extension to Cloud DC

In Figure 3, the optimal Cloud DC to host the workloads (due to proximity, capacity, pricing, or other criteria chosen by the enterprises) does not happen to have a direct connection to the PEs of the MPLS VPN that interconnects the enterprise's existing sites.

5.1. Gap Analysis of Floating PEs to connect to Remote CPEs

To extend MPLS VPNs to remote CPEs, it is necessary to establish secure tunnels (such as IPsec tunnels) between the Floating PEs and the remote CPEs.

Gap:

Even though a set of PEs can be manually selected to act as the floating PEs for a specific cloud data center, there are no standard protocols for those PEs to interact with the remote CPEs (most likely virtualized) instantiated in the third party cloud data centers (such as exchanging performance or route information).

When there is more than one fPE available for use (as there should be for resiliency or the ability to support multiple cloud DCs scattered geographically), it is not straightforward to designate an egress fPE to remote CPEs based on applications. There is too much applications' traffic traversing PEs, and it is not feasible for PEs to recognize applications from the payload of packets.

5.2. NAT Traversal

Most cloud DCs only assign private addresses to the workloads instantiated. Therefore, traffic to/from the workload usually need to traverse NAT.
A SD-WAN edge node can inquire STUN (Session Traversal of UDP Through Network Address Translation RFC 3489) Server to get the NAT property, the public IP address and the Public Port number to pass to peers.

5.3. Complication of using BGP between PEs and remote CPEs via Internet

Even though an EBGP (external BGP) Multi-hop design can be used to connect peers that are not directly connected to each other, there are still some complications/gaps in extending BGP from MPLS VPN PEs to remote CPEs via any access paths (e.g., Internet).

The path between the remote CPEs and VPN PE can traverse untrusted nodes.

EBGP Multi-hop scheme requires static configuration on both peers. To use EBGP between a PE and remote CPEs, the PE has to be manually configured with the "next-hop" set to the IP addresses of the CPEs. When remote CPEs, especially remote virtualized CPEs are dynamically instantiated or removed, the configuration on the PE Multi-Hop EBGP has to be changed accordingly.

Gap:

Egress peering engineering (EPE) is not enough. Running BGP on virtualized CPEs in Cloud DC requires GRE tunnels being established first, which in turn requires address and key management for the remote CPEs. RFC 7024 (Virtual Hub & Spoke) and Hierarchical VPN is not enough.

Also there is a need for a method to automatically trigger configuration changes on PE when remote CPEs' are instantiated or moved (leading to an IP address change) or deleted.

EBGP Multi-hop scheme does not have an embedded security mechanism. The PE and remote CPEs need secure communication channels when connecting via the public Internet.

Remote CPEs, if instantiated in Cloud DC, might have to traverse NAT to reach PE. It is not clear how BGP can be used between devices outside the NAT and the entities behind the NAT. It is not clear how to configure the Next Hop on the PEs to reach private IPv4 addresses.

5.4. Designated Forwarder to the remote edges

Among multiple floating PEs available for a remote CPE, multicast traffic from the remote CPE towards the MPLS VPN can be forwarded back to the remote CPE due to the PE receiving the multicast data frame forwarding the multicast/broadcast frame to other PEs that in turn send to all attached CPEs. This process may cause traffic loop.

Therefore, it is necessary to designate one floating PE as the CPE's Designated Forwarder, similar to TRILL's Appointed Forwarders [RFC6325].

Gap: the MPLS VPN does not have features like TRILL's Appointed Forwarders.

5.5. Traffic Path Management

When there are multiple floating PEs that have established IPsec tunnels to the remote CPE, the remote CPE can forward the outbound traffic to the Designated Forwarder PE, which in turn forwards the

traffic to egress PEs to the destinations. However, it is not straightforward for the egress PE to send back the return traffic to the Designated Forwarder PE.

Example of Return Path management using Figure 3 above.

- fPE-1 is DF for communication between App-1 <-> Host-a due to latency, pricing or other criteria.
- fPE-2 is DF for communication between App-1 <-> Host-b.

6. Manageability Considerations

Zero touch provisioning of SD-WAN edge nodes is expected in SD-WAN deployment. It is necessary for a newly powered up SD-WAN edges to establish a secure connection (such as TLS, DTLS, etc.) to its controller.

7. Security Considerations

The intention of this draft is to identify the gaps in current and proposed SD-WAN approaches that can address requirements identified in [Net2Cloud-problem].

Several of these approaches have gaps in meeting enterprise security requirements when tunneling their traffic over the Internet, as is the general intention of SD-WAN. See the individual sections above for further discussion of these security gaps.

8. IANA Considerations

This document requires no IANA actions. RFC Editor: Please remove this section before publication.

9. References

9.1. Normative References

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

9.2. Informative References

[RFC8192] S. Hares, et al, "Interface to Network Security Functions (I2NSF) Problem Statement and Use Cases", July 2017

[RFC5521] P. Mohapatra, E. Rosen, "The BGP Encapsulation Subsequent Address Family Identifier (SAFI) and the BGP Tunnel Encapsulation Attribute", April 2009.

[BGP-SDWAN-EXT]L. Dunbar, "BGP Extension for SDWAN Overlay Networks", draft-dunbar-idr-bgp-sdwan-overlay-ext-00, Oct 2018.

[BGP-SDWAN-Usage] L. Dunbar, et al, "Framework of Using BGP for SDWAN Overlay Networks", draft-dunbar-idr-sdwan-framework-00, work-in-progress, Feb 2019.

[Tunnel-Encap]E. Rosen, et al, "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-10, July 2018.

[VPN-over-Internet] E. Rosen, "Provide Secure Layer L3VPNs over Public Infrastructure", draft-rosen-bess-secure-l3vpn-00, work-in-progress, July 2018

[DMVPN] Dynamic Multi-point VPN: https://www.cisco.com/c/en/us/products/security/dynamic-multipoint-vpn-dmvpn/index.html

[DSVPN] Dynamic Smart VPN: http://forum.huawei.com/enterprise/en/thread-390771-1-1.html

[ITU-T-X1036] ITU-T Recommendation X.1036, "Framework for creation, storage, distribution and enforcement of policies for network security", Nov 2007.

[Net2Cloud-Problem] L. Dunbar and A. Malis, "Seamless Interconnect Underlay to Cloud Overlay Problem Statement", draft-dm-net2cloud-problem-statement-02, June 2018

10. Acknowledgments

Acknowledgements to xxx for his review and contributions.

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

Authors' Addresses

Linda Dunbar
Huawei
Email: Linda.Dunbar@huawei.com

Andrew G. Malis
Huawei
Email: agmalis@gmail.com

Christian Jacquenet
Orange
Rennes, 35000
France
Email: Christian.jacquenet@orange.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: July 2019

L. Dunbar
A. Malis
Huawei
C. Jacquenet
Orange
M. Toy
Verizon

February 6, 2019

Seamless Interconnect Underlay to Cloud Overlay Problem Statement
draft-dm-net2cloud-problem-statement-07

Abstract

This document describes the problems that enterprises face today when connecting their branch offices to dynamic workloads in third party data centers (a.k.a. Cloud DCs).

It examines some of the approaches interconnecting cloud DCs with enterprises' on-premises DCs & branch offices. This document also describes some of the (network) problems that many enterprises face when they have workloads & applications & data split among hybrid data centers, especially for those enterprises with multiple sites that are already interconnected by VPNs (e.g., MPLS L2VPN/L3VPN).

Current operational problems are examined to determine whether there is a need to improve existing protocols or whether a new protocol is necessary to solve them.

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

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

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

This Internet-Draft will expire on August 6, 2019.

Table of Contents

1. Introduction

The ever-increasing use of cloud applications for communication services change the way corporate business works and shares information. Such cloud applications use resources hosted in third party DCs that also host services for other customers.

With the advent of widely available third party cloud DCs in diverse geographic locations and the advancement of tools for monitoring and predicting application behaviors, it is technically feasible for enterprises to instantiate applications and workloads in locations that are geographically closest to their end-users. Such proximity improves end-to-end latency and overall user experience. Conversely, an enterprise can easily shutdown applications and workloads whenever end-users are in motion (thereby modifying the networking connection of subsequently relocated applications and workloads). In addition, an enterprise may wish to take advantage of more and more business applications offered by third party private cloud DCs.

Most of those enterprise branch offices & on-premises data centers are already connected via VPNs, such as MPLS-based L2VPNs and L3VPNs. Then connecting to the cloud-hosted resources may not be straightforward if the provider of the VPN service does not have direct connections to the corresponding cloud DCs. Under those circumstances, the enterprise can upgrade the CPEs deployed in its various premises to utilize SD-WAN techniques to reach cloud resources (without any assistance from the VPN service provider), or wait for their VPN service provider to make new agreements with data

center providers to connect to the cloud resources. Either way has additional infrastructure and operational costs.

In addition, it is an uptrend with more enterprises instantiating their apps & workloads in different cloud DCs to maximize the benefits of geographical proximity, elasticity and special features offered by different cloud DCs.

2. Definition of terms

Cloud DC: Third party Data Centers that usually host applications and workload owned by different organizations or tenants.

Controller: Used interchangeably with SD-WAN controller to manage SD-WAN overlay path creation/deletion and monitoring the path conditions between two or more sites.

DSVPN: Dynamic Smart Virtual Private Network. DSVPN is a secure network that exchanges data between sites without needing to pass traffic through an organization's headquarter virtual private network (VPN) server or router.

Heterogeneous Cloud: applications & workloads split among Cloud DCs owned & managed by different operators.

Hybrid Clouds: Hybrid Clouds (usually plural) refer to enterprises using their own premises DCs in addition to Cloud services provided by multiple cloud operators. For example, an enterprise not only have applications running in their own DCs, but also have applications hosted in multiple third party cloud DCs ((AWS, Azure, Google, Salesforces, SAP, etc). . ONUG also has a notion of heterogeneous cloud, refers to enterprises does not have its own DC, only uses services by 3rd party cloud operators.

SD-WAN: Software Defined Wide Area Network. In this document, "SD-WAN" refers to the solutions specified by ONUG (Open Network User Group), https://www.onug.net/software-

defined-wide-area-network-sd-wan/, which is about pooling WAN bandwidth from multiple underlay networks to get better WAN bandwidth management, visibility & control. When the underlay networks are private networks, traffic can traverse without additional encryption; when the underlay networks are public, such as Internet, some traffic needs to be encrypted when traversing through (depending on user provided policies).

VPC: Virtual Private Cloud. A service offered by Cloud DC operators to allocate logically-isolated cloud resources, including compute, networking and storage.

3. Current Practices in Interconnecting Enterprise Sites with Cloud DCs

3.1. Interconnect to Cloud DCs

Most Cloud operators offer some type of network gateway through which an enterprise can reach their workloads hosted in the Cloud DCs. For example, AWS (Amazon Web Services) offers the following options to reach workloads in AWS Cloud DCs:

- Internet gateway for any external entities to reach the workloads hosted in AWS Cloud DC via the Internet.
- Virtual gateway (vGW) where IPsec tunnels [RFC6071] are established between an enterprise's own gateway and AWS vGW, so that the communications between those gateways can be secured from the underlay (which might be the public Internet).
- Direct Connect, which allows enterprises to purchase direct connect from network service providers to get a private leased line interconnecting the enterprises gateway(s) and the AWS Direct Connect routers. Via Direct Connect, an AWS Transit Gateway can be used to interconnect multiple VPCs in different Availability Zones.

CPEs at one Enterprise branch office are connected to the Internet to reach AWS's vGW via IPsec tunnels. Other ports of such CPEs are connected to AWS DirectConnect via a private network (without any encryption).

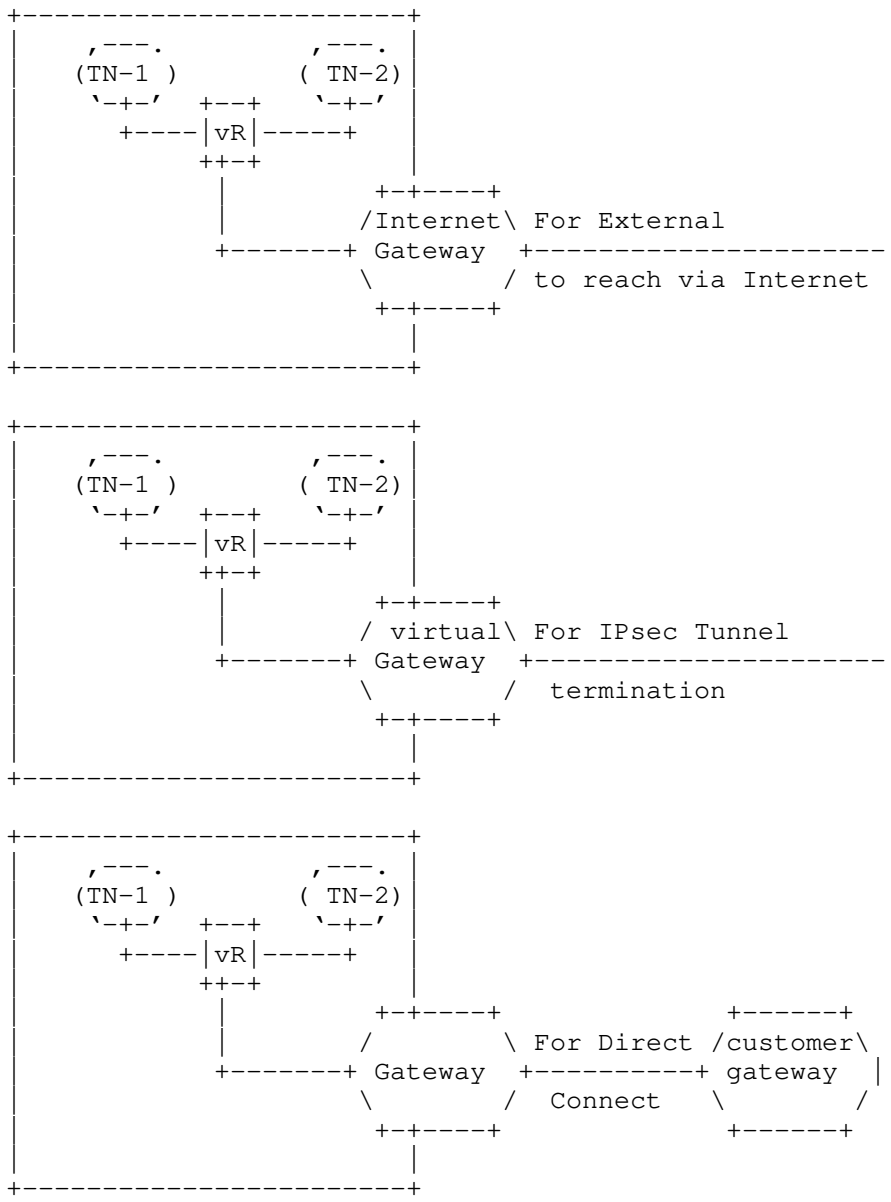


Figure 1: Examples of Cloud DC connections.

3.2. Interconnect to Hybrid Cloud DCs

According to Gartner, by 2020 "hybrid will be the most common usage of the cloud" as more enterprises see the benefits of integrating public and private cloud infrastructures. However, enabling the growth of hybrid cloud deployments in the enterprise requires fast and safe interconnection between public and private cloud services. For an enterprise to connect to applications & workloads hosted in multiple Cloud DCs, the enterprise can use IPsec tunnels established over the Internet or a (virtualized) leased line service to connect its on-premises gateways to each of the Cloud DC's gateways, virtual routers instantiated in the Cloud DCs, or any other suitable design (including a combination thereof).

Some enterprises prefer to instantiate their own virtual CPEs/routers inside the Cloud DC to connect the workloads within the Cloud DC. Then an overlay path is established between customer gateways to the virtual CPEs/routers for reaching the workloads inside the cloud DC.

3.3. Connecting workloads among hybrid Cloud DCs

There are multiple approaches to interconnect workloads among different Cloud DCs:

- Utilize Cloud DC provided transit gateways, which usually does not work if Cloud DCs are owned and managed by different Cloud providers.
- Hairpin all the traffic through the customer gateway, which creates additional transmission delay & incurs cost when exiting Cloud DCs, or
- Establish direct tunnels among different VPCs (Virtual Private Clouds) via client's own virtual routers instantiated within Cloud DCs. DMVPN (Dynamic Multipoint Virtual Private Network) or DSVPN (Dynamic Smart VPN) techniques can be used to establish direct Multi-point-to-Point or multi-point-to multi-point tunnels among those client's own virtual routers.

DMVPN & DSVPN use NHRP (Next Hop Resolution Protocol) [RFC2735] so that spoke nodes can register their IP addresses & WAN ports with the hub node. The IETF ION (Internetworking over NBMA (non-broadcast

multiple access) WG standardized NHRP for connection-oriented NBMA network (such as ATM) network address resolution more than two decades ago.

There are many differences between virtual routers in Public Cloud DCs and the nodes in an NBMA network. NHRP & DSVPN are not cannot be used for registering virtual routers in Cloud DCs unless an extension of such protocols is developed for that purpose. Other protocols such as BGP can be used, as described in [BGP-SDWAN].

4. Desired Properties for Networks that interconnect Hybrid Clouds

The networks that interconnect hybrid cloud DCs must address the following requirements:

- High availability at any time, whatever the duration of the connection to the cloud DC.
  Many enterprises include cloud infrastructures in their disaster recovery strategy, e.g., by enforcing periodic backup policies within the cloud, or by running backup applications in the Cloud, etc. Therefore, the connection to the cloud DCs may not be permanent, but rather needs to be on-demand.

- Global reachability from different geographical zones, thereby facilitating the proximity of applications as a function of the end users' location, to improve latency.
- Elasticity and mobility, to instantiate additional applications at Cloud DCs when end-users' usages increase and shut down applications at locations when there are fewer end-users.
  Some enterprises have front-end web portals running in cloud DCs and database servers in their on-premises DCs. Those Front-end web portals need to be reachable from the public Internet. The backend connection to the sensitive data in database servers hosted in the on-premises DCs might need secure connections.

- Scalable security management. IPsec is commonly used to interconnect cloud gateways with CPEs deployed in the enterprise premises. For enterprises with a large number or branch offices, managing the IPsec's Security Associations among many nodes can be very difficult.

5. Problems with MPLS-based VPNs extending to Hybrid Cloud DCs

Traditional MPLS-based VPNs have been widely deployed as an effective way to support businesses and organizations that require network performance and reliability. MPLS shifted the burden of managing a VPN service from enterprises to service providers. The CPEs attached to MPLS VPNs are also simpler and less expensive, since they do not need to manage routes to remote sites; they simply pass all outbound traffic to the MPLS VPN PEs to which the CPEs are attached (albeit multi-homing scenarios require more processing logic on CPEs). MPLS has addressed the problems of scale, availability, and fast recovery from network faults, and incorporated traffic-engineering capabilities.

However, traditional MPLS-based VPN solutions are sub-optimized for connecting end-users to dynamic workloads/applications in cloud DCs because:

- The Provider Edge (PE) nodes of the enterprise's VPNs might not have direct connections to third party cloud DCs that are used for hosting workloads with the goal of providing an easy access to enterprises' end-users.

- It usually takes some time to deploy provider edge (PE) routers at new locations. When enterprise's workloads are changed from one cloud DC to another (i.e., removed from one DC and re-instantiated to another location when demand changes), the enterprise branch offices need to be connected to the new cloud DC, but the network service provider might not have PEs located at the new location.

  One of the main drivers for moving workloads into the cloud is the widely available cloud DCs at geographically diverse locations, where apps can be instantiated so that they can be as close to their end-users as possible. When the user base changes, the applications may be migrated to a new cloud DC location closest to the new user base.

- Most of the cloud DCs do not expose their internal networks, so the MPLS-based VPNs can only reach Cloud DC's Gateways, not to the workloads hosted inside.

- Many cloud DCs use an overlay to connect their gateways to the workloads located inside the DC. There has not been any standard to address the interworking between the Cloud Overlay and the enterprise' existing underlay networks.

Another roadblock is the lack of a standard way to express and enforce consistent security policies for workloads that not only use virtual addresses, but in which are also very likely hosted in different locations within the Cloud DC [RFC8192]. The current VPN path computation and bandwidth allocation schemes may not be flexible enough to address the need for enterprises to rapidly connect to dynamically instantiated (or removed) workloads and applications regardless of their location/nature (i.e., third party cloud DCs).

6. Problem with using IPsec tunnels to Cloud DCs

As described in the previous section, many Cloud operators expose their gateways for external entities (which can be enterprises themselves) to directly establish IPsec tunnels. Enterprises can also instantiate virtual routers within Cloud DCs to connect to their on-premises devices via IPsec tunnels. If there is only one enterprise location that needs to reach the Cloud DC, an IPsec tunnel is a very convenient solution.

However, many medium-to-large enterprises usually have multiple sites and multiple data centers. For workloads and apps hosted in cloud DCs, multiple sites need to communicate securely with those cloud workloads and apps. This section documents some of the issues associated with using IPsec tunnels to connect enterprise premises with cloud gateways.

6.1. Complexity of multi-point any-to-any interconnection

The dynamic workload instantiated in cloud DC needs to communicate with multiple branch offices and on-premises data centers. Most enterprises need multi-point interconnection among multiple locations, which can be provided by means of MPLS L2/L3 VPNs.

Using IPsec overlay paths to connect all branches & on-premises data centers to cloud DCs requires CPEs to manage routing among Cloud DCs gateways and the CPEs located at other branch locations, which can dramatically increase the complexity of the design, possibly at the cost of jeopardizing the CPE performance.

The complexity of requiring CPEs to maintain routing among other CPEs is one of the reasons why enterprises migrated from Frame Relay based services to MPLS-based VPN services.

MPLS-based VPNs have their PEs directly connected to the CPEs. Therefore, CPEs only need to forward all traffic to the directly attached PEs, which are therefore responsible for enforcing the routing policy within the corresponding VPNs. Even for multi-homed CPEs, the CPEs only need to forward traffic among the directly connected PEs. However, when using IPsec tunnels between CPEs and Cloud DCs, the CPEs need to compute, select, establish and maintain routes for traffic to be forwarded to Cloud DCs, to remote CPEs via VPN, or directly.

6.2. Poor performance over long distance

When enterprise CPEs or gateways are far away from cloud DC gateways or across country/continent boundaries, performance of IPsec tunnels over the public Internet can be problematic and unpredictable. Even though there are many monitoring tools available to measure delay and various performance characteristics of the network, the measurement for paths over the Internet is passive and past measurements may not represent future performance.

Many cloud providers can replicate workloads in different available zones. An App instantiated in a cloud DC closest to clients may have to cooperate with another App (or its mirror image) in another region or database server(s) in the on-premises DC. This kind of coordination requires predicable networking behavior/performance among those locations.

6.3. Scaling Issues with IPsec Tunnels

IPsec can achieve secure overlay connections between two locations over any underlay network, e.g., between CPEs and Cloud DC Gateways.

If there is only one enterprise location connected to the cloud gateway, a small number of IPsec tunnels can be configured on-demand

between the on-premises DC and the Cloud DC, which is an easy and flexible solution.

However, for multiple enterprise locations to reach workloads hosted in cloud DCs, the cloud DC gateway needs to maintain multiple IPsec tunnels to all those locations (e.g., as a hub & spoke topology). For a company with hundreds or thousands of locations, there could be hundreds (or even thousands) of IPsec tunnels terminating at the cloud DC gateway, which is not only very expensive (because Cloud Operators usually charge their customers based on connections), but can be very processing intensive for the gateway. Many cloud operators only allow a limited number of (IPsec) tunnels & bandwidth to each customer. Alternatively, you could use a solution like group encryption where a single IPsec SA is necessary at the GW but the drawback here is key distribution and maintenance of a key server, etc.

7. Problems of Using SD-WAN to connect to Cloud DCs

SD-WAN can establish parallel paths over multiple underlay networks between two locations on-demand, for example, to support the connections established between two CPEs interconnected by a traditional MPLS VPN ([RFC4364] or [RFC4664]) or by IPsec [RFC6071] tunnels.

SD-WAN lets enterprises augment their current VPN network with cost-effective, readily available Broadband Internet connectivity, enabling some traffic offloading to paths over the Internet according to differentiated, possibly application-based traffic forwarding policies, or when the MPLS VPN connection between the two locations is congested, or otherwise undesirable or unavailable.

7.1. SD-WAN among branch offices vs. interconnect to Cloud DCs

SD-WAN interconnection of branch offices is not as simple as it appears. For an enterprise with multiple sites, using SD-WAN overlay paths among sites requires each CPE to manage all the addresses that local hosts have the potential to reach, i.e., map internal VPN addresses to appropriate SD-WAN paths. This is similar to the complexity of Frame Relay based VPNs, where each CPE needed to maintain mesh routing for all destinations if they were to avoid an extra hop through a hub router. Even though SD-WAN CPEs can get assistance from a central controller (instead of running a routing protocol) to resolve the mapping between destinations and SD-WAN paths, SD-WAN CPEs are still responsible for routing table

maintenance as remote destinations change their attachments, e.g., the dynamic workload in other DCs are de-commissioned or added.

Even though originally envisioned for interconnecting branch offices, SD-WAN offers a very attractive way for enterprises to connect to Cloud DCs.

The SD-WAN for interconnecting branch offices and the SD-WAN for interconnecting to Cloud DCs have some differences:

- SD-WAN for interconnecting branch offices usually have two end-points (e.g., CPEs) controlled by one entity (e.g., a controller or management system operated by the enterprise).
- SD-WAN for Cloud DC interconnects may consider CPEs owned or managed by the enterprise, while remote end-points are being managed or controlled by Cloud DCs (For the ease of description, let’s call such CPEs asymmetrically-managed CPEs).

- Cloud DCs may have different entry points (or devices) with one entry point that terminates a private direct connection (based upon a leased line for example) and other entry points being devices terminating the IPsec tunnels, as shown in Figure 2.

Therefore, the SD-WAN design becomes asymmetric.

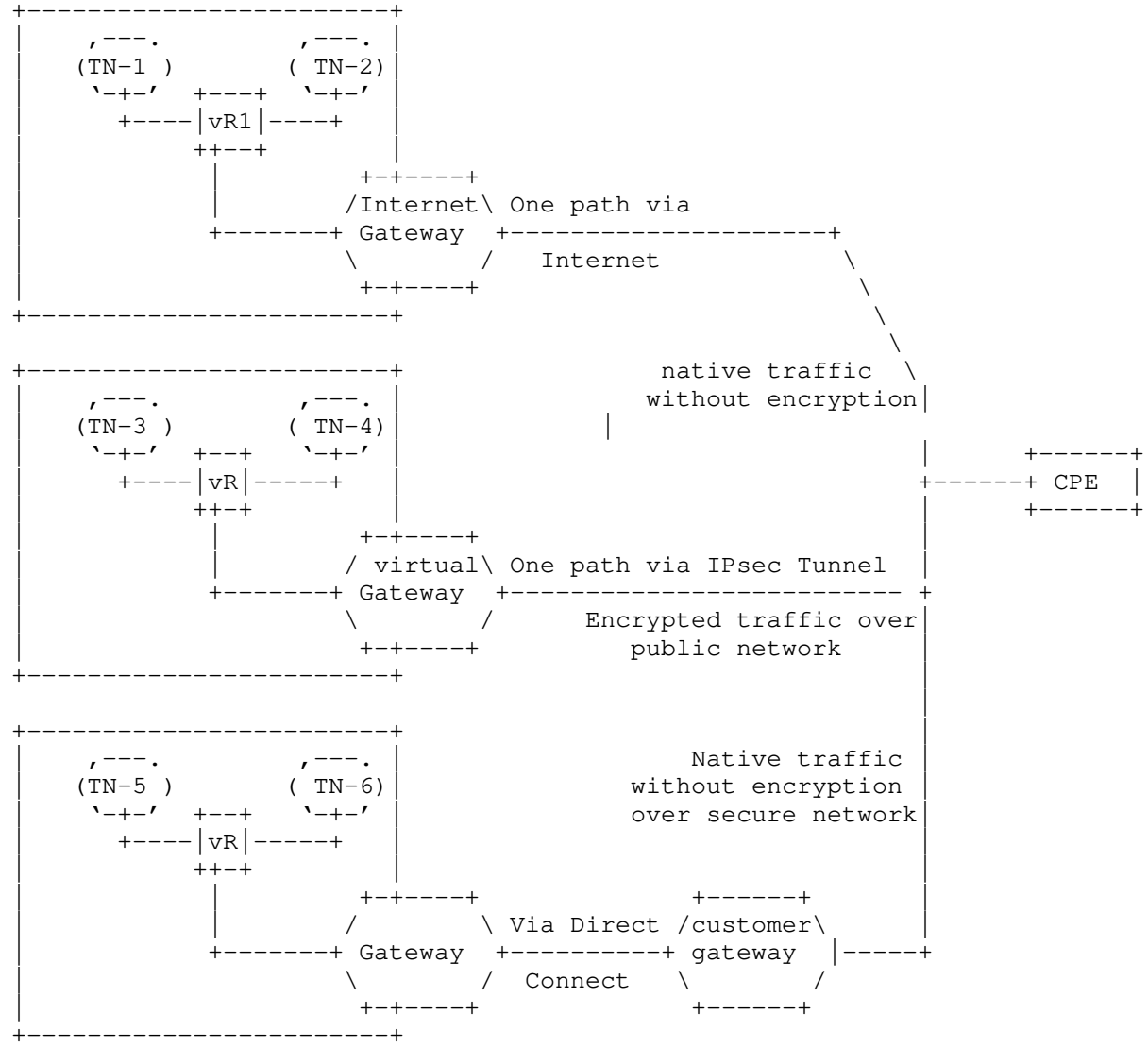


Figure 2: Different Underlays to Reach Cloud DC

8. End-to-End Security Concerns for Data Flows

When IPsec tunnels established from enterprise on-premises CPEs are terminated at the Cloud DC gateway where the workloads or applications are hosted, some enterprises have concerns regarding traffic to/from their workload being exposed to others behind the data center gateway (e.g., exposed to other organizations that have workloads in the same data center).
To ensure that traffic to/from workloads is not exposed to unwanted entities, IPsec tunnels may go all the way to the workload (servers, or VMs) within the DC.

9. Requirements for Dynamic Cloud Data Center VPNs

In order to address the aforementioned issues, any solution for enterprise VPNs that includes connectivity to dynamic workloads or applications in cloud data centers should satisfy a set of requirements:

- The solution should allow enterprises to take advantage of the current state-of-the-art in VPN technology, in both traditional MPLS-based VPNs and IPsec-based VPNs (or any combination thereof) that run over the public Internet.
- The solution should not require an enterprise to upgrade all their existing CPEs.
- The solution should support scalable IPsec key management among all nodes involved in DC interconnect schemes.
- The solution needs to support easy and fast, on-the-fly, VPN connections to dynamic workloads and applications in third party data centers, and easily allow these workloads to migrate both within a data center and between data centers.
- Allow VPNs to provide bandwidth and other performance guarantees.
- Be a cost-effective solution for enterprises to incorporate dynamic cloud-based applications and workloads into their existing VPN environment.

10. Security Considerations

The draft discusses security requirements as a part of the problem space, particularly in sections 4, 5, and 8.

Solution drafts resulting from this work will address security concerns inherent to the solution(s), including both protocol aspects and the importance (for example) of securing workloads in cloud DCs and the use of secure interconnection mechanisms.

IANA Considerations

This document requires no IANA actions. RFC Editor: Please remove this section before publication.

11. References

11.1. Normative References

11.2. Informative References

[RFC2735] B. Fox, et al "NHRP Support for Virtual Private networks". Dec. 1999.

[RFC8192] S. Hares, et al "Interface to Network Security Functions (I2NSF) Problem Statement and Use Cases", July 2017

[ITU-T-X1036] ITU-T Recommendation X.1036, "Framework for creation, storage, distribution and enforcement of policies for network security", Nov 2007.

[RFC6071] S. Frankel and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", Feb 2011.

[RFC4364] E. Rosen and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", Feb 2006

[RFC4664] L. Andersson and E. Rosen, "Framework for Layer 2 Virtual Private Networks (L2VPNs)", Sept 2006.

[BGP-SDWAN] L. Dunbar, et al. "BGP Extension for SDWAN Overlay Networks", draft-dunbar-idr-bgp-sdwan-overlay-ext-03, work-in-progress, Nov 2018.

12. Acknowledgments

Many thanks to Ignas Bagdonas, Michael Huang, Liu Yuan Jiao, Katherine Zhao, and Jim Guichard for the discussion and contributions.

Authors' Addresses

Linda Dunbar
Huawei
Email: Linda.Dunbar@huawei.com

Andrew G. Malis
Huawei
Email: agmalis@gmail.com

Christian Jacquenet
Orange
Rennes, 35000
France
Email: Christian.jacquenet@orange.com

Mehmet Toy
Verizon
One Verizon Way
Basking Ridge, NJ 07920
Email: mehmet.toy@verizon.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 18, 2019

Y. Gu
S. Zhuang
Z. Li
Huawei

July 17, 2018

Network Monitoring Protocol (NMP)
draft-gu-network-monitoring-protocol-00

Abstract

To evolve towards automated network OAM (Operations, administration and management), the monitoring of control plane protocols is a fundamental necessity. In this document, a network monitoring protocol (NMP) is proposed to provision the running status information of control plane protocols, e.g., IGP (Interior Gateway Protocol) and other protocols. By collecting the protocol monitoring data and reporting it to the NMP monitoring server in real-time, NMP can facilitate network troubleshooting. In this document, NMP for IGP troubleshooting are illustrated to showcase the necessity of NMP. IS-IS is used as the demonstration protocol, and the case of OSPF (Open Shortest Path First) and other control protocols will be elaborated in the future versions. The operations of NMP are described, and the NMP message types and message formats are defined in the document.

Requirements Language

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

Status of This Memo

This Internet-Draft will expire on January 3, 2019.

Table of Contents

1. Introduction

1.1. Motivation

The requirement for better network OAM approaches has been greatly driven by the network evolvement. The concept of network Telemetry has been proposed to meet the current and future OAM demands w.r.t., massive and real-time data storage, collection, process, exportion, and analysis, and an architectural framework of existing Telemetry approaches is introduced in [I-D.song-ntf]. Network Telemetry provides visibility to the network health conditions, and is beneficial for faster network troubleshooting, network OpEx (operating expenditure) reduction, and network optimization. Telemetry can be applied to the data plane, control plane and management plane. There have been various methods proposed for each plane:

- Management plane: For example, SNMP (Simple Network Management Protocol) [RFC1157], NETCONF (Network Configuration Protocol) [RFC6241] and gNMI (gRPC Network Management Interface) [I-D.openconfig-rtgwg-gnmi-spec] are three typical widely adopted management plane Telemetry approaches. Various YANG modules are defined for network operational state retrieval and configuration management. Subscription to specific YANG datastore can be realized in combination with gRPC/NETCONF.

- Data plane: For example, In-situ OAM (iOAM) [I-D.brockners-inband-oam-requirements] embeds an instruction header to the user data packets, and collects the requested data and adds it to the use packet at each network node along the forwarding path. Applications such as path verification, SLA (service-level agreement) assurance can be enabled with iOAM.

- Control Plane: BGP monitoring protocol (BMP) [RFC7854] is proposed to monitor BGP sessions and intended to provide a convenient interface for obtaining BGP route views. Date collected using BMP can be further analyzed with big data platforms for network health condition visualization, diagnose and prediction applications.

The general idea of most Telemetry approaches is to collect various information from devices and export to the centralized server for further analysis, and thus providing more network insight. It should not be surprising that any future and even current Telemetry applications may require the fusion of data acquired from more than one single approach/one single plane. For example, for network troubleshooting purposes, it requires the collection of comprehensive information from devices, such system ID/router ID, interface status, PDUs (protocol data units), device/protocol statistics and so on.

Information such as system ID/router ID can be reported by management plane Telemetry approaches, while the protocol related data (especially PDUs) are more fit to be monitored using the control plane Telemetry. With rich information collected in real time at the centralized server, network issues can be localized faster and more accurately, and the root cause analysis can be also provided.

The conventional troubleshooting logic is to log in a faulty router, physically or through Telnet, and by using CLI to display related information/logs for fault source localization and further analysis. There are several concerns with the conventional troubleshooting methods:

1. It requires rich OAM experience for the OAM operator to know what information to check on the device, and the operation is complex;

2. In a multi-vendor network, it requires the understanding and familiarity of vendor specific operations and configurations;

3. Locating the fault source device could be non-trivial work, and is often realized through network-wide device-by-device check, which is both time-consuming and labor-consuming; and finally,

4. The acquisition of troubleshooting data can be difficult under some cases, e.g., when auto recovery is used.

This document proposes the Network Monitoring Protocols (NMP) to monitor the running status of control protocols, e.g., PDUs, protocol statistics and peer status, which have not been systematically covered by any other Telemetry approach, to facilitate network troubleshooting.

1.2. Overview

Like BMP, an NMP session is established between each monitored router (NMP client) and the NMP monitoring station (NMP server) through TCP connection. Information are collected directly from each monitored router and reported to the NMP server. The NMP message can be both periodic and event-triggered, depending on the message type.

IS-IS [RFC1195], as one of the most commonly adopted network layer protocols, builds the fundamental network connectivity of an autonomous system (AS). The disfunction of IS-IS, e.g., IS-IS neighbor down, route flapping, MTU mismatch, and so on, could lead to network-wide instability and service interruption. Thus, it is critical to keep track of the health condition of IS-IS, and the availability of information, related to IS-IS running status, is the fundamental requirement. In this document, typical network issues

are illustrated as the use cases of NMP for IS-IS to showcase the necessity of NMP. Then the operations and the message formats of NMP for IS-IS are defined. In this document IS-IS is used as the illustration protocol, and the case of OSPF and other control protocols will be included in the future version.

2. Terminology

IGP: Interior Gateway Protocol

IS-IS: Intermediate System to Intermediate System

NMP: Network Monitoring Protocol

IMP: Network Monitoring Protocol for IGP

BMP: BGP monitoring protocol

IIH: IS-IS Hello Packet

LSP: Link State Packet

CSNP: Complete Sequence Number Packet

NSNP: Partial Sequence Number Packet

3. Use Cases

We have identified several typical network issues due to IS-IS disfunction that are currently difficult to detect or localize. The usage of NMP is not limited to the solve the following listed issues.

3.1. IS-IS Adjacency Issues

IS-IS adjacency issues are identified as top network issues and may take hours to localize. The adjacency issues can be classified into two situations:

1. An existing established adjacency goes down;

2. An adjacency fails to be established.

In Case 1, the adjacency down can be caused by factors such as circuit down, hold timer expiration, device memory low, user configuration change, and so on. Case 2 can be caused by mismatch link MTU, mismatch authentication, mismatch area ID, system ID conflict, and so on. Typically, such adjacency failure events are logged/recorded in the device, but currently there is no real-time

report/alarm of such issue. The conventional troubleshooting process for adjacency issue is to find the faulty devices and then log in to check the logs or the IIH statistics for further analysis.

Using NMP, the IS-IS adjacency status: up, down and initial, is reported to the NMP server in real time, together with the possible recorded reasons. Then the NMP server can solve such issue in about minutes. For example, for an adjacency set up failure due to different authentications, the NMP server can recognize the difference by comparing the IIHs collected from both devices.

3.2. Forwarding Path Disconnection

The PING test can be used to test the reachability of a destination address. However, there are cases of disconnection that cannot be detected by PING. The PING result may return a connected path, but the forwarding of certain-sized packets always fails. This could be caused by factors, such as mismatched MTU values for devices along the path. It can be quite common since vendors have different understanding and configurations of MTU. There are methods proposed to discover the path MTU. For example, router's link MTU is conveyed in the MPLS LDP/RSVP-TE path set up signaling, and the path MTU is decided at the ingress or egress node[RFC3988] [RFC3209]. For IPv4 packets, by setting the DF flag bit of the outgoing packet, any device along the path with smaller MTU will drop the packet, and send back an ICMP Fragmentation Needed message containing its MTU, allowing the source to reduce the MTU. The process is repeated until the MTU is small enough to traverse the entire path without fragmentation[RFC1191]. Apparently, such method is too time-consuming.

Using NMP, each device can report its link MTU to the monitoring station directly. The mismatch can be recognized at the NMP server in seconds.

3.3. IS-IS LSP Synchronization Failure

It happens that two IS-IS neighbors fail to learn the LSPs sent from each other in the following two cases: in Case 1, the LSP fails to be received, and in Case 2, the LSP is received but the LSP information shown in the receiver's LSDB is not the same as the one sent from the transmitter (e.g., one or more prefixes missing, the LSP sequence number modified). Case 1 can be caused by link failure, similar to the adjacency down issue. In Case 2, the received LSP can be processed incorrectly due to hardware/software bugs. In fact, the LSDB synchronization issue is usually hard to localize once happens.

Using NMP, the NMP server can detect the failure by comparing the sent/received LSP statistics from the two neighbors. In the case that the received LSPs are improperly processed within the device, the NMP monitoring station can recognize the LSP synchronization failure by comparing the LSPs sent out from the two neighbors.

4. NMP Message Format

4.1. Protocol Selection Options

Regarding the NMP/IMP monitoring data exportion, BMP has been a good option. First of all, BMP serves similar purposes of NMP that reports routes, route statistics and peer status. In addition, BMP has already been implemented in major vendor devices and utilized by operator. Thus, we propose the following two options for the NMP data exportion.

- Option 1: Extending BMP with new message types to carry NMP/IMP data: Reusing the BMP framework saves certain implementation cost for both vendors and operators. Besides, the monitoring data exportion of different routing protocols (e.g., BGP, ISIS, OSPF) can be unified.

- Option 2: Defining NMP to carry NMP/IMP data: This option defines a brand new framework to carry protocol monitoring data, similar to BMP. Defining a new framework provides advantages such as more flexible and customized features for IGP and other protocols, since the monitoring data and troubleshooting of different protocols vary from one another.

In this document, we take Option 2 as the illustration example to define the NMP message types and message formats. The decision of the protocol selection may be further clarified in futures versions.

4.2. Message Types

The variety of IS-IS troubleshooting use cases requires a systematic information report of NMP, so that the NMP server or any third party analyzer could efficiently utilize the reported messages to localize and recover various network issues. We define NMP messages for IS-IS uses the following types:

- Initiation Message: A message used for the monitored device to inform the NMP monitoring station of its capabilities, vendor, software version and so on. For example, the link MTU can be included within the message. The initiation message is sent once the TCP connection between the monitoring station and monitored

router is set up. During the monitoring session, any change of the initiation message could trigger an Initiation Message update.

- Adjacency Status Change Notification Message: A message used to inform the monitoring station of the adjacency status change of the monitored device, i.e., from up to down, from down/initiation to up, with possible alarms/logs recorded in the device. This message notifies the NMP server of the ongoing IS-IS adjacency change event and possible reasons. If no reason is provided or the provided reason is not specific enough, the NMP server can further analyze the IS-IS PDU or the IS-IS statistics.

- Statistic Report Message: A message used to report the statistics of the ongoing IS-IS process at the monitored device. For example, abnormal LSP count of the monitored device can be a sign of route flapping. This message can be sent periodically or event triggered. If sent periodically, the frequency can be configured by the operator depending on the monitoring requirement. If it's event triggered, it could be triggered by a counter/timer exceeding the threshold.

- IS-IS PDU Monitoring Message: A message used to update the NMP server of any PDU sent from and received at the monitored device. For example, the IIHs collected from two neighbors can be used for analyzing the adjacency set up failure issue. The LSPs collected from two neighbors can be analyzed for the LSP synchronization issue.

- Termination Message: A message for the monitored router to inform the monitoring station of why it is closing the NMP session. This message is sent when the monitoring session is to be closed.

4.3. Message Format

4.3.1. Common Header

The common header is encapsulated in all NMP messages. It includes the Version, Message Length and Message Type fields.

- Version (1 byte): Indicates the NMP version and is set to '1' for all messages.

- Message Length (4 bytes): Length of the message in bytes (including headers, data, and encapsulated messages, if any).

- Message Type (1 byte): This indicates the type of the NMP message, which are listed as follows.

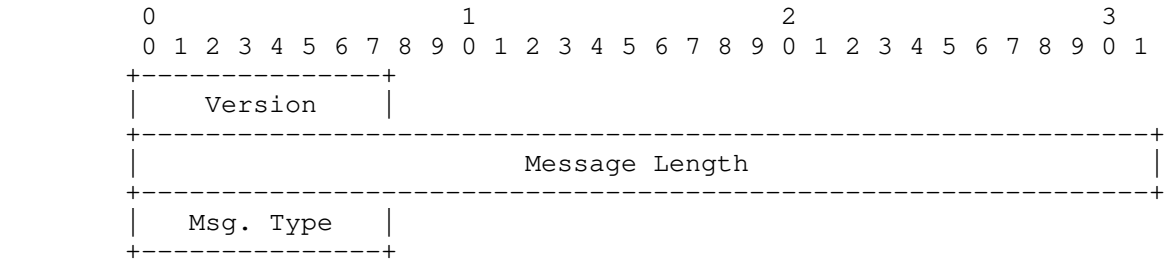
* Type = 0: Initiation

* Type = 1: Adjacency Status Change Notification

* Type = 2: Statistic Report

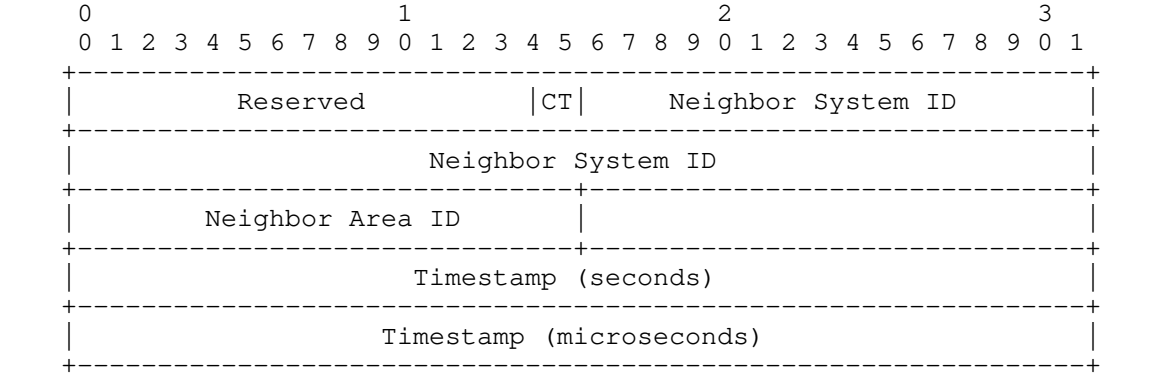
* Type = 3: IS-IS PDU Monitoring

* Type = 4: Termination Message



4.3.2. Per Adjacency Header

Except the Initiation and Termination Message, all the rest messages are per adjacency based. Thus, a per adjacency header is defined as follows.



o Adjacency Flag (2 bytes): The Circuit Type (2 bits) flag specifies if the router is an L1(01), L2(10), or L1/L2(11). If both bits are zeroes (00), the Per Adjacency Header SHALL be ignored. This configuration is used when the statistic is not per-adjacency based, e.g., when reporting the number of adjacencies.

- Neighbor System ID (6 bytes): identifies the system ID of the remote router.

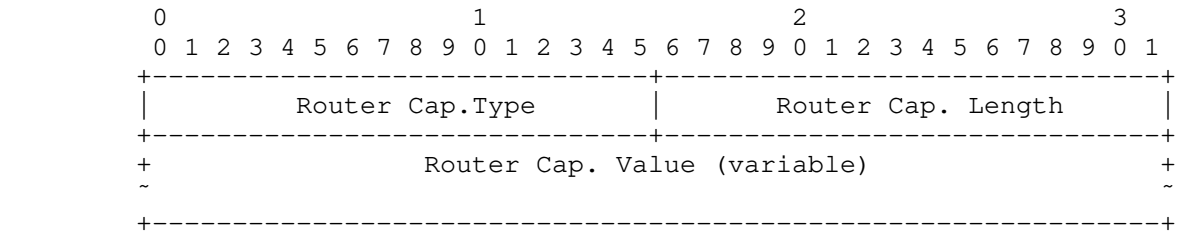
- Neighbor Area ID (2 bytes): identifies the area ID of the remote router.

- Timestamp (4 bytes): records the time when the message is sent/ received, expressed in seconds and microseconds since midnight (zero hour), January 1, 1970 (UTC).

4.3.3. Initiation Message

The Initiation Message indicates the monitored router's capabilities, vendor, software version and so on. It consists of the Common Header and the Router Capability TLV. The Common Header can be followed by multiple Router Capability TLVs.

The Router Capability TLV is defined as follows.



- Router Capability Type: provides the type of the router capability information. Currently defined types are:

  * Type = 0: sysDescr. The corresponding Router Capability Value field should contain an ASCII string whose value MUST be set to be equal to the value of the sysDescr MIB-II [RFC1213] object.

  * Type = 1: sysName. The corresponding Router Capability Value field should contain an ASCII string whose value MUST be set to be equal to the value of the sysName MIB-II [RFC1213] object.

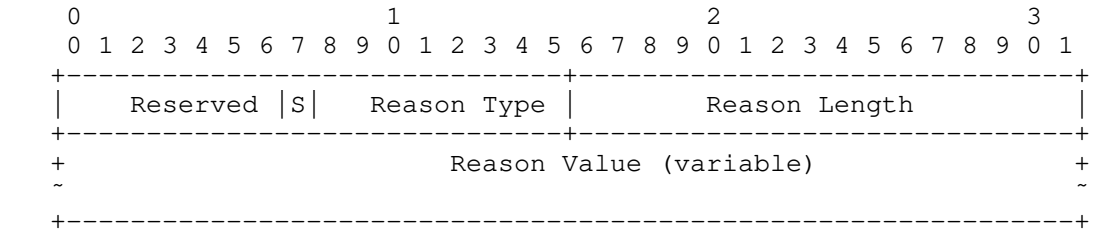
  * Type = 2: Local System ID. The corresponding Router Capability Value field SHALL indicate the router's System ID

  * Type = 3: Link MTU. The corresponding Router Capability Value field SHALL indicate the router's link MTU.

  * Type = 4: String. The corresponding Router Capability Value field contains a free-form UTF-8 string whose length is given by the Information Length field.

4.3.4. Adjacency Status Change Notification

The Adjacency Status Change Notification Message indicates an IS-IS adjacency status change: from up to down or from initiation/down to up. It consists of the Common Header, Per Adjacency Header and the Reason TLV. The Notification is triggered whenever the status changes. The Reason TLV is optional, and is defined as follows. More Reason types can be defined if necessary.



- Reason Flags (1 byte): The S flag (1 bit) indicates if the Adjacency status is from up to down (set to 0) or from down/ initial to up (set to 1). The rest bits of the Flag field are reserved. When the S flag is set to 1, the Reason Type SHALL be set to all zeroes (i.e., Type 0), the Reason Length fields SHALL be set to all zeroes, and the Reason Value field SHALL be set empty.

- Reason Type (1 byte): indicates the possible reason that caused the adjacency status change. Currently defined types are:

  * Type = 0: Adjacency Up. This type indicates the establishment of an adjacency. For this reason type, the S flag MUST be set to 1, indicating it's a adjacency-up event. There's no further reason to be provided. The reason Length field SHALL be set to all zeroes, and the Reason Value field SHALL be set empty.

  * Type = 1: Circuit Down. For this data type, the S flag MUST be set to 0, indicating it's a adjacency-down event. The length field is set to all zeroes, and the value field is set empty.

  * Type = 2: Memory Low. For this data type, the S flag MUST be set to 0, indicating it's a adjacency-down event. The length field is set to all zeroes, and the value field is set empty.

  * Type = 3: Hold timer expired. For this data type, the S flag MUST be set to 0, indicating it's a adjacency-down event. The length field is set to all zeroes, and the value field is set empty.

* Type = 4: String. For this data type, the S flag MUST be set to 0, indicating it's a adjacency-down event. The corresponding Reason Value field indicates the reason specified by the monitored router in a free-form UTF-8 string whose length is given by the Reason Length field.

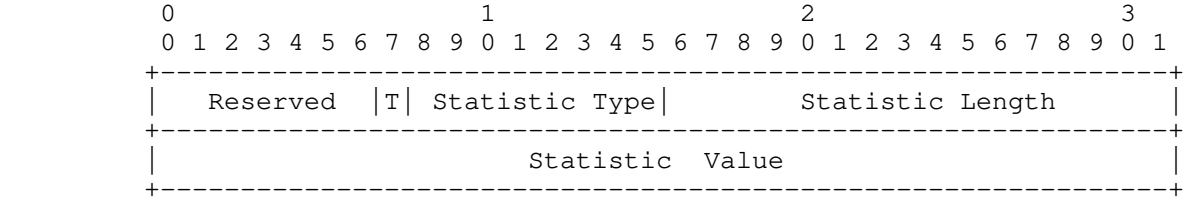
- Reason Length (2 bytes): indicates the length of the Reason Value field.

- Reason Value (variable): includes the possible reason why the Adjacency is down.

4.3.5. Statistic Report Message

The Statistic Report Message reports the statistics of the parameters that are of interest to the operator. The message consists of the NMP Common Header, the Per Adjacency Header and the Statistic TLV. The message include both per-adjacency based statistics and non per-adjacency based statistics. For example, the received/sent LSP counts are per-adjacency based statistics, and the local LSP change times count and the number of established adjacencies are non per-adjacency based statistics. For the non per-adjacency based statistics, the CT Flag (2 bits) in the Per Adjacency Header MUST be set to 00. Upon receiving any message with CT flag set to 00, the Per Adjacency Header SHALL be ignored (the total length of the Per Adjacency Header is 18 bytes as defined in Section 3.2.2, and the message reading/analysis SHALL resume from the Statistic TLV part.

The Statistic TLV is defined as follows.



- Statistic Flags (1 byte): provides information for the reported statistics.

  * T flag (1 bit): indicates if the statistic is for the received-from direction (set to 1) or sent-to direction the neighbor (set to 0)

- Statistic Type (1 byte): specifies the statistic type of the counter. Currently defined types are:

* Type = 0: IIH count. The T flag indicates if it's a sent or received Hello PDU. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 1: Incorrect IIH received count. For this type, the T flag MUST be set to 1. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 2: LSP count. The T flag indicates if it's a sent or received LSP. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 3: Incorrect LSP received count. For this type, the T flag MUST be set to 1. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 4: Retransmitted LSP count. For this type, the T flag MUST be set to 0. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 5: CSNP count. The T flag indicates if it's a sent or received CSNP. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 6: PSNP count. The T flag indicates if it's a sent or received PSNP. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 7: Number of established adjacencies. It's a non per-adjacency based statistic type, and thus for the monitoring station to recognize this type, the CT flag in the Per Adjacency Header MUST be set to 00.

* Type = 8: LSP change time count. It's a non per-adjacency based statistic type, and thus for the monitoring station to recognize this type, the CT flag in the Per Adjacency Header MUST be set to 00.

- Statistic Length (2 bytes): indicates the length of the Statistic Value field.

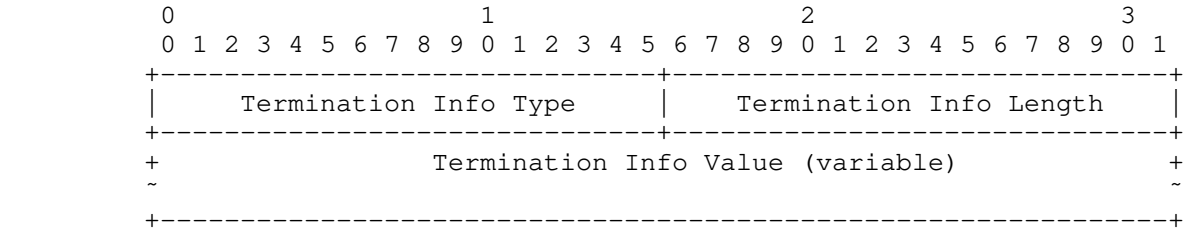
- Statistic Value (4 bytes): specifies the counter value, which is a non-negative integer.

4.3.6. IS-IS PDU Monitoring Message

The IS-IS PDU Monitoring Message is used to update the monitoring station of any PDU sent from and received at the monitored device per neighbor. Following the Common Header and the Per Adjacency Header is the IS-IS PDU. To tell whether it's a sent or received PDU, the monitoring station can analyze the source and destination addresses in the reported PDUs.

4.3.7. Termination Message

The Termination Message is sent when the NMP session is to be closed, and is used to indicate the termination reason to the monitoring station. The TCP session between the monitored router and the monitoring station SHALL be terminated upon receiving this message. It consists of the Common Header and the Termination Info TLVs, defined as follows.



- Termination Info Type (2 bytes): Provides the termination reason type. Currently defined types are:
  - Type = 0: Unknown. This reason type specifies that the NMP session is closed for an unknown or unspecified reason. For this data type, the length field is filled with all zeroes, and the value field is set empty.
  - Type = 1: Memory Low. This reason indicates that the monitored router lacks resources for the NMP session. For this data type, the length field is filled with all zeroes, and the value field is set empty.
  - Type = 2: Administratively Closed. This reason specifies that the session is closed due to administrative reasons. The corresponding Termination Info Value field may include more details about the reason expressed in a free-form UTF-8 string whose length is given by the Termination Info Length field.
  - Type = 3: String. The corresponding Termination Info Value field may include details about the reason expressed in a free-

form UTF-8 string whose length is given by the Termination Info Length field.

Termination Info Length (2 bytes): indicates the length of the Termination Info Reason Value field.

- Termination Info Value (variable): includes more detailed reason for the session termination.

5. IANA

TBD

6. Contributors

TBD

7. Acknowledgments

TBD

8. 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.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-17 (work in progress), July 2018.

[I-D.openconfig-rtgwg-gnmi-spec]
Shakir, R., Shaikh, A., Borman, P., Hines, M., Lebsack, C., and C. Morrow, "gRPC Network Management Interface (gNMI)", draft-openconfig-rtgwg-gnmi-spec-01 (work in progress), March 2018.

[I-D.song-ntf]
Song, H., Zhou, T., Li, Z., Fioccola, G., Li, Z., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Toward a Network Telemetry Framework", draft-song-ntf-02 (work in progress), July 2018.

[RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, DOI 10.17487/RFC1157, May 1990, <https://www.rfc-editor.org/info/rfc1157>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

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

[RFC1213] McCloghrie, K. and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, DOI 10.17487/RFC1213, March 1991, <https://www.rfc-editor.org/info/rfc1213>.

[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>.

[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, <https://www.rfc-editor.org/info/rfc3209>.

[RFC3988] Black, B. and K. Kompella, "Maximum Transmission Unit Signalling Extensions for the Label Distribution Protocol", RFC 3988, DOI 10.17487/RFC3988, January 2005, <https://www.rfc-editor.org/info/rfc3988>.

[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>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <https://www.rfc-editor.org/info/rfc7854>.

Authors' Addresses

Yunan Gu
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China

Email: guyunan@huawei.com

Shunwan Zhuang
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China

Email: zhuangshunwan@huawei.com

Zhenbin Li
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China

Email: lizhenbin@huawei.com

Routing Area Working Group J. Heitz
Internet-Draft K. Majumdar
Intended status: Standards Track Cisco
Expires: April 25, 2019 October 22, 2018

Automatic discovery and configuration of the network fabric in Massive Scale Data Centers
draft-heitz-idr-msdc-fabric-autoconf-00

Abstract

A switching fabric in a massive scale data center can comprise many 10,000's of switches and 100,000's of IP hosts. To connect and configure a network of such size needs automation to avoid errors. Zero Touch Provisioning (ZTP) protocols exist. These can configure IP devices that are reachable by the ZTP agents. A method to combine BGP, DHCPv6 and SRv6 with ZTP that can be used to configure an entire network of devices is described. It is designed to scale well, because each networked device is not required to know about more than its directly connected neighborhood.

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].

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 April 25, 2019.

Table of Contents

1. Introduction

[RFC7938] defines a massive scale data center as one that contains over one hundred thousand servers. It describes the advantages of using BGP [RFC4271] as a routing protocol in a Clos switching fabric that connects these servers. A fabric design that scales to one million servers is considered enough for the forseeable future and is the design goal of this document. Of course, the design should also work for smaller fabrics. A switch fabric to connect one million servers will consist of between 35000 and 130000 switches and 1.5 million to 8 million links, depending on how redundantly the servers are connected to the fabric and the level of oversubscription in the fabric. A switch that needs to store, send and operate on hundreds of routes is clearly cheaper than one that needs to store, send and operate on millions of links.

Such a network requires significant configuration on each switch and many cables to connect. This is an onerous task without automation.

2. Requirements

To configure a fabric network for massive scale data centers.

To detect every wiring error. For example, a spine switch that has a different number of links into one pod than into another pod in a Clos fabric.

One or multiple controllers exist to control a network. Multiple controllers are used for redundancy and to improve operation in partitioned networks.

Any devices with equivalent functionality should be interchangeable without requiring configuration changes. That means if a device breaks, it can be replaced by any other device of equivalent functionality without any changes to its configuration. Even if a replacement device already has configuration, it should still work in its new position.

A device may have configuration, but such configuration MUST NOT depend on the location of the device in the network. Therefore, no IP addresses should be pre-configured on any devices. No fabric tier should be needed.

For scalability, every device must not need to know how to reach every other device. Only a controller should be expected to know the entire topology.

If two such auto-discovering/auto-configuring networks are connected together, the function of discovery/configuration in one network must not disturb this function in the other network.

A device must accept configuration only from a well-defined set of controllers.

Separate cabling for a management network must not be required.

The network should function even if the controllers are disconnected. Link failures and restoration should be dealt with. Device failure should be dealt with. Device restoration should be dealt with as long as it does not require new configuration. A controller should only be needed to discover and configure new devices to the network.

The protocol does not need to be fast.

A controller must be able to reach any device if there is any way at all to reach it, even if that is multiple hops between spine switches or any other path that may be disallowed in a normal Clos network.

At the same time, normal traffic must remain restricted to allowable paths.

The routing protocol for normal traffic must be fast and efficient.

The network must scale to 1 million connected servers and 8 million links in the fabric.

3. Solution Overview

DHCPv6 [RFC3315] and ZTP are used to discover and configure devices reachable by the controller. As the controller configures devices, it configures them to be DHCP relay agents. This makes more devices reachable by the new DHCP relay agents, allowing the new devices to be configured. As this configuration process proceeds further away from the controller, it configures BGP to ensure reachabillity to all devices even if links were to fail. Reachability needs to be device to controller and controller to device. Every device does not need to be able to reach every other device during the discovery/ configuration process. Devices close to the controller will be used to forward packets to many more distant devices. These close devices should not store routes to reach all those more distant devices. A possible idea to reduce the routing table on close devices is to aggregate addresses of more distant devices. This is difficult and unreliable, because before discovery completes, the number of devices behind any given device is unknown. Also, if links fail, suddenly, a large number of devices could appear behind a different device, making the previous addressing structure non-aggregatable with the new topology. The chosen method to route traffic from controller to device is segment routing. The controller knows the topology. With that knowledge, it can build a segment list to reach any device.

In certain environments, it is required for devices to authenticate the network and for the network to authenticate devices. DHCPv6 provides a method to authenticate in both directions using shared keys. TCP-AO [RFC5925] can be used to authenticate BGP sessions. SZTP [I-D.ietf-netconf-zerotouch] provides for authentication during the ZTP process.

4. Solution Details

Each device needs a unique identifier. This may be printed on the device. For easy servicability, a device must have a single identifier, visible on the outside of the device and by the controller. This will be the DUID in the DHCPv6 Client Identifier Option.

In order to discover the topology, a controller needs to know every link in the topology. This means the device ID and interface ID or interface address at each end of every link. DHCPv6 can be used to obtain that information. For each link, one end of the link is the device that requests an address. The other end of the link is either the controller itself or a DHCP relay agent. The DHCP relay agent relays all client requests back to the controller.

Configuration proceeds in waves. Each controller may take part in configuring the network. The waves of configuration propagate away from each controller. In the first wave, a controller allocates a routable ipv6 address to each device directly connected to the controller. These devices comprise the first wave. The controller will then configure each of these devices using a ZTP protocol, such as [I-D.ietf-netconf-zerotouch]. The configuration for each device will include the following items:

- A routable Ipv6 address for each of its interfaces that have not already acquired one by DHCP.

- A routable Ipv6 address for the loopback interface.

- Configuration to act as a DHCPv6 relay agent for the next wave of devices.

- Configuration for a BGP session to each of its connected neighbors. That BGP session will initially be down, but will establish once the neighbors are connected and configured.

- Configuration for a BGP session to the controller.

The controller will allocate a different IP address for each interface for each device in the network. When the controller receives DHCP requests from DHCP relay agents, it will recognize the DHCP relay agent end of the link from the link-address field in the relay-forward message. The controller will note the DUID in the DHCP request to keep track of the device making the request. Because it already knows the DUID of the DHCP relay agent from its IP address, it can tie the two devices together by their DUID.

The controller must keep track of the DUID in every DHCP request, so that it can recognize different interfaces on the same device. This is needed to detect looped cables and to prevent the controller attempting to use ZTP to configure a single device through multiple links at the same time.

Two devices A and B may be connected by a link and be configured at the same time, each through a different link. At this time, the

controller does not yet know about the link A-B. In this case, neither A nor B will send a DHCP request across the link A-B. The interfaces on each end will not come up either, because the IP interface addresses will not have a common prefix. This case can be detected, because both A and B will send periodic router-advertisement messages on the link, announcing their interface IP addresses. The device with the lower address MUST send a DHCPv6 request to the other device to get a new address.

A device SHOULD use the DHCPv6 User Class Option to identify the network it is attempting to reach. This is to prevent the controller from configuring devices attached to the network that are not part of the network to be configured. A string should be used that is not likely to match that of any other network that this network is connecting to. However, even if it matches by some small chance, the DHCPv6 authentication key will likely not match or the subsequent ZTP will fail. Inadvertently getting an IP address is not a terrible thing.

The controller should allocate a different BGP AS number for each device. There are plenty of private 4-octet ASNs available.

The controller will advertise its own loopback address to all the directly connected BGP neighbors with a community to identify it as a controller address. This IP address will be advertised by all devices to their directly connected BGP neighbors. The devices will use this BGP route to route back to the controller.

Each device will announce its interface addresses to the BGP connections of its directly connected neighbors tagged with a community. These routes will be re-announced only to the BGP session to the controller and not to directly connected neighbors. The BGP connections can be made to fail upon interface down or BFD down. BFD should only operate on the BGP sessions to directly connected neighbors, not on the session to the controller.

The devices will be segment-routing V6 (SRv6) [I-D.ietf-6man-segment-routing-header] capable. When a device receives an Ipv6 packet, it will first inspect the SRv6 extension header and be able to forward the packet to the next segment. If there is no SRv6 extension header or no more segments, then the packet should be for itself or for a directly connected neighbor or for a controller. If none of those match, then it must drop the packet.

The controller, knowing the topology, will be able to send a packet to any device in the network by building the appropriate SRv6 SID

list. Thus each device in the network does not need to store a route for every other device.

Once the controller has learnt the whole network topology, or at least a large recognizable part of it, it can complete the configuration of the network. This depends on the network. The controller will be programmed with a description of the expected network and applicable constraints. As discovery proceeds, the controller will try to match the discovered topology with the programmed description. An example of a data center description is: "A number of pods. Each pod consists of 384 TORs and 32 spines. Each TOR has 32 south facing ports and 32 north facing ports. Each spine has 384 south facing ports and 192 north facing ports. Super-spines connect the pods. Some of the pods are DCI pods. The devices need aggregatable addresses and BGP sessions." The controller should be able to recognize all the switches, the servers and the DCI routers and match the discovered topology to the description. It should then create configurations for all the devices and report inconsistencies. How the controller does this is out of scope of this document.

When a new device joins the network, the controller will detect it, because it will receive a DHCP request from it, relayed by its neighboring DHCP relay agent.

5. Security Considerations

TBD

6. IANA Considerations

TBD

7. Acknowldgements

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>.

[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, <https://www.rfc-editor.org/info/rfc3315>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/info/rfc5925>.

8.2. Informative References

[I-D.ietf-6man-segment-routing-header] Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-14 (work in progress), June 2018.

[I-D.ietf-netconf-zerotouch] Watsen, K., Abrahamsson, M., and I. Farrer, "Zero Touch Provisioning for Networking Devices", draft-ietf-netconf-zerotouch-25 (work in progress), September 2018.

[RFC7938] Lapukhov, P., Premji, A., and J. Mitchell, Ed., "Use of BGP for Routing in Large-Scale Data Centers", RFC 7938, DOI 10.17487/RFC7938, August 2016, <https://www.rfc-editor.org/info/rfc7938>.

Authors' Addresses

Jakob Heitz
Cisco
170 West Tasman Drive
San Jose, CA, CA 95134
USA

Email: jheitz@cisco.com

Kausik Majumdar
Cisco
170 West Tasman Drive
San Jose, CA, CA 95134
USA

Email: kmajumda@cisco.com

RTGWG
Internet-Draft
Intended status: Standards Track
Expires: February 13, 2022

Y. Qu
Futurewei
J. Tantsura
Microsoft
A. Lindem
Cisco
X. Liu
Volta Networks
August 12, 2021

A YANG Data Model for Routing Policy
draft-ietf-rtgwg-policy-model-31

Abstract

This document defines a YANG data model for configuring and managing routing policies in a vendor-neutral way. The model provides a generic routing policy framework which can be extended for specific routing protocols using the YANG 'augment' mechanism.

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 February 13, 2022.

Copyright Notice

Table of Contents

1. Introduction

This document describes a YANG [RFC7950] data model for routing policy configuration based on operational usage and best practices in a variety of service provider networks. The model is intended to be vendor-neutral, to allow operators to manage policy configuration consistently in environments with routers supplied by multiple vendors.

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

1.1. Goals and approach

This model does not aim to be feature complete -- it is a subset of the policy configuration parameters available in a variety of vendor implementations, but supports widely used constructs for managing how routes are imported, exported, and modified across different routing protocols. The model development approach has been to examine actual policy configurations in use across several operator networks. Hence, the focus is on enabling policy configuration capabilities and structure that are in wide use.

Despite the differences in details of policy expressions and conventions in various vendor implementations, the model reflects the observation that a relatively simple condition-action approach can be readily mapped to several existing vendor implementations, and also gives operators a familiar and straightforward way to express policy. A side effect of this design decision is that other methods for expressing policies are not considered.

Consistent with the goal to produce a data model that is vendor neutral, only policy expressions that are deemed to be widely available in existing major implementations are included in the model. Those configuration items that are only available from a single implementation are omitted from the model with the expectation they will be available in separate vendor-provided modules that augment the current model.

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.

Routing policy: A routing policy defines how routes are imported, exported, modified, and advertised between routing protocol instances or within a single routing protocol instance.

Policy chain: A policy chain is a sequence of policy definitions. They can be referenced from different contexts.

Policy statement: Policy statements consist of a set of conditions and actions (either of which may be empty).

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. Tree Diagrams

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

2.2. 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]
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and Corresponding YANG Modules

3. Model overview

The routing policy module has three main parts:

- A generic framework is provided to express policies as sets of related conditions and actions. This includes match sets and actions that are useful across many routing protocols.

- A structure that allows routing protocol models to add protocol-specific policy conditions and actions though YANG augmentations is also provided. There is a complete example of this for BGP [RFC4271] policies in the proposed vendor-neutral BGP data model [I-D.ietf-idr-bgp-model]. Appendix A provides an example of how an augmentation for BGP policies might be accomplished. Note that this section is not normative as the BGP model is still evolving.

- Finally, a reusable grouping is defined for attaching import and export rules in the context of routing configuration for different protocols, VRFs, etc. This also enables creation of policy chains and expressing default policy behavior. In this document, policy chains are sequences of policy definitions that are applied in order (described in Section 4).

The module makes use of the standard Internet types, such as IP addresses, autonomous system numbers, etc., defined in RFC 6991 [RFC6991].

4. Route policy expression

Policies are expressed as a sequence of top-level policy definitions each of which consists of a sequence of policy statements. Policy statements in turn consist of simple condition-action tuples. Conditions may include multiple match or comparison operations, and similarly, actions may include multiple changes to route attributes, or indicate a final disposition of accepting or rejecting the route. This structure is shown below.

```
+--rw routing-policy
   +--ro match-modified-attributes?   boolean
   +--rw policy-definitions
      +--rw policy-definition* [name]
         +--rw name          string
         +--rw statements
            +--rw statement* [name]
               +--rw name          string
               +--rw conditions
               |     ...
               +--rw actions
                     ...
```

4.1. Defined sets for policy matching

The model provides a collection of generic sets that can be used for matching in policy conditions. These sets are applicable for route selection across multiple routing protocols. They may be further augmented by protocol-specific models which have their own defined sets. The defined sets include:

- prefix sets - Each prefix set defines a set of IP prefixes, each with an associated IP prefix and netmask range (or exact length).

- neighbor sets - Each neighbor set defines a set of neighboring nodes by their IP addresses. A neighbor set is used for selecting routes based on the neighbors advertising the routes.

- tag set - Each tag set defines a set of generic tag values that can be used in matches for filtering routes.

The model structure for defined sets is shown below.

```
      +--rw routing-policy
         +--rw defined-sets
         |  +--rw prefix-sets
         |  |  +--rw prefix-set* [name]
         |  |     +--rw name        string
         |  |     +--rw mode?       enumeration
         |  |     +--rw prefixes
         |  |        +--rw prefix-list* [ip-prefix mask-length-lower
         |  |                            mask-length-upper]
         |  |           +--rw ip-prefix           inet:ip-prefix
         |  |           +--rw mask-length-lower    uint8
         |  |           +--rw mask-length-upper    uint8
         |  +--rw neighbor-sets
         |  |  +--rw neighbor-set* [name]
         |  |     +--rw name       string
         |  |     +--rw address*   inet:ip-address
         |  +--rw tag-sets
         |     +--rw tag-set* [name]
         |        +--rw name         string
         |        +--rw tag-value*   tag-type
```

4.2. Policy conditions

Policy statements consist of a set of conditions and actions (either of which may be empty). Conditions are used to match route attributes against a defined set (e.g., a prefix set), or to compare attributes against a specific value. The default action is to reject-route.

Match conditions may be further modified using the match-set-options configuration which allows network operators to change the behavior of a match. Three options are supported:

- ALL - match is true only if the given value matches all members of the set.
- ANY - match is true if the given value matches any member of the set.
- INVERT - match is true if the given value does not match any member of the given set.

Not all options are appropriate for matching against all defined sets (e.g., match ALL in a prefix set does not make sense). In the model, a restricted set of match options is used where applicable.

Comparison conditions may similarly use options to change how route attributes should be tested, e.g., for equality or inequality, against a given value.

While most policy conditions will be added by individual routing protocol models via augmentation, this routing policy model includes several generic match conditions and the ability to test which protocol or mechanism installed a route (e.g., BGP, IGP, static, etc.). The conditions included in the model are shown below.

```
+--rw routing-policy
  +--rw policy-definitions
      +--rw policy-definition* [name]
         +--rw name          string
         +--rw statements
            +--rw statement* [name]
               +--rw conditions
               |  +--rw call-policy?
               |  +--rw source-protocol?
               |  +--rw match-interface
               |  |  +--rw interface?
               |  +--rw match-prefix-set
               |  |  +--rw prefix-set?
               |  |  +--rw match-set-options?
               |  +--rw match-neighbor-set
               |  |  +--rw neighbor-set?
               |  +--rw match-tag-set
               |  |  +--rw tag-set?
               |  |  +--rw match-set-options?
               |  +--rw match-route-type*   identityref
               |     +--rw route-type*
```

4.3. Policy actions

When policy conditions are satisfied, policy actions are used to set various attributes of the route being processed, or to indicate the final disposition of the route, i.e., accept or reject.

Similar to policy conditions, the routing policy model includes generic actions in addition to the basic route disposition actions. These are shown below.

```
      +--rw routing-policy
        +--rw policy-definitions
           +--rw policy-definition* [name]
              +--rw statements
                +--rw statement* [name]
                  +--rw actions
                      +--rw policy-result?   policy-result-type
                      +--rw set-metric
                      |  +--rw metric-modification?
                      |  |         metric-modification-type
                      |  +--rw metric?                 uint32
                      +--rw set-metric-type
                      |  +--rw metric-type?   identityref
                      +--rw set-route-level
                      |  +--rw route-level?   identityref
                      +--rw set-route-preference?      uint16
                      +--rw set-tag?               tag-type
                      +--rw set-application-tag?   tag-type
```

4.4. Policy subroutines

Policy 'subroutines' (or nested policies) are supported by allowing policy statement conditions to reference other policy definitions using the call-policy configuration. Called policies apply their conditions and actions before returning to the calling policy statement and resuming evaluation. The outcome of the called policy affects the evaluation of the calling policy. If the called policy results in an accept-route, then the subroutine returns an effective Boolean true value to the calling policy. For the calling policy, this is equivalent to a condition statement evaluating to a true value and evaluation of the policy continues (see Section 5). Note that the called policy may also modify attributes of the route in its action statements. Similarly, a reject-route action returns false and the calling policy evaluation will be affected accordingly. When the end of the subroutine policy statements is reached, the default route disposition action is returned (i.e., Boolean false for reject-route). Consequently, a subroutine cannot explicitly accept or reject a route. Rather, the called policy returns Boolean true if its outcome is accept-route or Boolean false if its outcome is reject-route. Route acceptance or rejection is solely determined by the top-level policy.

Note that the called policy may itself call other policies (subject to implementation limitations). The model does not prescribe a nesting depth because this varies among implementations. For example, an implementation may only support a single level of subroutine recursion. As with any routing policy construction, care must be taken with nested policies to ensure that the effective

return value results in the intended behavior. Nested policies are a convenience in many routing policy constructions but creating policies nested beyond a small number of levels (e.g., 2-3) is discouraged. Also, implementations MUST validate to ensure that there is no recursion among nested routing policies.

5. Policy evaluation

Evaluation of each policy definition proceeds by evaluating its individual policy statements in order that they are defined. When all the condition statements in a policy statement are satisfied, the corresponding action statements are executed. If the actions include either accept-route or reject-route actions, evaluation of the current policy definition stops, and no further policy statement is evaluated. If there are multiple policies in the policy chain, subsequent policies are not evaluated. Policy chains are sequences of policy definitions (as described in Section 4).

If the conditions are not satisfied, then evaluation proceeds to the next policy statement. If none of the policy statement conditions are satisfied, then evaluation of the current policy definition stops, and the next policy definition in the chain is evaluated. When the end of the policy chain is reached, the default route disposition action is performed (i.e., reject-route unless an alternate default action is specified for the chain).

Whether the route's pre-policy attributes are used for testing policy statement conditions is dependent on the implementation specific value of the match-modified-attributes leaf. If match-modified-attributes is false and actions modify route attributes, these modifications are not used for policy statement conditions. Conversely, if match-modified-attributes is true and actions modify the policy application-specific attributes, the attributes as modified by the policy are used for policy condition statements.

6. Applying routing policy

Routing policy is applied by defining and attaching policy chains in various routing contexts. Policy chains are sequences of policy definitions (described in Section 4). They can be referenced from different contexts. For example, a policy chain could be associated with a routing protocol and used to control its interaction with its protocol peers. Or it could be used to control the interaction between a routing protocol and the local routing information base. A policy chain has an associated direction (import or export), with respect to the context in which it is referenced.

The routing policy model defines an apply-policy grouping that can be imported and used by other models. As shown below, it allows definition of import and export policy chains, as well as specifying the default route disposition to be used when no policy definition in the chain results in a final decision.

```
      +--rw apply-policy
      |  +--rw import-policy*
      |  +--rw default-import-policy?   default-policy-type
      |  +--rw export-policy*
      |  +--rw default-export-policy?   default-policy-type
```

The default policy defined by the model is to reject the route for both import and export policies.

7. YANG Module and Tree

7.1. Routing Policy Model Tree

The tree of the routing policy model is shown below.

```
module: ietf-routing-policy
rw routing-policy
  +--rw defined-sets
  |  +--rw prefix-sets
  |  |  +--rw prefix-set* [name mode]
  |  |     +--rw name         string
  |  |     +--rw mode         enumeration
  |  |     +--rw prefixes
  |  |        +--rw prefix-list* [ip-prefix mask-length-lower
  |  |                            mask-length-upper]
  |  |           +--rw ip-prefix            inet:ip-prefix
  |  |           +--rw mask-length-lower    uint8
  |  |           +--rw mask-length-upper    uint8
  |  +--rw neighbor-sets
  |  |  +--rw neighbor-set* [name]
  |  |     +--rw name       string
  |  |     +--rw address*   inet:ip-address
  |  +--rw tag-sets
  |     +--rw tag-set* [name]
  |        +--rw name         string
  |        +--rw tag-value*   tag-type
  +--rw policy-definitions
   +--ro match-modified-attributes?   boolean
   +--rw policy-definition* [name]
     +--rw name          string
     +--rw statements
        +--rw statement* [name]
```

```
              +--rw name          string
              +--rw conditions
              |  +--rw call-policy?       -> ../../../../../..
              |                           /policy-definitions
              |                           /policy-definition/name
              |  +--rw source-protocol?      identityref
              |  +--rw match-interface
              |  |  +--rw interface?      -> /if:interfaces/interface
              |  |                        /name
              |  +--rw match-prefix-set
              |  |  +--rw prefix-set?     -> ../../../../../../..
              |  |                        /defined-sets/prefix-sets
              |  |                        /prefix-set/name
              |  |  +--rw match-set-options?   match-set-options-type
              |  +--rw match-neighbor-set
              |  |  +--rw neighbor-set?   -> ../../../../../../..
              |  |                        /defined-sets/neighbor-sets
              |  |                        /neighbor-set/name
              |  +--rw match-tag-set
              |  |  +--rw tag-set?        -> ../../../../../../..
              |  |                        /defined-sets/tag-sets
              |  |                        /tag-set/name
              |  |  +--rw match-set-options? match-set-options-type
              |  +--rw match-route-type*     identityref
              +--rw actions
                 +--rw policy-result?         policy-result-type
                 +--rw set-metric
                 |  +--rw metric-modification? metric-modification-type
                 |  +--rw metric?                uint32
                 +--rw set-metric-type
                 |  +--rw metric-type?   identityref
                 +--rw set-route-level
                 |  +--rw route-level?   identityref
                 +--rw set-route-preference?        uint16
                 +--rw set-tag?               tag-type
                 +--rw set-application-tag?   tag-type
```

7.2. Routing policy model

The following RFCs are not referenced in the document text but are referenced in the ietf-routing-policy.yang module: [RFC2328], [RFC3101], [RFC5130], [RFC5302], [RFC6991], and [RFC8343].

```
<CODE BEGINS> file "ietf-routing-policy@2021-08-12.yang"
module ietf-routing-policy {

  yang-version "1.1";
```

```
  namespace "urn:ietf:params:xml:ns:yang:ietf-routing-policy";
  prefix rt-pol;

  import ietf-inet-types {
    prefix "inet";
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-yang-types {
    prefix "yang";
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-interfaces {
    prefix "if";
    reference
      "RFC 8343: A YANG Data Model for Interface
                 Management (NMDA Version)";
  }

  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing
                 Management (NMDA Version)";
  }

  organization
    "IETF RTGWG - Routing Area Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/rtgwg/>
     WG List:  <mailto: rtgwg@ietf.org>

     Editor:   Yingzhen Qu
               <mailto: yingzhen.qu@futurewei.com>
               Jeff Tantsura
               <mailto: jefftant.ietf@gmail.com>
               Acee Lindem
               <mailto: acee@cisco.com>
               Xufeng Liu
               <mailto: xufeng.liu.ietf@gmail.com>";

  description
    "This module describes a YANG model for routing policy
     configuration. It is a limited subset of all of the policy
     configuration parameters available in the variety of vendor
```

```
      implementations, but supports widely used constructs for
      managing how routes are imported, exported, modified and
      advertised across different routing protocol instances or
      within a single routing protocol instance.  This module is
      intended to be used in conjunction with routing protocol
      configuration modules (e.g., BGP) defined in other models.

      This YANG module conforms to the Network Management
      Datastore Architecture (NMDA), as described in RFC 8342.

      Copyright (c) 2021 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
      (https://trustee.ietf.org/license-info).

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

      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 (RFC 2119) (RFC 8174) when,
      and only when, they appear in all capitals, as shown here.";

  reference "RFC XXXX: A YANG Data Model for Routing Policy.";

  revision "2021-08-12" {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for Routing Policy Management.";
  }

  /* Identities */

  identity metric-type {
    description
      "Base identity for route metric types.";
  }

  identity ospf-type-1-metric {
    base metric-type;
    description
```

```
      "Identity for the OSPF type 1 external metric types. It
       is only applicable to OSPF routes.";
    reference
      "RFC 2328: OSPF Version 2";
  }

  identity ospf-type-2-metric {
    base metric-type;
    description
      "Identity for the OSPF type 2 external metric types. It
       is only applicable to OSPF routes.";
    reference
      "RFC 2328: OSPF Version 2";
  }

  identity isis-internal-metric {
    base metric-type;
    description
      "Identity for the IS-IS internal metric types. It is only
       applicable to IS-IS routes.";
    reference
      "RFC 5302: Domain-Wide Prefix Distribution with
       Two-Level IS-IS";
  }

  identity isis-external-metric {
    base metric-type;
    description
      "Identity for the IS-IS external metric types. It is only
       applicable to IS-IS routes.";
    reference
      "RFC 5302: Domain-Wide Prefix Distribution with
       Two-Level IS-IS";
  }

  identity route-level {
    description
      "Base identity for route import level.";
  }

  identity ospf-normal {
    base route-level;
    description
      "Identity for OSPF importation into normal areas
       It is only applicable to routes imported
       into the OSPF protocol.";
    reference
      "RFC 2328: OSPF Version 2";
```

```
  }

  identity ospf-nssa-only {
    base route-level;
    description
      "Identity for the OSPF Not-So-Stubby Area (NSSA) area
       importation. It is only applicable to routes imported
       into the OSPF protocol.";
    reference
      "RFC 3101: The OSPF Not-So-Stubby Area (NSSA) Option";
  }

  identity ospf-normal-nssa {
    base route-level;
    description
      "Identity for OSPF importation into both normal and NSSA
       areas, it is only applicable to routes imported into
       the OSPF protocol.";
    reference
      "RFC 3101: The OSPF Not-So-Stubby Area (NSSA) Option";
  }

  identity isis-level-1 {
    base route-level;
    description
      "Identity for IS-IS Level 1 area importation. It is only
       applicable to routes imported into the IS-IS protocol.";
    reference
      "RFC 5302: Domain-Wide Prefix Distribution with
       Two-Level IS-IS";
  }

  identity isis-level-2 {
    base route-level;
    description
      "Identity for IS-IS Level 2 area importation. It is only
       applicable to routes imported into the IS-IS protocol.";
    reference
      "RFC 5302: Domain-Wide Prefix Distribution with
       Two-Level IS-IS";
  }

  identity isis-level-1-2 {
    base route-level;
    description
      "Identity for IS-IS importation into both Level 1 and Level 2
       areas. It is only applicable to routes imported into the IS-IS
       protocol.";
```

```
    reference
      "RFC 5302: Domain-Wide Prefix Distribution with
       Two-Level IS-IS";
  }

  identity proto-route-type {
    description
      "Base identity for route type within a protocol.";
  }

  identity isis-level-1-type {
    base proto-route-type;
    description
      "Identity for IS-IS Level 1 route type. It is only
       applicable to IS-IS routes.";
    reference
      "RFC 5302: Domain-Wide Prefix Distribution with
       Two-Level IS-IS";
  }

  identity isis-level-2-type {
    base proto-route-type;
    description
      "Identity for IS-IS Level 2 route type. It is only
       applicable to IS-IS routes.";
    reference
      "RFC 5302: Domain-Wide Prefix Distribution with
       Two-Level IS-IS";
  }

  identity ospf-internal-type {
    base proto-route-type;
    description
      "Identity for OSPF intra-area or inter-area route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 2328: OSPF Version 2";
  }

  identity ospf-external-type {
    base proto-route-type;
    description
      "Identity for OSPF external type 1/2 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 2328: OSPF Version 2";
  }
```

```
  identity ospf-external-t1-type {
    base ospf-external-type;
    description
      "Identity for OSPF external type 1 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 2328: OSPF Version 2";
  }

  identity ospf-external-t2-type {
    base ospf-external-type;
    description
      "Identity for OSPF external type 2 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 2328: OSPF Version 2";
  }

  identity ospf-nssa-type {
    base proto-route-type;
    description
      "Identity for OSPF NSSA type 1/2 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 3101: The OSPF Not-So-Stubby Area (NSSA) Option";
  }

  identity ospf-nssa-t1-type {
    base ospf-nssa-type;
    description
      "Identity for OSPF NSSA type 1 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 3101: The OSPF Not-So-Stubby Area (NSSA) Option";
  }

  identity ospf-nssa-t2-type {
    base ospf-nssa-type;
    description
      "Identity for OSPF NSSA type 2 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 3101: The OSPF Not-So-Stubby Area (NSSA) Option";
  }

  identity bgp-internal {
    base proto-route-type;
    description
```

```
      "Identity for routes learned from internal BGP (IBGP).
       It is only applicable to BGP routes.";
    reference
      "RFC 4271: A Border Gateway Protocol 4 (BGP-4)";
  }

  identity bgp-external {
    base proto-route-type;
    description
      "Identity for routes learned from external BGP (EBGP).
       It is only applicable to BGP routes.";
    reference
      "RFC 4271: A Border Gateway Protocol 4 (BGP-4)";
  }

  /* Type Definitions */

  typedef default-policy-type {
    type enumeration {
      enum accept-route {
        description
          "Default policy to accept the route.";
      }
      enum reject-route {
        description
          "Default policy to reject the route.";
      }
    }
    description
      "Type used to specify route disposition in
       a policy chain. This typedef is used in
       the default import and export policy.";
  }

  typedef policy-result-type {
    type enumeration {
      enum accept-route {
        description
          "Policy accepts the route.";
      }
      enum reject-route {
        description
          "Policy rejects the route.";
      }
    }
    description
      "Type used to specify route disposition in
       a policy chain.";
```

```
  }

  typedef tag-type {
    type union {
      type uint32;
      type yang:hex-string;
    }
    description
      "Type for expressing route tags on a local system,
       including IS-IS and OSPF; may be expressed as either decimal
       or hexadecimal integer.";
    reference
      "RFC 2328: OSPF Version 2
       RFC 5130: A Policy Control Mechanism in IS-IS Using
                 Administrative Tags";
  }

  typedef match-set-options-type {
    type enumeration {
      enum any {
        description
          "Match is true if given value matches any member
           of the defined set.";
      }
      enum all {
        description
          "Match is true if given value matches all
           members of the defined set.";
      }
      enum invert {
        description
          "Match is true if given value does not match any
           member of the defined set.";
      }
    }
    default any;
    description
      "Options that govern the behavior of a match statement.  The
       default behavior is any, i.e., the given value matches any
       of the members of the defined set.";
  }

  typedef metric-modification-type {
    type enumeration {
      enum set-metric {
        description
          "Set the metric to the specified value.";
      }
```

```
      enum  add-metric {
        description
          "Add the specified value to the existing metric.
           If the result overflows the maximum metric
           (0xffffffff), set the metric to the maximum.";
      }
      enum  subtract-metric {
        description
          "Subtract the specified value from the existing metric. If
           the result is less than 0, set the metric to 0.";
      }
    }
    description
      "Type used to specify how to set the metric given the
       specified value.";
  }

  /* Groupings */

  grouping prefix {
    description
      "Configuration data for a prefix definition.

       The combination of mask-length-lower and mask-length-upper
       define a range for the mask length, or single 'exact'
       length if mask-length-lower and mask-length-upper are
       equal.

       Example: 192.0.2.0/24 through 192.0.2.0/26 would be
       expressed as prefix: 192.0.2.0/24,
                    mask-length-lower=24,
                    mask-length-upper=26

       Example: 192.0.2.0/24 (an exact match) would be
       expressed as prefix: 192.0.2.0/24,
                    mask-length-lower=24,
                    mask-length-upper=24

       Example: 2001:DB8::/32 through 2001:DB8::/64 would be
       expressed as prefix: 2001:DB8::/32,
                    mask-length-lower=32,
                    mask-length-upper=64";

    leaf ip-prefix {
      type inet:ip-prefix;
      mandatory true;
      description
        "The IP prefix represented as an IPv6 or IPv4 network
```

```
          number followed by a prefix length with an intervening
          slash character as a delimiter. All members of the
          prefix-set MUST be of the same address family as the
          prefix-set mode.";
      }

      leaf mask-length-lower {
        type uint8 {
          range "0..128";
        }
        description
          "Mask length range lower bound. It MUST NOT be less than
           the prefix length defined in ip-prefix.";
      }
      leaf mask-length-upper {
        type uint8 {
          range "1..128";
        }
        must "../mask-length-upper >= ../mask-length-lower" {
          error-message "The upper bound MUST NOT be less"
                      + "than lower bound.";
        }
        description
          "Mask length range upper bound. It MUST NOT be less than
           lower bound.";
      }
    }

    grouping match-set-options-group {
      description
        "Grouping containing options relating to how a particular set
         will be matched.";

      leaf match-set-options {
        type match-set-options-type;
        description
          "Optional parameter that governs the behavior of the
           match operation.";
      }
    }

    grouping match-set-options-restricted-group {
      description
        "Grouping for a restricted set of match operation
         modifiers.";

      leaf match-set-options {
        type match-set-options-type {
```

```
        enum any {
          description
            "Match is true if given value matches any
             member of the defined set.";
        }
        enum invert {
          description
            "Match is true if given value does not match
             any member of the defined set.";
        }
      }
      description
        "Optional parameter that governs the behavior of the
         match operation. This leaf only supports matching on
         'any' member of the set or 'invert' the match.
         Matching on 'all' is not supported.";
    }
  }

  grouping apply-policy-group {
    description
      "Top level container for routing policy applications. This
       grouping is intended to be used in routing models where
       needed.";

    container apply-policy {
      description
        "Anchor point for routing policies in the model.
         Import and export policies are with respect to the local
         routing table, i.e., export (send) and import (receive),
         depending on the context.";

      leaf-list import-policy {
        type leafref {
          path "/rt-pol:routing-policy/rt-pol:policy-definitions/" +
          "rt-pol:policy-definition/rt-pol:name";
          require-instance true;
        }
        ordered-by user;
        description
          "List of policy names in sequence to be applied on
          receiving redistributed routes from another routing protocol
          or receiving a routing update in the current context, e.g.,
          for the current peer group, neighbor, address family, etc.";
      }

      leaf default-import-policy {
        type default-policy-type;
```

```
        default reject-route;
        description
          "Explicitly set a default policy if no policy definition
           in the import policy chain is satisfied.";
      }

      leaf-list export-policy {
        type leafref {
          path "/rt-pol:routing-policy/rt-pol:policy-definitions/" +
            "rt-pol:policy-definition/rt-pol:name";
          require-instance true;
        }
        ordered-by user;
        description
          "List of policy names in sequence to be applied on
          redistributing routes from one routing protocol to another
          or sending a routing update in the current context, e.g.,
          for the current peer group, neighbor, address family, etc.";
      }

      leaf default-export-policy {
        type default-policy-type;
        default reject-route;
        description
          "Explicitly set a default policy if no policy definition
           in the export policy chain is satisfied.";
      }
    }
  }

  container routing-policy {
    description
      "Top-level container for all routing policy.";

    container defined-sets {
      description
        "Predefined sets of attributes used in policy match
         statements.";

      container prefix-sets {
        description
          "Data definitions for a list of IPv4 or IPv6
          prefixes which are matched as part of a policy.";
        list prefix-set {
          key "name mode";
          description
            "List of the defined prefix sets";
```

```
          leaf name {
            type string;
            description
              "Name of the prefix set -- this is used as a label to
               reference the set in match conditions.";
          }

          leaf mode {
            type enumeration {
              enum ipv4 {
                description
                  "Prefix set contains IPv4 prefixes only.";
              }
              enum ipv6 {
                description
                  "Prefix set contains IPv6 prefixes only.";
              }
            }
            description
              "Indicates the mode of the prefix set, in terms of
               which address families (IPv4 or IPv6) are present.
               The mode provides a hint, all prefixes MUST be of
               the indicated type. The device MUST validate that
               all prefixes and reject the configuration if there
               is a discrepancy.";
          }

          container prefixes {
            description
              "Container for the list of prefixes in a policy
               prefix list. Since individual prefixes do not have
               unique actions, the order in which the prefix in
               prefix-list are matched has no impact on the outcome
               and is left to the implementation. A given prefix-set
               condition is satisfied if the input prefix matches
               any of the prefixes in the prefix-set.";

            list prefix-list {
              key "ip-prefix mask-length-lower mask-length-upper";
              description
                "List of prefixes in the prefix set.";

              uses prefix;
            }
          }
        }
      }
```

```
      container neighbor-sets {
        description
          "Data definition for a list of IPv4 or IPv6
           neighbors which can be matched in a routing policy.";

        list neighbor-set {
          key "name";
          description
            "List of defined neighbor sets for use in policies.";

          leaf name {
            type string;
            description
              "Name of the neighbor set -- this is used as a label
               to reference the set in match conditions.";
          }

          leaf-list address {
            type inet:ip-address;
            description
              "List of IP addresses in the neighbor set.";
          }
        }
      }

      container tag-sets {
        description
          "Data definitions for a list of tags which can
           be matched in policies.";

        list tag-set {
          key "name";
          description
            "List of tag set definitions.";

          leaf name {
            type string;
            description
              "Name of the tag set -- this is used as a label to
              reference the set in match conditions.";
          }

          leaf-list tag-value {
            type tag-type;
            description
              "Value of the tag set member.";
          }
        }
```

```
      }
    }

    container policy-definitions {
      description
        "Enclosing container for the list of top-level policy
         definitions.";

      leaf match-modified-attributes {
        type boolean;
        config false;
        description
          "This boolean value dictates whether matches are performed
           on the actual route attributes or route attributes
           modified by policy statements preceding the match.";
      }

      list policy-definition {
        key "name";
        description
          "List of top-level policy definitions, keyed by unique
           name.  These policy definitions are expected to be
           referenced (by name) in policy chains specified in
           import or export configuration statements.";

        leaf name {
          type string;
          description
            "Name of the top-level policy definition -- this name
             is used in references to the current policy.";
        }

        container statements {
          description
            "Enclosing container for policy statements.";

          list statement {
            key "name";
            ordered-by user;
            description
              "Policy statements group conditions and actions
               within a policy definition.  They are evaluated in
               the order specified.";

            leaf name {
              type string;
              description
                "Name of the policy statement.";
```

```
            }

            container conditions {
              description
                "Condition statements for the current policy
                 statement.";

              leaf call-policy {
                type leafref {
                  path "../../../../../../" +
                       "rt-pol:policy-definitions/" +
                       "rt-pol:policy-definition/rt-pol:name";
                  require-instance true;
                }
                description
                  "Applies the statements from the specified policy
                   definition and then returns control to the current
                   policy statement. Note that the called policy
                   may itself call other policies (subject to
                   implementation limitations). This is intended to
                   provide a policy 'subroutine' capability.  The
                   called policy SHOULD contain an explicit or a
                   default route disposition that returns an
                   effective true (accept-route) or false
                   (reject-route), otherwise the behavior may be
                   ambiguous.";
              }

              leaf source-protocol {
                type identityref {
                  base rt:control-plane-protocol;
                }
                description
                  "Condition to check the protocol / method used to
                  install the route into the local routing table.";
              }

              container match-interface {
                leaf interface {
                  type leafref {
                    path "/if:interfaces/if:interface/if:name";
                  }
                  description
                    "Reference to a base interface.";
                }
                description
                  "Container for interface match conditions";
              }
```

```
            container match-prefix-set {
              leaf prefix-set {
                type leafref {
                  path "../../../../../../../defined-sets/" +
                       "prefix-sets/prefix-set/name";
                }
                description
                  "References a defined prefix set.";
              }
              uses match-set-options-restricted-group;

              description
                "Match a referenced prefix-set according to the
                 logic defined in the match-set-options leaf.";
            }

            container match-neighbor-set {
              leaf neighbor-set {
                type leafref {
                  path "../../../../../../../defined-sets/" +
                       "neighbor-sets/neighbor-set/name";
                  require-instance true;
                }
                description
                  "References a defined neighbor set.";
              }

              description
                "Match a referenced neighbor set.";
            }

            container match-tag-set {
              leaf tag-set {
                type leafref {
                  path  "../../../../../../../defined-sets/" +
                        "tag-sets/tag-set/name";
                  require-instance true;
                }
                description
                  "References a defined tag set.";
              }
              uses match-set-options-group;

              description
                "Match a referenced tag set according to the logic
                 defined in the match-set-options leaf.";
            }
```

```
            container match-route-type {
              description
                "This container provides route-type match condition";

              leaf-list route-type {
                type identityref {
                  base proto-route-type;
                }
                description
                  "Condition to check the protocol-specific type
                   of route. This is normally used during route
                   importation to select routes or to set protocol
                   specific attributes based on the route type.";
              }
            }
          }

          container actions {
            description
              "Top-level container for policy action
               statements.";
            leaf policy-result {
              type policy-result-type;
              default reject-route;
              description
                "Select the final disposition for the route,
                 either accept or reject.";
            }
            container set-metric {
              leaf metric-modification {
                type metric-modification-type;
                description
                  "Indicates how to modify the metric.";
              }
              leaf metric {
                type uint32;
                description
                  "Metric value to set, add, or subtract.";
              }
              description
                "Set the metric for the route.";
            }
            container set-metric-type {
              leaf metric-type {
                type identityref {
                  base metric-type;
                }
                description
```

```
                    "Route metric type.";
                }
                description
                  "Set the metric type for the route.";
              }
              container set-route-level {
                leaf route-level {
                  type identityref {
                    base route-level;
                  }
                  description
                    "Route import level.";
                }
                description
                  "Set the level for importation or
                   exportation of routes.";
              }
              leaf set-route-preference {
                type uint16;
                description
                  "Set the preference for the route. It is also
                   known as 'administrative distance', allows for
                   selecting the preferred route among routes with
                   the same destination prefix. A smaller value is
                   more preferred.";
              }
              leaf set-tag {
                type tag-type;
                description
                  "Set the tag for the route.";
              }
              leaf set-application-tag {
                type tag-type;
                description
                  "Set the application tag for the route.
                   The application-specific tag is an additional tag
                   that can be used by applications that require
                   semantics and/or policy different from that of the
                   tag. For example, the tag is usually automatically
                   advertised in OSPF AS-External Link State
                   Advertisements (LSAs) while this application-specific
                   tag is not advertised implicitly.";
              }
            }
          }
        }
      }
    }
```

```
  }
}
<CODE ENDS>
```

8. Security Considerations

The YANG module specified in this document defines 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 [RFC8446].

The NETCONF Access Control Model (NACM) [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 this 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. These are the subtrees and data nodes and their sensitivity/vulnerability:

/routing-policy/defined-sets/prefix-sets -- Modification to prefix-sets could result in a Denial-of-Service (DoS) attack. An attacker may try to modify prefix-sets and redirect or drop traffic. Redirection of traffic could be used as part of a more elaborate attack to either collect sensitive information or masquerade a service. Additionally, a control-plane DoS attack could be accomplished by allowing a large number of routes to be leaked into a routing protocol domain (e.g., BGP).

/routing-policy/defined-sets/neighbor-sets -- Modification to the neighbor-sets could be used to mount a DoS attack or more elaborate attack as with prefix-sets. For example, a DoS attack could be mounted by changing the neighbor-set from which routes are accepted.

/routing-policy/defined-sets/tag-sets -- Modification to the tag-sets could be used to mount a DoS attack. Routes with certain tags might be redirected or dropped. The implications are similar to prefix-sets and neighbor-sets. However, the attack may be more difficult to detect as the routing policy usage of route tags and

intent must be understood to recognize the breach. Conversely, the implications of prefix-set or neighbor set modification are easier to recognize.

/routing-policy/policy-definitions/policy-definition /statements/statement/conditions -- Modification to the conditions could be used to mount a DoS attack or other attack. An attacker may change a policy condition and redirect or drop traffic. As with prefix-sets, neighbor-sets, or tag-sets, traffic redirection could be used as part of a more elaborate attack.

/routing-policy/policy-definitions/policy-definition /statements/statement/actions -- Modification to actions could be used to mount a DoS attack or other attack. Traffic may be redirected or dropped. As with prefix-sets, neighbor-sets, or tag-sets, traffic redirection could be used as part of a more elaborate attack. Additionally, route attributes may be changed to mount a second-level attack that is more difficult to detect.

Some of the readable data nodes in the 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. These are the subtrees and data nodes and their sensitivity/vulnerability:

/routing-policy/defined-sets/prefix-sets -- Knowledge of these data nodes can be used to ascertain which local prefixes are susceptible to a Denial-of-Service (DoS) attack.

/routing-policy/defined-sets/prefix-sets -- Knowledge of these data nodes can be used to ascertain local neighbors against whom to mount a Denial-of-Service (DoS) attack.

/routing-policy/policy-definitions/policy-definition /statements/ -- Knowledge of these data nodes can be used to attack the local router with a Denial-of-Service (DoS) attack. Additionally, policies and their attendant conditions and actions should be considered proprietary and disclosure could be used to ascertain partners, customers, and supplies. Furthermore, the policies themselves could represent intellectual property and disclosure could diminish their corresponding business advantage.

Routing policy configuration has a significant impact on network operations, and, as such, other YANG models that reference routing policies are also susceptible to vulnerabilities relating the YANG data nodes specified above.

9. IANA Considerations

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

```
URI: urn:ietf:params:xml:ns:yang:ietf-routing-policy
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-routing-policy
namespace: urn:ietf:params:xml:ns:yang:ietf-routing-policy
prefix: rt-pol
reference: RFC XXXX
```

10. Acknowledgements

The routing policy module defined in this document is based on the OpenConfig route policy model. The authors would like to thank to OpenConfig for their contributions, especially Anees Shaikh, Rob Shakir, Kevin D'Souza, and Chris Chase.

The authors are grateful for valuable contributions to this document and the associated models from: Ebben Aires, Luyuan Fang, Josh George, Stephane Litkowski, Ina Minei, Carl Moberg, Eric Osborne, Steve Padgett, Juergen Schoenwaelder, Jim Uttaro, Russ White, and John Heasley.

Thanks to Mahesh Jethanandani, John Scudder, Chris Bowers and Tom Petch for their reviews and comments.

11. References

11.1. 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>.

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

[RFC3101] Murphy, P., "The OSPF Not-So-Stubby Area (NSSA) Option", RFC 3101, DOI 10.17487/RFC3101, January 2003, <https://www.rfc-editor.org/info/rfc3101>.

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

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC5130] Previdi, S., Shand, M., Ed., and C. Martin, "A Policy Control Mechanism in IS-IS Using Administrative Tags", RFC 5130, DOI 10.17487/RFC5130, February 2008, <https://www.rfc-editor.org/info/rfc5130>.

[RFC5302] Li, T., Smit, H., and T. Przygienda, "Domain-Wide Prefix Distribution with Two-Level IS-IS", RFC 5302, DOI 10.17487/RFC5302, October 2008, <https://www.rfc-editor.org/info/rfc5302>.

[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>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

11.2. Informative references

[I-D.ietf-idr-bgp-model] Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", draft-ietf-idr-bgp-model-11 (work in progress), July 2021.

Appendix A. Routing protocol-specific policies

Routing models that require the ability to apply routing policy may augment the routing policy model with protocol or other specific policy configuration. The routing policy model assumes that additional defined sets, conditions, and actions may all be added by other models.

The example below provides an illustration of how another data model can augment parts of this routing policy data model. It uses

specific examples from draft-ietf-idr-bgp-model-09 to show in a concrete manner how the different pieces fit together. This example is not normative with respect to [I-D.ietf-idr-bgp-model]. The model similarly augments BGP-specific conditions and actions in the corresponding sections of the routing policy model. In the example below, the XPath prefix "bp:" specifies import from the ietf-bgp-policy sub-module and the XPath prefix "bt:" specifies import from the ietf-bgp-types sub-module [I-D.ietf-idr-bgp-model].

```
module: ietf-routing-policy
+--rw routing-policy
  +--rw defined-sets
  |  +--rw prefix-sets
  |  |  +--rw prefix-set* [name]
  |  |     +--rw name        string
  |  |     +--rw mode?       enumeration
  |  |     +--rw prefixes
  |  |        +--rw prefix-list* [ip-prefix mask-length-lower
  |  |                           mask-length-upper]
  |  |           +--rw ip-prefix            inet:ip-prefix
  |  |           +--rw mask-length-lower    uint8
  |  |           +--rw mask-length-upper    uint8
  |  +--rw neighbor-sets
  |  |  +--rw neighbor-set* [name]
  |  |     +--rw name       string
  |  |     +--rw address*   inet:ip-address
  |  +--rw tag-sets
  |  |  +--rw tag-set* [name]
  |  |     +--rw name         string
  |  |     +--rw tag-value*   tag-type
  |  +--rw bp:bgp-defined-sets
  |     +--rw bp:community-sets
  |     |  +--rw bp:community-set* [name]
  |     |     +--rw bp:name      string
  |     |     +--rw bp:member*   union
  |     +--rw bp:ext-community-sets
  |     |  +--rw bp:ext-community-set* [name]
  |     |     +--rw bp:name      string
  |     |     +--rw bp:member*   union
  |     +--rw bp:as-path-sets
  |        +--rw bp:as-path-set* [name]
  |           +--rw bp:name      string
  |           +--rw bp:member*   string
  +--rw policy-definitions
     +--ro match-modified-attributes?   boolean
     +--rw policy-definition* [name]
        +--rw name          string
        +--rw statements
```

```
            +--rw statement* [name]
               +--rw name          string
               +--rw conditions
               |  +--rw call-policy?
               |  +--rw source-protocol?          identityref
               |  +--rw match-interface
               |  |  +--rw interface?
               |  +--rw match-prefix-set
               |  |  +--rw prefix-set?       prefix-set/name
               |  |  +--rw match-set-options?   match-set-options-type
               |  +--rw match-neighbor-set
               |  |  +--rw neighbor-set?
               |  +--rw match-tag-set
               |  |  +--rw tag-set?
               |  |  +--rw match-set-options?   match-set-options-type
               |  +--rw match-route-type*   identityref
               |  +--rw bp:bgp-conditions
               |     +--rw bp:med-eq?         uint32
               |     +--rw bp:origin-eq?      bt:bgp-origin-attr-type
               |     +--rw bp:next-hop-in*    inet:ip-address-no-zone
               |     +--rw bp:afi-safi-in*    identityref
               |     +--rw bp:local-pref-eq?  uint32
               |     +--rw bp:route-type?     enumeration
               |     +--rw bp:community-count
               |     +--rw bp:as-path-length
               |     +--rw bp:match-community-set
               |     |  +--rw bp:community-set?
               |     |  +--rw bp:match-set-options?
               |     +--rw bp:match-ext-community-set
               |     |  +--rw bp:ext-community-set?
               |     |  +--rw bp:match-set-options?
               |     +--rw bp:match-as-path-set
               |        +--rw bp:as-path-set?
               |        +--rw bp:match-set-options?
               +--rw actions
                  +--rw policy-result?         policy-result-type
                  +--rw set-metric
                  |  +--rw metric-modification?
                  |  +--rw metric?                uint32
                  +--rw set-metric-type
                  |  +--rw metric-type?   identityref
                  +--rw set-route-level
                  |  +--rw route-level?   identityref
                  +--rw set-route-preference?        uint16
                  +--rw set-tag?               tag-type
                  +--rw set-application-tag?   tag-type
                  +--rw bp:bgp-actions
                     +--rw bp:set-route-origin?bt:bgp-origin-attr-type
```

```
                      +--rw bp:set-local-pref?   uint32
                      +--rw bp:set-next-hop?     bgp-next-hop-type
                      +--rw bp:set-med?          bgp-set-med-type
                      +--rw bp:set-as-path-prepend
                      |  +--rw bp:repeat-n?   uint8
                      +--rw bp:set-community
                      |  +--rw bp:method?      enumeration
                      |  +--rw bp:options?
                      |  +--rw bp:inline
                      |  |  +--rw bp:communities*   union
                      |  +--rw bp:reference
                      |     +--rw bp:community-set-ref?
                      +--rw bp:set-ext-community
                         +--rw bp:method?      enumeration
                         +--rw bp:options?
                         +--rw bp:inline
                         |  +--rw bp:communities*   union
                         +--rw bp:reference
                            +--rw bp:ext-community-set-ref?
```

Appendix B. Policy examples

Below we show examples of XML-encoded configuration data using the routing policy and BGP models to illustrate both how policies are defined, and how they can be applied. Note that the XML has been simplified for readability.

The following example shows how prefix-set and tag-set can be defined. The policy condition is to match a prefix-set and a tag-set, and the action is to accept routes that match the condition.

```
  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <routing-policy
     xmlns="urn:ietf:params:xml:ns:yang:ietf-routing-policy">

        <defined-sets>
          <prefix-sets>
            <prefix-set>
              <name>prefix-set-A</name>
              <mode>ipv4</mode>
              <prefixes>
                <prefix-list>
                  <ip-prefix>192.0.2.0/24</ip-prefix>
                  <mask-length-lower>24</mask-length-lower>
                  <mask-length-upper>32</mask-length-upper>
                </prefix-list>
                <prefix-list>
                  <ip-prefix>198.51.100.0/24</ip-prefix>
```

```
                    <mask-length-lower>24</mask-length-lower>
                    <mask-length-upper>32</mask-length-upper>
                  </prefix-list>
                </prefixes>
              </prefix-set>
              <prefix-set>
                <name>prefix-set-B</name>
                <mode>ipv6</mode>
                  <prefixes>
                  <prefix-list>
                    <ip-prefix>2001:DB8::/32</ip-prefix>
                    <mask-length-lower>32</mask-length-lower>
                    <mask-length-upper>64</mask-length-upper>
                  </prefix-list>
                </prefixes>
              </prefix-set>
             </prefix-sets>
             <tag-sets>
              <tag-set>
               <name>cust-tag1</name>
               <tag-value>10</tag-value>
              </tag-set>
            </tag-sets>
          </defined-sets>

          <policy-definitions>
           <policy-definition>
             <name>export-tagged-BGP</name>
             <statements>
               <statement>
                 <name>term-0</name>
                 <conditions>
                   <match-prefix-set>
                     <prefix-set>prefix-set-A</prefix-set>
                   </match-prefix-set>
                   <match-tag-set>
                     <tag-set>cust-tag1</tag-set>
                   </match-tag-set>
                 </conditions>
                 <actions>
                   <policy-result>accept-route</policy-result>
                 </actions>
               </statement>
             </statements>
           </policy-definition>
         </policy-definitions>

         </routing-policy>
```

```
</config>
```

In the following example, all routes in the RIB that have been learned from OSPF advertisements corresponding to OSPF intra-area and inter-area route types should get advertised into ISIS level-2 advertisements.

```
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing-policy
   xmlns="urn:ietf:params:xml:ns:yang:ietf-routing-policy">
   <policy-definitions>
    <policy-definition>
     <name>export-all-OSPF-prefixes-into-ISIS-level-2</name>
      <statements>
       <statement>
         <name>term-0</name>
         <conditions>
           <match-route-type>ospf-internal-type</match-route-type>
         </conditions>
         <actions>
           <set-route-level>
             <route-level>isis-level-2</route-level>
           </set-route-level>
           <policy-result>accept-route</policy-result>
         </actions>
       </statement>
      </statements>
    </policy-definition>
   </policy-definitions>
  </routing-policy>
</config>
```

Authors' Addresses

Yingzhen Qu
Futurewei
2330 Central Expressway
Santa Clara CA 95050
USA

Email: yingzhen.qu@futurewei.com

Jeff Tantsura
Microsoft

Email: jefftant.ietf@gmail.com

Acee Lindem
Cisco
301 Midenhall Way
Cary, NC 27513
US

Email: acee@cisco.com

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

gwg
Internet Draft
Intended status: Informational
Expires: September 11, 2019

S. Wadhwa
K. DeSmedt
Nokia
R. Shinde
Reliance Jio
J. Newton
Vodafone
R. Hoffman
TELUS
P. Muley
Nokia
Subrat Pani
Juniper Networks
Mar 11, 2019

Architecture for Control and User Plane Separation on BNG
draft-wadhwa-rtgwg-bng-cups-03.txt

Status of this Memo

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This document discusses separation of subscriber-management control plane and data-plane for BNG. Traditionally, the BNG provides aggregation of fixed access nodes (such as DSLAM and OLTs) over Ethernet and provides subscriber management and traffic management functions for residential subscribers. The BNG has however evolved to become a multi-access edge device that also provides termination of subscribers over fixed-wireless and hybrid access. Therefore, this document proposes interfaces between control and user-plane of a BNG that can support multi-access BNG.

Table of Contents

1. Introduction

This document describes requirements and architecture for separation of subscriber management control plane and user plane for the BNG. In rest of the document the control plane is referred to as CP, user plane as UP, and the separation is referred to as CUPS (control and user plane separation). The draft describes the functional decomposition between CP and UP, and applicability of CUPS to a BNG that can support multiple access technologies such as fixed (DSL or Fiber), fixed-wireless (LTE,5G) and hybrid access i.e. simultaneous fixed and wireless access described in BBF [WT378]. The subsequent sections of the draft also define the interfaces required between CP and UP and briefly discusses a candidate base protocol for these interfaces.

1.1. Requirements Language

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

2. CUPS for BNG

In a CUPS architecture, signaling to setup subscriber sessions CP terminates signaling to setup subscriber sessions, and interfaces with the UP to create forwarding state for these sessions on the UP.

For fixed access subscribers, the CP terminates the signaling protocols (e.g. DHCP, PPPoE, SLAAC) from the customer premise, performs authorization/authentication with AAA Server, participates in address assignment, and then interfaces with the UP to create state related to forwarding and SLA management for the subscriber sessions on the UP. A subscriber session is a single IP connection, such as an IPoE or PPPoE session. The session can be single-stack (IPv4 or IPv6 only), or dual-stack (both IPv4 and IPv6). A CPE can

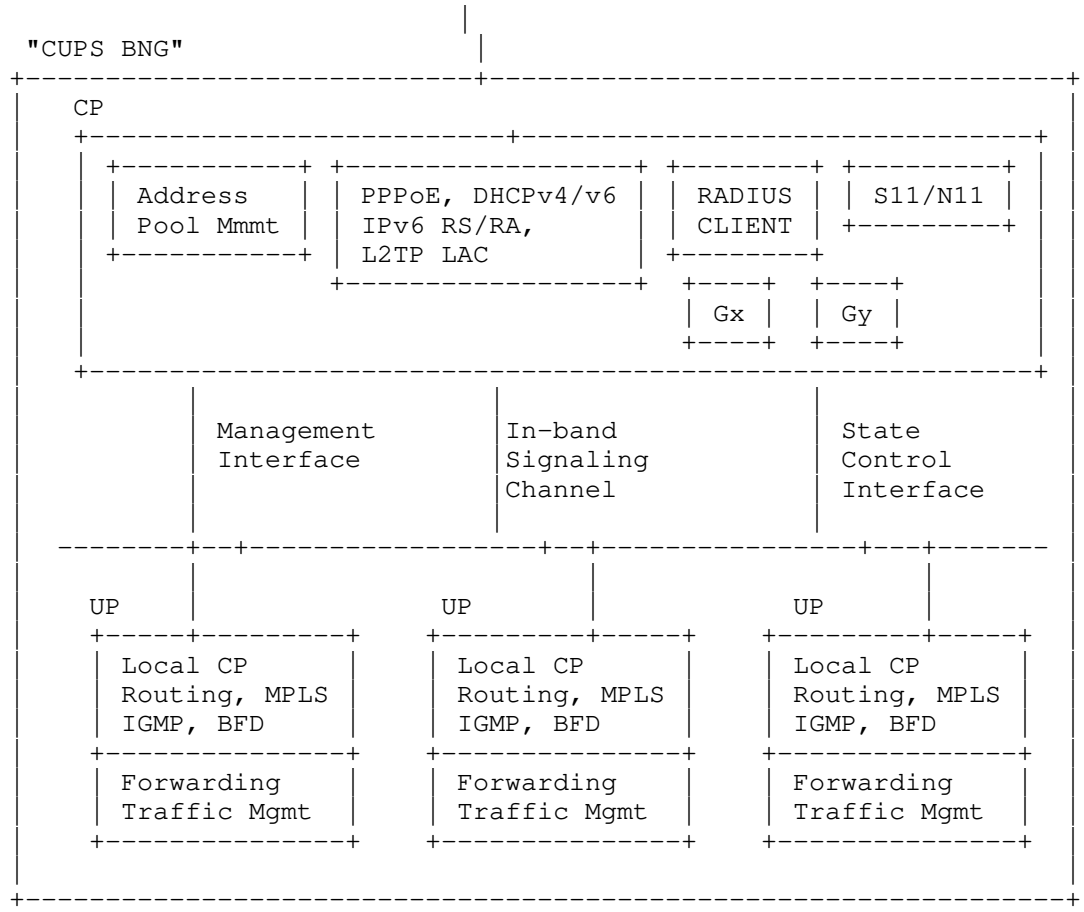
have multiple sessions, if multiple IP connections are required (e.g. on per service, or one per device behind the CPE).

The CP also processes solicited or unsolicited event notifications from the UP e.g. periodic accounting updates, usage reports, or session inactivity notifications. The interface between CP and UP that is used by the CP to manage session related forwarding state on the UP is being referred to as "state control interface". Asynchronous event notifications from UP to CP are also part of this interface.

In typical fixed access deployments, signaling (e.g. DHCPv4/v6, PPPoE, ICMPv6 RS/RA) to setup the subscriber sessions is in-band, and hence the UP receives the signaling messages from the customer premise. The UP should transparently forward (unmodified) in-band control messages as received from the customer premise to the CP and return messages from CP to the customer premise. Therefore, an in-band signaling channel is required between UP and CP. With a typical "CUPS BNG" deployment, the CP and UP are connected over a network, and the in-band signaling channel must be over a tunnel.

The UP performs forwarding and traffic management for the subscriber sessions. The infrastructure routing and signaling is done on the local control plane of the UP for fast convergence on network topology changes. In rest of the document the term "UP" is used generically for both functions performed by the local control plane on the UP and the data-plane.

A typical deployment architecture for CUPS includes a centralized CP running as a VNF interacting with multiple BNG UP instances that may be more distributed than the CP and could run as VNF or PNF. In this model, the CP and UP association is 1:N. This composite system containing CP VNF and one or more UP instances is referred to as a "CUPS BNG" in rest of the document. For operational ease, the CP MUST provide a single point for control and management for the entire "CUPS BNG". It MUST expose a single interface on behalf of the "CUPS BNG" to external systems such as AAA servers, OSS/BSS, Policy and charging servers. The CP VNF MUST support scale-out in order to cope with growth in number of subscriber sessions and/or increase in number of UP instances in the "CUPS BNG". Figure 1 below shows the functional components and interfaces for a "CUPS BNG".

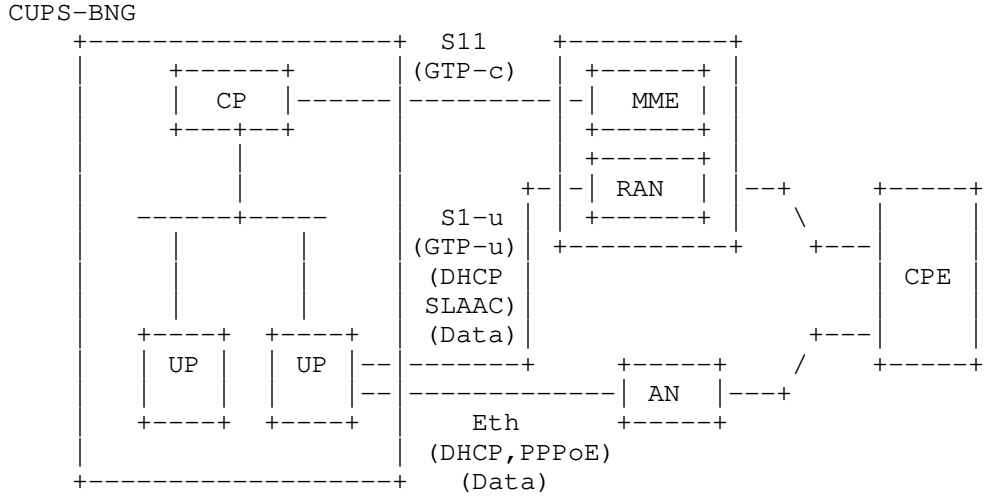


CUPS BNG System

2.1. Convergence

A single BNG can support subscribers over fixed, "fixed-wireless" or hybrid access. When a residential gateway has fixed-wireless access (LTE or 5G), then the BNG participates in 3GPP signaling with an MME

or AMF (i.e. support 3GPP S11 and N11 interfaces) to setup connections from (NG)RAN. With Hybrid access the customer premise initiates both fixed and wireless connections. The BNG in this case aggregates subscribers over Ethernet from fixed access nodes (DSLAMs and OLTs), but simultaneously terminates connections from (NG)RAN by participating in signaling with MME or AMF (S11/N11 interface). These deployment models are drivers for fixed-mobile convergence. It is important to ensure that the interfaces between CP and UP for CUPS can support not only fixed L2 access, but also the converged access scenario shown in Figure 2. One key requirement on the CP in these cases is the need to participate in 3GPP signaling (which is out-of-band) to setup the data-path. The data-path is a GTP-u (GPRS Tunneling protocol - User Plane) tunnel from the RAN (i.e. S1-u interface for LTE) as described in 3GPP [TS29281], and it terminates on the UP. It carries data traffic but also subscriber signaling messages (e.g. DHCPv4, DHCPv6, SLAAC) from the customer premise. The UP therefore still requires an in-band signaling channel to transport these protocol messages to the CP.



"CUPS BNG" with Converged Access

3. Interfaces for CUPS

A "CUPS BNG" MUST support the following interfaces between CP and UP, as shown in the figure in section 2.

3.1. In-band Signaling Channel

Section 2 describes the need for a signaling channel between CP and UP to transport in-band control messages between CP and the customer premise. Following are some key requirements for this interface.

- The UP MUST pass the access circuit identifier over which the signaling messages are received as meta-data to the CP. This includes port, VLAN tags, tunnel endpoint IPs, any tunnel identifiers such as GTP TEID, MPLS labels, L2TP tunnel-id etc. The UP MUST also pass the L2 or L3 transport service that the access circuit is associated with. In case the control message PDU is carried in an Ethernet frame, then the UP SHOULD pass the received Ethernet frame to the CP. Both access circuit identifier and information in the Ethernet header are required by the CP to construct successful response packet (control message) back towards the customer premise. The access circuit identifier MUST be reflected from CP to UP, so UP can identify the access circuit over which it needs to send the CP's response packet. In the control message sent from UP to CP, the UP MUST also include the local MAC address associated with access circuit. This is because certain control messages from the customer premise are destined to a broadcast MAC (e.g. DHCP DISCOVER) or multicast (e.g. ICMPv6 RS), so CP cannot infer the local MAC from these messages. Certain messages also require the local MAC address to be inserted in the message (e.g. Link-Layer address in ICMPv6 RA messages)

- The CP MUST be able to control the UP to forward only specific control messages to the CP.

- The CP MUST be able to control the UP to block certain control messages received on a particular access circuit.

- The CP MUST be able to control the UP to limit the rate of control messages (of specified type) to be sent by the UP.

- The CP MUST be able to prioritize reception of certain control messages over others in a granular manner (e.g. prioritize DHCP RENEWS over DISCOVERS or prioritize PPP Keepalive over other messages).

- The in-band signaling channel MUST support both fixed and converged access as described in section 2.1. The tunnel used for transporting these messages should therefore support both Ethernet and IP payloads.

3.2. State Control Interface

The CP and UP can exchange state at two levels using the "state control interface". One is at the node level and includes node-level information such as supported features, software releases, available resources, and operational state (e.g. active, failed, or overloaded). The other is at the subscriber session level. Subscriber session is described in section 2. The session level state includes basic forwarding and traffic management rules per session, that need to be provided by the CP to the UP in order to control per session forwarding and traffic management on the UP. It also includes state that triggers routing related actions on the UP. The session level state can include asynchronous event notifications from UP to CP, such as notifications to report per session usage (periodically or based on thresholds), notification to report session inactivity, and session liveness.

The interactions between CP and UP over "state control interface" can be categorized as:

- Session level state management
- Session level event notifications
- Node level management
- Node level event notifications

Following sub-sections provide more details on these interactions. The interactions between CP and UP over "state control interface" are modeled via abstract request/response messages between CP and UP. These messages will need to be defined as part of the protocol specification for this interface.

The protocol selected to implement this interface MUST support both fixed access and converged access (described in section 2.1) on BNG

3.2.1. Session level state management

Once the CP has successfully authorized and/or authenticated the subscriber session, and completed address assignment, it uses the "state control interface" to install forwarding and related state for the session on the forwarding path of the UP. This is abstracted as a "session create request" call from CP to UP, as shown in the figure below. The UP MUST ack or NACK via a response back to CP.

Since BNG can support different access types (e.g. fixed L2 access, or tunneled L3 in case of fixed-wireless, or a combination in case of hybrid access), it is important that the forwarding state information for the subscriber sessions, sent from CP to UP, can be specified as flexible packet matching rules and set of actions related to forwarding and traffic management. The UP should be able to use these match rules and actions to derive various lookup tables and processing in the forwarding path to forward traffic to and from the CPE.

The basic forwarding state in upstream direction (i.e. access to network) and downstream direction (i.e. network to access) fundamentally consists of session identification and one or more actions. Following shows a logical representation of a directive from CP to UP to install basic forwarding state on the UP for fixed L2 access (i.e. access from DSLAM or OLTs over Ethernet).

Direction Upstream - Access to Network:
Subscriber-identification: Port/VLAN-tag(s) + subscriber-MAC
Action: remove encapsulation, IP FIB lookup, forward to network.

Direction Downstream - Network to Access:
Subscriber-identification: IP address
Action: lookup IP DA, build encapsulation using Port/VLAN-tag(s)+ subscriber-MAC, forward to access.

Optionally, the IP address assigned to the CPE can also be provided for subscriber-identification (e.g. for anti-spoofing) in the upstream direction.
In case of PPPoE sessions, the subscriber-identification for upstream direction and encapsulation for downstream direction also includes the PPPoE session-id.

Based on the directive from CP to UP (as shown in the example above), the UP can then populate appropriate tables in the forwarding path, e.g. subscriber lookup tables, IP-FIB, and ARP or IPv6 Neighbor discovery table. It can also program the packet processing in both upstream and downstream direction based on the specified actions.

In case of "fixed-wireless" access, the access circuit is a GTP-u tunnel. In this case there is no physical interface (or port), and hence the CP MUST provide a tunnel definition to the UP to use as access circuit in upstream direction, and encapsulation in downstream direction. The tunnel definition will include the tunnel endpoint IP, and TEID that is established via out-of-band signaling

between the CP and the customer premise. It can also include the routing context for transporting the tunnel.

In addition to setting up the forwarding state as directed by the CP, the UP also needs to announce in routing the aggregate prefixes from which the CP assigns IPv4 and IPv6 addresses (or prefixes) to the CPEs. The CP SHOULD provide these aggregate prefixes to the UP as part session state. In case the aggregate prefixes are not provided, the UP MUST announce individual CPE addresses in routing, or it MAY try to aggregate in case addresses for multiple CPEs are from a contiguous address space.

The CPE can have a routed subnet behind it (aka framed-route). CP can learn the framed-routes during authentication/authorization. The CP should provide the framed-route to the UP as part of session state. The UP MUST install this route in the forwarding path and associate it with the forwarding state of the corresponding subscriber session. It should also announce this in routing towards the Network.

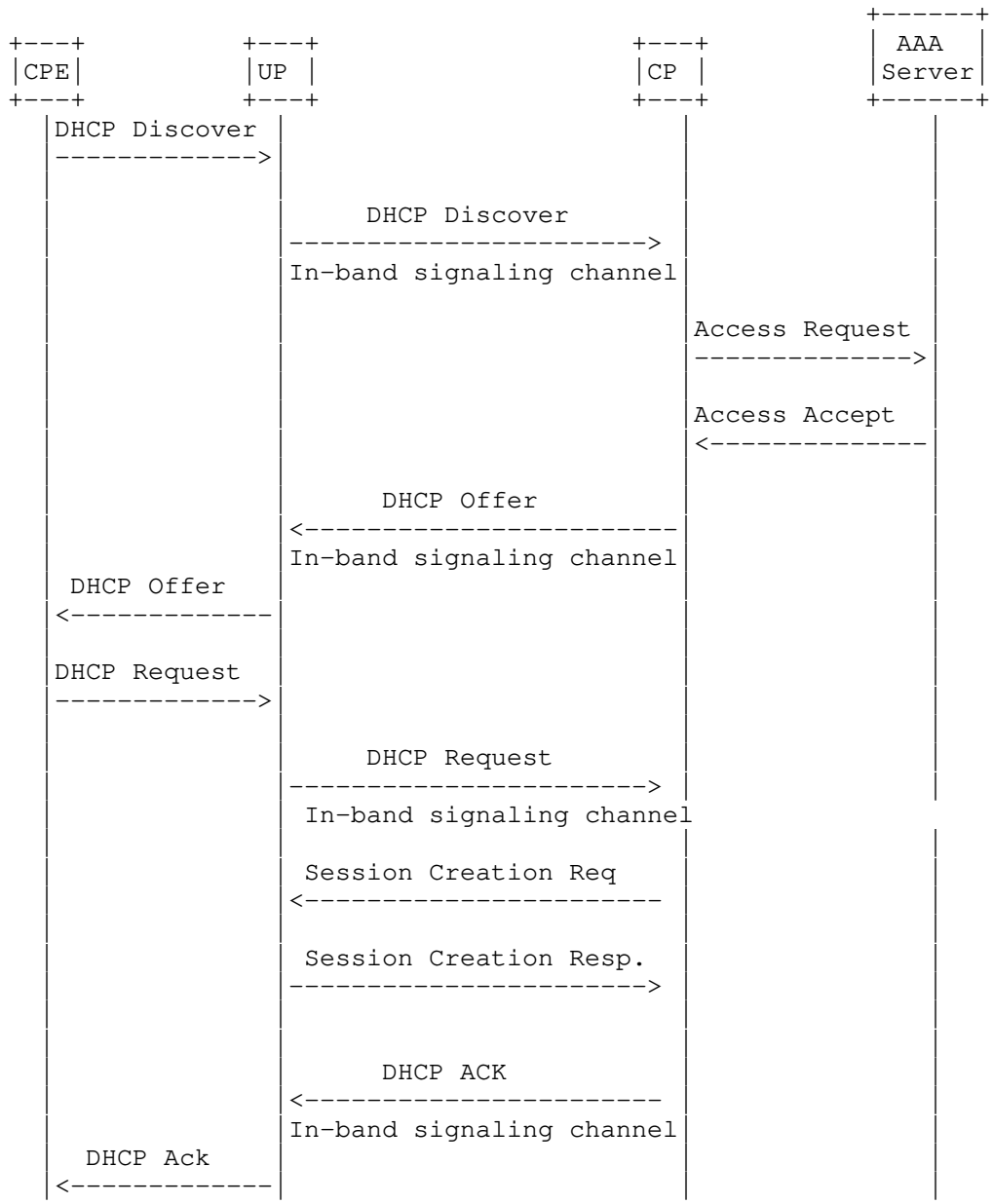
The CP MUST also provide to the UP the address assigned as IP gateway address to the CPEs in DHCP. The UP MUST locally configure this address appropriately, such that it can respond to ARP requests for this address from the CPEs.

The session sate on the UP is always controlled by the CP i.e. the UP just follows the directive from the CP to install, modify and delete the session state. In addition to the basic forwarding state, the CP can also associate, update and disassociate other related state with the session e.g. state related to:

- Filtering
- SLA management
- Statistics collection
- Credit control
- Traffic mirroring
- Traffic Steering
- NAT
- Application aware policies

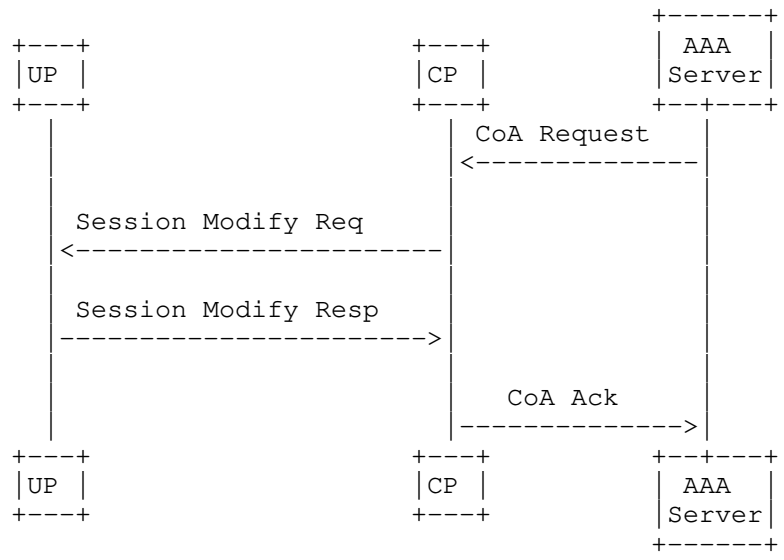
BNG deployments use hierarchical QoS (H-QOS) models which follows from a combination of link-layer over-subscription, multi-service networks and multiple layers of aggregation. For example, a common hierarchy exists of at least a QoS layer per access-node, and per

CPE. The CP MUST provide SLA management information to the UP per CPE. This includes applicable QoS parameters (e.g. rates, queues, markings) and the QoS hierarchy to which the CPE belongs. The CP may choose to signal this via a QoS policy that is locally pre-configured on the UP.



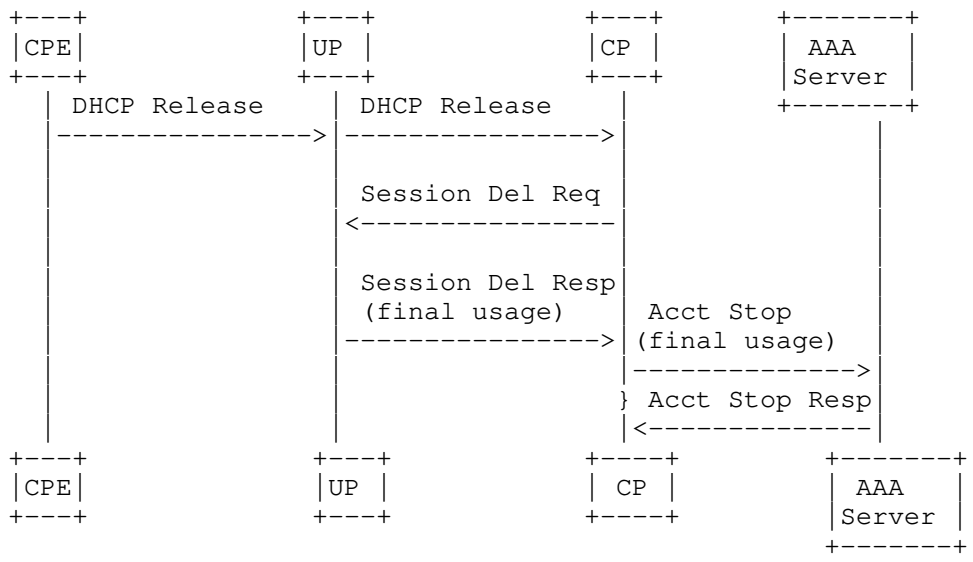
Session Creation Sequence

CP can trigger update of session state on the UP, triggered by re-authentication or COA from AAA or policy-server, as show in the figure below.



Session Modification

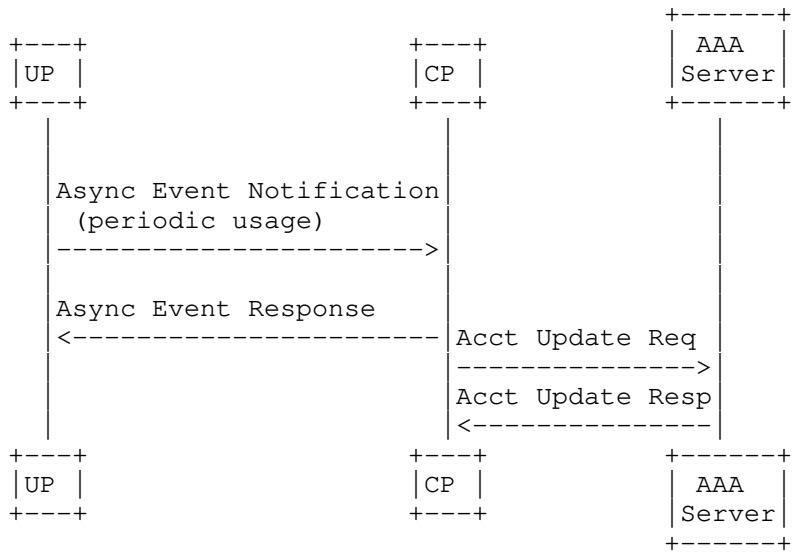
CP can trigger the deletion of session state based on signaling messages (as shown in the figure below), administrative action or disconnect-message initiated from the AAA server.



Session Deletion

3.2.2. Session level event notifications

UP can asynchronously generate Session level event notifications to the CP. An example of asynchronous notification is periodic usage reporting from UP to the CP, so that the CP can report the usage to a AAA server via interim accounting-updates. The CP can set the periodicity of this notification on the UP based on interim accounting interval configured by the operator on the CP.



Async Event Notification for periodic usage

Following are some other examples requiring asynchronous notifications from UP to CP.

- o Threshold based usage reporting
- o Inactivity timeout
- o Subscriber unreachability detection

The protocol for "state control interface" MUST support asynchronous notifications from UP to CP.

3.2.3. Node level management

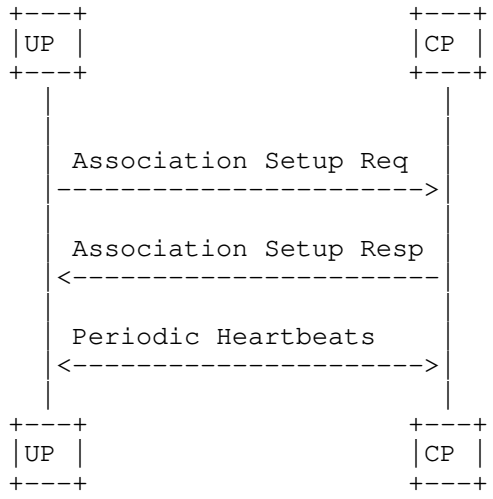
There needs to exist a concept of association between CP and UP. When the CP or UP comes online it should setup an association with configured or discovered peers via a message exchange. In association setup, the nodes should be able to exchange supported capabilities, version of software, load/overload information, and resource information. Also, any node-wide parameters can be exchanged during association setup.

No session state related messages should be accepted from the peer by either CP or UP unless an association exists.

Either node should be able to update the association to report changed feature capabilities, overload condition, resource exhaustion or any other node-wide parameters.

The UP should be able to request a graceful association release from the CP. In this case the CP should delete all sessions from that UP and process the final stats report for each session and send it in accounting-stop to the AAA server. During this process the CP MUST not create new sessions on the UP. Once all sessions are successfully deleted, the CP should release the association.

There needs to be a periodic node-level heartbeat exchange between CP and UP to detect if the peer is reachable and active. If peer is determined to be down based on heartbeat messages, then all the data-plane session state associated with the peer should be deleted.



Node Association Setup and Maintenance

3.2.4. Node level event notifications

There needs to be support for asynchronous node level event notifications from UP to CP. Example includes switchover

notification in case ports or UP failures when UP node level warm-standby redundancy is enabled. Based on this notification, the CP can create session state for all the sessions associated with the failure domain on the new primary UP.

3.3. Management Interface

The CP MUST provide a single point for local management of "CUPS BNG" system to the operator. This requires a management interface between CP and each of its associated UPs for pushing configuration to the UP and retrieving operational state from the UP. The interface MUST minimally include BNG specific configuration and state.

The Management interface SHOULD support transactional configuration from CP to UPs and SHOULD support state retrieval, both based on a well-defined data schema. The management interface SHOULD support unsolicited signaling of state changes (events) from UP to CP i.e. MUST provide telemetry for events. Either gNMI or NETCONF can be considered as acceptable candidates for model driven management interface.

4. Resiliency

"CUPS BNG" system MUST be protected against failure of CP VNF and MUST be able to recover the session state without operator intervention and reliance on CPEs. This can be achieved by providing redundancy for processing resources within CP VNF and maintaining redundant instance of session state.

Protection against UP failures based on 1:1 UP (hot-redundancy) and N:M (warm-redundancy) SHOULD be supported. For 1:1 hot-redundancy the CP needs to create data-plane state for sessions on both UPs that form a redundant pair, using the "state control interface". The CP needs to ensure the data-plane state for a session stays synchronized between the two nodes. A given session’s data-plane should only be active on one UP in the pair, which serves as active UP for the session. However, sessions that share the redundant UP pair can be distributed between the two UPs for active forwarding.

N:M warm-redundancy (N > M) can be supported via creation of data-plane state on the designated backup chassis after the failure has been detected. This would result in longer failover times than 1:1 hot-redundancy.

Redundant network connectivity between CP and UPs MUST be supported. In the "CUPS BNG" architecture, it is important to configure redundant connectivity that doesn't share fate.

5. Protocol Selection for CUPS Interfaces

It is important that the selected protocol for "state control interface" between CP and UP works not just for fixed access but also works for converged access on BNG. 3GPP has defined PFCP (Packet Forwarding Control Protocol) in [TS29244] as the interface between CP and UP for LTE gateways. This protocol is suited for large scale state management between CP and UP. Following are some of the key attributes of this protocol:

- o It supports management of forwarding and QOS enforcement state on the UP from CP. It also supports usage reporting from UP to CP.
- o It is over UDP transport and doesn't suffer from any HOL blocking.
- o It provides reliable operation based on request/response with message sequencing and retransmissions.
- o It provides an overload control procedure where overload on UP can be handled gracefully.
- o The protocol is extensible and allows addition of new IEs.

For fixed access BNG, the protocol requires simple extensions in form of additional IEs. The required extensions are mainly due to fact that typically a fixed access BNG requires tighter control over L2 behavior and manages access and subscriber using L2 identifiers (such as VLANs and MAC addresses), whereas mobile access works in terms of L3, either routed or tunneled.

The details of the protocol as applicable to the BNG and the required extensions will be defined in a separate draft.

[TS29244] also describes an in-band signaling channel based on GTP-u tunnel between CP and UP. GTP-u (GPRS Tunneling protocol - User Plane) is defined in 3GPP [TS29281] and defines a tunneling protocol which carries IP payloads. The protocol runs over a UDP/IP stack and uses UDP port number 2152. Data within a tunnel can be multiplexed based on Tunnel Endpoint Identifiers (TEIDs). The protocol supports optional sequence numbers. The protocol supports extension headers to allow development of new features. GTP-u tunnels are signaled between CP and UP, and it is possible

to associate filters to block certain control packets from being forwarded form UP to CP. The payload type carried by GTP-u can be extended to Ethernet (via payload type in extension header). The tunnel encapsulation can also be extended similarly to carry any required meta-data.

6. Address Pool Management

The CP MUST support management of IPv4 and IPv6 address pools, where each pool can contain one or more subnets. The pool management MUST support pool selection based on one or more of the following criteria:

- o UP
- o Access port on the UP.
- o Redundancy domain on the UP (e.g. set of access ports that share fate with respect to switchovers due to failures, when UP node level redundancy is enabled).
- o Service (e.g. HSI, VoIP, IPTV etc.).
- o Location (e.g. based on circuit-id/remote-id or part of circuit-id/remote-id in DHCP and PPPoE).

Pool management on CP SHOULD NOT statically link subnets to UPs but SHOULD dynamically allocate subnets to UP based on load i.e. on-demand, and signal allocated subnets using the "state control interface" as described in section 3.2.1. This allows for better IP resource utilization and less subnet fragmentation.

7. Security Considerations

For security between CP and UP, Network Domain Security (NDS) as defined in [TS33210] can be considered. As per NDS, the network can be split into security domains. Communication within a single security domain is considered secure, and protocols can operate without any additional security. When communication has to cross security domains, then IPSEC can be used.

8. IANA Considerations

None.

9. References

9.1. Normative References

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

[TS29244] 3GPP, "Interface between the Control Plane and the User Plane Nodes", TS 29.244 15.2.0, June 2018, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3111.

[TS29281] 3GPP, "General Packet Radio System (GPRS) Tunneling Protocol User Plane (GTPv1-U)", TS 29.281 15.3.0, June 2018, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1699.

[TS33210] 3GPP, "Network Domain Security (NDS); IP network layer security", TS 33.210 15.0.0, June 2018, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2279.

[WT378] BBF, "Nodal Requirements for Hybrid Access Broadband Networks", WT-378, 2018.

9.2. Informative References

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

[RFC2516] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D., and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)", RFC 2516, DOI 10.17487/RFC2516, February 1999, https://www.rfc-editor.org/info/rfc2516.

[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, https://www.rfc-editor.org/info/rfc3315.

Authors' Addresses

Sanjay Wadhwa
Nokia
777 East Middlefield Road
Mountain View
USA

Email: Sanjay.wadhwa@nokia.com

Killian De Smedt
Nokia
Copernicuslaan 50
Antwerp
Belgium

Email: Killian.de_smedt@nokia.com

Rajesh Shinde
Reliance Jio Infocomm Ltd.
Reliance Corporate Park
Thane Belapur Road, Ghansoli
Navi Mumbai 400710
India

Email: Rajesh.A.Shinde@ril.com

Jonathan Newton
Vodafone
Waterside House
Bracknell
United Kingdom

Email: jonathan.newton@vodafone.com

Ryan Hoffman
TELUS
1525 10th Ave SW
Calgary, Alberta
Canada

Email: ryan.hoffman@telus.com

Praveen Muley
Nokia
805. E. Middle Field Rd.
Mountain View, CA, 94043
USA

Email: praveen.muley@nokia.com

Subrat Pani
Juniper Networks

10 Technology Park Dr.
Westford, MA
USA

Email: spani@juniper.net

gwg
Internet Draft
Intended status: Informational
Expires: September 11, 2019

S. Wadhwa
K. DeSmedt
P. Muley
Nokia
R. Shinde
Reliance Jio
J. Newton
Vodafone
R. Hoffman
TELUS
S. Pani
Juniper Networks
March 11, 2019

Requirements for Protocol between Control and User Plane on BNG
draft-wadhwa-rtgwg-bng-cups-protocol-requirements-02.txt

Status of this Memo

Copyright Notice

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

Traditionally, the BNG provides aggregation of fixed access nodes (such as DSLAM and OLTs) over Ethernet and provides subscriber management and traffic management functions for residential subscribers. The BNG has however evolved to become a multi-access edge device that also provides termination of subscribers over fixed-wireless and hybrid access. An overall architecture and interfaces required between separated control and user-plane for a multi-access BNG are described in draft-wadhwa-rtgwg-bng-cups-01.txt. This document discusses requirements for protocol between subscriber-management control-plane and user-plane for BNG to achieve separation.

Contents

1. Introduction

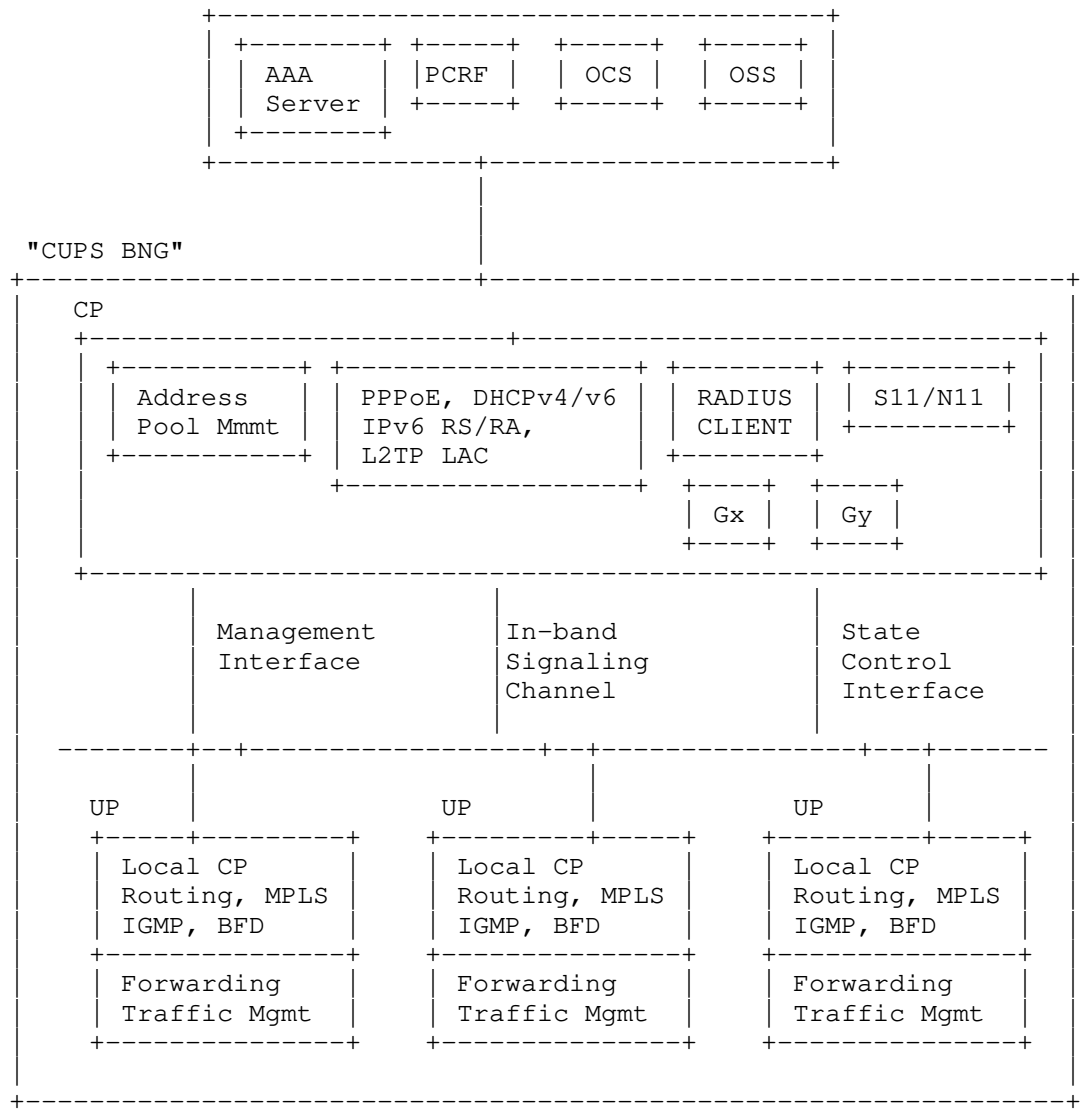
This document describes a set of requirements for protocol between subscriber-management control and user plane for BNG, that need to be met, in order to achieve separation. In rest of the document the control plane is referred to as CP, user plane as UP, and the separation is referred to as CUPS (control and user plane separation). The protocol between control and user-plane to achieve separation is referred to as "CUPS protocol". These requirements should form the basis for "CUPS protocol" selection. The functional decomposition between CP and UP, and applicability of CUPS to a BNG that can support multiple access technologies such as fixed (DSL or Fiber), fixed-wireless (LTE,5G) and hybrid access are described in [CUPS].

1.1. Requirements Language

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

2. Requirements for "CUPS protocol"

[CUPS] defines overall operation and architecture for control and user-plane separation on BNG. It also defines key functional interfaces between CP and UP, as shown in Fig 1, to realize the separation. "CUPS protocol" MUST provide support for information exchange to realize the "state control interface" and "in-band signaling channel" as defined in [CUPS].



CUPS BNG System

2.1. State Control Interface Requirements

. "CUPS protocol" MUST support convergence on BNG, where the CPEs terminating connections on the BNG can have fixed-access (e.g. xDSL/PON/Ethernet), fixed-wireless access (LTE/5G) or hybrid-access (i.e. combined fixed and wireless access).

. "CUPS protocol" MUST support messages and information exchange for node level management. There needs to exist a concept of association between CP and UP. When the CP or UP comes online it should setup an association with the configured or discovered peers via a message exchange. In association setup, the nodes should be able to exchange supported capabilities, version of software, load/overload information, and resource information. Also, any node-wide parameters can be exchanged during association setup.

. "CUPS protocol" MUST allow either node to update the association to report changed feature capabilities, overload condition, resource exhaustion or any other node-wide parameters.

. "CUPS protocol" MUST provide support for UP to request a graceful association release from the CP.

. "CUPS protocol" MUST support periodic node-level heartbeat exchange between CP and UP to detect if the peer is reachable and active.

. "CUPS protocol" MUST support exchange of messages and information elements (IEs) between CP and UP for session level state management on the UP.
A subscriber session is a single IP connection, such as an IPoE or PPPoE session. A CPE can have multiple sessions, if multiple IP connections are required (e.g. one per service, or one per device behind the CPE). The session level state on the UP, managed from the CP includes:

- Data-plane state for forwarding data traffic from subscriber sessions in upstream direction (access to network), and downstream direction (network to access).

- Forwarding state related to in-band control plane messages (such as messages for DHCP, PPPoE, SLAAC) that are forwarded

from CPE to CP via the UP (in upstream direction), and from CP to CPE via the UP (in downstream direction).

. In addition to the basic forwarding state, the "CUPS protocol" MUST support messages and information elements (IEs) for CP to associate, update and disassociate other data-plane related state with the session e.g. state related to:
  - o Filtering
  - o SLA management
  - o Statistics collection
  - o Credit control (usage monitoring and reporting)
  - o Traffic mirroring for legal intercept
  - o NAT
  - o Application (L4-L7) aware policies

. Depending on the type of access and the network between access-nodes and the BNG, the subscriber traffic from the CPEs can be encapsulated and transported over an L2 connection or over an L3 tunnel. Common scenarios for fixed access include Ethernet (q-in-q,.1q), L2oGRE, L2TPv3, VxLAN, and MPLS PW. For fixed-wireless the access is over a GTP tunnel (as defined in [CUPS]). The tunnel transport for L3 tunneled subscriber traffic can IPv4 or IPv6. The subscriber traffic itself can be IPv4, IPv6 or PPPoE. In case of PPPoE, the BNG can terminate PPPoE or tunnel it over L2TP to another gateway. The data-plane on the BNG decapsulates the upstream (access->network) traffic and routes it towards the network in appropriate routing-context, and optionally perform NAT before routing. It determines the subscriber for downstream (network->access) IP traffic, encapsulates it appropriately before forwarding towards the access. In addition, it does traffic-management and SLA management, maintains traffic statistics and optionally monitors and reports usage. The "CUPS protocol" MUST be able to carry state from CP to UP for IPv4, IPv6 and PPPoE sessions, for various flavors of transport connections mentioned above.

. Given the variety of access types on the CPE and type of transport networks between access-nodes and BNG (as outlined above) , the "CUPS protocol" MUST specify forwarding state information for the subscriber sessions, for both data and in-band control, as flexible packet matching rules and set of actions related to forwarding and traffic management, rather than just fixed-format lookup tables understood by particular UP implementation. Using the flexible match rules and actions conveyed in the "CUPS protocol" IEs, the UP should unambiguously be able to derive

various lookup tables and processing in the forwarding path to forward traffic to and from the CPE. The basic forwarding state in upstream direction (i.e. access to network) and downstream direction (i.e. network to access) fundamentally consists of session identification and one or more actions. Following shows a logical representation of a directive from CP to UP to install basic forwarding state on the UP for fixed L2 access (i.e. access from DSLAM or OLTs over Ethernet).

- Direction Upstream – Access to Network:
  - Subscriber-session identification: Port/VLAN-tag(s) + subscriber-MAC + Session IP address + PPPoE Session-ID
  - Action: remove encapsulation (i.e. Ethernet and PPPoE/PPP headers), apply policer, do IP FIB lookup, forward to network.

- Direction Downstream – Network to Access:
  - Subscriber-session identification: IP address
  - Action: apply subscriber-shaper, build encapsulation using (PPPoE session-id and Port/VLAN-tag(s)+ subscriber-MAC), forward to access.

Examples of actions and processing related to forwarding and traffic management include encapsulation/decapsulation, table lookups, drop, forward, mirror, count, redirect, police, classify, queue, shape etc.

. In addition to packet-matching rules and actions to setup data-path on the UP, the "CUPS protocol" MUST allow CP to specify subscriber routing and IP interface related information. This includes the following:

- Aggregate IPv4 subnets and IPv6 prefixes that are used for assigning addresses or prefixes (e.g. IPv6 delegated-prefix) to subscribers on a UP. These are announced in routing by the UP to draw downstream traffic.
- UE’s IP address and subnet mask.
- Default gateway IP address within the subscriber subnets. This is used to draw upstream traffic from the CPEs and the UP is required to respond to ICMP requests for this address from the CPEs.
- Subnets for network behind a CPE (also known as framed-routes).

. The "CUPS protocol" MUST provide support for CP to specify session level HQOS related information to the UP. A common QOS hierarchy on BNG consists of at least a QoS layer per access-node, and per CPE. "CUPS protocol" MUST provide support for CP to specify QoS parameters (e.g. rates, queues, markings) and the QoS hierarchy to which the CPE belongs, to the UP. The CP may choose to signal this via a QoS policy that is locally pre-configured on the UP. "CUPS protocol" MUST provide support for CP to specify HQOS-policy that the session is associated with.

. "CUPS protocol" MUST support asynchronous session level event notifications from UP to CP. Session level asynchronous notifications include:

  - Periodic usage-reports
  - Threshold based usage-reports
  - Inactivity timeout
  - Subscriber unreachability detection

. "CUPS protocol" MUST support asynchronous node level event notifications from UP to CP. Example includes switchover notification in case ports or UP failures when node level redundancy is enabled.

2.2. Extensibility

. "CUPS protocol" MUST support exchange of software version and feature capabilities when a node level association is setup between a CP and UP.

. "CUPS protocol" MUST encode information in messages as TLVs.

. "CUPS protocol" MUST allow extension to defined Information Elements (IEs) i.e. it MUST allow adding new information to existing IEs while maintaining backwards compatibility.

. "CUPS protocol" MUST allow addition of new IEs exchanged in protocol messages.

. "CUPS protocol" MUST support vendor specific IEs (modelled as TLVs) by carving out TLV space for vendor specific extensions.

. "CUPS protocol" processing on UP MUST support graceful handling when an unknown TLV is received. The UP MUST ignore unknown TLV and continue with normal message processing. This ensures the CP MAY send non-mandatory TLVs to the UP. However, CP MUST only send mandatory TLVs if it knows the UP will accept it (based on local configuration or based on capability exchange during association setup). A TLV is considered mandatory if session state cannot be installed or updated without it.

2.3. Scalability and Performance

. A single CP VNF can control multiple UP nodes. Each UP can support its maximum scale of subscriber sessions as allowed by its data-plane. External control plane running as a VNF can horizontally scale-out as needed with the growth in CUPS system-wide subscriber scale. In typical deployments CP may be centralized whereas the UPs may be distributed, with multiple L2 or L3 hops between CP and UPs. There are scenarios where a large number of sessions may be getting created or deleted close in time via "CUPS protocol". It is important that latency to bring subscribers online is minimized. The transport protocol chosen for "CUPS protocol" MUST NOT suffer from head-of-line (HOL) blocking where transport of messages related to one subscriber can be adversely impacted by messages being exchanged for other subscribers.

. "CUPS protocol" MUST limit chattiness by minimizing number of messages required to create fully functional subscriber on the UP with complete forwarding, traffic management, HQOS, and routing state. Ideally, a single request/response message exchange between CP and UP should be able to create subscriber with all the required state in the data-plane. The "CUPS protocol" message that creates the subscriber session MUST therefore be able to signal IEs for all the required subscriber state.

. To further reduce latency the protocol MUST be binary encoded.

. "CUPS protocol" MUST allow dynamic scale-out for control plane VNF with the growth in subscriber scale of the CUPS system, as more UPs are added to the CUPS system or more ports are enabled on a UP in a CUPS system.

. The "CUPS Protocol" MUST allow mechanism to provide balancing of processing load amongst compute resources of control-plane VNF that supports dynamic scale-out.

. "CUPS protocol" SHOULD support signaling of overload state and optionally overload mitigation parameters from UP to CP, when UP determines the incoming signaling from CP is exceeding (or about to exceed) its nominal processing capacity. Overload mitigation can include a temporary message throttling on CP towards UP. Mitigation parameters can include message rate and validity time for the specified rate.

2.4. Transport Protocol

. As mentioned in section 2.3, the transport protocol used for "CUPS protocol" MUST NOT suffer from HOL blocking. Therefore, TCP is not an option for the transport protocol.

. Ideally, the transport protocol SHOULD preserve message boundary with datagram semantics and should be available or easily implementable on any simple forwarding devices. Therefore, UDP is the preferred option.

. "CUPS protocol" MUST therefore support reliability and ordering for exchanged messages. The reliability and ordering can be based on request/response with message sequencing and re-transmissions.

2.5. In-band Control Channel Requirements

. "CUPS protocol" MUST support setting up of control channel between UP and CP for transporting in-band control messages (e.g. DHCPv4/v6 and PPPoE) received on the UP (from CPEs) to the CP, and for return messages sent from CP to the UP (destined to CPEs).

. There can be a L3 network between CP and UPs. Therefore, L3 tunneling is required between CP and UP to carry messages for in-band control plane protocols. "CUPS protocol" MUST support exchange of tunnel identifiers between CP and UP.

. Because L2 access setup is in-band, control plane messages will arrive on the UP before any per-session state is learned. Therefore, "CUPS protocol" MUST support messages and information exchange to install forwarding state related to in-band control plane messages that do not match any existing subscriber session. These messages should be forwarded to the CP over a common default control channel.

. The in-band control channel setup by "CUPS protocol" MUST have support for UP to pass access-circuit identifier over which the signaling messages are received from the CPEs. Based on type of access, access-circuit identifier can include port/VLAN tags or tunnel identifiers which includes tunnel endpoint IPs and de-multiplexers such as GTP TEID, MPLS labels, L2TP tunnel-id etc. "CUPS protocol" MUST support setting up logically separate control channels for in-band control messages per access-circuit.

. In case of fixed-access CPEs with Ethernet based network between access-nodes and BNG, the control messages are received in Ethernet frames. The Ethernet frame carrying the control messages received on UP MUST be carried over the control channel to the CP, as outlined in [CUPS]. In case of fixed-wireless access, control messages (e.g. DHCPv4 and DHCPv6) are received on the UP over GTP-u tunnel from the RAN. The GTP-u tunnel directly carries IP payload. Therefore, control channel setup via "CUPS protocol" MUST support transporting both Ethernet and IP payloads.

. "CUPS protocol" MUST provide support for CP to specify the control protocols that should be forwarded by the UP over in-band control channel to the CP.

. The "CUPS protocol" SHOULD have support for CP to specify rate-limits for specific control protocols and optionally specific messages within a control protocol, that the UP should enforce.

. The "CUPS protocol" SHOULD provide support for CP to direct the UP to drop certain control messages received on a particular access-circuit.

. The "CUPS protocol" SHOULD provide support for CP to prioritize reception of certain control messages over others.

2.6. Resiliency

. "CUPS protocol" MUST allow support for both 1:1 (hot standby) and N:M (warm standby) UP node level redundancy.

. "CUPS protocol" MUST provide support for CP to specify the "redundancy domain" that a subscriber session is associated with during session level state creation on the UP. The "redundancy domain" is set of resources that share fate with respect to switchover on failure, e.g. a set of VLANs on a port, or a set of ports on a UP, or entire UP. "CUPS protocol" MUST also provide support for CP to provide relevant parameters to UP about the "redundancy domains". The UPs can then locally preform failure detection and switchover for the redundancy domains.

. The "CUPS protocol" MUST provide support for UP to notify the CP about switchover event. This notification must be on the granularity of "redundancy domain" on a UP.

. For warm standby redundancy, "CUPS protocol" MUST provide support for CP to create session level state on the backup UP node(s) for all subscribers associated with the impacted "redundancy domain".

. "CUPS protocol" MUST support in-service software upgrade (ISSU) on UPs. The protocol MUST provide support for UP to notify CP when it is completed ISSU to the new software release.

2.7. Security

"CUPS protocol" MUST be compatible with proven security mechanisms such as IPSEC or DTLS to satisfy following security requirements:

- Data-integrity and confidentiality MUST be ensured for the information exchanged via "CUPS protocol".
- Protection against man-in-the-middle attacks MUST be provided.
- Anti-replay protection MUST be provided.

3. "CUPS protocol" candidate

3GPP has defined PFCP (Packet Forwarding Control Protocol) in [TS29244] as the interface between CP and UP for LTE gateways. This protocol is suited for large scale state management between CP and UP and can be extended for BNG providing converged access. The protocol provides a good base for satisfying the requirements outlined in this draft for BNG "CUPS protocol". Following are some of the key attributes of this protocol/

- It supports management of forwarding and QOS enforcement state on the UP from CP.
- It also supports usage reporting from UP to CP.
- It is over UDP transport and doesn’t suffer from any HOL blocking.
- It provides reliable operation based on request/response with message sequencing and retransmissions.
- It provides support for graceful handling of overload on UP.
- The protocol is extensible and allows addition of new IEs.
- For fixed access BNG, the protocol requires simple extensions in the form of additional IEs. The required extensions are mainly due to fact that typically a fixed access BNG requires tighter control over L2 behavior and manages access and subscriber using L2 identifiers (such as VLANs and MAC

addresses), whereas mobile access works in terms of L3, either routed or tunneled.

. [TS29244] also describes an in-band signaling channel based on GTP-u tunnel between CP and UP. GTP-u (GPRS Tunneling protocol User Plane) is defined in 3GPP [TS29281] and defines a tunneling protocol which carries IP payloads. The protocol runs over a UDP/IP stack and uses UDP port number 2152. Data within a tunnel can be multiplexed based on Tunnel Endpoint Identifiers (TEIDs). The protocol supports optional sequence numbers. The protocol supports extension headers to allow development of new features. GTP-u tunnels are signaled between CP and UP, and it is possible to associate filters to forward or block certain control packets from UP to CP. The payload type carried by GTP-u can be extended to Ethernet (via payload type in extension header). The tunnel encapsulation can also be extended by introducing an additional NSH (network services header) to carry any required meta-data.

4. Security Considerations

For security between CP and UP, Network Domain Security (NDS) as defined in [TS33210] can be considered. As per NDS, the network can be split into security domains. Communication within a single security domain is considered secure, and protocols can operate without any additional security. When communication has to cross security domains, then IPSEC can be used.

5. Management Interface Requirements

. The CP MUST provide a single point for management of "CUPS BNG" system to the operator.

. Management interface for the CUPS system MUST provide support for both configuration of UPs, and state retrieval. The interface MUST minimally support BNG specific configuration and state.

. Management interface SHOULD support transactional configuration from CP to UPs, based on a well-defined data schema. Transactional configuration may be achieved by editing a candidate configuration on the UP which is subsequently activated (commit) or by providing the whole transaction in a

single command. In case UP data-stores are used, it MUST be possible for the CP to lock a data-store for exclusive access.

- The management interface SHOULD support transaction confirmation, where an unconfirmed transaction gets reverted automatically after a timeout even if the transaction succeeded. This is to avoid configuration errors where a valid configuration breaks communication between UP and CP, requiring on-site intervention.

- The management interface SHOULD support state retrieval based on a well-defined data schema. This includes retrieval for any state that is not signaled via the state control interface.

- The management interface SHOULD support unsolicited signaling of state changes (events) from UP to CP i.e. SHOULD provide telemetry for events. Even while state changes are sent unsolicited, the CP SHOULD be able to subscribe to a specific subset of state it is interested in.

- The management interface MUST provide security through an existing mechanism such as (D)TLS or IPSEC to guarantee confidentiality and authenticity and protect against replay and man in the middle attacks.

6. IANA Considerations

None.

7. References

7.1. Normative References

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

[CUPS] Wadhwa, S. et al., "Architecture for control and user plane separation on BNG, July 2019. https://datatracker.ietf.org/doc/draft-wadhwa-rtgwg-bng-cups/

[TS29244] 3GPP, "Interface between the Control Plane and the User Plane Nodes", TS 29.244 15.2.0, June 2018, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3111.

[TS29281] 3GPP, "General Packet Radio System (GPRS) Tunneling Protocol User Plane (GTPv1-U)", TS 29.281 15.3.0, June 2018, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1699.

[TS33210] 3GPP, "Network Domain Security (NDS); IP network layer security", TS 33.210 15.0.0, June 2018, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2279.

7.2. Informative References

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

[RFC2516] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D., and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)", RFC 2516, DOI 10.17487/RFC2516, February 1999, https://www.rfc-editor.org/info/rfc2516.

[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, https://www.rfc-editor.org/info/rfc3315.

Authors' Addresses

Sanjay Wadhwa
Nokia
777 East Middlefield Road
Mountain View
USA

Email: Sanjay.wadhwa@nokia.com

Killian De Smedt
Nokia
Copernicuslaan 50
Antwerp
Belgium

Email: Killian.de_smedt@nokia.com

Praveen Muley
Nokia
805. E. Middle Field Rd.
Mountain View, CA, 94043
USA

Email: praveen.muley@nokia.com

Rajesh Shinde
Reliance Jio Infocomm Ltd.
Reliance Corporate Park
Thane Belapur Road, Ghansoli
Navi Mumbai 400710
India

Email: Rajesh.A.Shinde@ril.com

Jonathan Newton
Vodafone
Waterside House
Bracknell
United Kingdom

Email: jonathan.newton@vodafone.com

Ryan Hoffman
TELUS
1525 10th Ave SW
Calgary, Alberta
Canada

Email: ryan.hoffman@telus.com

Subrat Pani
Juniper Networks
10 Technology Park Dr.
Westford, MA
USA

Email: spani@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

F. Xu
Tencent
S. Zhuang
Y. Gu
Huawei

October 22, 2018

Real-Time Monitoring Link/Protocol Neighbor State
draft-xu-rtgwg-monitoring-neighbor-state-00

Abstract

Various protocols are deployed in today's networks, such as BGP / ISIS / OSPF etc. Link neighbor state changes and protocol neighbor state changes are the most important network events that need to be processed with the highest priority. In particular, the SDN controller needs to quickly sense the link neighbor & protocol neighbor state change information in the network. Thus, the various policies applied by the SDN controller to the network can quickly match the current state of the network. This document discusses some possible scenarios and the relevant 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 in RFC 2119 [RFC2119].

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at 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 April 25, 2019.

Table of Contents

1. Introduction

Various protocols are deployed in today's networks, such as BGP / ISIS / OSPF / LDP / BFD etc. When managing a network, one of the most important things to monitor is changes to the various protocols' neighbor states. Many times a protocol neighbor state change is indicative of a problem on the network, and it is an important basis for the SND controller to deploy the traffic steering policies. There are several ways to monitor these state changes, e.g. we can use command-line interface (CLI) to get them from the devices, but typically it's done with either SNMP based polling and/or SNMP traps. For BGP, we can use BMP (BGP Monitoring Protocol) [RFC7854] to collect BGP neighbor state change information.

Link neighbor state changes and protocol neighbor state changes are the most important network events that need to be processed with the highest priority. In particular, the SDN controller needs to quickly sense the link neighbor & protocol neighbor state change information in the network. Thus, the various policies applied by the SDN controller to the network can quickly match the current state of the network.

The problem of the current real-time data collection method: Collecting protocol neighbor state will also collect many other large amounts of information data associated with it and have a significant impact on the reception/processing of high priority protocol neighbor state data. E.g., the processing of the BMP Peer Up/Down message is not real-time, and is affected by the receiving and processing of other BMP messages, especially a large number of route monitoring messages.

At present, the SDN controller uses a single channel to receive real-time data from the network, and then classifies the data and processes it in order, which causes the delay of the neighbor state information processing to grow; and the neighbor information data structure of different protocols are different; these cases will increase the delay in which the SDN controller processes neighbor state data.

2. Requirements and Options

Summary of requirements are as follows:

Requirement 1: Network event prioritization, Set Link neighbor state changes and protocol neighbor state changes as the most important network events.

Requirement 2: The structure of the link neighbor & protocol neighbor state change information needs to use a normalized format, such as a unified TLV.

Requirement 3: Link Neighbor & Protocol Neighbor State Change Information requires a separate transport channel to be separated from other low priority data.

Requirement 4: SDN controller implements the convergence mechanism of the existing network protocol in milliseconds/second.

Some options to be discussed:

1) A new Neighbor State Monitoring Protocol.

2) Consider processing all neighbor states into LS information, flooding them through IGP, and collecting them on the controller through BGP-LS.

3) Consider processing all neighbor states into LS information, imorting them to the BGP-LS Local-RIB, and collecting them on the controller through BMP.

4) gRPC + YANG Model.

5) To be added...

3. Neighbor State Information Format

At present, the neighbor information data structure of different protocols are different. In order to speed up processing in controller or collector, this document proposes to use a normalized format as following:

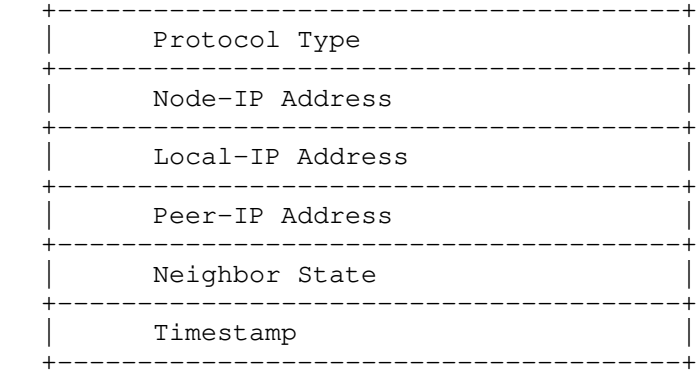


Figure 1 Link Neighbor/Protocol Neighbor State Information Format

Where:

Protocol Type: 1: ISIS / 2: OSPF / 3: BGP / 4: LDP / 5: BFD etc.

Node-IP Address: The IP Address of the monitored node, usually the router ID.

Local-IP Address: Local-IP Address of the Neighbor

Peer-IP Address: Peer-IP Address of the Neighbor

Neighbor State: The state of the Neighbor

Timestamp: The timestamp of the moment of the event

4. Acknowledgements

The authors would like to thank Haibo Wang, Zhongjia Wang for their contributions to this work.

5. Contributors

Mach Chen
mach.chen@huawei.com

6. IANA Considerations

TBD.

7. Security Considerations

TBD.

8. 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>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, DOI 10.17487/RFC5492, February 2009, <https://www.rfc-editor.org/info/rfc5492>.

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <https://www.rfc-editor.org/info/rfc7854>.

Authors' Addresses

Feng Xu
Tencent
Guangzhou
China

Email: oliverxu@tencent.com

Shunwan Zhuang
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com

Yunan Gu
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: guyunan@huawei.com