Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2021

A. Choudhary
Cisco Systems
I. Chen
The MITRE Corporation

October 22, 2020

YANG Model for QoS Operational Parameters
draft-asechoud-rtgwg-qos-oper-model-08

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 25, 2021.

Table of Contents

1. Introduction

This document defines a base YANG [RFC6020] [RFC7950] data module for Quality of Service (QoS) operational parameters. Remote Procedure Calls (RPC) or notification definition is currently not part of this document and will be added later if necessary. QoS configuration modules are defined by [I-D.ietf-rtgwg-qos-model].

This document doesn't include operational parameters for random-detect (RED), which is left to individual vendor to augment it.

Editorial Note: (To be removed by RFC Editor)

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

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

1.1. Tree Diagrams

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

2. Terminology

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

3. QoS Operational Model Design

QoS operational model include QoS policy applied to an interface in each direction of traffic. For each QoS policy applied to an interface the model further includes counters for associated Classifiers, Meters and Queues in a particular direction. To modularize and for reusability, grouping have been defined for various counters of classifier, Meters and Queues. The target is assumed to be interface but the groupings can be used for any other target type where QoS policy is applied.

[I-D.ietf-rtgwg-qos-model] defines various building blocks for applying a QoS Policy on a target. It includes QoS Policy configuration, which is a container of various classifiers and corresponding actions which are configured for traffic conditioning. This drafts defines the various counters for these building blocks. ietf-qos-oper module defined in this draft augments ietf-interfaces [RFC8343] module.

Classifier statistics contains counters for packets and bytes matched to the traffic in a direction and also average rate at which traffic is hitting a classifier. Classification criterion may be based on IP, MPLS or Ethernet. Counters defined in this draft are agnostic to underlying data plane technology.

Statistics of meter is modeled based on commonly used algorithms in industry, Single Rate Tri Color Marking (srTCM) [RFC2697] meter, Two Rate Tri Color Marking (trTCM) [RFC2698] meter. Metering statistics includes counters corresponding to various rates configured. A metering container is referred by a metering identifier. This identifier could be a classifier name if the metering configuration is inline with classifier or it could be metering template name if the metering is configured as separate entity and associated with the classifier.

Queuing statistics includes counters corresponding to various queues associated with the policy. A queuing container is referred by queuing identifier. This identifier could be a classifier name if the queuing configuration is inline with classifier and hence there is one-to-one mapping between a classifier and a queue or it could be

a separate queue identifier if one or more than one classifiers are associated with a queue.

4. Modules Tree Structure

This document defines counters for classifiers, meters and queues.

Classifier statistics consists of list of classifier entries identified by a classifier entry name. Classifier counters include matched packets, bytes and average rate of traffic matching a particular classifier.

Metering statistics consists of meters identified by an identifier. Metering counters include conform, exceed, violate and drop packets and bytes.

Queuing counters include instantaneous, peak, average queue length, as well as output conform, exceed, tail drop packets and bytes.

Named statistics is defined as statistics which are tagged by a name. This could be aggregated or non-aggregated. Aggregated named statistics is defined as counters which are aggregated across classifier entries in a policy applied to an interface in a particular direction. Non-aggregated named statistics are counters of classifier, metering or queuing which have the same tag name but maintained separately.

```
module: ietf-qos-oper
  augment /if:interfaces/if:interface:
    +--ro qos-interface-statistics
       +--ro stats-per-direction* []
          +--ro direction?               identityref
          +--ro policy-name?             string
          +--ro classifier-statistics* []
          |  +--ro classifier-entry-name?   string
          |  +--ro classified-pkts?         uint64
          |  +--ro classified-bytes?        uint64
          |  +--ro classified-rate?         uint64
          +--ro named-statistics* []
          |  +--ro stats-name?       string
          |  +--ro aggregated
          |  |  +--ro pkts?    uint64
          |  |  +--ro bytes?   uint64
          |  |  +--ro rate?    uint64
          |  +--ro non-aggregated
          |     +--ro classifier-statistics* []
          |     |  +--ro classifier-entry-name?   string
```

```
|      |  +--ro classified-pkts?         uint64
|      |  +--ro classified-bytes?        uint64
|      |  +--ro classified-rate?         uint64
|      +--ro metering-statistics* []
|      |  +--ro meter-id?           string
|      |  +--ro conform-pkts?       uint64
|      |  +--ro conform-bytes?      uint64
|      |  +--ro conform-rate?       uint64
|      |  +--ro exceed-pkts?        uint64
|      |  +--ro exceed-bytes?       uint64
|      |  +--ro exceed-rate?        uint64
|      |  +--ro violate-pkts?       uint64
|      |  +--ro violate-bytes?      uint64
|      |  +--ro violate-rate?       uint64
|      |  +--ro meter-drop-pkts?    uint64
|      |  +--ro meter-drop-bytes?   uint64
|      +--ro queueing-statistics* []
|         +--ro queue-id?                    string
|         +--ro output-conform-pkts?         uint64
|         +--ro output-conform-bytes?        uint64
|         +--ro output-exceed-pkts?          uint64
|         +--ro output-exceed-bytes?         uint64
|         +--ro queue-current-size-bytes?    uint64
|         +--ro queue-average-size-bytes?    uint64
|         +--ro queue-peak-size-bytes?       uint64
|         +--ro tailed-drop-pkts?            uint64
|         +--ro tailed-drop-bytes?           uint64
+--ro metering-statistics* []
|  +--ro meter-id?           string
|  +--ro conform-pkts?       uint64
|  +--ro conform-bytes?      uint64
|  +--ro conform-rate?       uint64
|  +--ro exceed-pkts?        uint64
|  +--ro exceed-bytes?       uint64
|  +--ro exceed-rate?        uint64
|  +--ro violate-pkts?       uint64
|  +--ro violate-bytes?      uint64
|  +--ro violate-rate?       uint64
|  +--ro meter-drop-pkts?    uint64
|  +--ro meter-drop-bytes?   uint64
+--ro queueing-statistics* []
   +--ro queue-id?                    string
   +--ro output-conform-pkts?         uint64
   +--ro output-conform-bytes?        uint64
   +--ro output-exceed-pkts?          uint64
   +--ro output-exceed-bytes?         uint64
   +--ro queue-current-size-bytes?    uint64
   +--ro queue-average-size-bytes?    uint64
```

```
               +--ro queue-peak-size-bytes?      uint64
               +--ro tailed-drop-pkts?           uint64
               +--ro tailed-drop-bytes?          uint64
```

5. Modules

5.1. IETF-QOS-OPER

<CODE BEGINS>file "ietf-qos-oper.yang"

```
module ietf-qos-oper {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-qos-oper";
  prefix oper;
  import ietf-interfaces {
    prefix if;
    reference
      "RFC8343: A YANG Data Model for Interface Management";
  }
  organization "IETF RTG (Routing Area) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>
     Editor:   Aseem Choudhary
               <mailto:asechoud@cisco.com>";
  description
    "This module contains a collection of YANG definitions for
     qos operational specification.
     Copyright (c) 2019 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.
     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).
     This version of this YANG module is part of RFC XXXX; see
     the RFC itself for full legal notices.";
  revision 2020-10-22 {
    description
      "Latest revision for qos operational statistics";
    reference
      "RFC XXXX: YANG Model for QOS Operational Parameters";
  }
  identity direction {
    description
      "This is identity of traffic direction";
```

```
    }
    identity inbound {
      base direction;
      description
        "Direction of traffic coming into the network entry";
    }
    identity outbound {
      base direction;
      description
        "Direction of traffic going out of the network entry";
    }
    grouping classifier-entry-stats {
      description
        "
         This group defines the classifier filter counters of
         each classifier entry
        ";
      leaf classified-pkts {
        type uint64;
        description
          " Number of total packets which filtered
            to a classifier-entry";
      }
      leaf classified-bytes {
        type uint64;
        description
          " Number of total bytes which filtered
            to a classifier-entry";
      }
      leaf classified-rate {
        type uint64;
        units "bits-per-second";
        description
          " Rate of average data flow through a
            classifier-entry";
      }
    }
    grouping named-stats {
      description
        "QoS matching statistics associated with a stats-name";
        leaf pkts {
          type uint64;
          description
            " Number of total matched packets associated
              to a statistics name";
        }
        leaf bytes {
          type uint64;
```

```
          description
            " Number of total matched bytes associated
              to a statistics name";
        }
        leaf rate {
          type uint64;
          units "bits-per-second";
          description
            " Rate of average matched data which is associated
              to a statistics name";
        }
    }
    grouping queue-stats {
      description
        "Queuing Counters";
      leaf output-conform-pkts {
        type uint64;
        description
          "Number of packets transmitted from queue ";
      }
      leaf output-conform-bytes {
        type uint64;
        description
          "Number of bytes transmitted from queue ";
      }
      leaf output-exceed-pkts {
        type uint64;
        description
          "Number of packets transmitted from queue ";
      }
      leaf output-exceed-bytes {
        type uint64;
        description
          "Number of bytes transmitted from queue ";
      }
      leaf queue-current-size-bytes {
        type uint64;
        description
          "Number of bytes currently buffered ";
      }
      leaf queue-average-size-bytes {
        type uint64;
        description
          "Average queue size in number of bytes";
      }
      leaf queue-peak-size-bytes {
        type uint64;
        description
```

```
            "Peak buffer queue size in bytes ";
        }
        leaf tailed-drop-pkts {
          type uint64;
          description
            "Total number of packets tail-dropped ";
        }
        leaf tailed-drop-bytes {
          type uint64;
          description
            "Total number of bytes tail-dropped ";
        }
      }
      grouping meter-stats {
        description
          "Metering counters";
        leaf conform-pkts {
          type uint64;
          description
            "Number of conform packets";
        }
        leaf conform-bytes {
          type uint64;
          description
            "Bytes of conform packets";
        }
        leaf conform-rate {
          type uint64;
          units "bits-per-second";
          description
            "Traffic Rate measured as conformimg";
        }
        leaf exceed-pkts {
          type uint64;
          description
            "Number of packets counted as exceeding";
        }
        leaf exceed-bytes {
          type uint64;
          description
            "Bytes of packets counted as exceeding";
        }
        leaf exceed-rate {
          type uint64;
          units "bits-per-second";
          description
            "Traffic Rate measured as exceeding";
        }
```

```
      leaf violate-pkts {
        type uint64;
        description
          "Number of packets counted as violating";
      }
      leaf violate-bytes {
        type uint64;
        description
          "Bytes of packets counted as violating";
      }
      leaf violate-rate {
        type uint64;
        units "bits-per-second";
        description
          "Traffic Rate measured as violating";
      }
      leaf meter-drop-pkts {
        type uint64;
        description
          "Number of packets dropped by meter";
      }
      leaf meter-drop-bytes {
        type uint64;
        description
          "Bytes of packets dropped by meter";
      }
    }
    grouping classifier-entry-statistics {
      description
        "Statistics for a classifier entry";
      leaf classifier-entry-name {
        type string;
        description
          "Classifier Entry Name";
      }
      uses classifier-entry-stats;
    }

    grouping queuing-stats {
      description
        "Statistics for a queue";
      leaf queue-id {
        type string;
        description
          "Queue Identifier";
      }
      uses queue-stats;
    }
```

```
    grouping metering-stats {
      description
        "Statistics for a meter";
      leaf meter-id {
        type string;
        description
          "Meter Identifier";
      }
      uses meter-stats;
    }

    augment "/if:interfaces/if:interface" {
      description
        "Augments Qos Target Entry to Interface module";

      container qos-interface-statistics {
        config false;
        description
          "Qos Interface statistics";

        list stats-per-direction {
          description
            "Qos Interface statistics for ingress or egress direction";

          leaf direction {
            type identityref {
              base direction;
            }
            description
              "Direction fo the traffic flow either inbound or
               outbound";
          }
          leaf policy-name {
            type string;
            description
              "Policy entry name for single level policy as well as
               for Hierarchical policies. For Hierarchical policies,
               this represent relative path as well as the last level
               policy name.";
          }

          list classifier-statistics {
            description
              "Classifier Statistics for each Classifier Entry in a
               Policy applied in a particular direction";
            reference
              "RFC3289: Section 6";
            uses classifier-entry-statistics;
```

```
          }
          list named-statistics {
            config false;
            description
              "Statistics for a statistics-name";
            leaf stats-name {
              type string;
              description
                "Statistics name";
            }
            container aggregated {
              description
                "Matched aggregated statistics for a statistics-name";
              uses named-stats;
            }
            container non-aggregated {
              description
                "Statistics for non-aggregated statistics-name";
              list classifier-statistics {
                description
                  "Classifier Statistics for each Classifier Entry in a
                   Policy applied in a particular direction";
                uses classifier-entry-statistics;
              }
              list metering-statistics {
                config false;
                description
                  "Statistics for each Meter associated with
                   the Policy";
                reference
                  "RFC2697: A Single Rate Three Color Marker
                   RFC2698: A Two Rate Three Color Marker";
                uses metering-stats;
              }
              list queueing-statistics {
                config false;
                description
                  "Statistics for each Queue associated with
                   the Policy";
                uses queuing-stats;
              }
            }
          }
          list metering-statistics {
            config false;
            description
              "Statistics for each Meter associated with the Policy";
            reference
```

```
              "RFC2697: A Single Rate Three Color Marker
               RFC2698: A Two Rate Three Color Marker";
            uses metering-stats;
          }
          list queueing-statistics {
            config false;
            description
              "Statistics for each Queue associated with the Policy";
            uses queuing-stats;
          }
        }
      }
    }
  }
```

<CODE ENDS>

6. Security Considerations

7. Acknowledgement

MITRE has approved this document for Public Release, Distribution Unlimited, with Public Release Case Number 20-0518. The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author.

8. References

8.1. Normative References

[I-D.ietf-rtgwg-qos-model]
Choudhary, A., Jethanandani, M., Strahle, N., Aries, E., and I. Chen, "YANG Model for QoS", draft-ietf-rtgwg-qos-model-02 (work in progress), July 2020.

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

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

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

[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>.

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

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

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

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

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>.

Authors' Addresses

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

Email: asechoud@cisco.com

Ing-Wher Chen
The MITRE Corporation

Email: ingwherchen@mitre.org

RTG Working Group
Internet-Draft
Intended status: Informational
Expires: December 26, 2020

C. Bookham, Ed.
A. Stone
Nokia
J. Tantsura
Apstra
M. Durrani
Equinix Inc
B. Decraene
Orange

June 24, 2020

An Architecture for Network Function Interconnect
draft-bookham-rtgwg-nfix-arch-01

Abstract

The emergence of technologies such as 5G, the Internet of Things (IoT), and Industry 4.0, coupled with the move towards network function virtualization, means that the service requirements demanded from networks are changing. This document describes an architecture for a Network Function Interconnect (NFIX) that allows for interworking of physical and virtual network functions in a unified and scalable manner across wide-area network and data center domains while maintaining the ability to deliver against SLAs.

Requirements Language

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

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 December 26, 2020.

Table of Contents

1. Introduction

With the introduction of technologies such as 5G, the Internet of Things (IoT), and Industry 4.0, service requirements are changing. In addition to the ever-increasing demand for more capacity, these services have other stringent service requirements that need to be met such as ultra-reliable and/or low-latency communication.

Parallel to this, there is a continued trend to move towards network function virtualization. Operators are building digitalized infrastructure capable of hosting numerous virtualized network functions (VNFs). Infrastructure that can scale in and scale out depending on the application demand and can deliver flexibility and service velocity. Much of this virtualization activity is driven by the afore-mentioned emerging technologies as new infrastructure is deployed in support of them. To try and meet the new service requirements some of these VNFs are becoming more dispersed, so it is common for networks to have a mix of centralized medium- or large-sized sized data centers together with more distributed smaller 'edge-clouds'. VNFs hosted within these data centers require seamless connectivity to each other, and to their existing physical network function (PNF) counterparts. This connectivity also needs to deliver against agreed SLAs.

Coupled with the deployment of virtualization is automation. Many of these VNFs are deployed within SDN-enabled data centers where automation is simply a must-have capability to improve service activation lead-times. The expectation is that services will be instantiated in an abstract point-and-click manner and be automatically created by the underlying network, dynamically adapting to service connectivity changes as virtual entities move between hosts.

This document describes an architecture for a Network Function Interconnect (NFIX) that allows for interworking of physical and virtual network functions in a unified and scalable manner. It describes a mechanism for establishing connectivity across multiple discreet domains in both the wide-area network (WAN) and the data

center (DC) while maintaining the ability to deliver against SLAs. To achieve this NFIX works with the underlying topology to build a unified over-the-top topology.

The NFIX architecture described in this document does not define any new protocols but rather outlines an architecture utilizing a collaboration of existing standards-based protocols.

2. Terminology

- A physical network function (PNF) refers to a network device such as a Provider Edge (PE) router that connects physically to the wide-area network.
- A virtualized network function (VNF) refers to a network device such as a provider edge (PE) router that is hosted on an application server. The VNF may be bare-metal in that it consumes the entire resources of the server, or it may be one of numerous virtual functions instantiated as a VM or number of containers on a given server that is controlled by a hypervisor or container management platform.
- A Data Center Border (DCB) router refers to the network function that spans the border between the wide-area and the data center networks, typically interworking the different encapsulation techniques employed within each domain.
- An Interconnect controller is the controller responsible for managing the NFIX fabric and services.
- A DC controller is the term used for a controller that resides within an SDN-enabled data center and is responsible for the DC network(s)

3. Motivation

Industrial automation and business-critical environments use applications that are demanding on the network. These applications present different requirements from low-latency to high-throughput, to application-specific traffic conditioning, or a combination. The evolution to 5G equally presents challenges for mobile back-, front- and mid-haul networks. The requirement for ultra-reliable low-latency communication means that operators need to re-evaluate their network architecture to meet these requirements.

At the same time, the service edge is evolving. Where the service edge device was historically a PNF, the adoption of virtualization means VNFs are becoming more commonplace. Typically, these VNFs are

hosted in some form of data center environment but require end-to-end connectivity to other VNFs and/or other PNFs. This represents a challenge because generally transport layer connectivity differs between the WAN and the data center environment. The WAN includes all levels of hierarchy (core, aggregation, access) that form the networks footprint, where transport layer connectivity using IP/MPLS is commonplace. In the data center native IP is commonplace, utilizing network virtualization overlay (NVO) technologies such as virtual extensible LAN (VXLAN) [RFC7348], network virtualization using generic routing encapsulation (NVGRE) [RFC7637], or generic network virtualization encapsulation (GENEVE) [I-D.ietf-nvo3-geneve]. There is a requirement to seamlessly integrate these islands and avoid heavy-lifting at interconnects as well as providing a means to provision end-to-end services with a single touch point at the edge.

The service edge boundary is also changing. Some functions that were previously reasonably centralized are now becoming more distributed. One reason for this is to attempt to deal with low latency requirements. Another reason is that operators seek to reduce costs by deploying low/medium-capacity VNFs closer to the edge. Equally, virtualization also sees some of the access network moving towards the core. Examples of this include cloud-RAN or Software-Defined Access Networks.

Historically service providers have architected data centers independently from the wide-area network, creating two independent domains or islands. As VNFs become part of the service landscape the service data-path must be extended across the WAN into the data center infrastructure, but in a manner that still allows operators to meet deterministic performance requirements. Methods for stitching WAN and DC infrastructures together with some form of service-interworking at the data center border have been implemented and deployed, but this service-interworking approach has several limitations:

- The data center environment typically uses encapsulation techniques such as VXLAN or NVGRE while the WAN typically uses encapsulation techniques such as MPLS [RFC3031]. Underlying optical infrastructure might also need to be programmed. These are incompatible and require interworking at the service layer.

- It typically requires heavy-touch service provisioning on the data center border. In an end-to-end service, midpoint provisioning is undesirable and should be avoided.

- Automation is difficult; largely due to the first two points but with additional contributing factors. In the virtualization world automation is a must-have capability.

- When a service is operating at Layer 3 in a data center with redundant interconnects the risk of routing loops exists. There is no inherent loop avoidance mechanism when redistributing routes between address families so extreme care must be taken. Proposals such as the Domain Path (D-PATH) attribute [I-D.ietf-bess-evpn-ipvpn-interworking] attempt to address this issue but as yet are not widely implemented or deployed.

- Some or all the above make the service-interworking gateway cumbersome with questionable scaling attributes.

Hence there is a requirement to create an open, scalable, and unified network architecture that brings together the wide-area network and data center domains. It is not an architecture e xclusively targeted at greenfield deployments, nor does it require a flag day upgrade to deploy in a brownfield network. It is an evolutionary step to a consolidated network that uses the constructs of seamless MPLS [I-D.ietf-mpls-seamless-mpls] as a baseline and extends upon that to include topologies that may not be link-state based and to provide end-to-end path control. Overall the NFIX architecture aims to deliver the following:

- Allows for an evolving service edge boundary without having to constantly restructure the architecture.

- Provides a mechanism for providing seamless connectivity between VNF to VNF, VNF to PNF, and PNF to PNF, with deterministic SLAs, and with the ability to provide differentiated SLAs to suit different service requirements.

- Delivers a unified transport fabric using Segment Routing (SR) [RFC8402] where service delivery mandates touching only the service edge without imposing additional encapsulation requirements in the DC.

- Embraces automation by providing an environment where any end-to-end connectivity can be instantiated in a single request manner while maintaining SLAs.

4. Requirements

The following section outlines the requirements that the proposed solution must meet. From an overall perspective, the proposed generic architecture must:

- Deliver end-to-end transport LSPs using traffic-engineering (TE) as required to meet appropriate SLAs for the service using(s)

using those LSPs. End-to-end refers to VNF and/or PNF connectivity or a combination of both.

- Provide a solution that allows for optimal end-to-end path placement; where optimal not only meets the requirements of the path in question but also meets the global network objectives.

- Support varying types of VNF physical network attachment and logical (underlay/overlay) connectivity.

- Facilitate automation of service provision. As such the solution should avoid heavy-touch service provisioning and decapsulation/ encapsulation at data center border routers.

- Provide a framework for delivering logical end-to-end networks using differentiated logical topologies and/or constraints.

- Provide a high level of stability; faults in one domain should not propagate to another domain.

- Provide a mechanism for homogeneous end-to-end OAM.

- Hide/localize instabilities in the different domains that participate in the end-to-end service.

- Provide a mechanism to minimize the label-stack depth required at path head-ends for SR-TE LSPs.

- Offer a high level of scalability.

- Although not considered in-scope of the current version of this document, the solution should not preclude the deployment of multicast. This subject may be covered in later versions of this document.

5. Theory of Operation

This section describes the NFIX architecture including the building blocks and protocol machinery that is used to form the fabric. Where considered appropriate rationale is given for selection of an architectural component where other seemingly applicable choices could have been made.

5.1. VNF Assumptions

For the sake of simplicity, references to VNF are made in a broad sense. Equally, the differences between VNF and Container Network Function (CNF) are largely immaterial for the purposes of this

document, therefore VNF is used to represent both. The way in which a VNF is instantiated and provided network connectivity will differ based on environment and VNF capability, but for conciseness this is not explicitly detailed with every reference to a VNF. Common examples of VNF variants include but are not limited to:

- A VNF that functions as a routing device and has full IP routing and MPLS capabilities. It can be connected simultaneously to the data center fabric underlay and overlay and serves as the NVO tunnel endpoint [RFC8014]. Examples of this might be a virtualized PE router, or a virtualized Broadband Network Gateway (BNG).

- A VNF that functions as a device (host or router) with limited IP routing capability. It does not connect directly to the data center fabric underlay but rather connects to one or more external physical or virtual devices that serve as the NVO tunnel endpoint(s). It may however have single or multiple connections to the overlay. Examples of this might be a mobile network control or management plane function.

- A VNF that has no routing capability. It is a virtualized function hosted within an application server and is managed by a hypervisor or container host. The hypervisor/container host acts as the NVO endpoint and interfaces to some form of SDN controller responsible for programming the forwarding plane of the virtualization host using, for example, OpenFlow. Examples of this might be an Enterprise application server or a web server running as a virtual machine and front-ended by a virtual routing function such as OVS/xVRS/VTF.

Where considered necessary exceptions to the examples provided above or focus on a particular scenario will be highlighted.

5.2. Overview

The NFIX architecture makes no assumptions about how the network is physically composed, nor does it impose any dependencies upon it. It also makes no assumptions about IGP hierarchies and the use of areas/levels or discrete IGP instances within the WAN is fully endorsed to enhance scalability and constrain fault propagation. This could apply for instance to a hierarchical WAN from core to edge or from WAN to LAN connections. The overall architecture uses the constructs of seamless MPLS as a baseline and extends upon that. The concept of decomposing the network into multiple domains is one that has been widely deployed and has been proven to scale in networks with large numbers of nodes.

The proposed architecture uses segment routing (SR) as its preferred choice of transport. Segment routing is chosen for construction of end-to-end LSPs given its ability to traffic-engineer through source-routing while concurrently scaling exceptionally well due to its lack of network state other than the ingress node. This document uses SR instantiated on an MPLS forwarding plane(SR-MPLS), although it does not preclude the use of SRv6 either now or at some point in the future. The rationale for selecting SR-MPLS is simply maturity and more widespread applicability across a potentially broad range of network devices. This document may be updated in future versions to include more description of SRv6 applicability.

5.3. Use of a Centralized Controller

It is recognized that for most operators the move towards the use of a controller within the wide-area network is a significant change in operating model. In the NFIX architecture it is a necessary component. Its use is not simply to offload inter-domain path calculation from network elements; it provides many more benefits:

- It offers the ability to enforce constraints on paths that originate/terminate on different network elements, thereby providing path diversity, and/or bidirectionality/co-routing, and/or disjointness.

- It avoids collisions, re-tries, and packing problems that has been observed in networks using distributed TE path calculation, where head-ends make autonomous decisions.

- A controller can take a global view of path placement strategies, including the ability to make path placement decisions over a high number of LSPs concurrently as opposed to considering each LSP independently. In turn, this allows for 'global' optimization of network resources such as available capacity.

- A controller can make decisions based on near-real-time network state and optimize paths accordingly. For example, if a network link becomes congested it may recompute some of the paths transiting that link to other links that may not be quite as optimal but do have available capacity. Or if a link latency crosses a certain threshold, it may select to reoptimize some latency-sensitive paths away from that link.

- The logic of a controller can be extended beyond pure path computation and placement. If the controller is aware of services, service requirements, and available paths within the network it can cross-correlate between them and ensure that the appropriate paths are used for the appropriate services.

- The controller can provide assurance and verification of the underlying SLA provided to a given service.

As the main objective of the NFIX architecture is to unify the data center and wide-area network domains, using the term controller is not sufficiently succinct. The centralized controller may need to interface to other controllers that potentially reside within an SDN-enabled data center. Therefore, to avoid interchangeably using the term controller for both functions, we distinguish between them simply by using the terms 'DC controller' which as the name suggests is responsible for the DC, and 'Interconnect controller' responsible for managing the extended SR fabric and services.

The Interconnect controller learns wide-area network topology information and allocation of segment routing SIDs within that domain using BGP link-state [RFC7752] with appropriate SR extensions. Equally it learns data center topology information and Prefix-SID allocation using BGP labeled unicast [RFC8277] with appropriate SR extensions, or BGP link-state if a link-state IGP is used within the data center. If Route-Reflection is used for exchange of BGP link-state or labeled unicast NLRI within one or more domains, then the Interconnect controller need only peer as a client with those Route-Reflectors in order to learn topology information.

Where BGP link-state is used to learn the topology of a data center (or any IGP routing domain) the BGP-LS Instance Identifier (Instance-ID) is carried within Node/Link/Prefix NLRI and is used to identify a given IGP routing domain. Where labeled unicast BGP is used to discover the topology of one or more data center domains there is no equivalent way for the Interconnect controller to achieve a level of routing domain correlation. The controller may learn some splintered connectivity map consisting of 10 leaf switches, four spine switches, and four DCB's, but it needs some form of key to inform it that leaf switches 1-5, spine switches 1 and 2, and DCB's 1 and 2 belong to data center 1, while leaf switches 6-10, spine switches 3 and 4, and DCB's 3 and 4 belong to data center 2. What is needed is a form of 'data center membership identification' to provide this correlation. Optionally this could be achieved at BGP level using a standard community to represent each data center, or it could be done at a more abstract level where for example the DC controller provides the membership identification to the Interconnect controller through an application programming interface (API).

Understanding real-time network state is an important part of the Interconnect controllers role, and only with this information is the controller able to make informed decisions and take preventive or corrective actions as necessary. There are numerous methods implemented and deployed that allow for harvesting of network state,

including (but not limited to) IPFIX [RFC7011], Netconf/YANG [RFC6241][RFC6020], streaming telemetry, BGP link-state [RFC7752] [I-D.ietf-idr-te-lsp-distribution], and the BGP Monitoring Protocol (BMP) [RFC7854].

5.4. Routing and LSP Underlay

This section describes the mechanisms and protocols that are used to establish end-to-end LSPs; where end-to-end refers to VNF-to-VNF, PNF-to-PNF, or VNF-to-PNF.

5.4.1. Intra-Domain Routing

In a seamless MPLS architecture domains are based on geographic dispersion (core, aggregation, access). Within this document a domain is considered as any entity with a captive topology; be it a link-state topology or otherwise. Where reference is made to the wide-area network domain, it refers to one or more domains that constitute the wide-area network domain.

This section discusses the basic building blocks required within the wide-area network and the data center, noting from above that the wide-area network may itself consist of multiple domains.

5.4.1.1. Wide-Area Network Domains

The wide-area network includes all levels of hierarchy (core, aggregation, access) that constitute the networks MPLS footprint as well as the data Center border routers. Each domain that constitutes part of the wide-area network runs a link-state interior gateway protocol (IGP) such as ISIS or OSPF, and each domain may use IGP-inherent hierarchy (OSPF areas, ISIS levels) with an assumption that visibility is domain-wide using, for example, L2 to L1 redistribution. Alternatively, or additionally, there may be multiple domains that are split by using separate and distinct instances of IGP. There is no requirement for IGP redistribution of any link or loopback addresses between domains.

Each IGP should be enabled with the relevant extensions for segment routing [RFC8667][RFC8665], and each SR-capable router should advertise a Node-SID for its loopback address, and an Adjacency-SID (Adj-SID) for every connected interface (unidirectional adjacency) belonging to the SR domain. SR Global Blocks (SRGB) can be allocated to each domain as deemed appropriate to specific network requirements. Border routers belonging to multiple domains have an SRGB for each domain.

The default forwarding path for intra-domain LSPs that do not require TE is simply an SR LSP containing a single label advertised by the destination as a Node-SID and representing the ECMP-aware shortest path to that destination. Intra-domain TE LSPs are constructed as required by the Interconnect controller. Once a path is calculated it is advertised as an explicit SR Policy [I-D.ietf-spring-segment-routing-policy] containing one or more paths expressed as one or more segment-lists, which may optionally contain binding SIDs if requirements dictate. An SR Policy is identified through the tuple [headend, color, endpoint] and this tuple is used extensively by the Interconnect controller to associate services with an underlying SR Policy that meets its objectives.

To provide support for ECMP the Entropy Label [RFC6790][RFC8662] should be utilized. Entropy Label Capability (ELC) should be advertised into the IGP using the IS-IS Prefix Attributes TLV [I-D.ietf-isis-mpls-elc] or the OSPF Extended Prefix TLV [I-D.ietf-ospf-mpls-elc] coupled with the Node MSD Capability sub-TLV to advertise Entropy Readable Label Depth (ERLD) [RFC8491][RFC8476] and the base MPLS Imposition (BMI). Equally, support for ELC together with the supported ERLD should be signaled in BGP using the BGP Next-Hop Capability [I-D.ietf-idr-next-hop-capability]. Ingress nodes and or DCBs should ensure sufficient entropy is applied to packets to exercise available ECMP links.

5.4.1.2. Data Center Domain

The data center domain includes all fabric switches, network virtualization edge (NVE), and the data center border routers. The data center routing design may align with the framework of [RFC7938] running eBGP single-hop sessions established over direct point-to-point links, or it may use an IGP for dissemination of topology information. This document focuses on the former, simply because the ue of an IGP largely makes the data centers behaviour analogous to that of a wide-area network domain.

The chosen method of transport or encapsulation within the data center for NFIX is SR-MPLS over IP/UDP [RFC8663] or, where possible, native SR-MPLS. The choice of SR-MPLS over IP/UDP or native SR-MPLS allows for good entropy to maximize the use of equal-cost Clos fabric links. Native SR-MPLS encapsulation provides entropy through use of the Entropy Label, and, like the wide-area network, support for ELC together with the support ERLD should be signaled using the BGP Next-Hop Capability attribute. As described in [RFC6790] the ELC is an indication from the egress node of an MPLS tunnel to the ingress node of the MPLS tunnel that is is capable of processing an Entropy Label. The BGP Next-Hop Capability is a non-transitive attribute which is modified or deleted when the next-hop is changed to reflect the

capabilities of the new next-hop. If we assume that the path of a BGP-signaled LSP transits through multiple ASNs, and/or a single ASN with multiple next-hops, then it is not possible for the ingress node to determine the ELC of the egress node. Without this end-to-end signaling capability the entropy label must only be used when it is explicitly known, through configuration or other means, that the egress node has support for it. Entropy for SR-MPLS over IP/UDP encapsulation uses the source UDP port for IPv4 and the Flow Label for IPv6. Again, the ingress network function should ensure sufficient entropy is applied to exercise available ECMP links.

Another significant advantage of the use of native SR-MPLS or SR-MPLS over IP/UDP is that it allows for a lightweight interworking function at the DCB without the requirement for midpoint provisioning; interworking between the data center and the wide-area network domains becomes an MPLS label swap/continue action.

Loopback addresses of network elements within the data center are advertised using labeled unicast BGP with the addition of SR Prefix SID extensions [RFC8669] containing a globally unique and persistent Prefix-SID. The data-plane encapsulation of SR-MPLS over IP/UDP or native SR-MPLS allows network elements within the data center to consume BGP Prefix-SIDs and legitimately use those in the encapsulation.

5.4.2. Inter-Domain Routing

Inter-domain routing is responsible for establishing connectivity between any domains that form the wide-area network, and between the wide-area network and data center domains. It is considered unlikely that every end-to-end LSP will require a TE path, hence there is a requirement for a default end-to-end forwarding path. This default forwarding path may also become the path of last resort in the event of a non-recoverable failure of a TE path. Similar to the seamless MPLS architecture this inter-domain MPLS connectivity is realized using labeled unicast BGP [RFC8277] with the addition of SR Prefix SID extensions.

Within each wide-area network domain all service edge routers, DCBs, and ABRs/ASBRs form part of the labeled BGP mesh, which can be either full-mesh, or more likely based on the use of route-reflection. Each of these routers advertises its respective loopback addresses into labeled BGP together with an MPLS label and a globally unique Prefix-SID. Routes are advertised between wide-area network domains by ABRs/ASBRs that impose next-hop-self on advertised routes. The function of imposing next-hop-self for labeled routes means that the ABR/ASBR allocates a new label for advertised routes and programs a

label-swap entry in the forwarding plane for received and advertised routes. In short it becomes part of the forwarding path.

DCB routers have labeled BGP sessions towards the wide-area network and labeled BGP sessions towards the data center. Routes are bidirectionally advertised between the domains subject to policy, with the DCB imposing itself as next-hop on advertised routes. As above, the function of imposing next-hop-self for labeled routes implies allocation of a new label for advertised routes and a label-swap entry being programmed in the forwarding plane for received and advertised labels. The DCB thereafter becomes the anchor point between the wide-area network domain and the data center domain.

Within the wide-area network next-hops for labeled unicast routes containing Prefix-SIDs are resolved to SR LSPs, and within the data center domain next-hops for labeled unicast routes containing Prefix-SIDs are resolved to SR LSPs or IP/UDP tunnels. This provides end-to-end connectivity without a traffic-engineering capability.

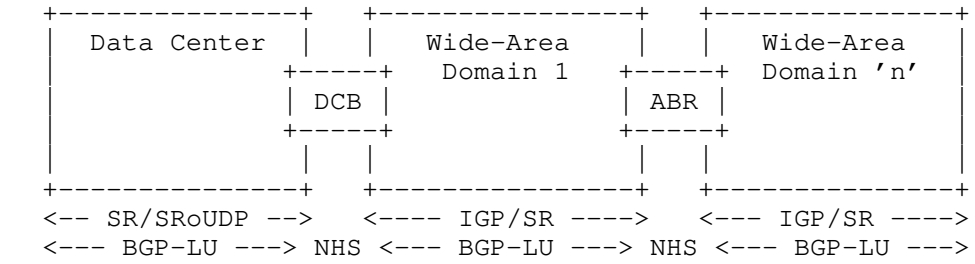


Figure 1

5.4.3. Intra-Domain and Inter-Domain Traffic-Engineering

The capability to traffic-engineer intra- and inter-domain end-to-end paths is considered a key requirement in order to meet the service objectives previously outlined. To achieve optimal end-to-end path placement the key components to be considered are path calculation, path activation, and FEC-to-path binding procedures.

In the NFIX architecture end-to-end path calculation is performed by the Interconnect controller. The mechanics of how the objectives of each path is calculated is beyond the scope of this document. Once a path is calculated based upon its objectives and constraints, the

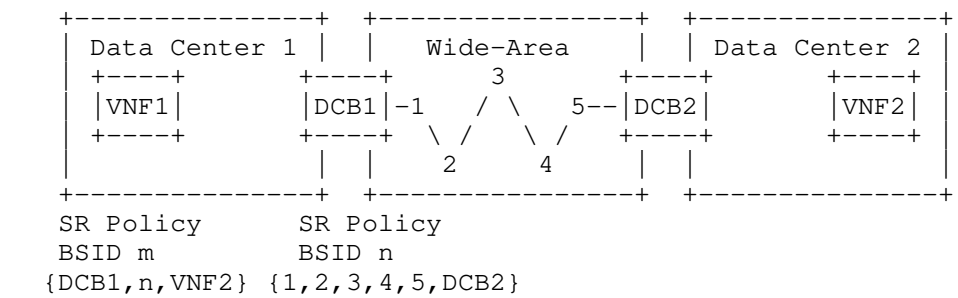
path is advertised from the controller to the LSP headend as an explicit SR Policy containing one or more paths expressed as one or more segment-lists. An SR Policy is identified through the tuple [headend, color, endpoint] and this tuple is used extensively by the Interconnect controller to associate services with an underlying SR Policy that meets its objectives.

The segment-list of an SR Policy encodes a source-routed path towards the endpoint. When calculating the segment-list the Interconnect controller makes comprehensive use of the Binding-SID (BSID), instantiating BSID anchors as necessary at path midpoints when calculating and activating a path. The use of BSID is considered fundamental to segment routing as described in [I-D.filsfils-spring-sr-policy-considerations]. It provides opacity between domains, ensuring that any segment churn is constrained to a single domain. It also reduces the number of segments/labels that the headend needs to impose, which is particularly important given that network elements within a data center generally have limited label imposition capabilities. In the context of the NFIX architecture it is also the vehicle that allows for removal of heavy midpoint provisioning at the DCB.

For example, assume that VNF1 is situated in data center 1, which is interconnected to the wide-area network via DCB1. VNF1 requires connectivity to VNF2, situated in data center 2, which is interconnected to the wide-area network via DCB2. Assuming there is no existing TE path that meet VNF1's requirements, the Interconnect controller will:

- Instantiate an SR Policy on DCB1 with BSID n and a segment-list containing the relevant segments of a TE path to DCB2. DCB1 therefore becomes a BSID anchor.

- Instantiate an SR Policy on VNF1 with BSID m and a segment-list containing segments {DCB1, n, VNF2}.



Traffic-Engineered Path using BSID

Figure 2

In the above figure a single DCB is used to interconnect two domains. Similarly, in the case of two wide-area domains the DCB would be represented as an ABR or ASBR. In some single operator environments domains may be interconnected using adjacent ASBRs connected via a distinct physical link. In this scenario the procedures outlined above may be extended to incorporate the mechanisms used in Egress Peer Engineering (EPE) [I-D.ietf-spring-segment-routing-central-epe] to form a traffic-engineered path spanning distinct domains.

5.4.3.1. Traffic-Engineering and ECMP

Where the Interconnect controller is used to place SR policies, providing support for ECMP requires some consideration. An SR Policy is described with one or more segment-lists, end each of those segment-lists may or may not provide ECMP as a sum instruction and each SID itself may or may not support ECMP forwarding. When an individual SID is a BSID, an ECMP path may or may not also be nested within. The Interconnect controller may choose to place a path consisting entirely of non-ECMP-aware Adj-SIDs (each SID representing a single adjacency) such that the controller has explicit hop-by-hop knowledge of where that SR-TE LSP is routed. This is beneficial to allow the controller to take corrective action if the criteria that was used to initially select a particular link in a particular path subsequently changes. For example, if the latency of a link increases or a link becomes congested and a path should be rerouted. If ECMP-aware SIDs are used in the SR policy segment-list (including Node-SIDs, Adj-SIDs representing parallel links, and Anycast SIDs) SR routers are able to make autonomous decisions about where traffic is forwarded. As a result, it is not possible for the controller to fully understand the impact of a change in network state and react to it. With this in mind there are a number of approaches that could be adopted:

- If there is no requirement for the Interconnect controller to explicitly track path on a hop-by-hop basis, ECMP-aware SIDs may be used in the SR policy segment-list. This approach may require multiple [ELI, EL] pairs to be inserted at the ingress node; for example, above and below a BSID to provide entropy in multiple domains.

- If there is a requirement for the Interconnect controller to explicitly track paths on a hop-by-hop to provide the capability to reroute them based on changes in network state, SR policy segment-lists should be constructed of non-ECMP-aware Adj-SIDs.

- A hybrid approach that allows for a level of ECMP (at the headend) together with the ability for the Interconnect controller to explicitly track paths is to instantiate an SR policy consisting of a set of segment-lists, each containing non-ECMP-aware Adj-SIDs. Each segment-list will be assigned a weight to allow for ECMP or UCMP. This approach does however imply computation and programing of two paths instead of one.

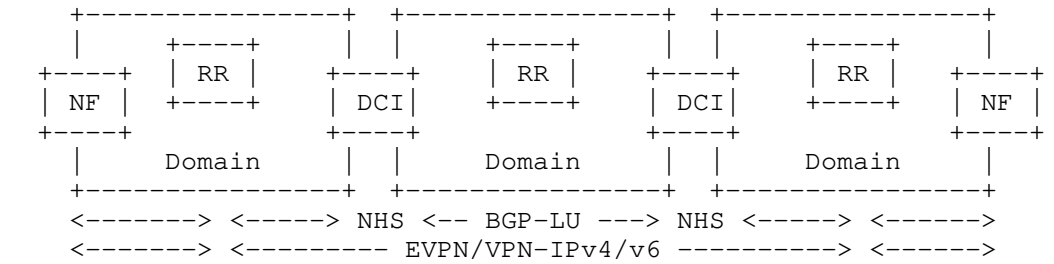
- Another hybrid approach might work as follows. Redundant DCBs advertise an Anycast-SID 'A' into the data center, and also instantiate an SR policy with a segment-list consisting of non-ECMP-aware Adj-SIDs meeting the required connectivity and SLA. The BSID value of this SR policy 'B' must be common to both redundant DCBs, but the calculated paths are diverse. Indeed, multiple segment-lists could be used in this SR policy. A VNF could then instantiate an SR policy with a segment-list of {A, B} to achieve ECMP in the data center and TE in the wide-area network with the option of ECMP at the BSID anchor

5.5. Service Layer

The service layer is intended to deliver Layer 2 and/or Layer 3 VPN connectivity between network functions to create an overlay utilizing the routing and LSP underlay described in section 5.4. To do this the solution employs the EVPN and/or VPN-IPv4/IPv6 address families to exchange Layer 2 and Layer 3 Network Layer Reachability Information (NLRI). When these NLRI are exchanged between domains it is typical for the border router to set next-hop-self on advertised routes. With the proposed routing and LSP underlay however, this is not required and EVPN/VPN-IPv4/IPv6 routes should be passed end-to-end without transit routers modifying the next-hop attribute.

Section 5.4.2 describes the use of labeled unicast BGP to exchange inter-domain routes to establish a default forwarding path. Labeled-unicast BGP is used to exchange prefix reachability between service edge routers, with domain border routes imposing next-hop-self on

routes advertised between domains. This provides a default inter-domain forwarding path and provides the required connectivity to establish inter-domain BGP sessions between service edges for the exchange of EVPN and/or VPN-IPv4/IPv6 NLRI. If route-reflection is used for the EVPN and/or VPN-IPv4/IPv6 address families within one or more domains, it may be desirable to create inter-domain BGP sessions between route-reflectors. In this case the peering addresses of the route-reflectors should also be exchanged between domains using labeled unicast BGP. This creates a connectivity model analogous to BGP/MPLS IP-VPN Inter-AS option C [RFC4364].



Inter-Domain Service Layer

Figure 3

EVPN and/or VPN-IPv4/v6 routes received from a peer in a different domain will contain a next-hop equivalent to the router that sourced the route. The next-hop of these routes can be resolved to labeled-unicast route (default forwarding path) or to an SR policy (traffic-engineered forwarding path) as appropriate to the service requirements. The exchange of EVPN and/or VPN-IPv4/IPv6 routes in this manner implies that Route-Distinguisher and Route-Target values remain intact end-to-end.

The use of end-to-end EVPN and/or VPN-IPv4/IPv6 address families without the imposition of next-hop-self at border routers complements the gateway-less transport layer architecture. It negates the requirement for midpoint service provisioning and as such provides the following benefits:

- Avoids the translation of MAC/IP EVPN routes to IP-VPN routes (and vice versa) that is typically associated with service interworking.

- Avoids instantiation of MAC-VRFs and IP-VPNs for each tenant resident in the DCB.

- Avoids provisioning of demarcation functions between the data center and wide-area network such as QoS, access-control, aggregation and isolation.

5.6. Service Differentiation

As discussed in section 5.4.3, the use of TE paths is a key capability of the NFIX solution framework described in this document. The Interconnect controller computes end-to-end TE paths between NFs and programs DC nodes, DCBs, ABR/ASBRs, via SR Policy, with the necessary label forwarding entries for each [headend, color, endpoint]. The collection of [headend, endpoint] pairs for the same color constitutes a logical network topology, where each topology satisfies a given SLA requirement.

The Interconnect controller discovers the endpoints associated to a given topology (color) upon the reception of EVPN or IPVPN routes advertised by the endpoint. The EVPN and IPVPN NLRIs are advertised by the endpoint nodes along with a color extended community which identifies the topology to which the owner of the NLRI belongs. At a coarse level all the EVPN/IPVPN routes of the same VPN can be advertised with the same color, and therefore a TE topology would be established on a per-VPN basis. At a more granular level IPVPN and especially EVPN provide a more granular way of coloring routes, that will allow the Interconnect controller to associate multiple topologies to the same VPN. For example:

- All the EVPN MAC/IP routes for a given VNF may be advertised with the same color. This would allow the Interconnect controller to associate topologies per VNF within the same VPN; that is, VNF1 could be blue (e.g., low-latency topology) and VNF2 could be green (e.g., high-throughput).

- The EVPN MAC/IP routes and Inclusive Multicast Ethernet Tag (IMET) route for VNF1 may be advertised with different colors, e.g., red and brown, respectively. This would allow the association of e.g., a low-latency topology for unicast traffic to VNF1 and best-effort topology for BUM traffic to VNF1.

- Each EVPN MAC/IP route or IP-Prefix route from a given VNF may be advertised with different color. This would allow the association of topologies at the host level or host route granularity.

5.7. Automated Service Activation

The automation of network and service connectivity for instantiation and mobility of virtual machines is a highly desirable attribute within data centers. Since this concerns service connectivity, it should be clear that this automation is relevant to virtual functions that belong to a service as opposed to a virtual network function that delivers services, such as a virtual PE router.

Within an SDN-enabled data center, a typical hierarchy from top to bottom would include a policy engine (or policy repository), one or more DC controllers, numerous hypervisors/container hosts that function as NVO endpoints, and finally the virtual machines(VMs)/containers, which we'll refer to generically as virtualization hosts.

The mechanisms used to communicate between the policy engine and DC controller, and between the DC controller and hypervisor/container are not relevant here and as such they are not discussed further. What is important is the interface and information exchange between the Interconnect controller and the data center SDN functions:

- The Interconnect controller interfaces with the data center policy engine and publishes the available colors, where each color represents a topological service connectivity map that meets a set of constraints and SLA objectives. This interface is a straightforward API.

- The Interconnect controller interfaces with the DC controller to learn overlay routes. This interface is BGP and uses the EVPN Address Family.

With the above framework in place, automation of network and service connectivity can be implemented as follows:

- The virtualization host is turned-up. The NVO endpoint notifies the DC controller of the startup.

- The DC controller retrieves service information, IP addressing information, and service 'color' for the virtualization host from the policy engine. The DC controller subsequently programs the associated forwarding information on the virtualization host. Since the DC controller is now aware of MAC and IP address information for the virtualization host, it advertises that information as an EVPN MAC Advertisement Route into the overlay.

- The Interconnect controller receives the EVPN MAC Advertisement Route (potentially via a Route-Reflector) and correlates it with

locally held service information and SLA requirements using Route Target and Color communities. If the relevant SR policies are not already in place to support the service requirements and logical connectivity, including any binding-SIDs, they are calculated and advertised to the relevant headends.

The same automated service activation principles can also be used to support the scenario where virtualization hosts are moved between hypervisors/container hosts for resourcing or other reasons. We refer to this simply as mobility. If a virtualization host is turned down the parent NVO endpoint notifies the DC controller, which in turn notifies the policy engine and withdraws any EVPN MAC Advertisement Routes. Thereafter all associated state is removed. When the virtualization host is turned up on a different hypervisor/ container host, the automated service connectivity process outlined above is simply repeated.

5.8. Service Function Chaining

Service Function Chaining (SFC) defines an ordered set of abstract service functions and the subsequent steering of traffic through them. Packets are classified at ingress for processing by the required set of service functions (SFs) in an SFC-capable domain and are then forwarded through each SF in turn for processing. The ability to dynamically construct SFCs containing the relevant SFs in the right sequence is a key requirement for operators.

To enable flexible service function deployment models that support agile service insertion the NFIX architecture adopts the use of BGP as the control plane to distribute SFC information. The BGP control plane for Network Service Header (NSH) SFC [I-D.ietf-bess-nsh-bgp-control-plane] is used for this purpose and defines two route types; the Service Function Instance Route (SFIR) and the Service Function Path Route (SFPR).

The SFIR is used to advertise the presence of a service function instance (SFI) as a function type (i.e. firewall, TCP optimizer) and is advertised by the node hosting that SFI. The SFIR is advertised together with a BGP Tunnel Encapsulation attribute containing details of how to reach that particular service function through the underlay network (i.e. IP address and encapsulation information).

The SFPRs contain service function path (SFP) information and one SFPR is originated for each SFP. Each SFPR contains the service path identifier (SPI) of the path, the sequence of service function types that make up the path (each of which has at least one instance advertised in an SFIR), and the service index (SI) for each listed service function to identify its position in the path.

Once a Classifier has determined which flows should be mapped to a given SFP, it imposes an NSH [RFC8300] on those packets, setting the SPI to that of the selected service path (advertised in an SFPR), and the SI to the first hop in the path. As NSH is encapsulation agnostic, the NSH encapsulated packet is then forwarded through the appropriate tunnel to reach the service function forwarder (SFF) supporting that service function instance (advertised in an SFIR). The SFF removes the tunnel encapsulation and forwards the packet with the NSH to the relevant SF based upon a lookup of the SPI/SI. When it is returned from the SF with a decremented SI value, the SFF forwards the packet to the next hop in the SFP using the tunnel information advertised by that SFI. This procedure is repeated until the last hop of the SFP is reached.

The use of the NSH in this manner allows for service chaining with topological and transport independence. It also allows for the deployment of SFIs in a condensed or dispersed fashion depending on operator preference or resource availability. Service function chains are built in their own overlay network and share a common underlay network, where that common underlay network is the NFIX fabric described in section 5.4. BGP updates containing an SFIR or SFPR are advertised in conjunction with one or more Route Targets (RTs), and each node in a service function overlay network is configured with one or more import RTs. As a result, nodes will only import routes that are applicable and that local policy dictates. This provides the ability to support multiple service function overlay networks or the construction of service function chains within L3VPN or EVPN services.

Although SFCs are constructed in a unidirectional manner, the BGP control plane for NSH SFC allows for the optional association of multiple paths (SFPRs). This provides the ability to construct a bidirectional service function chain in the presence of multiple equal-cost paths between source and destination to avoid problems that SFs may suffer with traffic asymmetry.

The proposed SFC model can be considered decoupled in that the use of SR as a transport between SFFs is completely independent of the use of NSH to define the SFC. That is, it uses an NSH-based SFC and SR is just one of many encapsulations that could be used between SFFs. A similar more integrated approach proposes encoding a service function as a segment so that an SFC can be constructed as a segment-list. In this case it can be considered an SR-based SFC with an NSH-based service plane since the SF is unaware of the presence of the SR. Functionally both approaches are very similar and as such both could be adopted and could work in parallel. Construction of SFCs based purely on SR (SF is SR-aware) are not considered at this time.

5.9. Stability and Availability

Any network architecture should have the capability to self-restore following the failure of a network element. The time to reconverge following the failure needs to be minimal to avoid evident disruptions in service. This section discusses protection mechanisms that are available for use and their applicability to the proposed architecture.

5.9.1. IGP Reconvergence

Within the construct of an IGP topology the Topology Independent Loop Free Alternate (TI-LFA) [I-D.ietf-rtgwg-segment-routing-ti-lfa] can be used to provide a local repair mechanism that offers both link and node protection.

TI-LFA is a repair mechanism, and as such it is reactive and initially needs to detect a given failure. To provide fast failure detection the Bidirectional Forwarding Mechanism (BFD) is used. Consideration needs to be given to the restoration capabilities of the underlying transmission when deciding values for message intervals and multipliers to avoid race conditions, but failure detection in the order of 50 milliseconds can reasonably be anticipated. Where Link Aggregation Groups (LAG) are used, micro-BFD [RFC7130] can be used to similar effect. Indeed, to allow for potential incremental growth in capacity it is not uncommon for operators to provision all network links as LAG and use micro-BFD from the outset.

5.9.2. Data Center Reconvergence

Clos fabrics are extremely common within data centers, and fundamental to a Clos fabric is the ability to load-balance using Equal Cost Multipath (ECMP). The number of ECMP paths will vary dependent on the number of devices in the parent tier but will never be less than two for redundancy purposes with traffic hashed over the available paths. In this scenario the availability of a backup path in the event of failure is implicit. Commonly within the DC, rather than computing protect paths (like LFA), techniques such as 'fast rehash' are often utilized. In this particular case, the failed next-hop is removed from the multi-path forwarding data structure and traffic is then rehashed over the remaining active paths.

In BGP-only data centers this relies on the implementation of BGP multipath. As network elements in the lower tier of a Clos fabric will frequently belong to different ASNs, this includes the ability to load-balance to a prefix with different AS_PATH attribute values

while having the same AS_PATH length; sometimes referred to as 'multipath relax' or 'multipath multiple-AS' [RFC7938].

Failure detection relies upon declaring a BGP session down and removing any prefixes learnt over that session as soon as the link is declared down. As links between network elements predominantly use direct point-to-point fiber, a link failure should be detected within milliseconds. BFD is also commonly used to detect IP layer failures.

5.9.3. Exchange of Inter-Domain Routes

Labeled unicast BGP together with SR Prefix-SID extensions are used to exchange PNF and/or VNF endpoints between domains to create end-to-end connectivity without TE. When advertising between domains we assume that a given BGP prefix is advertised by at least two border routers (DCBs, ABRs, ASBRs) making prefixes reachable via at least two next-hops.

BGP Prefix Independent Convergence (PIC) [I-D.ietf-rtgwg-bgp-pic] allows failover to a pre-computed and pre-installed secondary next-hop when the primary next-hop fails and is independent of the number of destination prefixes that are affected by the failure. When the primary BGP next-hop fails, it should be clear that BGP PIC depends on the availability o f a secondary next-hop in the Pathlist. To ensure that multiple paths to the same destination are visible the BGP ADD-PATH [RFC7911] can be used to allow for advertisement of multiple paths for the same address prefix. Dual-homed EVPN/IP-VPN prefixes also have the alternative option of allocating different Route-Distinguishers (RDs). To trigger the switch from primary to secondary next-hop PIC needs to detect the failure and many implementations support 'next-hop tracking' for this purpose. Next-hop tracking monitors the routing-table and if the next-hop prefix is removed will immediately invalidate all BGP prefixes learnt through that next-hop. In the absence of next-hop tracking, multihop BFD [RFC5883] could optionally be used as a fast failure detection mechanism.

5.9.4. Controller Redundancy

With the Interconnect controller providing an integral part of the networks' capabilities a redundant controller design is clearly prudent. To this end we can consider both availability and redundancy. Availability refers to the survivability of a single controller system in a failure scenario. A common strategy for increasing the availability of a single controller system is to build the system in a high-availability cluster such that it becomes a confederation of redundant constituent parts as opposed to a single monolithic system. Should a single part fail, the system can still

survive without the requirement to failover to a standby controller system. Methods for detection of a failure of one or more member parts of the cluster are implementation specific.

To provide contingency for a complete system failure a geo-redundant standby controller system is required. When redundant controllers are deployed a coherent strategy is needed that provides a master/standby election mechanism, the ability to propagate the outcome of that election to network elements as required, an inter-system failure detection mechanism, and the ability to synchronize state across both systems such that the standby controller is fully aware of current state should it need to transition to master controller.

Master/standby election, state synchronisation, and failure detection between geo-redundant sites can largely be considered a local implementation matter. The requirement to propagate the outcome of the master/standby election to network elements depends on a) the mechanism that is used to instantiate SR policies, and b) whether the SR policies are controller-initiated or headend-initiated, and these are discussed in the following sub-sections. In either scenario, state of SR policies should be advertised northbound to both master/standby controllers using either PCEP LSP State Report messages or SR policy extensions to BGP link-state [I-D.ietf-idr-te-lsp-distribution].

5.9.4.1. SR Policy Initiator

Controller-initiated SR policies are suited for auto-creation of tunnels based on service route discovery and policy-driven route/flow programming and are ephemeral. Headend-initiated tunnels allow for permanent configuration state to be held on the headend and are suitable for static services that are not subject to dynamic changes. If all SR policies are controller-initiated, it negates the requirement to propagate the outcome of the master/standby election to network elements. This is because headends have no requirement for unsolicited requests to a controller, and therefore have no requirement to know which controller is master and which one is standby. A headend may respond to a message from a controller, but it is not unsolicited.

If some or all SR policies are headend-initiated, then the requirement to propagate the outcome of the master/standby election exists. This is further discussed in the following sub-section.

5.9.4.2. SR Policy Instantiation Mechanism

While candidate paths of SR policies may be provided using BGP, PCEP, Netconf, or local policy/configuration, this document primarily considers the use of PCEP or BGP.

When PCEP [RFC5440][RFC8231][RFC8281] is used for instantiation of candidate paths of SR policies [I-D.barth-pce-segment-routing-policy-cp] every headend/PCC should establish a PCEP session with the master and standby controllers. To signal standby state to the PCC the standby controller may use a PCEP Notification message to set the PCEP session into overload state. While in this overload state the standby controller will accept path computation LSP state report (PCRpt) messages without delegation but will reject path computation requests (PCReq) and any path computation reports (PCRpt) with the delegation bit set. Further, the standby controller will not path computation originate initiate messages (PCInit) or path computation update request messages (PCUpd). In the event of the failure of the master controller, the standby controller will transition to active and remove the PCEP overload state. Following expiration of the PCEP redelegation timeout at the PCC any LSPs will be redelegated to the newly transitioned active controller. LSP state is not impacted unless redelegation is not possible before the state timeout interval expires.

When BGP is used for instantiation of SR policies every headend should establish a BGP session with the master and standby controller capable of exchanging SR TE Policy SAFI. Candidate paths of SR policies are advertised only by the active controller. If the master controller should experience a failure, then SR policies learnt from that controller may be removed before they are re-advertised by the standby (or newly-active) controller. To minimize this possibility BGP speakers that advertise and instantiate SR policies can implement Long Lived Graceful Retart (LLGR) [I-D.ietf-idr-long-lived-gr], also known as BGP persistence, to retain existing routes treated as least-preferred until the new route arrives. In the absence of LLGR, two other alternatives are possible:

- Provide a static backup SR policy.
- Fallback to the default forwarding path.

5.9.5. Path and Segment Liveliness

When using traffic-engineered SR paths only the ingress router holds any state. The exception here is where BSIDs are used, which also implies some state is maintained at the BSID anchor. As there is no

control plane set-up, it follows that there is no feedback loop from transit nodes of the path to notify the headend when a non-adjacent point of the SR path fails. The Interconnect controller however is aware of all paths that are impacted by a given network failure and should take the appropriate action. This action could include withdrawing an SR policy if a suitable candidate path is already in place, or simply sending a new SR policy with a different segment-list and a higher preference value assigned to it.

Verification of data plane liveliness is the responsibility of the path headend. A given SR policy may be associated with multiple candidate paths and for the sake of clarity, we'll assume two for redundancy purposes (which can be diversely routed). Verification of the liveliness of these paths can be achieved using seamless BFD (S-BFD)[RFC7880], which provides an in-band failure detection mechanism capable of detecting failure in the order of tens of milliseconds. Upon failure of the active path, failover to a secondary candidate path can be activated at the path headend. Details of the actual failover and revert mechanisms are a local implementation matter.

S-BFD provides a fast and scalable failure detection mechanism but is unlikely to be implemented in many VNFs given their inability to offload the process to purpose-built hardware. In the absence of an active failure detection mechanism such as S-BFD the failover from active path to secondary candidate path can be triggered using continuous path validity checks. One of the criteria that a candidate path uses to determine its validity is the ability to perform path resolution for the first SID to one or more outgoing interface(s) and next-hop(s). From the perspective of the VNF headend the first SID in the segment-list will very likely be the DCB (as BSID anchor) but could equally be another Prefix-SID hop within the data center. Should this segment experience a non-recoverable failure, the headend will be unable to resolve the first SID and the path will be considered invalid. This will trigger a failover action to a secondary candidate path.

Injection of S-BFD packets is not just constrained to the source of an end-to-end LSP. When an S-BFD packet is injected into an SR policy path it is encapsulated with the label stack of the associated segment-list. It is possible therefore to run S-BFD from a BSID anchor for just that section of the end-to-end path (for example, from DCB to DCB). This allows a BSID anchor to detect failure of a path and take corrective action, while maintaining opacity between domains.

5.10. Scalability

There are many aspects to consider regarding scalability of the NFIX architecture. The building blocks of NFIX are standards-based technologies individually designed to scale for internet provider networks. When combined they provide a flexible and scalable solution:

- BGP has been proven to scale and operate with millions of routes being exchanged. Specifically, BGP labeled unicast has been deployed and proven to scale in existing seamless-MPLS networks.

- By placing forwarding instructions in the header of a packet, segment routing reduces the amount of state required in the network allowing the scale of greater number of transport tunnels. This aids in the feasibility of the NFIX architecture to permit the automated aspects of SR policy creation without having an impact on the state in the core of the network.

- The choice of utilizing native SR-MPLS or SR over IP in the data center continues to permit horizontal scaling without introducing new state inside of the data center fabric while still permitting seamless end to end path forwarding integration.

- BSIDs play a key role in the NFIX architecture as their use provides the ability to traffic-engineer across large network topologies consisting of many hops regardless of hardware capability at the headend. From a scalability perspective the use of BSIDs facilitates better scale due to the fact that detailed information about the SR paths in a domain has been abstracted and localized to the BSID anchor point only. When BSIDs are re-used amongst one or many headends they reduce the amount of path calculation and updates required at network edges while still providing seamless end to end path forwarding.

- The architecture of NFIX continues to use an independent DC controller. This allows continued independent scaling of data center management in both policy and local forwarding functions, while off-loading the end-to-end optimal path placement and automation to the Interconnect controller. The optimal path placement is already a scalable function provided in a PCE architecture. The Interconnect controller must compute paths, but it is not burdened by the management of virtual entity lifecycle and associated forwarding policies.

It must be acknowledged that with the amalgamation of the technology building blocks and the automation required by NFIX, there is an additional burden on the Interconnect controller. The scaling

considerations are dependent on many variables, but an implementation of a Interconnect controller shares many overlapping traits and scaling concerns as PCE, where the controller and PCE both must:

- Discover and listen to topological state changes of the IP/MPLS topology.
- Compute traffic-engineered intra and inter domain paths across large service provider topologies.
- Synchronize, track and update thousands of LSPs to network devices upon network state changes.

Both entail topologies that contain tens of thousands of nodes and links. The Interconnect controller in an NFIX architecture takes on the additional role of becoming end to end service aware and discovering data center entities that were traditionally excluded from a controllers scope. Although not exhaustive, an NFIX Interconnect controller is impacted by some of the following:

- The number of individual services, the number of endpoints that may exist in each service, the distribution of endpoints in a virtualized environment, and how many data centers may exist. Medium or large sized data centers may be capable to host more virtual endpoints per host, but with the move to smaller edge-clouds the number of headends that require inter-connectivity increases compared to the density of localized routing in a centralized data center model. The outcome has an impact on the number of headend devices which may require tunnel management by the Interconnect controller.
- Assuming a given BSID satisfies SLA, the ability to re-use BSIDs across multiple services reduces the number of paths to track and manage. However, the number of color or unique SLA definitions, and criteria such as bandwidth constraints impacts WAN traffic distribution requirements. As BSIDs play a key role for VNF connectivity, this potentially increases the number of BSID paths required to permit appropriate traffic distribution. This also impacts the number of tunnels which may be re-used on a given headend for different services.
- The frequency of virtualized hosts being created and destroyed and the general activity within a given service. The controller must analyze, track, and correlate the activity of relevant BGP routes to track addition and removal of service host or host subnets, and determine whether new SR policies should be instantiated, or stale unused SR policies should be removed from the network.

- The choice of SR instantiation mechanism impacts the number of communication sessions the controller may require. For example, the BGP based mechanism may only require a small number of sessions to route reflectors, whereas PCEP may require a connection to every possible leaf in the network and any BSID anchors.

- The number of hops within one or many WAN domains may affect the number of BSIDs required to provide transit for VNF/PNF, PNF/PNF, or VNF/VNF inter-connectivity.

- Relative to traditional WAN topologies, traditional data centers are generally topologically denser in node and link connectivity which is required to be discovered by the Interconnect controller, resulting in a much larger, dense link-state database on the Interconnect controller.

5.10.1. Asymmetric Model B for VPN Families

With the instantiation of multiple TE paths between any two VNFs in the NFIX network, the number of SR Policy (remote endpoint, color) routes, BSIDs and labels to support on VNFs becomes a choke point in the architecture. The fact that some VNFs are limited in terms of forwarding resources makes this aspect an important scale issue.

As an example, if VNF1 and VNF2 in Figure 1 are associated to multiple topologies 1..n, the Interconnect controller will instantiate n TE paths in VNF1 to reach VNF2:

[VNF1,color-1,VNF2] --> BSID 1

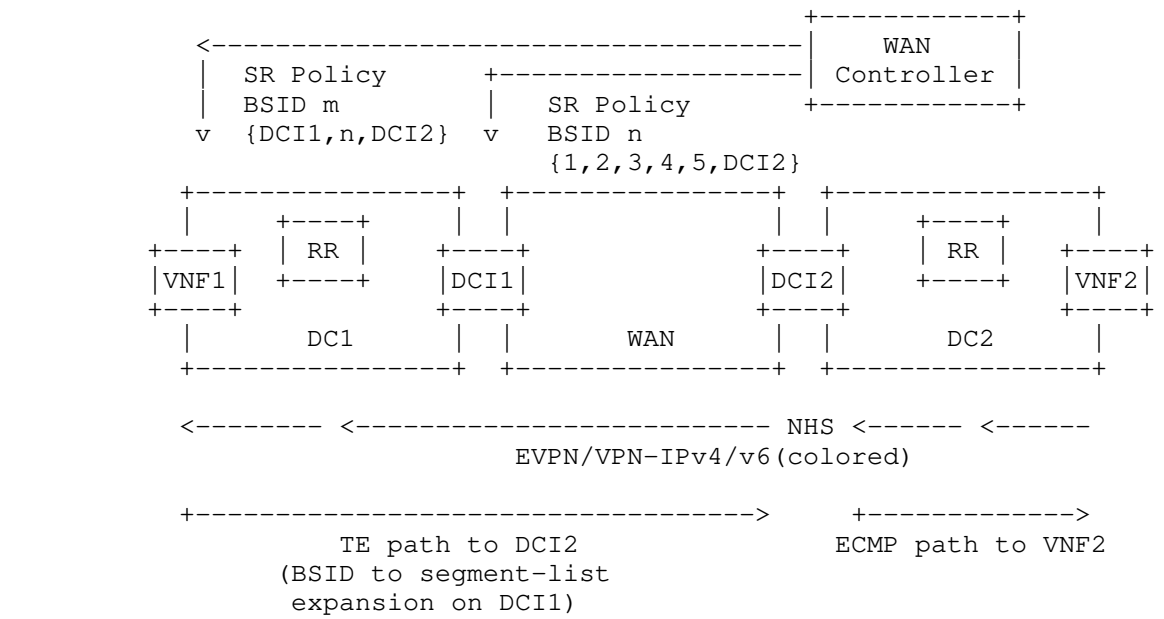
[VNF1,color-2,VNF2] --> BSID 2

...

[VNF1,color-n,VNF2] --> BSID n

Similarly, m TE paths may be instantiated on VNF1 to reach VNF3, another p TE paths to reach VNF4, and so on for all the VNFs that VNF1 needs to communicate with in DC2. As it can be observed, the number of forwarding resources to be instantiated on VNF1 may significantly grow with the number of remote [endpoint, color] pairs, compared with a best-effort architecture in which the number forwarding resources in VNF1 grows with the number of endpoints only.

This scale issue on the VNFs can be relieved by the use of an asymmetric model B service layer. The concept is illustrated in Figure 3.



Asymmetric Model B Service Layer

Figure 4

Consider the different n topologies needed between VNF1 and VNF2 are really only relevant to the different TE paths that exist in the WAN. The WAN is the domain in the network where there can be significant differences in latency, throughput or packet loss depending on the sequence of nodes and links the traffic goes through. Based on that assumption, for traffic from VNF1 to DCB2 in Figure 4, traffic from DCB2 to VNF2 can simply take an ECMP path. In this case an asymmetric model B Service layer can significantly relieve the scale pressure on VNF1.

From a service layer perspective, the NFIX architecture described up to now can be considered 'symmetric', meaning that the EVPN/IPVPN advertisements from e.g., VNF2 in Figure 2, are received on VNF1 with the next-hop of VNF2, and vice versa for VNF1's routes on VNF2. SR Policies to each VNF2 [endpoint, color] are then required on the VNF1.

In the 'asymmetric' service design illustrated in Figure 4, VNF2's EVPN/IPVPN routes are received on VNF1 with the next-hop of DCB2, and VNF1's routes are received on VNF2 with next-hop of DCB1. Now SR

policies instantiated on VNFs can be reduced to only the number of TE paths required to reach the remote DCB. For example, considering n topologies, in a symmetric model VNF1 has to be instantiated with n SR policy paths per remote VNF in DC2, whereas in the asymmetric model of Figure 4, VNF1 only requires n SR policy paths per DC, i.e., to DCB2.

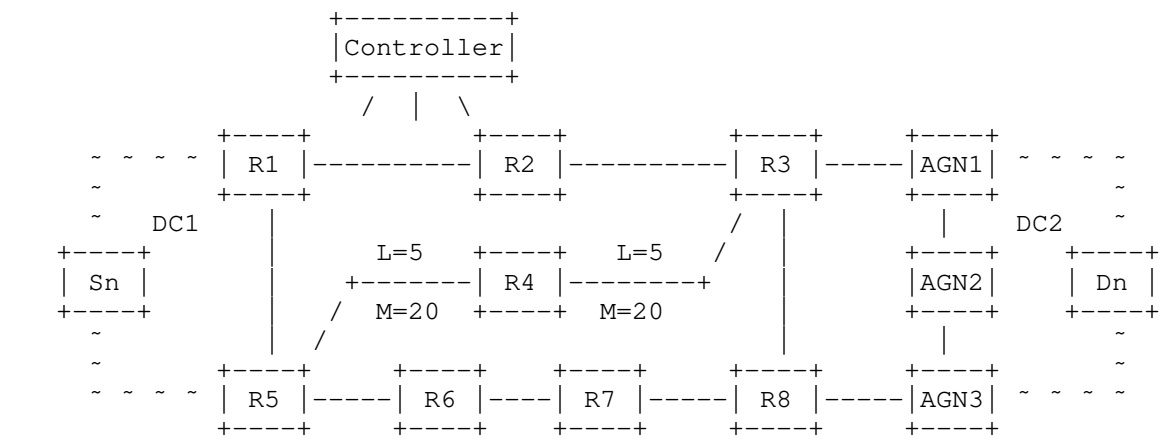
Asymmetric model B is a simple design choice that only requires the ability (on the DCB nodes) to set next-hop-self on the EVPN/IPVPN routes advertised to the WAN neighbors and not do next-hop-self for routes advertised to the DC neighbors. With this option, the Interconnect controller only needs to establish TE paths from VNFs to remote DCBs, as opposed to VNFs to remote VNFs.

6. Illustration of Use

For the purpose of illustration, this section provides some examples of how different end-to-end tunnels are instantiated (including the relevant protocols, SID values/label stacks etc.) and how services are then overlaid onto those LSPs.

6.1. Reference Topology

The following network diagram illustrates the reference network topology that is used for illustration purposes in this section. Within the data centers leaf and spine network elements may be present but are not shown for the purpose of clarity.



Reference Topology

Figure 5

The following applies to the reference topology in figure 5:

- Data center 1 and data center 2 both run BGP/SR. Both data centers run leaf/spine topologies, which are not shown for the purpose of clarity.

- R1 and R5 function as data center border routers for DC 1. AGN1 and AGN3 function as data center border routers for DC 2.

- Routers R1 through R8 form an independent ISIS-OSPF/SR instance.

- Routers R3, R8, AGN1, AGN2, and AGN2 form an independent ISIS-OSPF/SR instance.

- All IGP link metrics within the wide area network are metric 10 except for links R5-R4 and R4-R3 which are both metric 20.

- All links have a unidirectional latency of 10 milliseconds except for links R5-R4 and R4-R3 which both have a unidirectional latency of 5 milliseconds.

- Source 'Sn' and destination 'Dn' represent one or more network functions.

6.2. PNF to PNF Connectivity

The first example demonstrates the simplest form of connectivity; PNF to PNF. The example illustrates the instantiation of a unidirectional TE path from R1 to AGN2 and its consumption by an EVPN service. The service has a requirement for high-throughput with no strict latency requirements. These service requirements are catalogued and represented using the color blue.

- An EVPN service is provisioned at R1 and AGN2.
- The Interconnect controller computes the path from R1 to AGN2 and calculates that the optimal path based on the service requirements and overall network optimization is R1-R5-R6-R7-R8-AGN3-AGN2. The segment-list to represent the calculated path could be constructed in numerous ways. It could be strict hops represented by a series of Adj-SIDs. It could be loose hops using ECMP-aware Node-SIDs, for example {R7, AGN2}, or it could be a combination of both Node-SIDs and Adj-SIDs. Equally, BSIDs could be used to reduce the number of labels that need to be imposed at the headend. In this example, strict Adj-SID hops are used with a BSID at the area border router R8, but this should not be interpreted as the only way a path and segment-list can be represented.
- The Interconnect controller advertises a BGP SR Policy to R8 with BSID 1000, and a segment-list containing segments {AGN3, AGN2}.
- The Interconnect controller advertises a BGP SR Policy to R1 with BSID 1001, and a segment-list containing segments {R5, R6, R7, R8, 1000}. The policy is identified using the tuple [headed = R1, color = blue, endpoint = AGN2].
- AGN2 advertises an EVPN MAC Advertisement Route for MAC M1, which is learned by R1. The route has a next-hop of AGN2, an MPLS label of L1, and it carries a color extended community with the value blue.
- R1 has a valid SR policy [color = blue, next-hop = AGN2] with segment-list {R5, R6, R7, R8, 1000}. R1 therefore associates the MAC address M1 with that policy and programs the relevant information into the forwarding path.
- The Interconnect controller also learns the EVPN MAC Route advertised by AGN2. The purpose of this is two-fold. It allows the controller to correlate the service overlay with the underlying transport LSPs, thus creating a service connectivity map. It also allows the controller to dynamically create LSPs

based upon service requirements if they do not already exist, or to optimize them if network conditions change.

6.3. VNF to PNF Connectivity

The next example demonstrates VNF to PNF connectivity and illustrates the instantiation of a unidirectional TE path from S1 to AGN2. The path is consumed by an IP-VPN service that has a basic set of service requirements and as such simply uses IGP metric as a path computation objective. These basic service requirements are cataloged and represented using the color red.

In this example S1 is a VNF with full IP routing and MPLS capability that interfaces to the data center underlay/overlay and serves as the NVO tunnel endpoint.

- An IP-VPN service is provisioned at S1 and AGN2.
- The Interconnect controller computes the path from S1 to AGN2 and calculates that the optimal path based on IGP metric is R1-R2-R3-AGN1-AGN2.
- The Interconnect controller advertises a BGP SR Policy to R1 with BSID 1002, and a segment-list containing segments {R2, R3, AGN1, AGN2}.
- The Interconnect controller advertises a BGP SR Policy to S1 with BSID 1003, and a segment-list containing segments {R1, 1002}. The policy is identified using the tuple [headend = S1, color = red, endpoint = AGN2].
- Source S1 learns an VPN-IPv4 route for prefix P1, next-hop AGN2. The route has an VPN label of L1, and it carries a color extended community with value red.
- S1 has a valid SR policy [color = red, endpoint = AGN2] with segment-list {R1, 1002} and BSID 1003. S1 therefore associates the VPN-IPv4 prefix P1 with that policy and programs the relevant information into the forwarding path.
- As in the previous example the Interconnect controller also learns the VPN-IPv4 route advertised by AGN2 in order to correlate the service overlay with the underlying transport LSPs, creating or optimizing them as required.

6.4. VNF to VNF Connectivity

The last example demonstrates VNF to VNF connectivity and illustrates the instantiation of a unidirectional TE path from S2 to D2. The path is consumed by an EVPN service that requires low latency as a service requirement and as such uses latency as a path computation objective. This service requirement is cataloged and represented using the color green.

In this example S2 is a VNF that has no routing capability. It is hosted by hypervisor H1 that in turn has an interface to a DC controller through which forwarding instructions are programmed. H1 serves as the NVO tunnel endpoint and overlay next-hop.

D2 is a VNF with partial routing capability that is connected to a leaf switch L1. L1 connects to underlay/overlay in data center 2 and serves as the NVO tunnel endpoint for D2. L1 advertises BGP Prefix-SID 9001 into the underlay.

- The relevant details of the EVPN service are entered in the data center policy engines within data center 1 and 2.

- Source S2 is turned-up. Hypervisor H1 notifies its parent DC controller, which in turn retrieves the service (EVPN) information, color, IP and MAC information from the policy engine and subsequently programs the associated forwarding entries onto S2. The DC controller also dynamically advertises an EVPN MAC Advertisement Route for S2's IP and MAC into the overlay with next-hop H1. (This would trigger the return path set-up between L1 and H2 not covered in this example.)

- The DC controller in data center 1 learns an EVPN MAC Advertisement Route for D2, MAC M, next-nop L1. The route has an MPLS label of L2, and it carries a color extended community with the value green.

- The Interconnect controller computes the path between H1 and L1 and calculates that the optimal path based on latency is R5-R4-R3-AGN1.

- The Interconnect controller advertises a BGP SR Policy to R5 with BSID 1004, and a segment-list containing segments {R4, R3, AGN1}.

- The Interconnect controller advertises a BGP SR Policy to the DC controller in data center 1 with BSID 1005 and a segment-list containing segments {R5, 1004, 9001}. The policy is identified using the tuple [headend = H1, color = green, endpoint = L1].

- The DC controller in data center 1 has a valid SR policy [color = green, endpoint = L1] with segment-list {R5, 1004, 9001} and BSID 1005. The controller therefore associates the MAC Advertisement Route with that policy, and programs the associated forwarding rules into S2.

- As in the previous example the Interconnect controller also learns the MAC Advertisement Route advertised by D2 in order to correlate the service overlay with the underlying transport LSPs, creating or optimizing them as required.

7. Conclusions

The NFIX architecture provides an evolutionary path to a unified network fabric. It uses the base constructs of seamless-MPLS and adds end-to-end LSPs capable of delivering against SLAs, seamless data center interconnect, service differentiation, service function chaining, and a Layer-2/Layer-3 infrastructure capable of interconnecting PNF-to-PNF, PNF-to-VNF, and VNF-to-VNF.

NFIX establishes a dynamic, seamless, and automated connectivity model that overcomes the operational barriers and interworking issues between data centers and the wide-area network and delivers the following using standards-based protocols:

- A unified routing control plane: Multiprotocol BGP (MP-BGP) to acquire inter-domain NLRI from the IP/MPLS underlay and the virtualized IP-VPN/EVPN service overlay.

- A unified forwarding control plane: SR provides dynamic service tunnels with fast restoration options to meet deterministic bandwidth, latency and path diversity constraints. SR utilizes the appropriate data path encapsulation for seamless, end-to-end connectivity between distributed edge and core data centers across the wide-area network.

- Service Function Chaining: Leverage SFC extensions for BGP and segment routing to interconnect network and service functions into SFPs, with support for various data path implementations.

- Service Differentiation: Provide a framework that allows for construction of logical end-to-end networks with differentiated logical topologies and/or constraints through use of SR policies and coloring.

- Automation: Facilitates automation of service provisioning and avoids heavy service interworking at DCBs.

NFIX is deployable on existing data center and wide-area network infrastructures and allows the underlying data forwarding plane to evolve with minimal impact on the services plane.

8. Security Considerations

The NFIX architecture based on SR-MPLS is subject to the same security concerns as any MPLS network. No new protocols are introduced, hence security issues of the protocols encompassed by this architecture are addressed within the relevant individual standards documents. It is recommended that the security framework for MPLS and GMPLS networks defined in [RFC5920] are adhered to. Although [RFC5920] focuses on the use of RSVP-TE and LDP control plane, the practices and procedures are extendable to an SR-MPLS domain.

The NFIX architecture makes extensive use of Multiprotocol BGP, and it is recommended that the TCP Authentication Option (TCP-AO) [RFC5925] is used to protect the integrity of long-lived BGP sessions and any other TCP-based protocols.

Where PCEP is used between controller and path headend the use of PCEPS [RFC8253] is recommended to provide confidentiality to PCEP communication using Transport Layer Security (TLS).

9. Acknowledgements

The authors would like to acknowledge Mustapha Aissaoui, Wim Henderickx, and Gunter Van de Velde.

10. Contributors

The following people contributed to the content of this document and should be considered co-authors.

Juan Rodriguez
Nokia
United States of America

Email: juan.rodriguez@nokia.com

Jorge Rabadan
Nokia
United States of America

Email: jorge.rabadan@nokia.com

Nick Morris
Verizon
United States of America

Email: nicklous.morris@verizonwireless.com

Eddie Leyton
Verizon
United States of America

Email: edward.leyton@verizonwireless.com

Figure 6

11. IANA Considerations

This memo does not include any requests to IANA for allocation.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://xml.resource.org/public/rfc/html/rfc2119.html>.

[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>.

12.2. Informative References

[I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-16 (work in progress), March 2020.

[I-D.ietf-mpls-seamless-mpls]
Leymann, N., Decraene, B., Filsfils, C., Konstantynowicz, M., and D. Steinberg, "Seamless MPLS Architecture", draft-ietf-mpls-seamless-mpls-07 (work in progress), June 2014.

[I-D.ietf-bess-evpn-ipvpn-interworking]
Rabadan, J., Sajassi, A., Rosen, E., Drake, J., Lin, W., Uttaro, J., and A. Simpson, "EVPN Interworking with IPVPN", draft-ietf-bess-evpn-ipvpn-interworking-03 (work in progress), May 2020.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-07 (work in progress), May 2020.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Decraene, B., Francois, P., Voyer, D., Clad, F., and P. Camarillo, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-03 (work in progress), March 2020.

[I-D.ietf-bess-nsh-bgp-control-plane]
Farrel, A., Drake, J., Rosen, E., Uttaro, J., and L. Jalil, "BGP Control Plane for the Network Service Header in Service Function Chaining", draft-ietf-bess-nsh-bgp-control-plane-15 (work in progress), June 2020.

[I-D.ietf-idr-te-lsp-distribution]
Previdi, S., Talaulikar, K., Dong, J., Chen, M., Gredler, H., and J. Tantsura, "Distribution of Traffic Engineering (TE) Policies and State using BGP-LS", draft-ietf-idr-te-lsp-distribution-13 (work in progress), April 2020.

[I-D.barth-pce-segment-routing-policy-cp]
Koldychev, M., Sivabalan, S., Barth, C., Peng, S., and H. Bidgoli, "PCEP extension to support Segment Routing Policy Candidate Paths", draft-barth-pce-segment-routing-policy-cp-06 (work in progress), June 2020.

[I-D.filsfils-spring-sr-policy-considerations]
Filsfils, C., Talaulikar, K., Krol, P., Horneffer, M., and P. Mattes, "SR Policy Implementation and Deployment Considerations", draft-filsfils-spring-sr-policy-considerations-05 (work in progress), April 2020.

[I-D.ietf-rtgwg-bgp-pic]
Bashandy, A., Filsfils, C., and P. Mohapatra, "BGP Prefix Independent Convergence", draft-ietf-rtgwg-bgp-pic-11 (work in progress), February 2020.

[I-D.ietf-isis-mpls-elc]
Xu, X., Kini, S., Psenak, P., Filsfils, C., Litkowski, S., and M. Bocci, "Signaling Entropy Label Capability and Entropy Readable Label Depth Using IS-IS", draft-ietf-isis-mpls-elc-13 (work in progress), May 2020.

[I-D.ietf-ospf-mpls-elc]
Xu, X., Kini, S., Psenak, P., Filsfils, C., Litkowski, S., and M. Bocci, "Signaling Entropy Label Capability and Entropy Readable Label Depth Using OSPF", draft-ietf-ospf-mpls-elc-15 (work in progress), June 2020.

[I-D.ietf-idr-next-hop-capability]
Decraene, B., Kompella, K., and W. Henderickx, "BGP Next-Hop dependent capabilities", draft-ietf-idr-next-hop-capability-05 (work in progress), June 2019.

[I-D.ietf-spring-segment-routing-central-epe]
Filsfils, C., Previdi, S., Dawra, G., Aries, E., and D. Afanasiev, "Segment Routing Centralized BGP Egress Peer Engineering", draft-ietf-spring-segment-routing-central-epe-10 (work in progress), December 2017.

[I-D.ietf-idr-long-lived-gr]
Uttaro, J., Chen, E., Decraene, B., and J. Scudder, "Support for Long-lived BGP Graceful Restart", draft-ietf-idr-long-lived-gr-00 (work in progress), September 2019.

[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>.

[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>.

[RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017, <https://www.rfc-editor.org/info/rfc8277>.

[RFC8667] Previdi, S., Ed., Ginsberg, L., Ed., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", RFC 8667, DOI 10.17487/RFC8667, December 2019, <https://www.rfc-editor.org/info/rfc8667>.

[RFC8665] Psenak, P., Ed., Previdi, S., Ed., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", RFC 8665, DOI 10.17487/RFC8665, December 2019, <https://www.rfc-editor.org/info/rfc8665>.

[RFC8669] Previdi, S., Filsfils, C., Lindem, A., Ed., Sreekantiah, A., and H. Gredler, "Segment Routing Prefix Segment Identifier Extensions for BGP", RFC 8669, DOI 10.17487/RFC8669, December 2019, <https://www.rfc-editor.org/info/rfc8669>.

[RFC8663] Xu, X., Bryant, S., Farrel, A., Hassan, S., Henderickx, W., and Z. Li, "MPLS Segment Routing over IP", RFC 8663, DOI 10.17487/RFC8663, December 2019, <https://www.rfc-editor.org/info/rfc8663>.

[RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <https://www.rfc-editor.org/info/rfc7911>.

[RFC7880] Pignataro, C., Ward, D., Akiya, N., Bhatia, M., and S. Pallagatti, "Seamless Bidirectional Forwarding Detection (S-BFD)", RFC 7880, DOI 10.17487/RFC7880, July 2016, <https://www.rfc-editor.org/info/rfc7880>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010, <https://www.rfc-editor.org/info/rfc5920>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

[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>.

[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>.

[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>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <https://www.rfc-editor.org/info/rfc5440>.

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

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

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

[RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016, <https://www.rfc-editor.org/info/rfc8014>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883, June 2010, <https://www.rfc-editor.org/info/rfc5883>.

[RFC8231] Crabbe, E., Minei, I., Medved, J., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE", RFC 8231, DOI 10.17487/RFC8231, September 2017, <https://www.rfc-editor.org/info/rfc8231>.

[RFC8281] Crabbe, E., Minei, I., Sivabalan, S., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for PCE-Initiated LSP Setup in a Stateful PCE Model", RFC 8281, DOI 10.17487/RFC8281, December 2017, <https://www.rfc-editor.org/info/rfc8281>.

[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>.

[RFC8253] Lopez, D., Gonzalez de Dios, O., Wu, Q., and D. Dhody, "PCEPS: Usage of TLS to Provide a Secure Transport for the Path Computation Element Communication Protocol (PCEP)", RFC 8253, DOI 10.17487/RFC8253, October 2017, <https://www.rfc-editor.org/info/rfc8253>.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <https://www.rfc-editor.org/info/rfc6790>.

[RFC8662] Kini, S., Kompella, K., Sivabalan, S., Litkowski, S., Shakir, R., and J. Tantsura, "Entropy Label for Source Packet Routing in Networking (SPRING) Tunnels", RFC 8662, DOI 10.17487/RFC8662, December 2019, <https://www.rfc-editor.org/info/rfc8662>.

[RFC8491] Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling Maximum SID Depth (MSD) Using IS-IS", RFC 8491, DOI 10.17487/RFC8491, November 2018, <https://www.rfc-editor.org/info/rfc8491>.

[RFC8476] Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling Maximum SID Depth (MSD) Using OSPF", RFC 8476, DOI 10.17487/RFC8476, December 2018, <https://www.rfc-editor.org/info/rfc8476>.

Authors' Addresses

Colin Bookham (editor)
Nokia
740 Waterside Drive
Almondsbury, Bristol
UK

Email: colin.bookham@nokia.com

Andrew Stone
Nokia
600 March Road
Kanata, Ontario
Canada

Email: andrew.stone@nokia.com

Jeff Tantsura
Apstra
333 Middlefield Road #200
Menlo Park, CA 94025
USA

Email: jefftant.ietf@gmail.com

Muhammad Durrani
Equinix Inc
1188 Arques Ave
Sunnyvale CA
USA

Email: mdurrani@equinix.com

Bruno Decraene
Orange
38-40 Rue de General Leclerc
92794 Issey Moulineaux cedex 9
France

Email: bruno.decraene@orange.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 18, 2021

H. Chen
China Telecom
Z. Hu
Huawei Technologies
H. Chen
Futurewei
X. Geng
Huawei Technologies

October 15, 2020

SRv6 Midpoint Protection
draft-chen-rtgwg-srv6-midpoint-protection-03

Abstract

The current local repair mechanism, e.g., TI-LFA, allows local repair actions on the direct neighbors of the failed node to temporarily route traffic to the destination. This mechanism could not work properly when the failure happens in the destination point or the link connected to the destination. In SRv6 TE, the IPv6 destination address in the outer IPv6 header could be the dedicated endpoint of the TE path rather than the destination of the TE path. When the endpoint fails, local repair couldn't work on the direct neighbor of the failed endpoint either. This document defines midpoint protection, which enables the direct neighbor of the failed endpoint to do the function of the endpoint, replace the IPv6 destination address to the other endpoint, and choose the next hop based on the new destination address.

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 18, 2021.

Table of Contents

1. Introduction

The current mechanism, e.g., TI-LFA ([I-D.ietf-rtgwg-segment-routing-ti-lfa]), allows local repair actions on the direct neighbors of the failed node to temporarily route traffic to the destination. This mechanism could not work properly when the failure happens in the destination point or the link connected to the destination. In SRv6 TE, the IPv6 destination address in the outer IPv6 header could be the dedicated endpoint of the TE path rather than the destination of the TE path

([I-D.ietf-spring-srv6-network-programming]). When the endpoint fails, local repair couldn't work on the direct neighbor of the failed endpoint either. This document defines midpoint protection, which enables the direct neighbor of the failed endpoint to do the function of the endpoint, replace the IPv6 destination address to the other endpoint, and choose the next hop based on the new destination address.

2. SRv6 Midpoint Protection Mechanism

When an endpoint node fails, the packet needs to bypass the failed endpoint node and be forwarded to the next endpoint node of the failed endpoint. On the Repair Node (i.e., the previous hop of the failed endpoint node), it performs the proxy forwarding as follows :

- Outbound interface failure happens in the Repair Node;

Case 1: Route to the failed endpoint could be found in the FIB of Repair Node:

- If the Repair Node is not directly connected to the failed endpoint, the normal Ti-LFA is executed;

- If the Repair Node is directly connected to the failed endpoint, the Repair Node forwards the packets through a bypass to the failed endpoint, changing the IPv6 destination address with the IPv6 address of the next, the last or other reasonable endpoint nodes, which could avoid going throw the failed endpoint.

Case 2: Route to the failed endpoint could not be found in the FIB of Repair Node:

- Repair Node forwards the packets through a bypass of the failed endpoint to the next, the last or other reasonable endpoint node directly . There is no need to check whether the failed endpoint node is directly connected to the Repair Node or not.

3. SRv6 Midpoint Protection Example

The topology shown in Figure 1 illustrates an example of network topology with SRv6 enabled on each node.

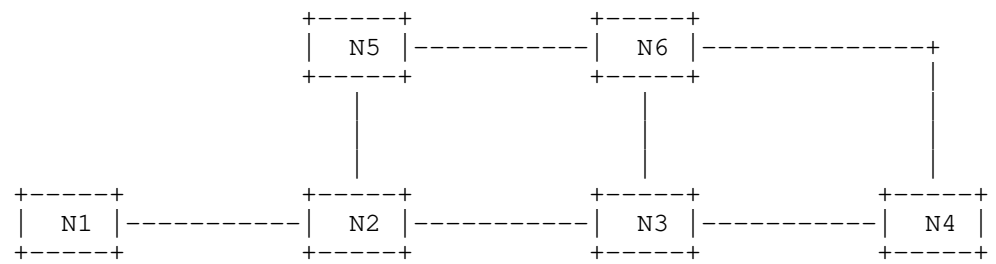


Figure 1: An example of midpoint protection

In this document, an end SID at node n with locator block B is represented as B:n. An end.x SID at node n towards node k with locator block B is represented as B:n:k. A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SRv6 TE path.

In the reference topology:

Node N1 is an ingress node of SRv6 domain. Node N1 steers a packet into a segment list < B:3, B:4>.

When Node N3 fails, the packet needs to bypass the failed endpoint node and be forwarded to the next endpoint node after the failed endpoint in the TE path. When outbound interface failure happens in the Repair Node (which is not limited to the previous hop node of the failed endpoint node), it performs the proxy forwarding as follows,:

For node N2, if the outbound interface to the endpoint B:3 is failed before IGP converges:

- Because node N2, as a Repair Node, is connected to the failed endpoint B:3 directly, node N2 forwards the packets through a bypass of the failed endpoint, changing the IPv6 destination address with the next sid B:4. N2 detects the failure of outbound interface to B:4 in the current route, it could use the normal Ti-LFA repait path to forward the packet, because it is not directly connected to the node N4. N2 encapsulates the packet with the segment list < B:5:6> as a repair path.

For node N1, route to the failed endpoint N3 could not be found in the FIB after IGP converges:

- Node N1, as a Repair Node, forwards the packets through a bypass of the failed endpoint to the next or endpoint node (e.g., N4) directly. There is no need to check whether the failed endpoint

node is directly connected to N1. N1 changes the IPv6 destination address with the next sid B:4. Since IGP has completed convergence, it forwards packets directly based on the IGP SPF path

4. SRv6 Midpoint Protection Behavior

4.1. Transit Node as Repair Node

When the Repair Node is a transit node, it provides fast protection against the endpoint node failure as follows after looking up the FIB.

```
IF the primary outbound interface used to forward the packet failed
  IF NH = SRH && SL != 0, and
     the failed endpoint is directly connected to the Repair Node THEN
    SL decreases*; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    forward the packet according to the backup nexthop;
ELSE // there is no FIB entry for forwarding the packet
  IF NH = SRH && SL != 0 THEN
    SL decreases*; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    drop the packet;
```

*: SL could decrease any dedicated value from [1-N], where N is the current value of SL.
The case is similar in the following examples.

4.2. Endpoint Node as Repair Node

When a node N receives a packet, if the destination address (DA) of the packet is a local END SID, then node N is an endpoint node. When the Repair Node is an endpoint node, it provides fast protections for the failure through executing the following procedure after looking up the FIB for the updated DA.

```
IF the primary outbound interface used to forward the packet failed
  IF NH = SRH && SL != 0, and
     the failed endpoint is directly connected to the Repair Node THEN
    SL decreases; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    forward the packet according to the backup nexthop;
ELSE // there is no FIB entry for forwarding the packet
  IF NH = SRH && SL != 0 THEN
    SL decreases; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    drop the packet;
ELSE
  forward accordingly to the matched entry;
```

4.3. Endpoint x Node as Repair Node

An endpoint node with cross-connect (End.X for short) is an endpoint node with an array of layer 3 adjacencies. When a node N receives a packet, if the destination address (DA) of the packet is a local END.X SID, then node N as Repair Node provides fast protections for the failure through executing the following procedure after updating DA.

```
IF the layer-3 adjacency interface is down THEN
  FIB lookup on the updated DA;
  IF the primary interface used to forward the packet failed THEN
    IF NH = SRH && SL != 0, and
       the failed endpoint directly connected to the Repair Node THEN
      SL decreases; update the IPv6 DA with SRH[SL];
      FIB lookup on the updated DA;
      forward the packet according to the matched entry;
    ELSE
      forward the packet according to the backup nexthop;
  ELSE // there is no FIB entry for forwarding the packet
    IF NH = SRH && SL != 0 THEN
      SL decreases; update the IPv6 DA with SRH[SL];
      FIB lookup on the updated DA;
      forward the packet according to the matched entry;
    ELSE
      drop the packet;
ELSE
  forward accordingly to the matched entry;
```

5. Determining whether the Endpoint could Be Bypassed

SRv6 Midpoint Protection provides a mechanism to bypass a failed endpoint. But in some scenarios, some important functions may be implemented in the bypassed failed endpoints that should not be bypassed, such as firewall functionality or In-situ Flow Information Telemetry of a specified path. Therefore, a mechanism is needed to indicate whether an endpoint can be bypassed or not. [I-D.li-rtgwg-enhanced-ti-lfa] provides method to determine whether enbale SRv6 midpoint protection or not by defining a "no bypass" flag for the SIDs in IGP.

6. Security Considerations

This section reviews security considerations related to SRv6 Midpoint protection processing discussed in this document.To ensure that the Repair node does not modify the SRH header Encapsulated by nodes outside the SRv6 Domain.Only the segment within the SRH is same domain as the repair node. So it is necessary to check the skipped segment have same block as repair node.

7. IANA Considerations

This document makes no request of IANA.

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

8. Acknowledgements

9. References

9.1. Normative References

[I-D.hu-spring-segment-routing-proxy-forwarding]
Hu, Z., Chen, H., Yao, J., Bowers, C., and Y. Zhu, "SR-TE Path Midpoint Protection", draft-hu-spring-segment-routing-proxy-forwarding-11 (work in progress), August 2020.

[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", draft-ietf-isis-segment-routing-extensions-25 (work in progress), May 2019.

[I-D.ietf-lsr-isis-srv6-extensions]
Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extension to Support Segment Routing over IPv6 Dataplane", draft-ietf-lsr-isis-srv6-extensions-11 (work in progress), October 2020.

[I-D.ietf-lsr-ospfv3-srv6-extensions]
Li, Z., Hu, Z., Cheng, D., Talaulikar, K., and P. Psenak, "OSPFv3 Extensions for SRv6", draft-ietf-lsr-ospfv3-srv6-extensions-01 (work in progress), August 2020.

[I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions-27 (work in progress), December 2018.

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-24 (work in progress), October 2020.

[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>.

[RFC7356] Ginsberg, L., Previdi, S., and Y. Yang, "IS-IS Flooding Scope Link State PDUs (LSPs)", RFC 7356, DOI 10.17487/RFC7356, September 2014, <https://www.rfc-editor.org/info/rfc7356>.

9.2. Informative References

[I-D.hegde-spring-node-protection-for-sr-te-paths]
Hegde, S., Bowers, C., Litkowski, S., Xu, X., and F. Xu, "Node Protection for SR-TE Paths", draft-hegde-spring-node-protection-for-sr-te-paths-07 (work in progress), July 2020.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Decraene, B., Francois, P., Voyer, D., Clad, F., and P. Camarillo, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-04 (work in progress), August 2020.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-08 (work in progress), July 2020.

[I-D.li-rtgwg-enhanced-ti-lfa]
Li, C. and Z. Hu, "Enhanced Topology Independent Loop-free Alternate Fast Re-route", draft-li-rtgwg-enhanced-ti-lfa-02 (work in progress), August 2020.

[I-D.sivabalan-pce-binding-label-sid]
Sivabalan, S., Filsfils, C., Tantsura, J., Hardwick, J., Previdi, S., and C. Li, "Carrying Binding Label/Segment-ID in PCE-based Networks.", draft-sivabalan-pce-binding-label-sid-07 (work in progress), July 2019.

[RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <https://www.rfc-editor.org/info/rfc5462>.

Authors' Addresses

Huanan Chen
China Telecom
109, West Zhongshan Road, Tianhe District
Guangzhou 510000
China

Email: chenhuan6@chinatelecom.cn

Zhibo Hu
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: huzhibo@huawei.com

Huaimo Chen
Futurewei
Boston, MA
USA

Email: Huaimo.chen@futurewei.com

Xuesong Geng
Huawei Technologies

Email: gengxuesong@huawei.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: January 26, 2021

L. Dunbar
Futurewei
A. Malis
Malis Consulting
C. Jacquenet
Orange

July 26, 2020

Networks Connecting to Hybrid Cloud DCs: Gap Analysis
draft-ietf-rtgwg-net2cloud-gap-analysis-07

Abstract

This document analyzes the IETF routing area technical gaps that may affect the dynamic connection to workloads and applications hosted in hybrid Cloud Data Centers from enterprise premises.

Table of Contents

1. Introduction

[Net2Cloud-Problem] describes the problems enterprises face today when interconnecting their branch offices with dynamic workloads hosted in third party data centers (a.k.a. Cloud DCs). In particular, this document analyzes the available routing protocols to identify whether there are any gaps that may impede such interconnection which may for example justify additional specification effort to define proper protocol extensions.

For the sake of readability, an edge, C-PE, or CPE are used interchangeably throughout this document. More precisely:

- Edge: may include multiple devices (virtual or physical);
- C-PE: provider-owned edge, e.g. for SECURE-EVPN's PE-based BGP/MPLS VPN, where PE is the edge node;
- CPE: device located in enterprise premises.

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 Overlay controller to manage overlay path creation/deletion and monitor the path conditions between sites.

CPE-Based VPN: Virtual Private Network designed and deployed from CPEs. This is to differentiate from most commonly used PE-based VPNs a la RFC 4364.

OnPrem: On Premises data centers and branch offices

3. Gap Analysis for Accessing Cloud Resources

Because of the ephemeral property of the selected Cloud DCs for specific workloads/Apps, an enterprise or its network service provider may not have direct physical connections to the Cloud DCs that are optimal for hosting the enterprise's specific workloads/Apps. Under those circumstances, an overlay network design can be an option to interconnect the enterprise's on-premises data centers & branch offices to its desired Cloud DCs.

However, overlay paths established over the public Internet can have unpredictable performance, especially over long distances. Therefore, it is highly desirable to minimize the distance or the number of segments that traffic had to be forwarded over the public Internet.

The Metro Ethernet Forum's Cloud Service Architecture [MEF-Cloud] also describes a use case of network operators using Overlay paths over an LTE network or the public Internet for the last mile access where the VPN service providers cannot always provide the required physical infrastructure.

In some scenarios, some overlay edge nodes may not be directly attached to the PEs that participate to the delivery and the operation of the enterprise's VPN.

When using an overlay network to connect the enterprise's sites to the workloads hosted in Cloud DCs, the existing C-PEs at enterprise's sites have to be upgraded to connect to the said overlay network. If the workloads hosted in Cloud DCs need to be connected to many sites, the upgrade process can be very expensive.

[Net2Cloud-Problem] describes a hybrid network approach that extends the existing MPLS-based VPNs to the Cloud DC Workloads over the access paths that are not under the VPN provider's control. To make it work properly, a small number of the PEs of the BGP/MPLS VPN can be designated to connect to the remote workloads via 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 hosted in Cloud DCs can be reached by the enterprise's VPN without upgrading all of the enterprise's CPEs. The

only CPE that needs to connect to the overlay network would be a virtualized CPE instantiated within the cloud DC.

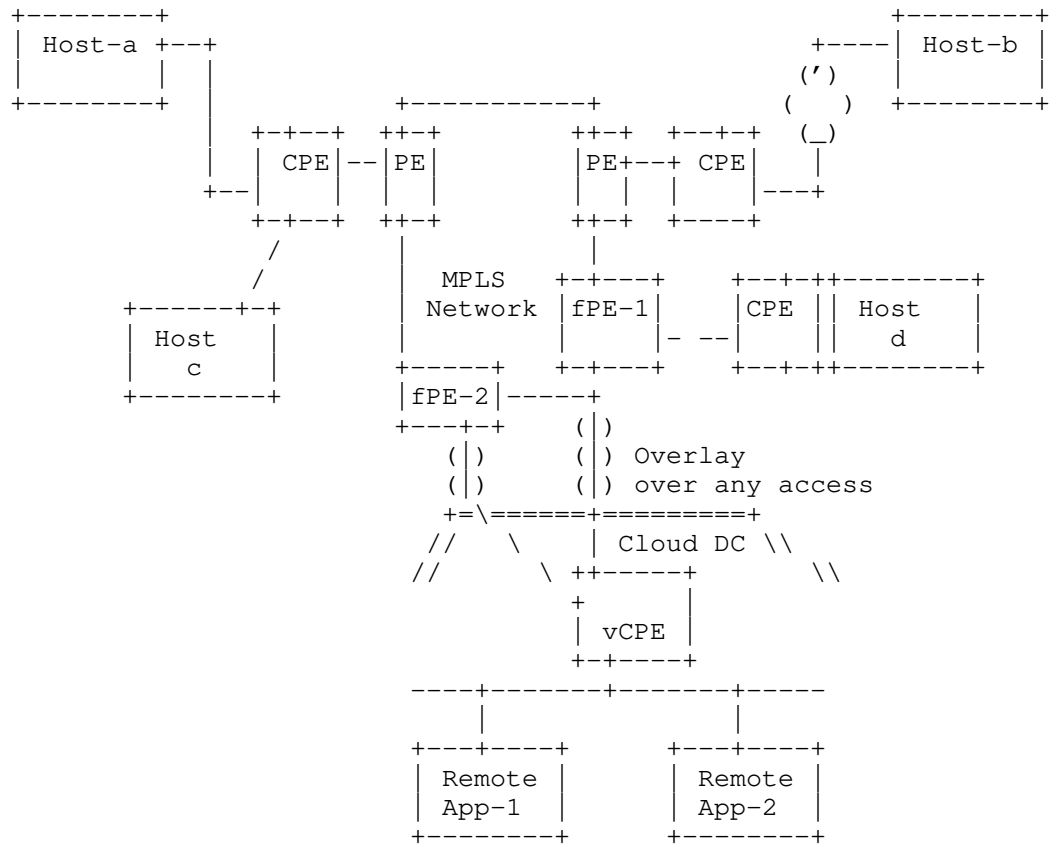


Figure 1: VPN Extension to Cloud DC

In Figure 1, the optimal Cloud DC to host the workloads (as a function of the proximity, capacity, pricing, or any other criteria chosen by the enterprises) does not have a direct connection to the PEs of the NGP/MPLS VPN that interconnects the enterprise's sites.

3.1. Multiple PEs connecting to virtual CPEs in Cloud DCs

To extend BGP/MPLS VPNs to virtual CPEs in Cloud DCs, it is necessary to establish secure tunnels (such as IPsec tunnels) between the PEs and the vCPEs.

Even though a set of PEs can be manually selected for a specific cloud data center, there are no standard protocols for those PEs to interact with the vCPEs instantiated in the third party cloud data centers over unsecure networks. The interaction includes exchanging performance, route information, etc..

When there is more than one PE available for use (as there should be for resiliency purposes or because of the need to support multiple cloud DCs geographically scattered), it is not straightforward to designate an egress PE to remote vCPEs based on applications. It might not be possible for PEs to recognize all applications because too much traffic traversing the PEs.

When there are multiple floating PEs that have established IPsec tunnels with a remote CPE, the remove CPE can forward outbound traffic to the optimal PE, which in turn forwards traffic to egress PEs to reach the final destinations. However, it is not straightforward for the ingress PE to select which egress PEs to send traffic. For example, in Figure 1:

- fPE-1 is the optimal PE for communication between App-1 <-> Host-a due to latency, pricing or other criteria.

- fPE-2 is the optimal PE for communication between App-1 <-> Host-b.

3.2. Access Control for workloads in the Cloud DCs

There is widespread diffusion of access policy for Cloud Resource, some of which is not easy for verification and validation. Because there are multiple parties involved in accessing Cloud Resources, policy enforcement points are not easily visible for policy refinement, monitoring, and testing.

The current state of the art for specifying access policies for Cloud Resources could be improved by having automated and reliable tools to map the user-friendly (natural language) rules into machine readable policies and to provide interfaces for enterprises to self-manage policy enforcement points for their own workloads.

3.3. NAT Traversal

Cloud DCs that only assign private IPv4 addresses to the instantiated workloads assume that traffic to/from the workload usually needs to traverse NATs.

There is no automatic way for an enterprise's network controller to be informed of the NAT properties for its workloads in Cloud DCs

One potential solution could be utilizing the messages sent during initialization of an IKE VPN when NAT Traversal option is enabled. There are some inherent problems while sending IPSec packets through NAT devices. One way to overcome these problems is to encapsulate IPSec packets in UDP. To do this effectively, there is a discovery phase in IKE (Phase1) that tries to determine if either of the IPSec gateways is behind a NAT device. If a NAT device is found, IPSec-over-UDP is proposed during IPSec (Phase 2) negotiation. If there is no NAT device detected, IPSec is used

Another potential solution could be allowing the virtual CPE in Cloud DCs to solicit a STUN (Session Traversal of UDP Through Network Address Translation, [RFC3489]) Server to get the information about the NAT property, the public IP addresses and port numbers so that such information can be communicated to the relevant peers.

3.4. 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 issues about extending BGP from MPLS VPN PEs to remote CPEs in cloud DCs via any access path (e.g., Internet).

The path between the remote CPEs and VPN PEs that maintain VPN routes can traverse untrusted segments.

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

Egress peering engineering (EPE) is not sufficient. Running BGP on virtualized CPEs in Cloud DCs requires GRE tunnels to be established first, which requires the remote CPEs to support address and key management capabilities. RFC 7024 (Virtual Hub & Spoke) and Hierarchical VPN do not support the required properties.

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

EBGP Multi-hop design does not include a security mechanism by default. The PE and remote CPEs need secure communication channels when connecting via the public Internet.

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

3.5. Multicast traffic from/to the remote edges

Among the multiple floating PEs that are reachable from a remote CPE in a Cloud DC, multicast traffic sent by the remote CPE towards the MPLS VPN can be forwarded back to the remote CPE due to the PE receiving the multicast packets forwarding the multicast/broadcast frame to other PEs that in turn send to all attached CPEs. This process may cause traffic loops.

This problem can be solved by selecting one floating PE as the CPE's Designated Forwarder, similar to TRILL's Appointed Forwarders [RFC6325].

BGP/MPLS VPNs do not have features like TRILL's Appointed Forwarders.

4. Gap Analysis of Traffic over Multiple Underlay Networks

Very often the Hybrid Cloud DCs are interconnected by multiple types of underlay networks, such as VPN, public Internet, wireless and wired infrastructures, etc. Sometimes the enterprises' VPN providers do not have direct access to the Cloud DCs that host some specific applications or workloads operated by the enterprise.

When reached by an untrusted network, all sensitive data to/from this virtual CPE have to be encrypted, usually by means of IPsec tunnels. When reached by a trusted direct connect paths, sensitive data can be forwarded without encryption for better performance.

If a virtual CPE in Cloud DC can be reached by both trusted and untrusted paths, better performance can be achieved to have a mixed encrypted and unencrypted traffic depending which paths the traffic is forwarded. However, there is no appropriate control plane protocol to achieve this automatically.

Some networks achieve the IPsec tunnel automation by using the modified NHRP protocol [RFC2332] to register network facing ports of the edge nodes with their Controller (or NHRP server), which then maps a private VPN address to a public IP address of the destination node/port. DSVPN [DSVPN] or DMVPN [DMVPN] are used to establish tunnels between WAN ports of SDWAN edge nodes.

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

- Interworking with the MPLS VPN control plane. An overlay edge can have some ports facing the MPLS VPN network over which packets can be forwarded without any encryption and some ports facing the

public Internet over which sensitive traffic needs to be encrypted.
- Scalability: NHRP/DSVPN/DMVPN work fine with small numbers of edge nodes. When a network has more than 100 nodes, these protocols do not scale well.
- NHRP does not have the IPsec attributes, which are needed for peers to build Security Associations over the public Internet.
- NHRP messages do not have any field to encode the C-PE supported encapsulation types, such as IPsec-GRE or IPsec-VxLAN.
- NHRP messages do not have any field to encode C-PE Location identifiers, such as Site Identifier, System ID, and/or Port ID.
- NHRP messages do not have any field to describe the gateway(s) to which the C-PE is attached. When a C-PE is instantiated in a Cloud DC, it is desirable for the C-PE's owner to be informed about how and where the C-PE is attached.
- NHRP messages do not have any field to describe C-PE's NAT properties if the C-PE is using private IPv4 addresses, such as the NAT type, Private address, Public address, Private port, Public port, etc.

5. Aggregating VPN paths and Internet paths

Most likely, enterprises (especially the largest ones) already have their C-PEs interconnected by VPNs, based upon VPN techniques like EVPN, L2VPN, or L3VPN. Their VPN providers might have direct paths/links to the Cloud DCs that host their workloads and applications.

When there is short term high traffic volume that can't justify increasing the VPNs capacity, enterprises can utilize public internet to reach their Cloud vCPEs. Then it is necessary for the vCPEs to communicate with the controller on how traffic is distributed among multiple heterogeneous underlay networks and to manage secure tunnels over untrusted networks.

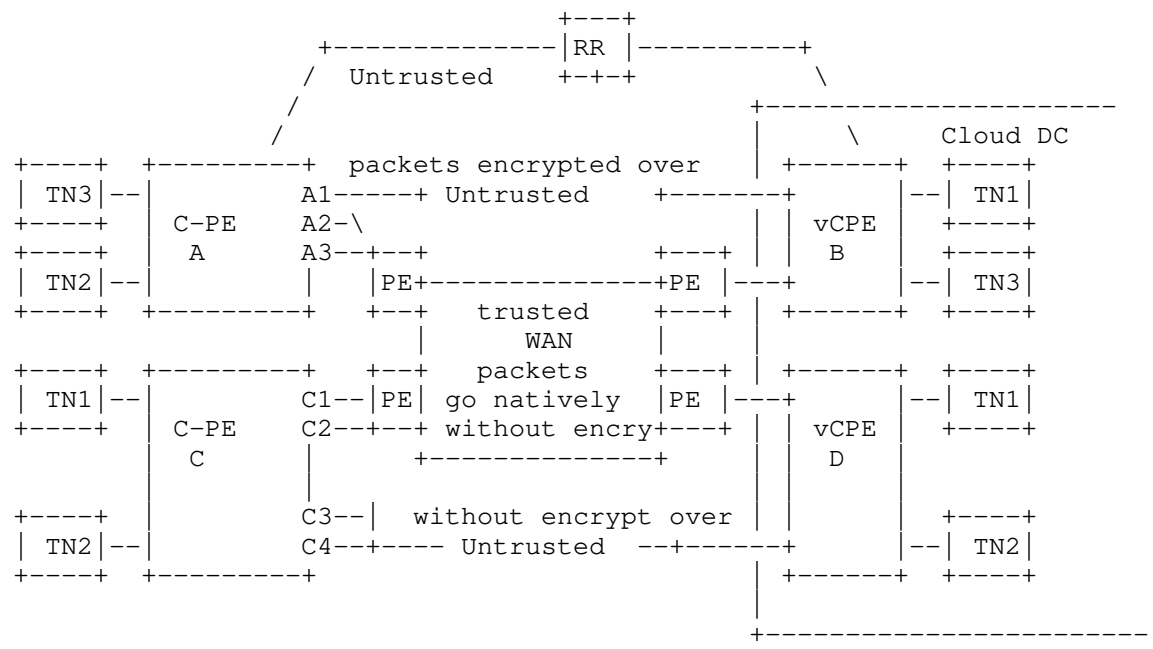


Figure 2: vCPEs reached by Hybrid Paths

5.1. Control Plane for Cloud Access via Heterogeneous Networks

The Control Plane for managing applications and workloads in cloud DCs reachable by heterogeneous networks need to include the following properties:

- vCPE in a cloud DCs needs to communicate with its controller of the properties of the directly connected underlay networks.

- Need Controller-facilitated IPsec SA attributes and NAT information distribution
  - o The controller facilitates and manages the peer authentication for all IPsec tunnels terminated at the vCPEs.

- Establishing and Managing the topology and reachability for services attached to the vCPEs in Cloud DCs.
  - o This is for the overlay layer's route distribution, so that a vCPE can populate its overlay routing table with

entries that identify the next hop for reaching a specific route/service attached to the vCPEs.

5.2. Using BGP UPDATE Messages

5.2.1. Lack ways to differentiate traffic in Cloud DCs

One enterprise can have different types of applications in one Cloud DC. Some can be production applications, some can be testing applications, and some can belong to one specific departments. The traffic to/from different applications might need to traverse different network paths or need to be differentiated by Control plane and data plane.

BGP already has built-in mechanisms, like Route Target, to differentiate different VPNs. But Route Target (RT) is for MPLS based VPNs, therefore RT is not appropriate to directly apply to virtual paths laid over mixed VPNs, IPsec or public underlay networks.

5.2.2. Miss attributes in Tunnel-Encap

[Tunnel-Encap] describes the BGP UPDATE Tunnel Path Attribute that advertises endpoints' tunnel encapsulation capabilities for the respective attached client routes encoded in the MP-NLRI Path Attribute. The receivers of the BGP UPDATE can use any of the supported encapsulations encoded in the Tunnel Path Attribute for the routes encoded in the MP-NLRI Path Attribute.

Here are some of the issues raised by using [Tunnel-Encap] to distribute the property of client routes be carried by mixed of hybrid networks:

- [Tunnel-Encap] doesn't have encoding methods to advertise that a route can be carried by mixed of IPsec tunnels and other already supported tunnels.
- The mechanism defined in [Tunnel-Encap] does not facilitate the exchange of IPsec SA-specific attributes.

5.3. SECURE-EVPN/BGP-EDGE-DISCOVERY

[SECURE-EVPN] describes a solution that utilize BGP as control plane for the Scenario #1 described in [BGP-SDWAN-Usage]. It relies upon a

BGP cluster design to facilitate the key and policy exchange among PE devices to create private pair-wise IPsec Security Associations. [Secure-EVPN] attaches all the IPsec SA information to the actual client routes.

[BGP-Edge-DISCOVERY] proposes BGP UPDATEs from client routers only include the IPsec SA identifiers (ID) to reference the IPsec SA attributes being advertised by separate Underlay Property BGP UPDATE messages. If a client route can be encrypted by multiple IPsec SAs, then multiple IPsec SA IDs are included in the Tunnel-Encap Path attribute for the client route.

[BGP-Edge-DISCOVERY] proposes detailed IPsec SA attributes are advertised in a separate BGP UPDATE for the underlay networks.

[Secure-EVPN] and [BGP-Edge-Discovery] differs in the information included in the client routes. [Secure-EVPN] attaches all the IPsec SA information to the actual client routes, whereas the [BGP-Edge-Discovery] only includes the IPsec SA IDs for the client routes. The IPsec SA IDs used by [BGP-Edge-Discovery] is pointing to the SA-Information which are advertised separately, with all the SA-Information attached to routes which describe the SDWAN underlay, such as WAN Ports or Node address.

5.4. SECURE-L3VPN

[SECURE-L3VPN] describes a method to enrich BGP/MPLS VPN [RFC4364] capabilities to allow some PEs to connect to other PEs via public networks. [SECURE-L3VPN] introduces the concept of Red Interface & Black Interface used by PEs, where the RED interfaces are used to forward traffic into the VPN, and the Black Interfaces are used between WAN ports through which only IPsec-formatted packets are forwarded to the Internet or to any other backbone network, thereby eliminating the need for MPLS transport in the backbone.

[SECURE-L3VPN] assumes PEs use MPLS over IPsec when sending traffic through the Black Interfaces.

[SECURE-L3VPN] is useful, but it misses the aspects of aggregating VPN and Internet underlays. In addition:

- The [SECURE-L3VPN] assumes that a 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. For overlay networks to connect Hybrid Cloud DCs, Zero Touch Provisioning is expected. Manual configuration is not an option.

- The [SECURE-L3VPN] assumes that C-PEs and RRs are connected via an IPsec tunnel. For management channel, TLS/DTLS is more economical than IPsec. The following assumption made by [SECURE-L3VPN] can be difficult to meet in the environment where zero touch provisioning is expected:

  > 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 [SECURE-L3VPN] does not provide any information about how to synchronize the refreshment among multiple nodes.
- IPsec usually sends configuration parameters to two endpoints only and lets these endpoints negotiate the key. The [SECURE-L3VPN] assumes that the RR is responsible for creating/managing the key for all endpoints. When one endpoint is compromised, all other connections may be impacted.

5.5. Preventing attacks from Internet-facing ports

When C-PEs have Internet-facing ports, additional security risks are raised.

To mitigate security risks, in addition to requiring Anti-DDoS features on C-PEs, it is necessary for C-PEs to support means to determine whether traffic sent by remote peers is legitimate to prevent spoofing attacks, in particular.

6. Gap Summary

Here is the summary of the technical gaps discussed in this document:

- For Accessing Cloud Resources

  a) Traffic Path Management: when a remote vCPE can be reached by multiple PEs of one provider VPN network, it is not

straightforward to designate which egress PE to the remote vCPE based on applications or performance.

b) NAT Traversal: There is no automatic way for an enterprise's network controller to be informed of the NAT properties for its workloads in Cloud DCs.

c) There is no loop prevention for the multicast traffic to/from remote vCPE in Cloud DCs.

Needs a feature like Appointed Forwarder specified by TRILL to prevent multicast data frames from looping around.

d) BGP between PEs and remote CPEs via untrusted networks.

- Missing control plane to manage the propagation of the property of networks connected to the virtual nodes in Cloud DCs.

BGP UPDATE propagate client's routes information, but don't distinguish underlay networks.

- Issues of aggregating traffic over private paths and Internet paths

a) Control plane messages for different overlay segmentations needs to be differentiated. User traffic belonging to different segmentations need to be differentiated.

b) BGP Tunnel Encap doesn't have ways to indicate a route or prefix that can be carried by both IPsec tunnels and VPN tunnels

c) Missing clear methods in preventing attacks from Internet-facing ports

7. Manageability Considerations

Zero touch provisioning of overlay networks to interconnect Hybrid Clouds is highly desired. It is necessary for a newly powered up edge node to establish a secure connection (by means of TLS, DTLS, etc.) with its controller.

8. Security Considerations

Cloud Services are built upon shared infrastructures, therefore not secure by nature.

Secure user identity management, authentication, and access control mechanisms are important. Developing appropriate security measurements can enhance the confidence needed by enterprises to fully take advantage of Cloud Services.

9. IANA Considerations

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

10. References

10.1. Normative References

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

10.2. 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-EDGE-DISCOVERY] L. Dunbar, et al, "BGP UPDATE for SDWAN Edge Discovery ", draft-dunbar-idr-sdwan-edge-discovery-00, Work-in-progress, July 2020.

[BGP-SDWAN-Usage] L. Dunbar, et al, "BGP Usage for SDWAN Overlay Networks ", draft-dunbar-bess-bgp-sdwan-usage-08, work-in-progress, July 2020.

[Tunnel-Encap] K. Patel, et al, "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-17, July 2020.

[SECURE-EVPN] A. Sajassi, et al, draft-sajassi-bess-secure-evpn-01, work in progress, March 2019.

[SECURE-L3VPN] 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

11. Acknowledgments

Acknowledgements to John Drake for his review and contributions. Many thanks to John Scudder for stimulating the clarification discussion on the Tunnel-Encap draft so that our gap analysis can be more accurate.

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

Authors' Addresses

Linda Dunbar
Futurewei
Email: ldunbar@futurewei.com

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

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

Network Working Group
Internet Draft
Intended status: Informational
Expires: January 26, 2021

L. Dunbar
Futurewei
Andy Malis
Malis Consulting
C. Jacquenet
Orange
M. Toy
Verizon

July 26, 2020

Dynamic Networks to Hybrid Cloud DCs Problem Statement
draft-ietf-rtgwg-net2cloud-problem-statement-11

Abstract

This document describes the problems that enterprises face today when interconnecting their branch offices with dynamic workloads in third party data centers (a.k.a. Cloud DCs). There can be many problems associated with network connecting to or among Clouds, many of which probably are out of the IETF scope. The objective of this document is to identify some of the problems that need additional work in IETF Routing area. Other problems are out of the scope of this document.

This document focuses on the network problems that many enterprises face when they have workloads & applications & data split among different 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.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that

other groups may also distribute working documents as Internet-Drafts.

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

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

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

This Internet-Draft will expire on January 26, 2009.

Table of Contents

1. Introduction

1.1. Key Characteristics of Cloud Services:

Key characteristics of Cloud Services are on-demand, scalable, highly available, and usage-based billing. Cloud Services, such as, compute, storage, network functions (most likely virtual), third party managed applications, etc. are usually hosted and managed by third parties Cloud Operators. Here are some examples of Cloud network functions: Virtual Firewall services, Virtual private network services, Virtual PBX services including voice and video conferencing systems, etc. Cloud Data Center (DC) is shared infrastructure that hosts the Cloud Services to many customers.

1.2. Connecting to Cloud Services

With the advent of widely available third-party cloud DCs and services in diverse geographic locations and the advancement of tools for monitoring and predicting application behaviors, it is very attractive for enterprises to instantiate applications and workloads in locations that are geographically closest to their end-users. Such proximity can improve 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, enterprises may wish to take advantage of more and more business applications offered by cloud operators.

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

- to access all workloads in the desired cloud DCs.
  Many enterprises include cloud in their disaster recovery strategy, such as enforcing periodic backup policies within the cloud, or running backup applications in the Cloud.

- Global reachability from different geographical zones, thereby facilitating the proximity of applications as a function of the end users' location, to improve latency.
- Elasticity: prompt connection to newly instantiated applications at Cloud DCs when usages increase and prompt release of connection after applications at locations being removed when demands change.
- Scalable security management.

1.3. Reaching App instances in the optimal Cloud DC locations

Many applications have multiple instances instantiated in different Cloud DCs. The current state of the art solutions are typically based on DNS assisted with load balancer by responding a FQDN (Fully Qualified Domain Name) inquiry with an IP address of the closest or lowest cost DC that can reach the instance. Here are some problems associated with DNS based solutions:

- Dependent on client behavior
  - Client can cache results indefinitely
  - Client may not receive service even though there are servers available (before cache timeout) in other Cloud DCs.
- No inherent leverage of proximity information present in the network (routing) layer, resulting in loss of performance

- Client on the west coast can be mapped to a DC on the east coast
- Inflexible traffic control:
  - Local DNS resolver become the unit of traffic management

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 and workloads split among Cloud DCs owned or managed by different operators.

Hybrid Clouds: Hybrid Clouds refers to an enterprise using its own on-premises DCs in addition to Cloud services provided by one or more cloud operators. (e.g. AWS, Azure, Google, Salesforces, SAP, etc).

VPC: Virtual Private Cloud is a virtual network dedicated to one client account. It is logically isolated from other virtual networks in a Cloud DC. Each client can launch his/her desired resources, such as compute, storage, or network functions into his/her VPC. Most Cloud operators' VPCs only support private addresses, some support IPv4 only, others support IPv4/IPv6 dual stack.

3. High Level Issues of Connecting to Multi-Cloud

There are many problems associated with connecting to hybrid Cloud Services, many of which are out of the IETF scope. This section is to identify some of the high-level problems that can be addressed by IETF, especially by Routing area. Other problems are out of the scope of this document. By no means has this section covered all problems for connecting to Hybrid Cloud Services, e.g. difficulty in managing cloud spending is not discussed here.

3.1. Security Issues

Cloud Services is built upon shared infrastructure, therefore not secure by nature. Security has been a primary, and valid, concern from the start of cloud computing, e.g. not being able to see the exact location where the data are stored or trace of access. Headlines highlighting data breaches, compromised credentials, and broken authentication, hacked interfaces and APIs, account hijacking haven't helped alleviate concerns.

Many Cloud operators offer monitoring services for data stored in Clouds, such as AWS CloudTrail, Azure Monitor, and many third-party monitoring tools to improve visibility to data stored in Clouds. But there is still underline security concerns on illegitimate data and workloads access.

Secure user identity management, authentication, and access control mechanisms are important. Developing appropriate security measurements can enhance the confidence needed by enterprises to fully take advantage of Cloud Services.

3.2. Authorization and Identity Management

One of the more prominent challenges for Cloud Services is Identity Management and Authorization. The Authorization not only includes user authorization, but also the authorization of API calls by applications from different Cloud DCs managed by different Cloud Operators. In addition, there are authorization for Workload Migration, Data Migration, and Workload Management.

There are many types of users in cloud environments, e.g. end users for accessing applications hosted in Cloud DCs, Cloud-resource users

who are responsible for setting permissions for the resources based on roles, access lists, IP addresses, domains, etc.

There are many types of Cloud authorizations: including MAC (Mandatory Access Control) - where each app owns individual access permissions, DAC (Discretionary Access Control) - where each app requests permissions from an external permissions app, RBAC (Role-based Access Control) - where the authorization service owns roles with different privileges on the cloud service, and ABAC (Attribute-based Access Control) - where access is based on request attributes and policies.

IETF hasn't yet developed comprehensive specification for Identity management and data models for Cloud Authorizations.

3.3. API abstraction

Different Cloud Operators have different APIs to access their Cloud resources, security functions, the NAT, etc.

It is difficult to move applications built by one Cloud operator's APIs to another. However, it is highly desirable to have a single and consistent way to manage the networks and respective security policies for interconnecting applications hosted in different Cloud DCs.

The desired property would be having a single network fabric to which different Cloud DCs and enterprise's multiple sites can be attached or detached, with a common interface for setting desired policies.

The difficulty of connecting applications in different Clouds might be stemmed from the fact that they are direct competitors. Usually traffic flow out of Cloud DCs incur charges. Therefore, direct communications between applications in different Cloud DCs can be more expensive than intra Cloud communications.

It is desirable to have a common API shim layer or abstraction for different Cloud providers to make it easier to move applications from one Cloud DC to another.

3.4. DNS for Cloud Resources

DNS name resolution is essential for on-premises and cloud-based resources. For customers with hybrid workloads, which include on-premises and cloud-based resources, extra steps are necessary to configure DNS to work seamlessly across both environments.

Cloud operators have their own DNS to resolve resources within their Cloud DCs and to well-known public domains. Cloud's DNS can be configured to forward queries to customer managed authoritative DNS servers hosted on-premises, and to respond to DNS queries forwarded by on-premises DNS servers.

For enterprises utilizing Cloud services by different cloud operators, it is necessary to establish policies and rules on how/where to forward DNS queries to. When applications in one Cloud need to communication with applications hosted in another Cloud, there could be DNS queries from one Cloud DC being forwarded to the enterprise's on premise DNS, which in turn be forwarded to the DNS service in another Cloud. Needless to say, configuration can be complex depending on the application communication patterns.

However, even with carefully managed policies and configurations, collisions can still occur. If you use an internal name like .cloud and then want your services to be available via or within some other cloud provider which also uses .cloud, then it can't work. Therefore, it is better to use the global domain name even when an organization does not make all its namespace globally resolvable. An organization's globally unique DNS can include subdomains that cannot be resolved at all outside certain restricted paths, zones that resolve differently based on the origin of the query, and zones that resolve the same globally for all queries from any source.

Globally unique names do not equate to globally resolvable names or even global names that resolve the same way from every perspective. Globally unique names do prevent any possibility of collision at the present or in the future and they make DNSSEC trust manageable. Consider using a registered and fully qualified domain name (FQDN) from global DNS as the root for enterprise and other internal namespaces.

3.5. NAT for Cloud Services

Cloud resources, such as VM instances, are usually assigned with private IP addresses. By configuration, some private subnets can have the NAT function to reach out to external network and some private subnets are internal to Cloud only.

Different Cloud operators support different levels of NAT functions. For example, AWS NAT Gateway does not currently support connections towards, or from VPC Endpoints, VPN, AWS Direct Connect, or VPC Peering. https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-nat-gateway.html#nat-gateway-other-services. AWS Direct Connect/VPN/VPC Peering does not currently support any NAT functionality.

Google's Cloud NAT allows Google Cloud virtual machine (VM) instances without external IP addresses and private Google Kubernetes Engine (GKE) clusters to connect to the Internet. Cloud NAT implements outbound NAT in conjunction with a default route to allow instances to reach the Internet. It does not implement inbound NAT. Hosts outside of VPC network can only respond to established connections initiated by instances inside the Google Cloud; they cannot initiate their own, new connections to Cloud instances via NAT.

For enterprises with applications running in different Cloud DCs, proper configuration of NAT has to be performed in Cloud DC and in their own on-premise DC.

3.6. Cloud Discovery

One of the concerns of using Cloud services is not aware where the resource is actually located, especially Cloud operators can move application instances from one place to another. When applications in Cloud communicate with on-premise applications, it may not be clear where the Cloud applications are located or to which VPCs they belong.

It is highly desirable to have tools to discover cloud services in much the same way as you would discover your on-premises infrastructure. A significant difference is that cloud discovery uses the cloud vendor's API to extract data on your cloud services, rather than the direct access used in scanning your on-premises infrastructure.

Standard data models, APIs or tools can alleviate concerns of enterprise utilizing Cloud Resources, e.g. having a Cloud service scan that connects to the API of the cloud provider and collects information directly.

4. Interconnecting Enterprise Sites with Cloud DCs

Considering that many enterprises already have existing VPNs (e.g. MPLS based L2VPN or L3VPN) interconnecting branch offices & on-premises data centers, connecting to Cloud services will be mixed of different types of networks. When an enterprise's existing VPN service providers do not have direct connections to the corresponding cloud DCs that the enterprise prefers to use, the enterprise has to face additional infrastructure and operational costs to utilize Cloud services.

4.1. Sites to Cloud DC

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

- AWS Internet gateway allows communication between instances in AWS VPC and the internet.
- AWS 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).
- AWS 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. In addition, an AWS Transit Gateway can be used to interconnect multiple VPCs in different Availability Zones. AWS Transit Gateway acts as a hub that controls how traffic is forwarded among all the connected networks which act like spokes.

Microsoft's ExpressRoute allows extension of a private network to any of the Microsoft cloud services, including Azure and Office365. ExpressRoute is configured using Layer 3 routing. Customers can opt for redundancy by provisioning dual links from their location to two Microsoft Enterprise edge routers (MSEEs) located within a third-party ExpressRoute peering location. The BGP routing protocol is then setup over WAN links to provide redundancy to the cloud. This redundancy is maintained from the peering data center into Microsoft's cloud network.

Google's Cloud Dedicated Interconnect offers similar network connectivity options as AWS and Microsoft. One distinct difference, however, is that Google's service allows customers access to the entire global cloud network by default. It does this by connecting your on-premises network with the Google Cloud using BGP and Google Cloud Routers to provide optimal paths to the different regions of the global cloud infrastructure.

Figure below shows an example of some of a tenant's workloads are accessible via a virtual router connected by AWS Internet Gateway; some are accessible via AWS vGW, and others are accessible via AWS Direct Connect.

Different types of access require different level of security functions. Sometimes it is not visible to end customers which type of network access is used for a specific application instance. To get better visibility, separate virtual routers (e.g. vR1 & vR2) can be deployed to differentiate traffic to/from different cloud GWs. It is important for some enterprises to be able to observe the specific behaviors when connected by different connections.

Customer Gateway can be customer owned router or ports physically connected to AWS Direct Connect GW.

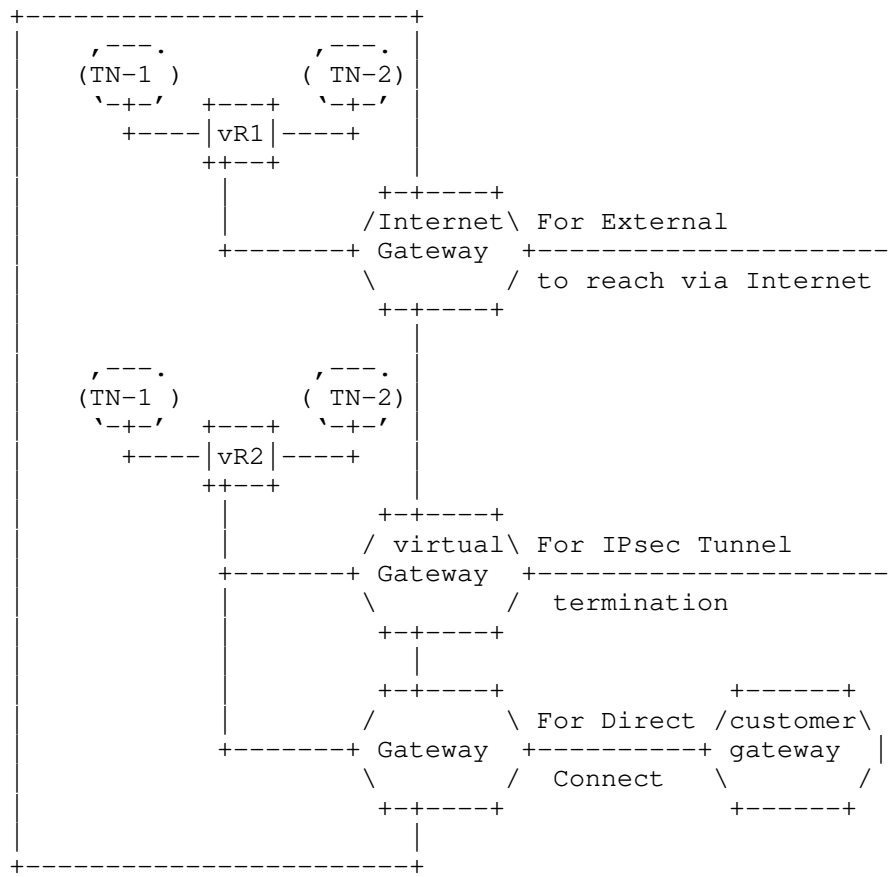


Figure 1: Examples of Multiple Cloud DC connections.

4.2. Inter-Cloud Interconnection

The connectivity options to Cloud DCs described in the previous section are for reaching Cloud providers' DCs, but not between cloud DCs. When applications in AWS Cloud need to communicate with applications in Azure, today's practice requires a third-party gateway (physical or virtual) to interconnect the AWS's Layer 2 DirectConnect path with Azure's Layer 3 ExpressRoute.

Enterprises can also instantiate their own virtual routers in different Cloud DCs and administer IPsec tunnels among them, which by itself is not a trivial task. Or by leveraging open source VPN software such as strongSwan, you create an IPSec connection to the Azure gateway using a shared key. The StrongSwan instance within AWS

not only can connect to Azure but can also be used to facilitate traffic to other nodes within the AWS VPC by configuring forwarding and using appropriate routing rules for the VPC.

Most Cloud operators, such as AWS VPC or Azure VNET, use non-globally routable CIDR from private IPv4 address ranges as specified by RFC1918. To establish IPsec tunnel between two Cloud DCs, it is necessary to exchange Public routable addresses for applications in different Cloud DCs.

In summary, here are some approaches, available now (which might change in the future), to interconnect workloads among different Cloud DCs:

a) Utilize Cloud DC provided inter/intra-cloud connectivity services (e.g., AWS Transit Gateway) to connect workloads instantiated in multiple VPCs. Such services are provided with the cloud gateway to connect to external networks (e.g., AWS DirectConnect Gateway).

b) Hairpin all traffic through the customer gateway, meaning all workloads are directly connected to the customer gateway, so that communications among workloads within one Cloud DC must traverse through the customer gateway.

c) Establish direct tunnels among different VPCs (AWS' Virtual Private Clouds) and VNET (Azure's Virtual Networks) 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.

Approach a) usually does not work if Cloud DCs are owned and managed by different Cloud providers.

Approach b) creates additional transmission delay plus incurring cost when exiting Cloud DCs.

For the Approach c), DMVPN or 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 cannot be used for registering virtual routers in Cloud DCs unless an extension of such protocols is developed for that purpose, e.g. taking NAT or dynamic addresses into consideration. Therefore, DMVPN and/or DSVPN cannot be used directly for connecting workloads in hybrid Cloud DCs.

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, because 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 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. An enterprise with a hybrid cloud deployment can use an MPLS-VPN to connect to a Cloud provider at multiple locations. The connection locations often correspond to gateways of different Cloud DC locations from the Cloud provider. The different Cloud DCs are interconnected by the Cloud provider's own internal network. At each connection location (gateway), the Cloud provider uses BGP to advertise all of the prefixes in the enterprise's VPC, regardless of which Cloud DC a given prefix is actually in. This can result in inefficient routing for the end-to-end data path.

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.

6.1. Scaling Issues with 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 have multiple sites and multiple data centers. For multiple sites to communicate with workloads and apps hosted in cloud DCs, Cloud DC gateways have to maintain many IPsec tunnels to all those locations. In addition, each of those IPsec Tunnels requires pair-wise periodic key refreshment. 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 very processing intensive. That is why 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 is key distribution and maintenance of a key server, etc.

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.

7. 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.

8. 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.

9. 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.

10. 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.

12. Acknowledgments

Many thanks to Alia Atlas, Chris Bowers, Paul Vixie, Paul Ebersman, Timothy Morizot, Ignas Bagdonas, Michael Huang, Liu Yuan Jiao, Katherine Zhao, and Jim Guichard for the discussion and contributions.

Authors' Addresses

Linda Dunbar
Futurewei
Email: Linda.Dunbar@futurewei.com

Andrew G. Malis
Malis Consulting
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 9, 2021

A. Choudhary
Cisco Systems
M. Jethanandani
VMware
N. Strahle
E. Aries
Juniper Networks
I. Chen
The MITRE Corporation

July 08, 2020

YANG Model for QoS
draft-ietf-rtgwg-qos-model-02

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2021.

Copyright Notice

Table of Contents

1. Introduction

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

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

Editorial Note: (To be removed by RFC Editor)

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

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

1.1. Tree Diagrams

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

2. Terminology

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

3. QoS Model Design

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

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

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

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

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

This module imports definitions from "Common YANG Data Types" [RFC6991] and "A YANG Data Model for Interface Management" [RFC8343].

4. DiffServ Model Design

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

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

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

5. Modules Tree Structure

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
          |     +--rw protocol-max    uint8
          +--:(traffic-group)
             +--rw traffic-group-cfg
                +--rw traffic-group-name?   string
     augment /policy:policies/policy:policy-entry +
             /policy:classifier-entry/policy:filter-entry:
       +--rw (filter-params)?
          +--:(dscp)
          |  +--rw dscp-cfg* [dscp-min dscp-max]
          |     +--rw dscp-min    inet:dscp
          |     +--rw dscp-max    inet:dscp
          +--:(source-ipv4-address)
          |  +--rw source-ipv4-address-cfg* [source-ipv4-addr]
          |     +--rw source-ipv4-addr    inet:ipv4-prefix
          +--:(destination-ipv4-address)
          |  +--rw destination-ipv4-address-cfg* [destination-ipv4-addr]
          |     +--rw destination-ipv4-addr    inet:ipv4-prefix
          +--:(source-ipv6-address)
          |  +--rw source-ipv6-address-cfg* [source-ipv6-addr]
          |     +--rw source-ipv6-addr    inet:ipv6-prefix
          +--:(destination-ipv6-address)
          |  +--rw destination-ipv6-address-cfg* [destination-ipv6-addr]
          |     +--rw destination-ipv6-addr    inet:ipv6-prefix
          +--:(source-port)
          |  +--rw source-port-cfg* [source-port-min source-port-max]
          |     +--rw source-port-min    inet:port-number
          |     +--rw source-port-max    inet:port-number
          +--:(destination-port)
          |  +--rw destination-port-cfg*
                      [destination-port-min destination-port-max]
          |     +--rw destination-port-min    inet:port-number
          |     +--rw destination-port-max    inet:port-number
          +--:(protocol)
          |  +--rw protocol-cfg* [protocol-min protocol-max]
          |     +--rw protocol-min    uint8
          |     +--rw protocol-max    uint8
          +--:(traffic-group)
             +--rw traffic-group-cfg
                +--rw traffic-group-name?   string
     augment /policy:policies/policy:policy-entry +
             /policy:classifier-entry +
             /policy:classifier-action-entry-cfg +
             /policy:action-cfg-params:
       +--:(dscp-marking)
       |  +--rw dscp-cfg
       |     +--rw dscp?   inet:dscp
       +--:(meter-inline) {action:meter-inline-feature}?
       |  +--rw (meter-type)?
```

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

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

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

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

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

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

6. Modules

6.1. IETF-QOS-CLASSIFIER

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

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

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

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

  feature classifier-template-feature {
```

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

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

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

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

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

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

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

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

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

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

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

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

6.2. IETF-QOS-POLICY

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

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

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

```
            uses classifier-action-entry-cfg;
          }
        }
      }
    }
    <CODE ENDS>
```

6.3. IETF-QOS-ACTION

```
    <CODE BEGINS>file "ietf-qos-action@2019-03-13.yang"
    module ietf-qos-action {
      yang-version 1.1;
      namespace "urn:ietf:params:xml:ns:yang:ietf-qos-action";
      prefix action;
      import ietf-inet-types {
        prefix inet;
        reference "RFC 6991: Common YANG Data Types";
      }
      import ietf-qos-policy {
        prefix policy;
        reference "RFC XXXX: YANG Model for QoS";
      }
      organization "IETF RTG (Routing Area) Working Group";
      contact
        "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
         WG List:  <mailto:rtgwg@ietf.org>
         Editor:   Aseem Choudhary
                   <mailto:asechoud@cisco.com>
         Editor:   Mahesh Jethanandani
                   <mailto:mjethanandani@gmail.com>
         Editor:   Norm Strahle
                   <mailto:nstrahle@juniper.net>";
      description
        "This module contains a collection of YANG definitions for
         configuring qos specification implementations.
         Copyright (c) 2019 IETF Trust and the persons identified as
         authors of the code.  All rights reserved.
         Redistribution and use in source and binary forms, with or
         without modification, is permitted pursuant to, and subject
         to the license terms contained in, the Simplified BSD License
         set forth in Section 4.c of the IETF Trust's Legal Provisions
         Relating to IETF Documents
         (http://trustee.ietf.org/license-info).
         This version of this YANG module is part of RFC XXXX; see
         the RFC itself for full legal notices.";
      revision 2019-03-13 {
        description
          "Latest revision for qos actions";
```

```
      reference "RFC XXXX: YANG Model for QoS";
    }
    feature meter-template-support {
      description
        " This feature allows support of meter-template.";
    }
    feature meter-inline-feature {
      description
        "This feature allows support of meter-inline configuration.";
    }
    feature meter-reference-feature {
      description
        "This feature allows support of meter by reference
         configuration.";
    }
    feature queue-action-support {
      description
        " This feature allows support of queue action configuration
          in policy.";
    }
    feature scheduler-action-support {
      description
        " This feature allows support of scheduler configuration
          in policy.";
    }
    feature child-policy-feature {
      description
        " This feature allows configuration of hierarchical policy.";
    }
    feature count-feature {
      description
        "This feature allows action configuration to enable
         counter in a classifier";
    }
    feature named-counter-feature {
      description
        "This feature allows action configuration to enable
         named counter in a classifier";
    }
    feature traffic-group-feature {
      description
        "traffic-group action support";
    }
    feature burst-time-unit-support {
      description
        "This feature allows burst unit to be configured as
         time duration.";
    }
```

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

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

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

    identity meter-type {
      description
        "This base identity type defines meter types";
    }
    identity one-rate-two-color-meter-type {
      base meter-type;
      description
        "one rate two color meter type";
    }
    identity one-rate-tri-color-meter-type {
      base meter-type;
      description
        "one rate three color meter type";
      reference
          "RFC2697: A Single Rate Three Color Marker";
    }
    identity two-rate-tri-color-meter-type {
      base meter-type;
      description
        "two rate three color meter action type";
      reference
        "RFC2698: A Two Rate Three Color Marker";
    }

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

```
        "tail drop algorithm";
    }

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

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

```
      leaf burst-unit {
        type identityref {
          base burst-unit-type;
        }
        description
          "burst unit";
      }
    }

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

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

```
      description
        "always drop grouping";
    }

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

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

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

```
          description
            "meter action type";
        }
        choice exceed-2color-meter-action-val {
          description
            " meter action based on choice of meter action type";
        }
      }
    }


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

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

```
        choice exceed-3color-meter-action-val {
          description
            " meter action based on choice of meter action type";
        }
      }
    }

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

    grouping one-rate-two-color-meter {
      container one-rate-two-color-meter {
        description
          "single rate two color marker meter";
        reference
          "RFC 2475, 2.3.3.1: An Architecture for Differentiated
           Services: Meters.";
        leaf committed-rate-value {
          type uint64;
          description
            "committed rate value";
        }
        leaf committed-rate-unit {
          type identityref {
            base rate-unit-type;
          }
          description
            "committed rate unit";
        }
```

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

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

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

    grouping two-rate-tri-color-meter {
      container two-rate-tri-color-meter {
        description
          "two rate three color meter";
        reference
          "RFC2698: A Two Rate Three Color Marker";
      leaf committed-rate-value {
```

```
        type uint64;
        units "bits-per-second";
        description
          "committed rate";
      }
      leaf committed-rate-unit {
        type identityref {
          base rate-unit-type;
        }
        description
          "committed rate unit";
      }
      leaf committed-burst-value {
        type uint64;
        description
          "commited burst size";
      }
      leaf committed-burst-unit {
        type identityref {
          base burst-unit-type;
        }
        description
          "committed burst unit";
      }
      leaf peak-rate-value {
        type uint64;
        description
          "peak rate";
      }
      leaf peak-rate-unit {
        type identityref {
          base rate-unit-type;
        }
        description
          "committed rate unit";
      }
      leaf peak-burst-value {
        type uint64;
        description
          "commited burst size";
      }
      leaf peak-burst-unit {
        type identityref {
          base burst-unit-type;
        }
        description
          "peak burst unit";
      }
```

```
        container conform-action {
          uses conform-3color-meter-action-params;
          description
            "conform, or green action";
        }
        container exceed-action {
          uses exceed-3color-meter-action-params;
          description
            "exceed, or yellow action";
        }
        container violate-action {
          uses violate-3color-meter-action-params;
          description
            "exceed, or red action";
        }
      }
        description
          "two-rate-tri-color-meter attributes";
      }

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

      container meter-template {
        description
          "list of meter templates";
        list meter-entry {
```

```
        if-feature meter-template-support;
        key "meter-name";
        description
          "meter entry template";
        leaf meter-name {
          type string;
          description
            "meter identifier";
        }
        uses meter;
      }
    }

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

    grouping count {
      container count-cfg {
      if-feature count-feature;
        leaf count-action {
          type empty;
          description
            "count action";
        }
        description
          "the count action";
      }
      description
```

```
        "the count action grouping";
    }

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

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

    grouping priority {
      container priority-cfg {
        leaf priority-level {
          type uint8;
          description
            "priority level";
        }
        description
          "priority attributes";
      }
      description
        "priority attributes grouping";
    }
    grouping min-rate {
      container min-rate-cfg {
        uses rate-value-unit;
```

```
        description
          "min guaranteed bandwidth";
        reference
          "RFC3289, section 3.5.3";
      }
      description
        "minimum rate grouping";
    }
    grouping dscp-marking {
      container dscp-cfg {
        leaf dscp {
          type inet:dscp;
          description
            "dscp marking";
        }
        description
          "dscp marking container";
      }
      description
        "dscp marking grouping";
      reference
        "RFC 2474: Definition of the Differentiated Services
                   Field (DS Field) in the IPv4 and IPv6 Headers.";
    }
    grouping traffic-group-marking {
      container traffic-group-cfg {
        leaf traffic-group {
          type string;
          description
            "traffic group marking";
        }
        description
          "traffic group marking container";
      }
      description
        "traffic group marking grouping";
    }
    grouping child-policy {
      container child-policy-cfg {
        if-feature child-policy-feature;
        leaf policy-name {
          type string;
          description
            "Hierarchical Policy";
        }
        description
          "Hierarchical Policy configuration container";
      }
```

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

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

6.4. IETF-QOS-TARGET

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

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

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

```
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).
        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

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

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

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

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

    augment "/if:interfaces/if:interface" {
      description
        "Augments QoS Target Entry to Interface module.
         If an interface is not capable of applying the QoS parameters,
         the server must not allow the client to configure these QoS
         parameters.";

      list qos-target-entry {
        key "direction policy-type";
        description
          "policy target for inbound or outbound direction";
        leaf direction {
          type identityref {
            base direction;
          }
          description
            "Direction fo the traffic flow either inbound or outbound";
        }
        leaf policy-type {
          type identityref {
```

```
            base policy:policy-type;
          }
          description
            "Policy entry type";
        }
        leaf policy-name {
          type string;
          mandatory true;
          description
            "Policy entry name";
        }
      }
    }
  }
  <CODE ENDS>
```

6.5. IETF-DIFFSERV

```
  <CODE BEGINS>file "ietf-diffserv@2020-03-17.yang"
  module ietf-diffserv {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-diffserv";
    prefix diffserv;

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

    organization "IETF RTG (Routing Area) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
       WG List:  <mailto:rtgwg@ietf.org>
       Editor:   Aseem Choudhary
                 <mailto:asechoud@cisco.com>
       Editor:   Mahesh Jethanandani
```

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

     revision 2020-03-17 {
       description
         "Latest revision of diffserv based classifier";
       reference "RFC XXXX: YANG Model for QoS";
     }

     feature diffserv-queue-inline-support {
       description
         "Queue inline support in diffserv policy";
     }
     feature diffserv-scheduler-inline-support {
       description
         "scheduler inline support in diffserv policy";
     }
     identity diffserv-policy-type {
       base policy:policy-type;
       description
         "This defines ip policy-type";
     }
     identity ipv4-diffserv-policy-type {
       base policy:policy-type;
       description
         "This defines ipv4 policy-type";
     }
     identity ipv6-diffserv-policy-type {
       base policy:policy-type;
       description
         "This defines ipv6 policy-type";
     }

     identity dscp {
```

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

    identity meter-type {
      description
        "This base identity type defines meter types";
```

```
    }
    identity one-rate-two-color-meter-type {
      base meter-type;
      description
        "one rate two color meter type";
    }
    identity one-rate-tri-color-meter-type {
      base meter-type;
      description
        "one rate three color meter type";
    }
    identity two-rate-tri-color-meter-type {
      base meter-type;
      description
        "two rate three color meter action type";
    }
    grouping dscp-cfg {
      list dscp-cfg {
        key "dscp-min dscp-max";
        description
          "list of dscp ranges";
        leaf dscp-min {
          type inet:dscp;
          description
            "Minimum value of dscp min-max range";
        }
        leaf dscp-max {
          type inet:dscp;
          must ". >= ../dscp-min" {
            error-message
            "The dscp-max must be greater than or equal to dscp-min";
          }
          description
            "maximum value of dscp min-max range";
        }
      }
      description
        "Filter grouping containing list of dscp ranges";
      reference
        "RFC 2474: Definition of the Differentiated Services
                   Field (DS Field) in the IPv4 and IPv6 Headers.";
    }
    grouping source-ipv4-address-cfg {
      list source-ipv4-address-cfg {
        key "source-ipv4-addr";
        description
          "list of source ipv4 address";
        leaf source-ipv4-addr {
```

```
          type inet:ipv4-prefix;
          description
            "source ipv4 prefix";
        }
      }
      description
        "Filter grouping containing list of source ipv4 addresses";
      reference
        "RFC 791: Internet Protocol.";
    }
    grouping destination-ipv4-address-cfg {
      list destination-ipv4-address-cfg {
        key "destination-ipv4-addr";
        description
          "list of destination ipv4 address";
        leaf destination-ipv4-addr {
          type inet:ipv4-prefix;
          description
            "destination ipv4 prefix";
        }
      }
      description
        "Filter grouping containing list of destination ipv4 address";
      reference
        "RFC 791: Internet Protocol.";
    }
    grouping source-ipv6-address-cfg {
      list source-ipv6-address-cfg {
        key "source-ipv6-addr";
        description
          "list of source ipv6 address";
        leaf source-ipv6-addr {
          type inet:ipv6-prefix;
          description
            "source ipv6 prefix";
        }
      }
      description
        "Filter grouping containing list of source ipv6 addresses";
      reference
        "RFC 4291: IP Version 6 Addressing Architecture.";
    }
    grouping destination-ipv6-address-cfg {
      list destination-ipv6-address-cfg {
        key "destination-ipv6-addr";
        description
          "list of destination ipv4 or ipv6 address";
        leaf destination-ipv6-addr {
```

```
          type inet:ipv6-prefix;
          description
            "destination ipv6 prefix";
        }
      }
      description
        "Filter grouping containing list of destination ipv6 address";
      reference
        "RFC 4291: IP Version 6 Addressing Architecture.";
    }
    grouping source-port-cfg {
      list source-port-cfg {
        key "source-port-min source-port-max";
        description
          "list of ranges of source port";
        leaf source-port-min {
          type inet:port-number;
          description
            "minimum value of source port range";
        }
        leaf source-port-max {
          type inet:port-number;
          must ". >= ../source-port-min" {
            error-message
            "The source-port-max must be greater than or equal to
             source-port-min";
          }
          description
            "maximum value of source port range";
        }
      }
      description
        "Filter grouping containing list of source port ranges";
      reference
        "RFC 768: User Datagram Protocol
         RFC 793: TRANSMISSION CONTROL PROTOCOL";
    }
    grouping destination-port-cfg {
      list destination-port-cfg {
        key "destination-port-min destination-port-max";
        description
          "list of ranges of destination port";
        leaf destination-port-min {
          type inet:port-number;
          description
            "minimum value of destination port range";
        }
        leaf destination-port-max {
```

```
          type inet:port-number;
          must ". >= ../destination-port-min" {
            error-message
            "The destination-port-max must be greater than or equal to
             destination-port-min";
          }
          description
            "maximum value of destination port range";
        }
      }
      description
        "Filter grouping containing list of destination port ranges";
      reference
        "RFC 768: User Datagram Protocol
         RFC 793: TRANSMISSION CONTROL PROTOCOL";
    }
    grouping protocol-cfg {
      list protocol-cfg {
        key "protocol-min protocol-max";
        description
          "list of ranges of protocol values.
           Internet Protocol number refers to the protocol of the
           payloadi in ipv4 header. In IPv6, this field is known as
           'next-header', and if extension headers are present, the
           protocol is present in the 'upper-layer' header.";
        leaf protocol-min {
          type uint8 {
            range "0..255";
          }
          description
            "minimum value of protocol range";
        }
        leaf protocol-max {
          type uint8 {
            range "0..255";
          }
          must ". >= ../protocol-min" {
            error-message
            "The protocol-max must be greater than or equal to
             protocol-min";
          }
          description
            "maximum value of protocol range";
        }
      }
      description
        "Filter grouping containing list of Protocol ranges";
      reference
```

```
        "RFC 791: Internet Protocol.
         RFC 8200: Internet Protocol, Version 6 (IPv6) Specification.";
    }
    grouping traffic-group-cfg {
      container traffic-group-cfg {
        leaf traffic-group-name {
          type string ;
          description
            "This leaf defines name of the traffic group referenced";
        }
      description
        "traffic group container";
      }
      description
        "traffic group grouping";
    }

    augment "/classifier:classifiers/classifier:classifier-entry" +
             "/classifier:filter-entry" {
      choice filter-param {
        description
          "Choice of filter types";
        case dscp {
          uses dscp-cfg;
          description
            "Filter containing list of dscp ranges";
        }
        case source-ipv4-address {
          uses source-ipv4-address-cfg;
          description
            "Filter containing list of source ipv4 addresses";
        }
        case destination-ipv4-address {
          uses destination-ipv4-address-cfg;
          description
            "Filter containing list of destination ipv4 address";
        }
        case source-ipv6-address {
          uses source-ipv6-address-cfg;
          description
            "Filter containing list of source ipv6 addresses";
        }
        case destination-ipv6-address {
          uses destination-ipv6-address-cfg;
          description
            "Filter containing list of destination ipv6 address";
        }
        case source-port {
```

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

```
            description
               "If policy type is v6, this filter cannot be used.";
          }
          uses source-ipv4-address-cfg;
          description
            "Filter containing list of source ipv4 addresses";
        }
        case destination-ipv4-address {
          when "../../policy:policy-type !=
                              'diffserv:ipv6-diffserv-policy-type'" {
            description
               "If policy type is v6, this filter cannot be used.";
          }
          uses destination-ipv4-address-cfg;
          description
            "Filter containing list of destination ipv4 address";
        }
        case source-ipv6-address {
          when "../../policy:policy-type !=
                              'diffserv:ipv4-diffserv-policy-type'" {
            description
               "If policy type is v4, this filter cannot be used.";
          }
          uses source-ipv6-address-cfg;
          description
            "Filter containing list of source ipv6 addresses";
        }
        case destination-ipv6-address {
          when "../../policy:policy-type !=
                              'diffserv:ipv4-diffserv-policy-type'" {
            description
               "If policy type is v4, this filter cannot be used.";
          }
          uses destination-ipv6-address-cfg;
          description
            "Filter containing list of destination ipv6 address";
        }
        case source-port {
          uses source-port-cfg;
          description
            "Filter containing list of source-port ranges";
        }
        case destination-port {
          uses destination-port-cfg;
          description
            "Filter containing list of destination-port ranges";
        }
        case protocol {
```

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

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

6.6. IETF-QUEUE-POLICY

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

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

    organization "IETF RTG (Routing Area) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
       WG List:  <mailto:rtgwg@ietf.org>
       Editor:   Aseem Choudhary
                 <mailto:asechoud@cisco.com>
       Editor:   Mahesh Jethanandani
                 <mailto:mjethanandani@gmail.com>
       Editor:   Norm Strahle
                 <mailto:nstrahle@juniper.net>";
    description
      "This module contains a collection of YANG definitions for
       configuring diffserv specification implementations.
```

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

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

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

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

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

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

    augment "/policy:policies/policy:policy-entry" +
            "/policy:classifier-entry/policy:filter-entry" {
      when "../../policy:policy-type =
                          'queue-policy:queue-policy-type'" {
        description
          "If policy type is v6, this filter cannot be used.";
      }
      if-feature queue-policy-support;
      choice filter-params {
```

```
        description
          "Choice of action types";
        case traffic-group-name {
          uses diffserv:traffic-group-cfg;
          description
            "traffic group name";
        }
      }
      description
        "Augments Queue policy Classifier with common filter types";
    }

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

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

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


    augment "/policy:policies/policy:policy-entry" +
            "/policy:classifier-entry" +
```

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

6.7. IETF-SCHEDULER-POLICY

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

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

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

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

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

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

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

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

```
          "the match-all action";
      }
      description
        "the match-all filter grouping";
    }

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

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

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

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

7. IANA Considerations

TBD

8. Security Considerations

9. Acknowledgement

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

MITRE has approved this document for Public Release, Distribution Unlimited, with Public Release Case Number 19-3027.

10. References

10.1. Normative References

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

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

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

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

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

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

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

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

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

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

10.2. Informative References

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

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

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

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

A.1. Example of Company A Diffserv Model

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

- use of template based classifier definitions

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

- use of inline actions in a policy

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

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

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

  organization "Company A";
  contact
    "Editor:   XYZ
```

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

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

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

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

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

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

```
            description
              "maximum value of qos-group range";
          }
        }
        description
          "Filter containing list of qos-group ranges";
      }

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

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

      augment "/classifier:classifiers/" +
              "classifier:classifier-entry/" +
              "classifier:filter-entry/diffserv:filter-param" {
        case qos-group {
          uses qos-group-cfg;
          description
            "Filter containing list of qos-group ranges.
```

```
            Qos-group represent packet metadata information
            in a device. ";
        }
        description
          "augmentation of classifier filters";
      }
      augment "/policy:policies/policy:policy-entry/" +
              "policy:classifier-entry/" +
              "policy:classifier-action-entry-cfg/" +
              "policy:action-cfg-params" {
        case random-detect {
          uses randomdetect;
        }
        description
          "Augment the actions to policy entry";
      }

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

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

```
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:one-rate-two-color-meter-type" +
                "/diffserv:one-rate-two-color-meter" +
                "/diffserv:exceed-action" +
                "/diffserv:exceed-2color-meter-action-params" +
                "/diffserv:exceed-2color-meter-action-val" {

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

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

```
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/diffserv:meter-inline" +
                "/diffserv:meter-type" +
                "/diffserv:one-rate-tri-color-meter-type" +
                "/diffserv:one-rate-tri-color-meter" +
                "/diffserv:exceed-action" +
                "/diffserv:exceed-3color-meter-action-params" +
                "/diffserv:exceed-3color-meter-action-val" {

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

```
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }

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

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

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

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

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

A.2. Example of Company B Diffserv Model

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

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

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

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

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

- use of internal meta data for classification and marking

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

```
       prefix compb-filter-policy;

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

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

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

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

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


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

```
 ***********************************************/

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

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

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

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

```
          }
          description
            "Loss-priority";
        }
      }
      description
        "Filter containing list of loss priorities";
    }

    augment "/policy:policies" +
            "/policy:policy-entry" +
            "/policy:classifier-entry" +
            "/policy:filter-entry" +
            "/diffserv:filter-params" {
      case forwarding-class {
          uses forwarding-class-cfg;
        description
          "Filter Type Internal-loss-priority";
      }
      case internal-loss-priority {
          uses loss-priority-cfg;
        description
          "Filter Type Internal-loss-priority";
      }
      description
        "Augments Diffserv Classifier with vendor" +
        " specific types";
    }

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

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

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

    grouping mark-fwd-class {
      container mark-fwd-class-cfg {
        leaf forwarding-class {
```

```
          type string;
          description
            "Forwarding class name";
        }
        description
          "mark-fwd-class container";
      }
      description
        "mark-fwd-class grouping";
    }

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

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

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

```
    }

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

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

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


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

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

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

      case exceed-2color-meter-action-drop {
        description
            "meter drop";
             uses action:drop;
      }
      case meter-action-mark-fwd-class {
        uses mark-fwd-class;
        description
          "Mark forwarding class in the packet";
      }
      case meter-action-mark-loss-priority {
          uses mark-loss-priority;
        description
          "Mark loss priority in the packet";
      }
    }

    augment "/action:meter-template" +
            "/action:meter-entry" +
            "/action:meter-type" +
            "/action:two-rate-tri-color-meter-type" +
            "/action:two-rate-tri-color-meter" +
            "/action:violate-action" +
            "/action:violate-3color-meter-action-params" +
```

```
            "/action:violate-3color-meter-action-val" {
      case exceed-3color-meter-action-drop {
        description
            "meter drop";
        uses action:drop;
      }

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

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

      description
        "Augment the actions to basic meter";
    }

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

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

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

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

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

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

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

  augment "/policy:policies" +
          "/policy:policy-entry" +
          "/policy:classifier-entry" +
          "/policy:filter-entry" {
    /* Does NOT support "logical-not" of forwarding class.
       Use "must"? */
    choice filter-params {
      description
        "Choice of filters";
      case forwarding-class-cfg {
        uses forwarding-class-cfg;
        description
```

```
            "Filter Type Internal-loss-priority";
        }
      }
      description
        "Augments Diffserv Classifier with fwd class filter";
    }

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

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

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

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

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

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

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

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

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

container compb-queue {
  description
    "Queue used in compb architecture";
    leaf name {
        type string;
        description
          "A unique name identifying this queue";
    }
    uses action:queue;
    container excess-rate {
      choice excess-rate-type {
         case percent {
           leaf excess-rate-percent {
             type uint32 {
               range "1..100";
             }
             description
               "excess-rate-percent";
           }
         }
         case proportion {
           leaf excess-rate-poroportion {
             type uint32 {
               range "1..1000";
```

```
                }
                description
                  "excess-rate-poroportion";
              }
             }
             description
               "Choice of excess-rate type";
          }
          description
              "Excess rate value";
        }
        leaf excess-priority {
          type enumeration {
            enum high {
             description "High Loss Priority";
            }
            enum medium-high {
             description "Medium-high Loss Priority";
            }
            enum medium-low {
             description "Medium-low Loss Priority";
            }
            enum low {
             description "Low Loss Priority";
            }
            enum none {
             description "No excess priority";
            }
          }
          description
            "Priority of excess (above guaranted rate) traffic";
        }
        container buffer-size {
          choice buffer-size-type {
             case percent {
               leaf buffer-size-percent {
                 type uint32 {
                   range "1..100";
                 }
                 description
                   "buffer-size-percent";
               }
             }
             case temporal {
               leaf buffer-size-temporal {
                 type uint64;
                 units "microsecond";
                 description
```

```
                    "buffer-size-temporal";
                }
              }
              case remainder {
                leaf buffer-size-remainder {
                  type empty;
                  description
                    "use remaining of buffer";
                }
              }
              description
                "Choice of buffer size type";
            }
            description
                "Buffer size value";
          }
        }

        augment
          "/compb-queue" +
          "/queue-cfg" +
          "/algorithmic-drop-cfg" +
          "/drop-algorithm" {
          case random-detect {
            list drop-profile-list {
              key "priority";
              description
                "map of priorities to drop-algorithms";
              leaf priority {
                type enumeration {
                  enum any {
                    description "Any priority mapped here";
                  }
                  enum high {
                    description "High Priority Packet";
                  }
                  enum medium-high {
                    description "Medium-high Priority Packet";
                  }
                  enum medium-low {
                    description "Medium-low Priority Packet";
                  }
                  enum low {
                    description "Low Priority Packet";
                  }
                }
                description
                  "Priority of guaranteed traffic";
```

```
          }
          leaf drop-profile {
            type string;
            description
              "drop profile to use for this priority";
          }
        }
      }
      description
        "compb random detect drop algorithm config";
    }
  }

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

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

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

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

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

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

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

```
      }

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

    augment "/policy:policies" +
            "/policy:policy-entry" +
            "/policy:classifier-entry" +
            "/policy:classifier-action-entry-cfg" {
      choice action-cfg-params {
        case schedular {
          uses action:schedular;
        }
        case queue-policy {
          uses queue-policy-name;
        }
        description
          "Augment the scheduler policy with a queue policy";
      }
    }
  }
```

A.3. Example of Company C Diffserv Model

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

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

within a policing or metering policy, as is defined in ietf-diffserv.yang.

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

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

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

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

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

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

  /* identities */

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

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

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

  identity policing-policy {
```

```
      base compc-qos-policy;
    }

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

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

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

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

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

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

    /* groupings */

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

    /* deviations */

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

    deviation "/pol:policies/pol:policy-entry/pol:classifier-entry" {
```

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

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

    /* augments */

    augment "/pol:policies/pol:policy-entry" {
      when "pol:type = 'compc-qos-policy')" {
        description
          "Additional nodes only for diffserv-policy";
      }
      leaf sub-type {
        type identityref {
          base compc-qos-policy;
        }
        mandatory true;
```

```
        /* The value of this leaf must not change once configured */
      }
      leaf per-class-action {
        mandatory true;
        type boolean;
        must "(((. = 'true') and " +
             "  ((../sub-type = 'policing-policy') or " +
             "   (../sub-type = 'metering-policy') or " +
             "   (../sub-type = 'mdrr-queuing-policy') or " +
             "   (../sub-type = 'pwfq-queuing-policy') or " +
             "   (../sub-type = 'forwarding-policy'))) or " +
             " ((. = 'false') and " +
             "  ((../sub-type = 'overhead-profile-policy') or " +
             "   (../sub-type = 'resource-profile-policy') or " +
             "   (../sub-type = 'protocol-rate-limit-policy')))" {
          description
            "Only certain policies have per-class action";
        }
      }
      container traffic-classifier {
        presence true;
        when "../sub-type = 'policing-policy' or " +
             "../sub-type = 'metering-policy' or " +
             "../sub-type = 'forwarding-policy'" {
          description
            "A classifier for policing-policy or metering-policy";
        }
        leaf name {
          type string;
          mandatory true;
          description
            "Traffic classifier name";
        }
        leaf type {
          type enumeration {
            enum 'internal-dscp-only-classifier' {
              value 0;
              description
                "Classify traffic based on (internal) dscp only";
            }
            enum 'ipv4-header-based-classifier' {
              value 1;
              description
                "Classify traffic based on IPv4 packet header fields";
            }
            enum 'ipv6-header-based-classifier' {
              value 2;
              description
```

```
                "Classify traffic based on IPv6 packet header fields";
            }
          }
          mandatory true;
          description
            "Traffic classifier type";
        }
      }
      container traffic-queue {
        when "(../sub-type = 'mdrr-queuing-policy') or " +
             "(../sub-type = 'pwfq-queuing-policy')" {
          description
            "Queuing policy properties";
        }
        leaf queue-map {
          type string;
          description
            "Traffic queue map for queuing policy";
        }
      }
      container overhead-profile {
        when "../sub-type = 'overhead-profile-policy'" {
          description
            "Overhead profile policy properties";
        }
      }
      container resource-profile {
        when "../sub-type = 'resource-profile-policy'" {
          description
            "Resource profile policy properties";
        }
      }
      container protocol-rate-limit {
        when "../sub-type = 'protocol-rate-limit-policy'" {
          description
            "Protocol rate limit policy properties";
        }
      }
    }

    augment "/pol:policies/pol:policy-entry/pol:classifier-entry" +
            "/pol:classifier-action-entry-cfg/pol:action-cfg-params" {
      when "../../../pol:type = 'compc-qos-policy')" {
        description
          "Configurations for a classifier-policy-type policy";
      }
      case metering-or-policing-policy {
        when "../../../sub-type = 'policing-policy' or "
```

```
           + "../../../sub-type = 'metering-policy'" {
        }
        container dscp-marking {
          uses action:dscp-marking;
        }
        container precedence-marking {
          uses action:dscp-marking;
        }
        container priority-marking {
          uses action:priority;
        }
        container rate-limiting {
          uses action:one-rate-two-color-meter;
        }
      }
      case mdrr-queuing-policy {
        when "../../../sub-type = 'mdrr-queuing-policy'" {
          description
            "MDRR queue handling properties for the traffic " +
            "classified into current queue";
        }
        leaf mdrr-queue-weight {
          type uint8 {
            range "20..100";
          }
          units percentage;
        }
      }
      case pwfq-queuing-policy {
        when "../../../sub-type = 'pwfq-queuing-policy'" {
          description
            "PWFQ queue handling properties for traffic " +
            "classified into current queue";
        }
        leaf pwfq-queue-weight {
          type uint8 {
            range "20..100";
          }
          units percentage;
        }
        leaf pwfq-queue-priority {
          type uint8;
        }
        leaf pwfq-queue-rate {
          type uint8;
        }
      }
      case forwarding-policy {
```

```
          when "../../../sub-type = 'forwarding-policy'" {
            description
              "Forward policy handling properties for traffic " +
              "in this classifier";
          }
          uses redirect-action-grp;
        }
        description
          "Add the classify action configuration";
      }

    }
```

Authors' Addresses

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

Email: asechoud@cisco.com

Mahesh Jethanandani
VMware

Email: mjethanandani@gmail.com

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

Email: nstrahle@juniper.net

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

Email: exa@juniper.net

Ing-Wher Chen
The MITRE Corporation

Email: ingwherchen@mitre.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2020

Z. Hu
Huawei
H. Chen
Futurewei
H. Chen
China Telecom
P. Wu
Huawei
M. Toy
Verizon
C. Cao
T. He
China Unicom
L. Liu
Fujitsu
X. Liu
Volta Networks

March 18, 2020

SRv6 Path Egress Protection
draft-ietf-rtgwg-srv6-egress-protection-00

Abstract

This document describes protocol extensions for protecting the egress node of a Segment Routing for IPv6 (SRv6) path or tunnel.

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 September 19, 2020.

Table of Contents

1. Introduction

The fast protection of a transit node of a Segment Routing (SR) path or tunnel is described in [I-D.ietf-rtgwg-segment-routing-ti-lfa] and [I-D.hu-spring-segment-routing-proxy-forwarding]. [RFC8400] specifies the fast protection of egress node(s) of an MPLS TE LSP tunnel including P2P TE LSP tunnel and P2MP TE LSP tunnel in details. However, these documents do not discuss the fast protection of the egress node of a Segment Routing for IPv6 (SRv6) path or tunnel.

This document fills that void and presents protocol extensions for the fast protection of the egress node of an SRv6 path or tunnel. Egress node and egress, fast protection and protection as well as SRv6 path and SRv6 tunnel will be used exchangeably below.

There are a number of topics related to the egress protection, which include the detection of egress node failure, the relation between egress protection and global repair, and so on. These are discussed in details in [RFC8679].

2. Terminologies

The following terminologies are used in this document.

SR: Segment Routing

SRv6: SR for IPv6

SRH: Segment Routing Header

SID: Segment Identifier

LSA: Link State Advertisement in OSPF

LSP: Label Switched Path in MPLS or Link State Protocol PDU in IS-IS

PDU: Protocol Data Unit

LS: Link Sate, which is LSA in OSPF or LSP in IS-IS

TE: Traffic Engineering

SA: Source Address

DA: Destination Address

P2MP: Point-to-MultiPoint

P2P: Point-to-Point

CE: Customer Edge

PE: Provider Edge

LFA: Loop-Free Alternate

TI-LFA: Topology Independent LFA

BFD: Bidirectional Forwarding Detection

VPN: Virtual Private Network

L3VPN: Layer 3 VPN

VRF: Virtual Routing and Forwarding

FIB: Forwarding Information Base

PLR: Point of Local Repair

BGP: Border Gateway Protocol

IGP: Interior Gateway Protocol

OSPF: Open Shortest Path First

IS-IS: Intermediate System to Intermediate System

3. SR Path Egress Protection

This section describes the mechanism of SR path egress protection and illustrates it through an example.

3.1. Mechanism

Figure 1 is used to explain the mechanism of SR path egress node protection.

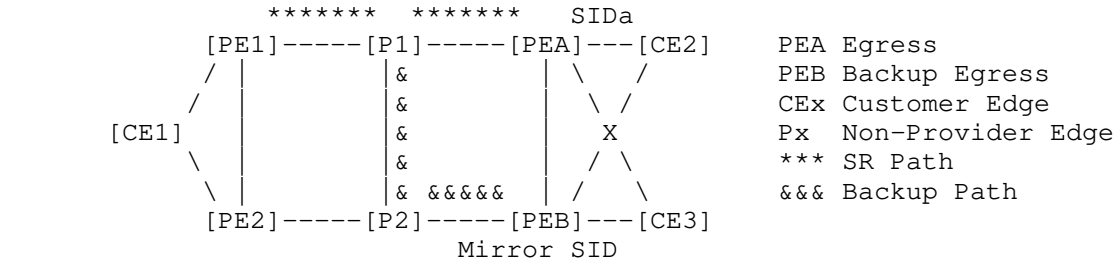


Figure 1: PEB Protects Egress PEA of SR Path

Where node PEA is the egress of the SR path from PE1 to PEA, and has SIDa which is the active segment in the packet from the SR path at PEA. Node PEB is the backup egress (or say protector) to provide the protection for egress (or say primary egress) PEA. Node P1 is the direct previous hop of egress PEA and acts as PLR to support the protection for PEA.

When PEB is selected as a backup egress to protect the egress PEA, a Mirror SID is configured on PEB to protect PEA. PEB advertises this information through IGP, which includes the Mirror SID and the egress PEA. The information is represented by <PEB, PEA, Mirror SID>, which indicates that PEB protects PEA with Mirror SID.

After PEA receives the information <PEB, PEA, Mirror SID>, it may send the forwarding behavior of the SIDa at PEA to PEB with the Mirror SID using some protocols such as BGP if PEB can not obtain this behavior from other approaches if PEB wants to protect SIDa of PEA. How to send the forwarding behavior of the SIDa to PEB is out scope of this document.

When PEB gets the forwarding behavior of the SIDa of PEA from PEA or other means, it adds a forwarding entry for the SIDa according to the behavior into the forwarding table for node PEA. This table is identified by the Mirror SID, which indicates node PEA's context. Using the forwarding entry for SIDa in this table, a packet with SIDa will be transmitted by PEB to the same destination as it is transmitted by PEA. For example, assume that the packet with SIDa is transmitted by PEA to CE2 through the forwarding behavior of the SIDa in PEA. The packet will be transmitted by PEB to the same CE2 through looking up the table identified by the Mirror SID.

After P1 as PLR receives the information <PEB, PEA, Mirror SID> and knows that PEB wants to protect SIDa of PEA, it computes a shortest path to PEB. A Repair List RL is obtained based on the path. It is one of the followings:

- RL = <Mirror SID> if the path does not go through PEA; or

- RL = <S1, ..., Sn, Mirror SID> if the path goes through PEA, where <S1, ..., Sn> is the TI-LFA Repair List to PEB computed by P1.

When PEA fails, P1 as PLR sends the packet with SIDa carried by the SR path to PEB, but encapsulates the packet before sending it by executing H.Encaps with the Repair List RL and a Source Address T.

Suppose that the packet received by P1 is represented by Pkt = (S, SIDa)Pkt0, where SA = S and DA = SIDa, and Pkt0 is the rest of the packet.

The execution of H.Encaps pushes an IPv6 header to Pkt and sets some fields in the outer and inner IPv6 header to produce an encapsulated packet Pkt'. Pkt' will be one of the followings:

- Pkt' = (T, Mirror SID) (S, SIDa)Pkt0 if RL = <Mirror SID>; or

- Pkt' = (T, S1)(Mirror SID, Sn, ..., S1; SL=n) (S, SIDa)Pkt0 if RL = <S1, ..., Sn, Mirror SID>.

When PEB receives the re-routed packet, which is (T, Mirror SID) (S, SIDa)Pkt0, it decapsulates the packet and forwards the decapsulated packet using the forwarding table identified by Mirror SID through executing End.DT6.

It obtains the Mirror SID in the outer IPv6 header of the packet, removes this outer IPv6 header with all its extension headers, and then processes the inner IPv6 packet (i.e., (S, SIDa)Pkt0, the packet without the outer IPv6 header). PEB finds the forwarding table for node PEA using the Mirror SID as the context ID, and submits the packet to this forwarding table lookup and transmission to the same destination as PEA does.

3.2. Example

Figure 2 shows an example of protecting egress PE3 of a SR path, which is from ingress PE1 to egress PE3.

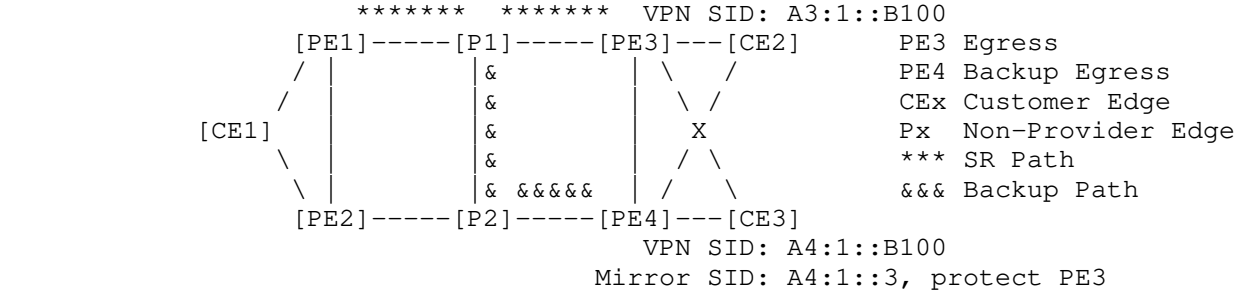


Figure 2: PE4 Protects Egress PE3 of SR Path

Where node P1's pre-computed backup path for PE3 is from P1 to PE4 via P2. In normal operations, after receiving a packet with destination PE3, P1 forwards the packet to PE3 according to its FIB. When PE3 receives the packet, it sends the packet to CE2.

When PE3 fails, P1 as PLR detects the failure through using a failure detection mechanism such as BFD and forwards the packet to PE4 via the backup path. When PE4 receives the packet, it sends the packet to the same CE2.

In Figure 2, Both CE2 and CE3 are dual home to PE3 and PE4. PE3 has a VPN SID A3:1::B100. PE4 has a VPN SID A4:1::B100. A Mirror SID A4:1::3 is configured on PE4 for protecting PE3.

After the configuration, PE4 advertises this information through an IGP LS (i.e., LSA in OSPF or LSP in IS-IS), which includes PE4's ID, PE3's ID and Mirror SID A4:1::3. Every node in the SR domain will receive this IGP LS, which indicates that PE4 wants to protect PE3 with Mirror SID A4:1::3.

When PE4 (e.g., BGP on PE4) receives a prefix whose VPN SID belongs to PE3 that is protected by PE4 through Mirror SID A4:1::3, it finds PE4's VPN SID corresponding to PE3's VPN SID. For example, local PE4 has Prefix 1.1.1.1 with VPN SID A4:1::B100, when PE4 receives prefix 1.1.1.1 with remote PE3's VPN SID A3:1::B100, it knows that they are for the same VPN.

The forwarding behaviors for these two VPN SIDs are the same from function's point of view. If the behavior for PE3's VPN SID in PE3 forwards the packet with it to CE2, then the behavior for PE4's VPN SID in PE4 forwards the packet to the same CE2; and vice versa. PE4 creates a forwarding entry for PE3's VPN SID A3:1::B100 in the table (or FIB) identified by Mirror SID A4:1::3 according to the forwarding behavior for PE4's VPN SID A4:1::B100.

Node P1's pre-computed backup path for destination PE3 is from P1 to PE4 having mirror SID A4:1::3. When P1 receives a packet destined to PE3's VPN SID A3:1::B100, in normal operations, it forwards the packet with source A1:1:: and destination PE3's VPN SID A3:1::B100 according to the FIB using the destination PE3's VPN SID A3:1::B100.

When PE3 fails, P1 as PLR sends the packet to PE4 via the backup path pre-computed. P1 encapsulates the packet using H.Encaps before sending it to PE4.

Suppose that the packet received by P1 is represented by Pkt = (SA = A1:1::, DA = A3:1::B100)Pkt0, where DA = A3:1::B100 is PE3's VPN SID, and Pkt0 is the rest of the packet. The encapsulated packet Pkt' will be one of the followings:

- Pkt' = (T, Mirror SID A4:1::3) (A1:1::, A3:1::B100)Pkt0 if backup path not via PE3; or (otherwise)

- Pkt' = (T, S1)(Mirror SID A4:1::3, Sn, ..., S1; SL=n) (A1:1::, A3:1::B100)Pkt0.

where T is a Source Address, <S1, ..., Sn> is the TI-LFA Repair List to PE4 computed by P1 when the backup path to PE4 goes through PE3.

When PE4 receives the re-routed packet, it decapsulates the packet and forwards the decapsulated packet by End.DT6. The packet received

by PE4 is (T, Mirror SID A4:1::3) (A1:1::, PE3's VPN SID A3:1::B100)Pkt0.

PE4 obtains Mirror SID A4:1::3 in the outer IPv6 header of the packet, removes this outer IPv6 header, and then processes the inner IPv6 packet (A1:1::, A3:1::B100)Pkt0. It finds the forwarding table for PE3 using Mirror SID A4:1::3 as the context ID, gets the forwarding entry for PE3's VPN SID A3:1::B100 from the table, and forwards the packet to CE2 using the entry.

4. Extensions to IGP for Egress Protection

This section describes extensions to IS-IS and OSPF for advertising the information about SRv6 path egress protection.

4.1. Extensions to IS-IS

A new sub-TLV, called IS-IS SRv6 Mirror SID sub-TLV, is defined. It is used in the SRv6 Locator TLV defined in [I-D.ietf-lsr-isis-srv6-extensions] to advertise SRv6 Mirror SID and the ID of the node to be protected. The SRv6 Mirror SID inherit the topology/algorithm from the parent locator. The format of the sub-TLV is illustrated below.

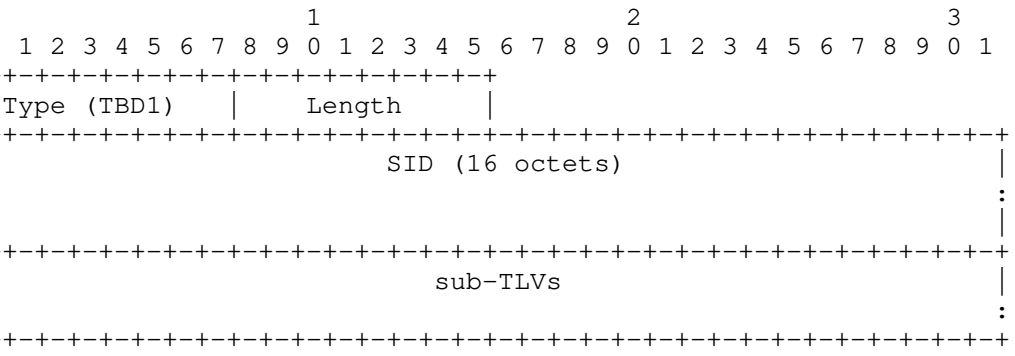


Figure 3: IS-IS SRv6 Mirror SID sub-TLV

Type: TBD1 (suggested value 8) is to be assigned by IANA.

Length: variable.

SID: 16 octets. This field contains the SRv6 Mirror SID to be advertised.

Two sub-TLVs are defined. One is the protected node sub-TLV, and the other is the protected SIDs sub-TLV.

A protected node sub-TLV is used to carry the ID of the node to be protected by the SRv6 Mirror SID. It has the following format.

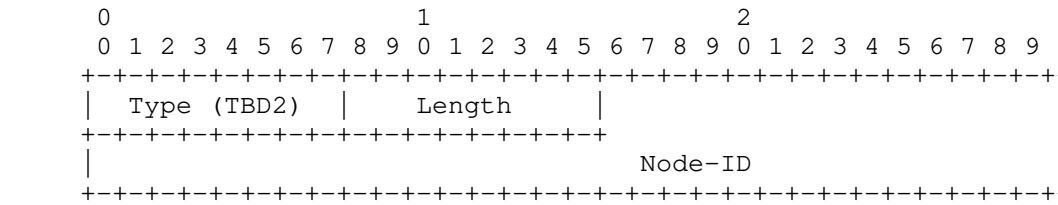


Figure 4: IS-IS Protected Node sub-TLV

Type: TBD2 (suggested value 1) is to be assigned by IANA.

Length: 1 octet. Its value is 6.

Node-ID: 6 octets. It contains a 6-octet ISO Node-ID (ISO system-ID).

A protected SIDs sub-TLV is used to carry the SIDs to be protected by the SRv6 Mirror SID. It has the following format.

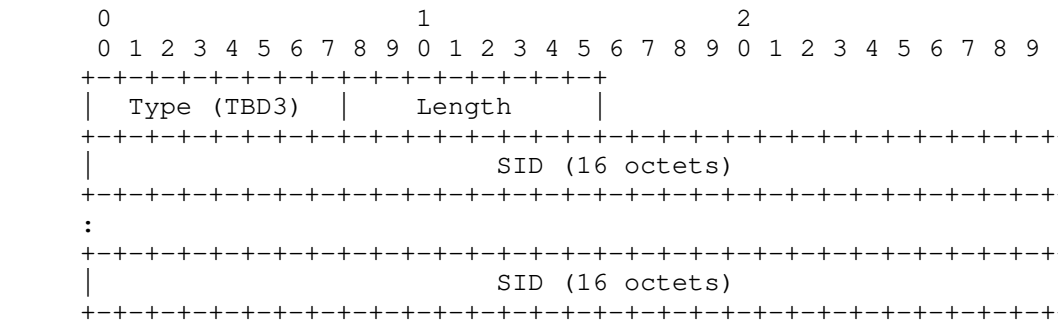


Figure 5: IS-IS Protected SIDs sub-TLV

Type: TBD3 (suggested value 2) is to be assigned by IANA.

Length: variable.

SID: 16 octets. This field encodes an SRv6 SID to be protected.

When node B advertises that B wants to protect node A with a Mirror SID through an LSP, the LSP contains an IS-IS SRv6 Mirror SID sub-TLV, which includes the Mirror SID and the node A's ID in an IS-IS Protected Node sub-TLV. If B wants to protect just a specific set of SIDs of node A, the Mirror SID sub-TLV includes these SIDs in an IS-

IS Protected SIDs sub-TLV; otherwise (i.e., B wants to protect all the SIDs of A) it does not contain any IS-IS Protected SIDs sub-TLV.

Note: the IS-IS extensions for SR MPLS is described in [RFC8667]. It says that the SID/Label Binding TLV may also be used to advertise a Mirror SID. For B to protect egress A of SR MPLS path, B may also use this TLV to advertise the node A's ID and a specific set of SIDs of A to be protected. An IS-IS SR MPLS Mirror SID sub-TLV may be obtained from an IS-IS SRv6 Mirror SID sub-TLV by replacing each SID field in the latter with an SID/Label sub-TLV. B may advertise a SID/Label Binding TLV including this IS-IS SR MPLS Mirror SID sub-TLV.

Alternatively, an IS-IS SR MPLS Mirror Supplement sub-TLV is defined from an IS-IS SRv6 Mirror SID sub-TLV by removing the SID field in the top level and replacing each other SID field with an SID/Label sub-TLV. That is that an IS-IS SR MPLS Mirror Supplement sub-TLV just contains a Protected Node sub-TLV and a Protected SIDs sub-TLV, which includes SID/Label sub-TLVs. When the SID/Label Binding TLV contains an SID/Label sub-TLV for the Mirror SID, it includes an IS-IS SR MPLS Mirror Supplement sub-TLV.

4.2. Extensions to OSPF

Similarly, a new sub-TLV, called OSPF Mirror SID sub-TLV, is defined. It is used to advertise SRv6 Mirror SID and the ID of the node to be protected. Its format is illustrated below.

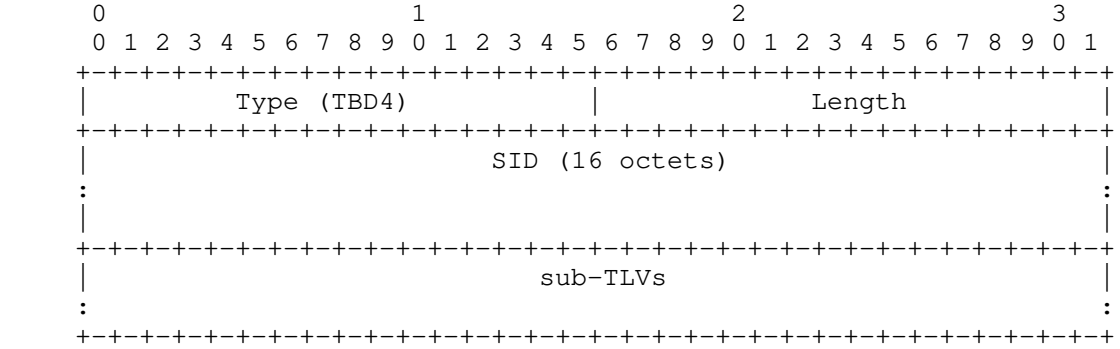


Figure 6: OSPF SRv6 Mirror SID sub-TLV

Type: TBD4 (suggested value 8) is to be assigned by IANA.

Length: variable.

SID: 16 octets. This field contains the SRv6 Mirror SID to be advertised.

Two sub-TLVs are defined. One is the protected node sub-TLV, and the other is the protected SIDs sub-TLV.

A protected node sub-TLV is used to carry the ID of the node to be protected by the SRv6 Mirror SID. It has the following format.

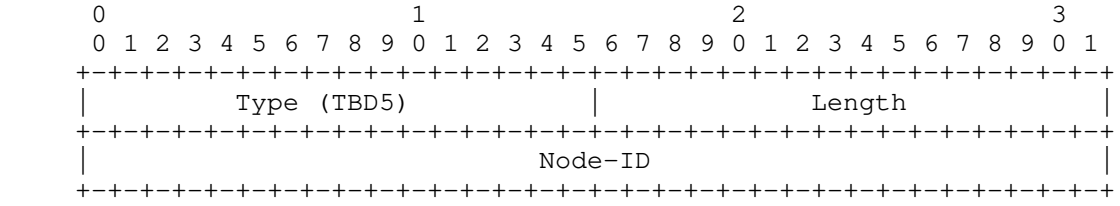


Figure 7: OSPF Protected Node sub-TLV

Type: TBD5 (suggested value 1) is to be assigned by IANA.

Length: 2 octets. Its value is 4.

Node-ID: 4 octets. It contains the ID of the OSPF node or router to be protected.

A protected SIDs sub-TLV is used to carry the SIDs to be protected by the SRv6 Mirror SID. It has the following format.

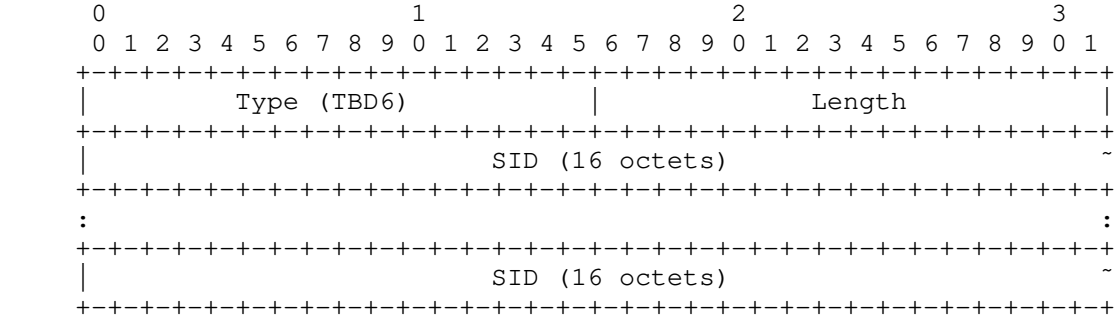


Figure 8: OSPF Protected SIDs sub-TLV

Type: TBD6 (suggested value 2) is to be assigned by IANA.

Length: variable.

SID: 16 octets. This field encodes an SRv6 SID to be protected.

5. Security Considerations

The security about the egress protection is described in in details in [RFC8679]. The extensions to OSPF and IS-IS described in this document for SRv6 path egress protection should not cause extra security issues.

6. IANA Considerations

6.1. IS-IS

Under "Sub-TLVs for TLVs 27, 135, 235, 236 and 237 registry" [I-D.ietf-lsr-isis-srv6-extensions], IANA is requested to add the following new Sub-TLV:

Sub-TLV Type	Sub-TLV Name	Reference
8	SRv6 Mirror SID Sub-TLV	This document

IANA is requested to create and maintain a new registry for sub-sub-TLVs of the SRv6 Mirror SID Sub-TLV. The suggested registry name is

o Sub-Sub-TLVs for SRv6 Mirror SID Sub-TLV

Initial values for the registry are given below. The future assignments are to be made through IETF Review [RFC5226].

Value	Sub-Sub-TLV Name	Definition
0	Reserved	
1	Protected Node Sub-Sub-TLV	This Document
2	Protected SIDs Sub-Sub-TLV	
3-255	Unassigned	

6.2. OSPFv3

Under registry "OSPFv3 Locator LSA Sub-TLVs" [I-D.li-ospf-ospfv3-srv6-extensions], IANA is requested to assign the following new Sub-TLV:

Sub-TLV Type	Sub-TLV Name	Reference
8	SRv6 Mirror SID Sub-TLV	This document

IANA is requested to create and maintain a new registry for sub-sub-TLVs of the SRv6 Mirror SID Sub-TLV. The suggested registry name is

o Sub-Sub-TLVs for SRv6 Mirror SID Sub-TLV

Initial values for the registry are given below. The future assignments are to be made through IETF Review [RFC5226].

Value	Sub-Sub-TLV Name	Definition
0	Reserved	
1	Protected Node Sub-Sub-TLV	This Document
2	Protected SIDs Sub-Sub-TLV	
3-65535	Unassigned	

7. Acknowledgements

The authors would like to thank Peter Psenak, Yimin Shen, Zhenqiang Li, Alexander Vainshtein, Greg Mirsky, Bruno Decraene and Jeff Tantsura for their comments to this work.

8. References

8.1. Normative References

[I-D.ietf-lsr-isis-srv6-extensions]
Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extension to Support Segment Routing over IPv6 Dataplane", draft-ietf-lsr-isis-srv6-extensions-05 (work in progress), February 2020.

[I-D.li-ospf-ospfv3-srv6-extensions]
Li, Z., Hu, Z., Cheng, D., Talaulikar, K., and P. Psenak, "OSPFv3 Extensions for SRv6", draft-li-ospf-ospfv3-srv6-extensions-07 (work in progress), November 2019.

[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>.

[RFC7356] Ginsberg, L., Previdi, S., and Y. Yang, "IS-IS Flooding Scope Link State PDUs (LSPs)", RFC 7356, DOI 10.17487/RFC7356, September 2014, <https://www.rfc-editor.org/info/rfc7356>.

[RFC7490] Bryant, S., Filsfils, C., Previdi, S., Shand, M., and N. So, "Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)", RFC 7490, DOI 10.17487/RFC7490, April 2015, <https://www.rfc-editor.org/info/rfc7490>.

[RFC8400] Chen, H., Liu, A., Saad, T., Xu, F., and L. Huang, "Extensions to RSVP-TE for Label Switched Path (LSP) Egress Protection", RFC 8400, DOI 10.17487/RFC8400, June 2018, <https://www.rfc-editor.org/info/rfc8400>.

[RFC8665] Psenak, P., Ed., Previdi, S., Ed., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", RFC 8665, DOI 10.17487/RFC8665, December 2019, <https://www.rfc-editor.org/info/rfc8665>.

[RFC8667] Previdi, S., Ed., Ginsberg, L., Ed., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", RFC 8667, DOI 10.17487/RFC8667, December 2019, <https://www.rfc-editor.org/info/rfc8667>.

[RFC8679] Shen, Y., Jeganathan, M., Decraene, B., Gredler, H., Michel, C., and H. Chen, "MPLS Egress Protection Framework", RFC 8679, DOI 10.17487/RFC8679, December 2019, <https://www.rfc-editor.org/info/rfc8679>.

8.2. Informative References

[I-D.hegde-spring-node-protection-for-sr-te-paths] Hegde, S., Bowers, C., Litkowski, S., Xu, X., and F. Xu, "Node Protection for SR-TE Paths", draft-hegde-spring-node-protection-for-sr-te-paths-05 (work in progress), July 2019.

[I-D.hu-spring-segment-routing-proxy-forwarding] Hu, Z., Chen, H., Yao, J., Bowers, C., and Y. Zhu, "SR-TE Path Midpoint Protection", draft-hu-spring-segment-routing-proxy-forwarding-07 (work in progress), January 2020.

[I-D.ietf-rtgwg-segment-routing-ti-lfa] Litkowski, S., Bashandy, A., Filsfils, C., Decraene, B., Francois, P., Voyer, D., Clad, F., and P. Camarillo, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-02 (work in progress), January 2020.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-06 (work in progress), December 2019.

[I-D.sivabalan-pce-binding-label-sid]
Sivabalan, S., Filsfils, C., Tantsura, J., Hardwick, J., Previdi, S., and C. Li, "Carrying Binding Label/Segment-ID in PCE-based Networks.", draft-sivabalan-pce-binding-label-sid-07 (work in progress), July 2019.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <https://www.rfc-editor.org/info/rfc5462>.

Authors' Addresses

Zhibo Hu
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: huzhibo@huawei.com

Huaimo Chen
Futurewei
Boston, MA
USA

Email: Huaimo.chen@futurewei.com

Huanan Chen
China Telecom
109, West Zhongshan Road, Tianhe District
Guangzhou 510000
China

Email: chenhuan6@chinatelecom.cn

Peng Wu
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: baggio.wupeng@huawei.com

Mehmet Toy
Verizon
USA

Email: mehmet.toy@verizon.com

Chang Cao
China Unicom
Beijing China

Email: caoc15@chinaunicom.cn

Tao He
China Unicom
Beijing China

Email: het21@chinaunicom.cn

Lei Liu
Fujitsu
USA

Email: liulei.kddi@gmail.com

Xufeng Liu
Volta Networks
McLean, VA
USA

Email: xufeng.liu.ietf@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2020

Z. Li
S. Peng
Huawei Technologies
D. Voyer
Bell Canada
C. Li
China Telecom
L. Geng
China Mobile
C. Cao
China Unicom
K. Ebisawa
Toyota Motor Corporation
S. Previdi
Individual
J. Guichard
Futurewei Technologies Ltd.

March 6, 2020

Application-aware Networking (APN) Framework
draft-li-apn-framework-00

Abstract

A multitude of applications are carried over the network, which have varying needs for network bandwidth, latency, jitter, and packet loss, etc. Some new emerging applications (e.g. 5G) have very demanding performance requirements. However, in current networks, the network and applications are decoupled, that is, the network is not aware of the applications' requirements in a fine granularity. Therefore, it is difficult to provide truly fine-granularity traffic operations for the applications and guarantee their SLA requirements.

This document proposes a new framework, named Application-aware Networking (APN), where application characteristic information such as application identification and its network performance requirements is carried in the packet encapsulation in order to facilitate service provisioning, perform application-level traffic steering and network resource adjustment.

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 7, 2020.

Table of Contents

1. Introduction

A multitude of applications are carried over the network, which have varying needs for network bandwidth, latency, jitter, and packet loss, etc. Some applications such as online gaming and live video streaming has very demanding network requirements and therefore require special treatment in the network. However, in current networks, the network and applications are decoupled, that is, the network is not aware of the applications' requirements in a fine granularity. Therefore, it is difficult to provide truly fine-granularity traffic operations for the applications and guarantee their SLA requirements accordingly.
[I-D.li-apn6-problem-statement-usecases] describes the challenges of traditional differentiated service provisioning methods, such as five tuples used for ACL/PBR causing coarse granularity, DPI imposing high CAPEX & OPEX and security issues, as well as orchestration and SDN-based solution causing long control loops.

This document proposes a new framework, named Application-aware Networking (APN), aiming to guarantee fine-granularity SLA requirements of applications, where application characteristic information such as application identification and its network performance requirements is carried in the packet encapsulation in order to determine the path, steer traffic, and perform network resource adjustment.

2. Specification of Requirements

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

This document is not a protocol specification and the key words in this document are used for clarity and emphasis of requirements language.

3. Terminology

ACL: Access Control List

APN: Application-aware Networking

APN6: Application-aware Networking for IPv6/SRv6

DPI: Deep Packet Inspection

MPLS: Multiprotocol Label Switching

PBR: Policy Based Routing

QoE: Quality of Experience

SDN: Software Defined Networking

SLA: Service Level Agreement

SR: Segment Routing

SR-MPLS: Segment Routing over MPLS dataplane

SRv6: Segment Routing over IPv6 dataplane

4. APN Framework and Key Components

The APN framework is shown in Figure 1. The key components include Service-aware App, App-aware Edge Device, App-aware-process Head-End, App-aware-process Mid-Point, and App-aware-process End-Point.

Packets carry application characteristic information (i.e. application-aware information) which includes the following information:

- Application-aware identification information: identifies the application, the user of the application, i.e, the packets of the traffic flow belonging to a specific SLA level/Application/User;

- Network performance requirements information that specify at least one of the following parameters: bandwidth, delay, delay variation, packet loss ratio, security, etc.

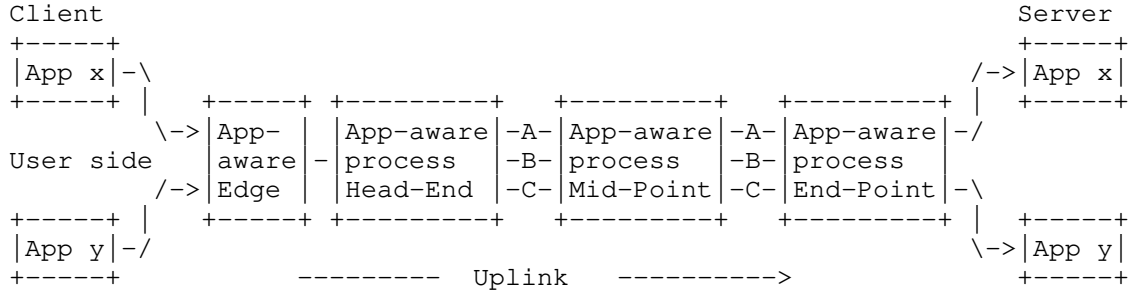


Figure 1: Framework and Key Components

The key components are introduced as follows.

1. Service-aware App: the host obtains the application characteristic information of the Service-aware App and generates the packets which carry the application characteristic information in the encapsulation. If carried in the packets, this information is used by the App-aware-process Head-End to determine the path between the App-aware-process Head-End and the App-aware-process End-Point for forwarding the packets to their destination, that is, to steer the packet into a given policy which satisfies the application requirements. In the APN framework, the application is not mandatory to be service-aware.

2. App-aware Edge Device: this network device receives packets from applications and obtains the application characteristic information. If the application is not Service-aware App, the application characteristic information can be retrieved by packet inspection, derived from services information such as double VLAN tagging (C-VLAN and S-VLAN), or added according to the local policies which is out of the scope of this document. The App-aware Edge Device adds the application characteristic information in the encapsulation on behalf of the application. The packets carrying the application characteristic information will be sent to the App-aware-process Head-End, and the application characteristic information will be used to determine the path between the App-aware-process Head-End and the App-aware-process End-Point for forwarding the packets.

3. App-aware-process Head-End: This network device receives packets and obtains the application characteristic information. A set of paths, tunnels or SR policy, exist between the App-aware-process Head-End and the App-aware-process End-Point. The App-aware-process Head-End maintains the matching relationship between the application characteristic information and the paths between the App-aware-process Head-End and the App-aware-process End-Point. The App-aware-process Head-End determines the path between the App-aware-process Head-End and the App-aware-process End-Point according to the application characteristic information carried in the packets and the matching relationship with it, which satisfies the service requirements of the application. If there is no such matching path found, the App-aware-process Head-End can set up a path towards the App-aware-process End-Point, and the matching relationship will be stored. The App-aware-process Head-End forwards the packets along the path. The application information conveyed by the packet received from the App-aware Edge Device can also be copied or be mapped to the outgoing packet header, e.g, IPv6 header followed by an extension header for further application-aware process.

4. App-aware-process Mid-Point: the Mid-Point provides the path service according to the path set up by the App-aware-process Head-End which satisfies the service requirements conveyed by the packets. The Mid-Point may also adjust the resource locally to guarantee the service requirements depending on a specific policy and the application-aware information conveyed by the packet. Policy definitions and mechanisms are out of the scope of this document.

5. App-aware-process End-Point: the process of the specific service path will end at the End-Point. The service requirements information can be removed at the End-Point together with the outer encapsulation or go on to be conveyed with the packets.

In this way the network is aware of the service requirements expressed by the applications explicitly. According to the service requirement information carried in the packets the network is able to adjust its resources fast in order to satisfy the service requirement of applications. The flow-driven method also reduces the challenges of interoperability and long control loop.

5. APN Requirements

APN doesn't mandate a specific encapsulation however it is reasonable to assume that most of the APN benefits are achieved when utilizing IPv6 encapsulation (e.g. IPv6 header as well as, possibly, extension headers). APN6 (the APN architecture applied to the IPv6/SRv6 data plane) consists of the application-aware information conveyed into the network through the use of IPv6 header and Extension Headers and where the network performs service provisioning, traffic steering, and SLA guarantee according to such information. This section specifies the requirements for supporting the APN framework, including the requirements for conveying and handling the application-aware information and related security requirements. Other encapsulation may be used with some obvious constraint such as, as in the case of MPLS, the limited space available in the header (i.e., 20-bit label size).

5.1. Application-aware Information Conveying Requirements

The application-aware information includes application-aware identification information and network performance requirements information.

1. Application-aware identification information includes the following identifiers (IDs),

* SLA level: indicates the level of SLA requirement of the application such as Gold, Silver, Bronze. In some cases, color (e.g. red, green) can be used to indicate the SLA level.

* Application ID: identifies an application.

* User ID: identifies the user of the application.

* Flow ID: identifies the flow which the application traffic belongs to.

The different combinations of the IDs can be used to provide different granularity of the service provisioning and SLA guarantee for the traffic.

2. Network performance requirements information includes the following parameters:

* Bandwidth: the bandwidth requirement of the application traffic

* Latency: the latency requirement of the application

* Packet loss ratio: the packet loss ratio requirement of the application

* Jitter: the jitter requirement of the application

The different combinations of the parameters are for further expressing the more detailed service requirements of an application, conveyed together with the Application-aware identifiers, which can be used to match to appropriate tunnels/SR Policies, queues that can satisfy these service requirements. If not available, new tunnels/SR Policies can also be triggered to be set up.

[REQ 1a]. Application-aware identification information MUST include Application ID to indicate the application that generates the packet.

[REQ 1b]. SLA level is RECOMMENDED to be included in the Application-aware identification information.

[REQ 1c]. User ID and Flow ID are OPTIONAL to be included in the Application-aware identification information.

[REQ 1d]. Network performance requirements information is OPTIONAL.

[REQ 1e]. All the nodes along the path SHOULD be able to process the application-aware information if needed.

[REQ 1f]. The application-aware information can be generated directly by application, or by the application-aware edge devices though packet inspection or local policy.

[REQ 1g]. The application-aware information SHOULD be kept intact when directly copied from the application-aware edge devices and carried in the packet.

5.2. Application-aware Information Handling Requirements

The app-aware-process Head-End and app-aware-process Mid-Point perform matching operation against the application-aware information, that is, to match IDs and/or service requirements to the corresponding network resources (tunnels/SR policies, queues).

5.2.1. App-aware SLA Guarantee

In order to achieve better Quality of Experience (QoE) of end users and engage customers, the network needs to be able to provide fine-granularity and even application-level SLA guarantee [I-D.li-apn6-problem-statement-usecases].

[REQ 2-1a]. With the application-aware information, the App-aware-process Head-End SHOULD be able to steer the traffic to the tunnel/SR policy that satisfies the matching operation.

[REQ 2-1b]. With the application-aware information, the App-aware-process Head-End SHOULD be able to trigger the setup of the tunnel/SR policy that satisfies the matching operation.

[REQ 2-1c]. With the application-aware information, the App-aware-process Head-End and Mid-Point SHOULD be able to steer the traffic to the queue that satisfies the matching operation.

[REQ 2-1d]. With the application-aware information, the App-aware-process Head-End and Mid-Point SHOULD be able to trigger the configuration of the queue that satisfies the matching operation.

5.2.2. App-aware network slicing

Network slicing provides ways to partition the network infrastructure in either control plane or data plane into multiple network slices that are running in parallel. The resources on each node need to be associated to corresponding slices.

[REQ 2-2a]. With the application-aware information, the App-aware-process Head-End SHOULD be able to steer the traffic to the slice that satisfies the matching operation.

[REQ 2-2a]. With the application-aware information, the App-aware-process Mid-Point SHOULD be able to associate the traffic to the resources in the slice that satisfies the matching operation.

5.2.3. App-aware deterministic networking

Along the path each node needs to provide guaranteed bandwidth, bounded latency, and other properties relevant to the transport of time-sensitive data for the Detnet flows that coexist with the best-effort traffic.

[REQ 2-3a]. With the application-aware information, the App-aware-process Head-End SHOULD be able to steer the traffic to the appropriate path that satisfies the matching operation.

[REQ 2-3b]. With the application-aware information, the App-aware-process Head-End SHOULD be able to trigger the setup of the appropriate path that satisfies the matching operation for the Detnet flows.

[REQ 2-3c]. With the application-aware information, the App-aware-process Mid-Point SHOULD be able to associate the traffic to the resources along the path that satisfies the performance guarantee.

[REQ 2-3d]. With the application-aware information, the App-aware-process Mid-Point SHOULD be able to reserve the resources for the Detnet flows along the path that satisfies the performance guarantee.

5.2.4. App-aware service function chaining

The end-to-end service delivery often needs to go through various service functions, including traditional network service functions such as firewalls, DPI as well as new application-specific functions, both physical and virtual. SFC is applicable to both fixed and mobile networks as well as data center networks.

[REQ 2-4a]. With the application-aware information, the App-aware-process devices SHOULD be able to steer the traffic to the appropriate service function.

[REQ 2-4b]. The App-aware-process devices SHOULD be able to process the application-aware information carried in the packets.

5.2.5. App-aware network measurement

Network measurement can be used for locating silent failure and predicting QoE satisfaction, which enables real-time SLA awareness/ proactive OAM.

[REQ 2-5a]. With the application-aware identification information, the App-aware-process devices SHOULD be able to perform IOAM based on the Application ID.

[REQ 2-5a]. With the application-aware information, the network measurement results can be reported based on the Application ID and verify whether the performance requirements of the application are satisfied.

5.3. Security requirements

[REQ 3a]. The security mechanism defined for APN MUST allow an operator to prevent applications sending arbitrary application-aware information without agreement with the operator.

[REQ 3b]. The security mechanism defined for APN MUST prevent an application requesting a service which it is not entitled to get.

6. IANA Considerations

This document does not include an IANA request.

7. Security Considerations

[I-D.li-apn6-problem-statement-usecases] and describe the security considerations and requirements for APN.

8. Acknowledgements

The authors would like to acknowledge Robert Raszuk (Bloomberg LP) and Yukito Ueno (NTT Communications Corporation) for their valuable reviews and comments.

9. Contributors

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Chongfeng Xie
China Telecom
China

Email: xiechf@chinatelecom.cn

Peng Liu
China Mobile
China

Email: liupengyjy@chinamobile.com

Zhuangzhuang Qin
China Unicom
China

Email: qinzhuangzhuang@chinaunicom.cn

Chang Liu
China Unicom
China

Email: liuc131@chinaunicom.cn

10. References

10.1. Normative References

[I-D.li-apn6-problem-statement-usecases]
Li, Z., Peng, S., Voyer, D., Xie, C., Liu, P., Liu, C., Ebisawa, K., Previdi, S., and J. Guichard, "Problem Statement and Use Cases of Application-aware IPv6 Networking (APN6)", draft-li-apn6-problem-statement-usecases-01 (work in progress), November 2019.

[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>.

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

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8578] Grossman, E., Ed., "Deterministic Networking Use Cases", RFC 8578, DOI 10.17487/RFC8578, May 2019, <https://www.rfc-editor.org/info/rfc8578>.

10.2. Informative References

[RFC3272] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and X. Xiao, "Overview and Principles of Internet Traffic Engineering", RFC 3272, DOI 10.17487/RFC3272, May 2002, <https://www.rfc-editor.org/info/rfc3272>.

Authors' Addresses

Zhenbin Li
Huawei Technologies
China

Email: lizhenbin@huawei.com

Shuping Peng
Huawei Technologies
China

Email: pengshuping@huawei.com

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Cong Li
China Telecom
China

Email: licong@chinatelecom.cn

Liang Geng
China Mobile
China

Email: gengliang@chinamobile.com

Chang Cao
China Unicom
China

Email: caoc15@chinaunicom.cn

Kentaro Ebisawa
Toyota Motor Corporation
Japan

Email: ebisawa@toyota-tokyo.tech

Stefano Previdi
Individual
Italy

Email: stefano@previdi.net

James N Guichard
Futurewei Technologies Ltd.
USA

Email: jguichar@futurewei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2020

Z. Li
S. Peng
Huawei Technologies
D. Voyer
Bell Canada
C. Xie
China Telecom
P. Liu
China Mobile
Z. Qin
China Unicom
K. Ebisawa
Toyota Motor Corporation
S. Previdi
Individual
J. Guichard
Futurewei Technologies Ltd.

March 6, 2020

Problem Statement and Use Cases of Application-aware Networking (APN) draft-li-apn-problem-statement-usecases-00

Abstract

Network operators are facing the challenge of providing better network services for users. As the ever developing 5G and industrial verticals evolve, more and more services that have diverse network requirements such as ultra-low latency and high reliability are emerging, and therefore differentiated service treatment is desired by users. However, network operators are typically unaware of which applications are traversing their network infrastructure, which means that only coarse-grained services can be provided to users. As a result, network operators are only evolving their infrastructure to be large but dumb pipes without corresponding revenue increases that might be enabled by differentiated service treatment. As network technologies evolve including deployments of IPv6, SRv6, Segment Routing over MPLS dataplane, the programmability provided by IPv6 and Segment Routing can be augmented by conveying application related information into the network. Adding application knowledge to the network layer allows applications to specify finer granularity requirements to the network operator.

This document analyzes the existing problems caused by lack of application awareness, and outlines various use cases that could benefit from an Application-aware Networking (APN) architecture.

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 September 7, 2020.

Table of Contents

1. Introduction

Due to the requirement for differentiated traffic treatment driven by diverse new services, the ability to convey the characteristics of an application's traffic flow and program the network infrastructure accordingly to provide fine-grained service assurance is becoming increasingly necessary for network operators. The Application-aware Networking (APN) architecture is being defined to address the requirements and use cases described in this document. APN takes advantage of network programmability by conveying application related information in the data plane allowing applications to specify finer grained requirements to the network infrastructure.

2. Terminology

ACL: Access Control List

APN: Application-aware Networking

APN6: Application-aware Networking for IPv6/SRv6

DPI: Deep Packet Inspection

PBR: Policy Based Routing

QoE: Quality of Experience

SDN: Software Defined Networking

SLA: Service Level Agreement

MPLS: Multiprotocol Label Switching

SR: Segment Routing

SRv6: Segment Routing over IPv6 dataplane

SR-MPLS: Segment Routing over MPLS dataplane

VPN: Virtual Private Network

TE: Traffic Engineering

FRR: Fast Reroute

CAPEX: Capital expenditures

OPEX: Operating expenditures

3. Problem Statement

This section summarizes the challenges currently faced by network operators when attempting to provide fine-grained traffic operations to satisfy the various application-awareness requirements demanded by new services that require differentiated service treatment.

3.1. Large but Dumb Pipe

In today's networks, the infrastructure through which user traffic is forwarded is not able to determine information about the packet, including which application the traffic belongs to, without the introduction of middleware such as DPI, that is, the network and applications are decoupled. It is therefore difficult for network operators to provide fine-grained traffic operations for performance-demanding applications. In order to satisfy the SLA requirements network operators continue to increase the network bandwidth but only carrying very light traffic load (around 30%-40% of its capacity). This situation greatly increases the CAPEX and OPEX but only brings very little revenue from the carried services.

3.2. Network on Its Own

As the network evolves, technologies such as VPN, TE, FRR, SFC, Network Slicing, etc play important roles in satisfying service isolation, SLA guarantee, and high reliability, etc. These network technologies have themselves been evolving, introducing new features that forces the network operator to be continuously upgrading their

network infrastructure. However, none of these network technologies make the network aware of which application traffic belongs to and the fine granularity requirements of the application. Therefore, such continuous network infrastructure upgrade doesn't always enable true fine-grained traffic operation, therefore reducing the ability to bring corresponding revenue increase.

3.3. Decoupling of Network and Applications

MPLS played a very important role in helping the network enter the generation of All-IP successfully. However, MPLS alone doesn't allow a close interworking with the application layer since MPLS encapsulation is, typically, not used by the packet source.

As new services continuously evolve, more encapsulations are required, and this isolation and decoupling has further become the blockage towards the seamless convergence of the network and applications.

3.4. Challenges of Traditional Differentiated Service Provisioning

Several IETF activities have been reviewed which are primarily intended to evolve the IP architecture to support new service definitions which allow preferential or differentiated treatment to be accorded to certain types of traffic. The challenge when using traditional ways to guarantee an SLA is that the packets are not able to carry enough information for indicating applications and expressing their service/SLA requirements. The network devices mainly rely on the 5-tuple of the packets or DPI. However, there are some challenges for these traditional methods in differentiated service provisioning:

1. Five Tuples used for ACL/PBR: five tuples are widely used for ACL/PBR matching of traffic. However, these features cannot provide enough information for the fine-grained service process, and can only provide indirect application information which needs to be translated in order to indicate a specific application.

2. Deep Packet Inspection (DPI): If more information is needed, it must be extracted using DPI which can inspect deep into the packets for application specific information. However, this will introduce more CAPEX and OPEX for the network operator and impose security challenges.

3. Orchestration and SDN-based Solution: In the era of SDN, typically, an SDN controller is used to manage and operate the network infrastructure and orchestrator elements introduce application requirements so that the network is programmed

accordingly. The SDN controller can be aware of the service requirements of the applications on the network through the interface with the orchestrator, and the service requirement is used by the controller for traffic management over the network. However, this method raises the following problems:

A. The whole loop is long and time-consuming which is not suitable for fast service provisioning for critical applications;

B. Too many interfaces are involved in the loop, as shown in Figure 1, which introduce challenges of standardization and inter-operability.

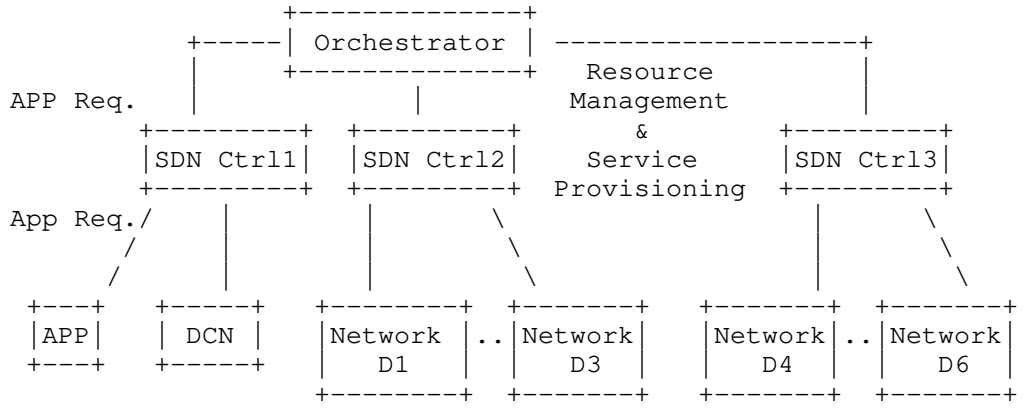


Figure 1: Multiple interfaces involved in the long service-provisioning loop

3.5. Challenges of Supporting New 5G and Edge Computing Technologies

New technologies such as 5G, IoT, and edge computing, are continuously developing leading to more and more new types of services accessing the network. Large volumes of network traffic with diverse requirements such as low latency and high reliability are therefore rapidly increasing. If traditional methods for differentiation of traffic continue to be utilized, it will cause much higher CAPEX and OPEX to satisfy the ever-developing applications' diverse requirements.

4. Key Elements of Application-aware Networking (APN)

Application-aware Networking (APN) aims to address the problems mentioned in Section 3, associated with fine-grained traffic operations that are required in order to satisfy the various

application-awareness requirements demanded by new services that need differentiated service treatment. APN aims to implement a mechanism through which application information is conveyed into the network infrastructure and that describes characteristics of the application associated with a traffic flow (e.g., application identification, network performance requirements), allowing the network to quickly adapt and perform the necessary resource adjustments so to maintain SLA performance guarantees, and hence better serve application fine-grained service requirements.

APN has the following key elements:

1. Application information is conveyed in the data plane through augmentation of existing encapsulations such as IPv6, SRv6 and MPLS. The conveyed application characteristic information (application-aware information) includes application identification and/or its network performance requirements. This element is not intended to be enforced but rather it provides an open option for applications to decide whether to input this application-aware information into their data stream. When a data packet uses APN and conveys the application information, it is referred in this document as an APN packet.

2. Application information and network service provisioning matching providing fine-granularity network service provisioning (traffic operations) and SLA guarantee based on the application-aware information carried in APN packets. This element provides the network capabilities to applications. According to the application-aware information, appropriate network services are selected, provisioned, and provided to the demanding applications to satisfy their performance requirements.

3. Network measurement of network performance and update the match between the applications and corresponding network services for better fine-granularity SLA compliance. The network measurement methods include in-band and out-of-band, passive, active, per-packet, per-flow, per node, end-to-end, etc. These methods can also be integrated.

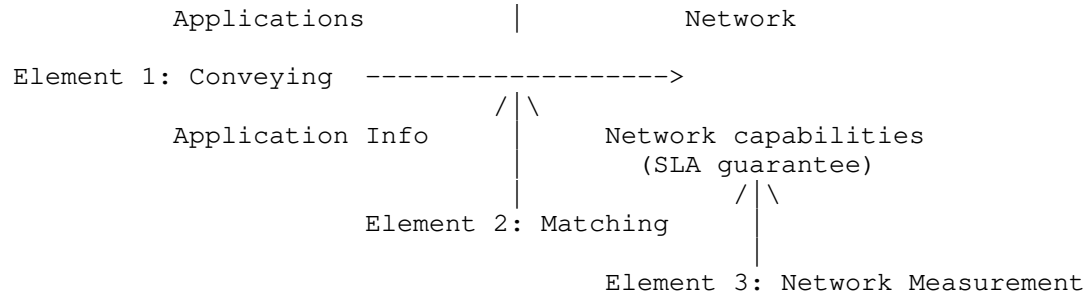


Figure 2: Illustration of the key elements of APN

4.1. Use cases for Application-aware Networking (APN)

This section provides the use cases that can benefit from the application awareness introduced by APN. The corresponding requirements for APN are also outlined.

4.1.1. Application-aware SLA Guarantee

One of the key objectives of APN is for network operators to provide fine-granularity SLA guarantees instead of coarse-grain traffic operations. This will enable them to provide differentiated services for different applications and increase revenue accordingly. Among various applications being carried and running in the network, some revenue-producing applications such as online gaming, video streaming, and enterprise video conferencing have much more demanding performance requirements such as low network latency and high bandwidth. In order to achieve better Quality of Experience (QoE) for end users and engage customers, the network needs to be able to provide fine-granularity and even application-level SLA guarantee. Differentiated service provisioning is also desired.

The APN architecture MUST address the following requirements:

- APN needs to perform the three key elements as described in Section 4.

- Support application-level fine-granularity traffic operation that may include finer QoS scheduling.

4.1.2. Application-aware network slicing

More and more applications/services with diverse requirements are being carried over and sharing the network operators' network infrastructure. However, it is still desirable to have customized network transport that can support some application's specific

requirements, taking into consideration service and resource isolation, which drives the concept of network slicing.

Network slicing provides ways to partition the network infrastructure in either the control plane or data plane into multiple network slices that are running in parallel. These network slices can serve diverse services and fulfill their various requirements at the same time. For example, the mission critical application that requires ultra-low latency and high reliability can be provisioned over a separate network slice.

The APN architecture MUST address the following requirements:

- o APN needs to perform the three key elements as described in Section 4 in the context of network slicing.
- o For the element 2, the APN architecture MUST allow to assign a given traffic flow to specific network slice according to the application information carried in the APN packet.
- o For the element 3, the APN architecture MUST allow the network measurement of each network slice.

4.1.3. Application-aware Deterministic Networking

[RFC8578] documents use cases for diverse industry applications that require deterministic flows over multi-hop paths. Deterministic flows provide guaranteed bandwidth, bounded latency, and other properties relevant to the transport of time-sensitive data, and can coexist on an IP network with best-effort traffic. It also provides for highly reliable flows through provision for redundant paths.

The APN architecture MUST address the following requirements:

- o APN needs to perform the three key elements as described in Section 4 in the context of deterministic networking.
- o For the element 2, the APN architecture MUST allow to assign a given traffic flow to a specific deterministic path according to the application information carried in the APN packet.
- o For the element 3, the APN architecture MUST allow the network measurement of each application-aware deterministic path.

4.1.4. Application-aware Service Function Chaining

End-to-end service delivery often needs to go through various service functions, including traditional network service functions such as firewalls, DPIs as well as new application-specific functions, both physical and virtual. The definition and instantiation of an ordered set of service functions and subsequent steering of the traffic through them is called Service Function Chaining (SFC) [RFC7665]. SFC is applicable to both fixed and mobile networks as well as data center networks.

Generally, in order to manipulate a specific application traffic along the SFC, a DPI needs to be deployed as the first service function of the chain to detect the application, which will impose high CAPEX and consume long processing time. For encrypted traffic, it even becomes impossible to inspect the application.

The APN architecture MUST address the following requirements:

- APN needs to perform the three key elements as described in Section 4 in the context of service function chaining.
- For the element 1, class information can be conveyed.
- For the element 2, the APN architecture MUST allow to assign a given traffic flow to a specific service function chain and MUST allow the subsequent steering according to the application information carried in the APN packets.
- For the element 3, the APN architecture MUST allow the network measurement of each application-aware service function chain.

4.1.5. Application-aware Network Measurement

Network measurement can be used for locating silent failure and predicting QoE satisfaction, which enables real-time SLA awareness/ proactive OAM. Operations, Administration, and Maintenance (OAM) refers to a toolset for fault detection and isolation, and network performance measurement. In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network.

The APN architecture MUST address the following requirements:

- APN needs to perform the two key elements as described in Section 4 in the context of network measurement. The network measurement in the element 3 does not need to be considered here.

5. Application-aware IPv6 Networking (APN6)

As mentioned in Section 3.3, MPLS dataplane is not (or rarely) used at the packet origin (i.e., where the packet is sourced) and therefore it is not possible to assume the MPLS encapsulation is available end-to-end in the traffic flow journey. This scenario is still supported by APN with the ability to classify the packet at the ingress node of the MPLS domain. Of course, it reduces the seamless inter-working between applications and network layer but still APN will improve the resources utilization of the network layer.

APN is intended to be dataplane agnostic. Hence, APN architecture, functions and elements are applicable to both IPv6/SRv6 and MPLS dataplanes. However, it is obvious that IPv6/SRv6 dataplane delivers a better option for APN due to its flexibility, address space and later developments of SRv6 as of [I-D.ietf-6man-segment-routing-header] and [I-D.ietf-spring-srv6-network-programming]. Therefore, this document is mostly focused on the IPv6/SRv6 dataplane. MPLS dataplane is also supported by APN but with some limitations such as backward compatibility and limited address space (20 bits label size).

In this document we refer to APN6 when APN applies to the IPv6/SRv6 dataplane. Application-aware IPv6 Networking (APN6) aims to address APN problems described in Section 3 in the IPv6/SRv6 dataplane. APN6 conveys information into the network infrastructure about the characteristics of the application associated with a traffic flow (including application identification and network performance requirements), using IPv6/SRv6 encapsulation allowing the network to quickly adapt and perform the necessary network resource adjustments to maintain SLA performance guarantees, and hence better serve application fine-grained service requirements.

The advantages of using IPv6/SRv6 to support APN include,

1. Simplicity: Conveying application information with IPv6 encapsulation can just be based on IP reachability.

2. Seamless convergence: Much easier to achieve seamless convergence between applications and network since both are based on IPv6.

3. Great extensibility: IPv6 encapsulation including its extension headers can be used to carry very rich information relevant to applications.

4. Backward compatibility: On-demand network upgrade and service provisioning. If the application information is not recognized

by the node, the packet will be forwarded based on pure IPv6, which ensure backward compatibility.

5. Little dependency: Information conveying and service provisioning are only based on the forwarding plane of devices, which is different from the Orchestration and SDN-based solution which involves multiple elements and diverse interfaces.

6. Quick response: Flow-driven and direct response from devices since it is based on the forwarding plane.

6. IANA Considerations

This document does not include an IANA request.

7. Security Considerations

Since the application information is conveyed into the network, it does involve some security and privacy issues.

First, APN only provides the capability to the applications to provide their profiles and requirements to the network, but it leaves the applications to decide whether to input this information. If the applications decide not to provide any information, they will be treated in the same way as today's network and cannot get the benefits from APN.

Once the application information has been carried in the IPv6 packets and conveyed into the network, the IPv6 extension headers, AH and ESP, can be used to guarantee the authenticity of the added application information.

Any scheme involving an information exchange between layers (application and network layers in this case) will obviously require an accurate valuation of security mechanism in order to prevent any leak of critical information. Some additional considerations may be required for multi-domain use cases. For example, how to agree upon which application information/ID to use and guarantee authenticity for packets traveling through multiple domains (network operators).

8. Acknowledgements

The authors would like to acknowledge Robert Raszuk (Bloomberg LP) and Yukito Ueno (NTT Communications Corporation) for their valuable review and comments.

9. Contributors

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Liang Geng
China Mobile
China

Email: gengliang@chinamobile.com

Chang Cao
China Unicom
China

Email: caoc15@chinaunicom.cn

Chang Liu
China Unicom
China

Email: liuc131@chinaunicom.cn

Cong Li
China Telecom
China

Email: licong@chinatelecom.cn

10. References

10.1. Normative References

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

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

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8578] Grossman, E., Ed., "Deterministic Networking Use Cases", RFC 8578, DOI 10.17487/RFC8578, May 2019, <https://www.rfc-editor.org/info/rfc8578>.

10.2. Informative References

[I-D.ietf-6man-segment-routing-header] Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-26 (work in progress), October 2019.

[I-D.ietf-spring-srv6-network-programming] Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-10 (work in progress), February 2020.

Authors' Addresses

Zhenbin Li
Huawei Technologies
China

Email: lizhenbin@huawei.com

Shuping Peng
Huawei Technologies
China

Email: pengshuping@huawei.com

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Chongfeng Xie
China Telecom
China

Email: xiechf@chinatelecom.cn

Peng Liu
China Mobile
China

Email: liupengyjy@chinamobile.com

Zhuangzhuang Qin
China Unicom
China

Email: qinzhuangzhuang@chinaunicom.cn

Kentaro Ebisawa
Toyota Motor Corporation
Japan

Email: ebisawa@toyota-tokyo.tech

Stefano Previdi
Individual
Italy

Email: stefano@previdi.net

James N Guichard
Futurewei Technologies Ltd.
USA

Email: jguichar@futurewei.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standard Track
Expires: April 20, 2020

Khaled Omar
The Road

October 20, 2020

KHALED Routing Protocol (KRP) Specification draft-omar-krp-07

Status of this Memo

Copyright Notice

Abstract

This document specifies KHALED Routing Protocol (KRP), an Exterior Gateway Protocol (EGP) that introduces a new way of routing IP packets from the source to the destination through different Autonomous Systems (ASs).

The enhancements that KRP adds are:

a) Decreasing the BGP routing table (using the regions concept).
b) Enhancing the routing function (using the stored information).
c) Choosing a better path (using the AS Weight)
c) Enhancing the QoS (using the information separation).

KRP sometimes referred to as Regional Routing Protocol (RRP).

Table of Contents

This contribution has been withdrawn.

Security Considerations

Acknowledgments

The author would like to thank Lee Howard and B. Raveendran for the useful inputs and discussions about KRP.

Author Address

Khaled Omar Ibrahim Omar
The Road
6th of October City, Giza
Egypt

Phone: +2 01003620284
E-mail: eng.khaled.omar@hotmail.com
National ID No.: 28611262102992

IANA Considerations

TBD

References

TBD

Full Copyright Statement

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 10, 2021

H. Tian
F. Zhao
CAICT
C. Xie
China Telecom
T. Li
J. Ma
China Unicom
R. Mwehaire
MTN Uganda Ltd.
E. Chingwena
MTN Group Limited
S. Peng, Ed.
Z. Li
Y. Xiao
Huawei Technologies
July 9, 2020

SRv6 Deployment Consideration
draft-tian-spring-srv6-deployment-consideration-03

Abstract

SRv6 has significant advantages over SR-MPLS and has attracted more and more attention and interest from network operators and verticals. Smooth network migration towards SRv6 is a key focal point and this document provides network design and migration guidance and recommendations on solutions in various scenarios. Deployment cases with SRv6 are also introduced.

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

Table of Contents

1. Introduction

SRv6 is the instantiation of Segment Routing deployed on the IPv6 data plane [RFC8200]. Therefore, in order to support SRv6, the network must first be enabled for IPv6. Over the past several years, IPv6 has been actively promoted all over the world, and the deployments of IPv6 have been ever-increasing which provides the basis for the deployments of SRv6.

With IPv6 as its data plane, for network migration towards SRv6, both software and hardware need to be upgraded. Compared with other new protocols, only IGP and BGP need to be extended to support SRv6, which significantly simplifies the software upgrade required. While the hardware needs to support the new SRv6 header SRH [RFC8754], the design of SRv6 assures compatibility with the existing IPv6 network as an SRv6 SID is designed as a 128-bit IPv6 address and the encapsulation of an SRv6 packet is the same as an IPv6 packet. When only L3VPN over SRv6 BE (Best-Effort) is deployed, there will be no SRH. Therefore, no additional hardware capabilities are required but only software upgrade for protocol extensions.

As the number of services supported by SRv6 increase, e.g. SFC, network slicing, iOAM etc., more SIDs in the SRH may impose new requirements on the hardware. Besides upgrading the hardware, various solutions have already been proposed to relieve the imposed pressure on the hardware, such as Binding SID (BSID) etc. to guarantee the compatibility with the existing network. On the other hand SRv6 has many more advantages over SR-MPLS for the network migration to support new services.

This document summarizes the advantages of SRv6 and provides network migration guidance and recommendations on solutions in various scenarios.

2. Advantages of SRv6

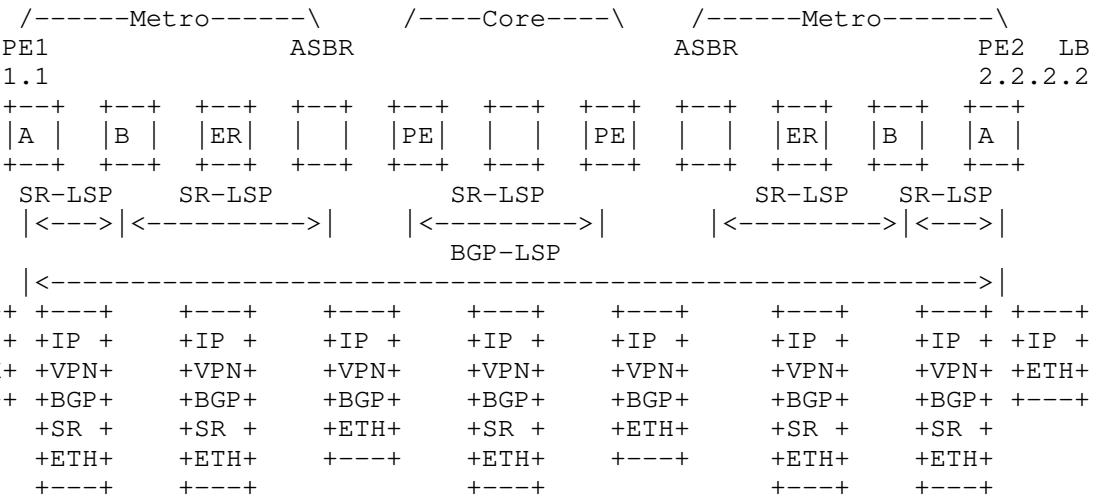
Compared with SR-MPLS, SRv6 has significant advantages especially in large scale networking scenarios.

2.1. IP Route Aggregation

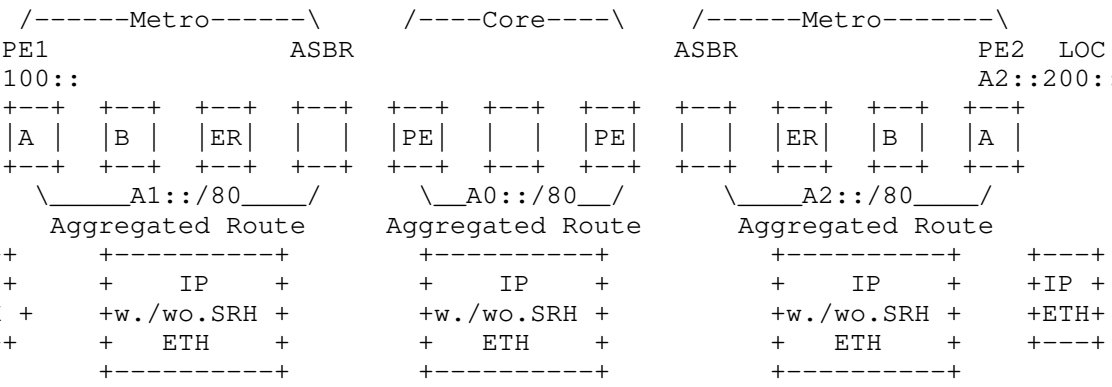
The increasing complexity of service deployment is of concern for network operators, especially in large-scale networking scenarios. With solutions such as multi-segment PW and Option A [RFC4364], the number of service-touch points has increased, and the services, with associated OAM features cannot be deployed end-to-end.

- With Seamless MPLS or SR-MPLS, since the MPLS label itself does not have reachability information, it must be attached to a routable address. The 32-bit host route needs to leak across domains. For an extreme case, as shown in Figure 1a, in a large scale networking scenario, millions of host route LSPs might need to be imported, which places big challenges on the capabilities of the edge nodes.

- With SRv6, owing to its native IP feature of route aggregation as shown in Figure 1b, the aggregated routes can be imported across network domains. For large scale networking, only very few aggregated routes are needed in order to start end-to-end services, which also reduces the scalability requirements on the edge nodes.



(a) SR-MPLS



(b) SRv6

Figure 1. Large-scale Networking with (a) SR-MPLS vs. (b) SRv6

2.2. End-to-end Service Auto-start

In the SR cross-domain scenario, in order to set up end-to-end SR tunnels, the SIDs in each domain need to be imported to other domains.

- With SR-MPLS, SRGB and Node SID need overall network-wide planning, and in the cross-domain scenario, it is difficult or sometimes even impossible to perform as the node SIDs in different

domains may collide. BGP Prefix SID can be used for the cross-domain SID import, but the network operator must be careful when converting the SID to avoid SID collision. Moreover, the pre-allocated SRGB within each domain needs to consider the total number of devices in all other domains, which raises difficulties for the network-wide planning.

- With SRv6, owing to its native IP feature of route reachability, if the IPv6 address space is carefully planned, and the aggregated routes are imported by using BGP4+ (BGP IPv6), the services will auto-start in the cross-domain scenario.

2.3. On-Demand Upgrade

The MPLS label itself does not hold any reachability information, so it must be attached to a routable address, which means that the matching relationship between the label and FEC needs to be maintained along the path.

SR-MPLS uses the MPLS data plane. When the network migrates to SR-MPLS, there are two ways, as shown in Figure 2:

1. MPLS/SR-MPLS Dual stack: the entire network is upgraded first and then deploy SR-MPLS.

2. MPLS and SR-MPLS interworking: mapping servers are deployed at some of the intermediate nodes and then removed once the entire network is upgraded

Regardless of which migration option is chosen, big changes in a wide area is required at the initial stage therefore causing a long time-to-market.

In contrast, the network can be migrated to SRv6 on demand. Wherever the services need to be turned on, only the relevant devices need to be upgraded to enable SRv6, and all other devices only need to support IPv6 forwarding and need not be aware of SRv6. When Traffic Engineering (TE) services are needed, only the key nodes along the path need to be upgraded to support SRv6.

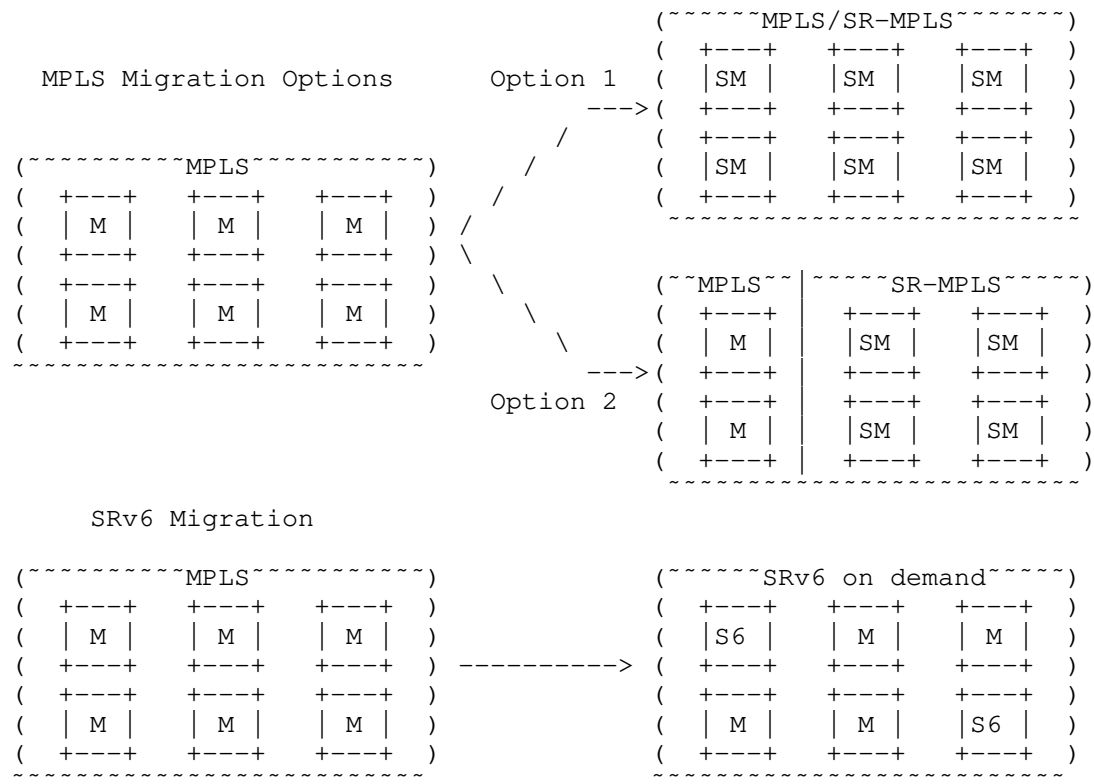


Figure 2. MPLS Domain Migration vs. SRv6 On-Demand Upgrade

2.4. Simplified Service Deployment

With SRv6, the service deployment can be significantly simplified in some scenarios.

2.4.1. Carrier's Carrier

When the customer of the VPN service carrier (Provider Carrier) is itself a VPN service carrier (Customer Carrier), it becomes the scenario of Carrier's Carrier. For this scenario, with SRv6, the service deployment can be significantly simplified.

To achieve better scalability, the CEs of the Provider Carrier (i.e. the PEs of the Customer Carriers) only distribute the internal network routes to the PEs of the Provider Carrier. The customers' routes of the Customer Carriers (i.e. from CE3 and CE4) will not be distributed into the network of the Provide Carrier. Therefore, LDP or Labeled BGP will be run between the CEs of the Provider Carrier

(i.e. CE1 and CE2 in the Figure 3) and the PEs of the Provider Carrier (i.e. PE1 and PE2 in the Figure 3), and LDP will be run between the CEs of the Provider Carrier (i.e. the PEs of the Customer Carriers) and the PEs of the Customer Carrier (i.e. PE3 and PE4 in the Figure 3). MP-BGP will be run between the PEs of the Customer Carrier. The overall service deployment is very complex.

If SRv6 is deployed by the Customer Carrier and the Provider Carrier, no LDP will be ever needed. The Locator routes and Loopback routes of the Customer Carriers can be distributed into the network of the Provider Carrier via BGP, and within each carrier's network only IGP is needed. The end-to-end VPN services can be provided just based on the IPv6 interconnections, and the customer carrier is just like a normal CE to the provider carrier, which significantly simplified the VPN service deployment.

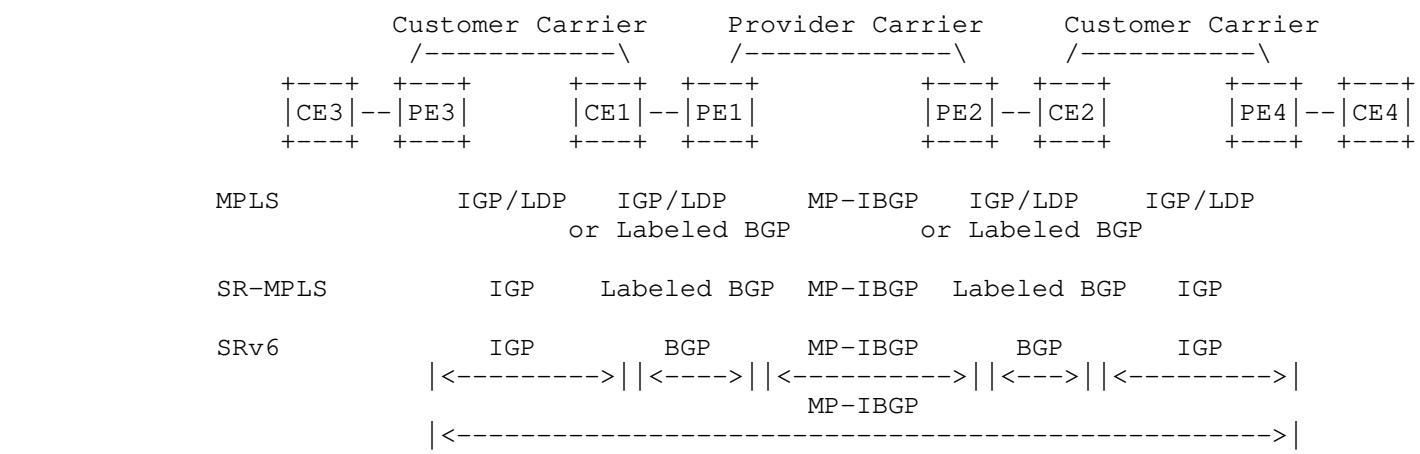


Figure 3. Service deployment with MPLS, SR-MPLS and SRv6

2.4.2. LDP over TE

In a MPLS network, generally RSVP-TE is deployed in the P nodes of the network, and LDP is running between these P nodes and the PE nodes. Customers access to VPN services via the PE nodes. This scenario is called LDP over TE, which is a typical deployment for carriers who want to achieve the TE capability over MPLS network while keep scalability. However, such network configuration and service deployment are very complex.

With SRv6 which can provide both TE capability and IP reachability, the service deployment can be significantly simplified. Only IGP and BGP are needed in the network to launch VPN services.

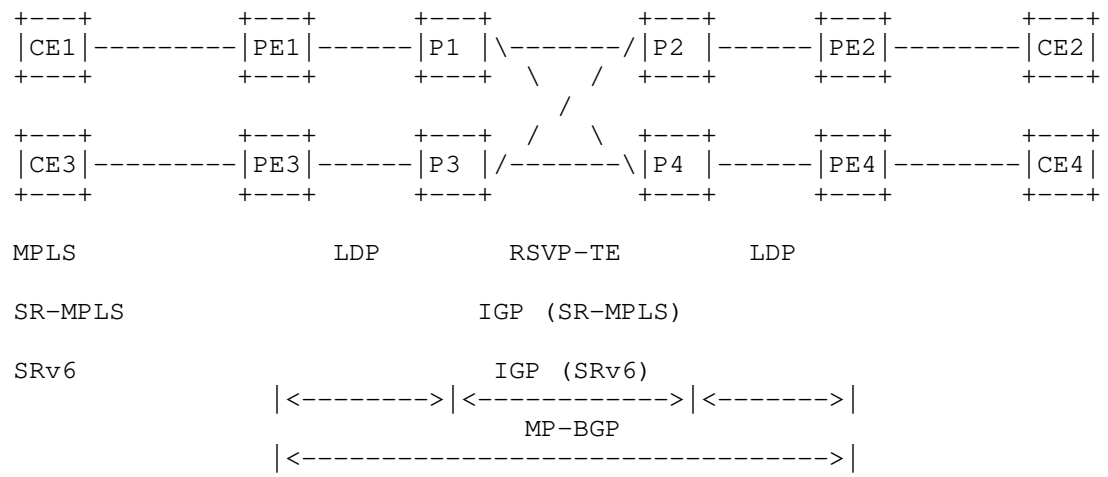


Figure 4. Service deployment with (a) MPLS/SR-MPLS vs. (b) SRv6

3. Compatibility Challenges

By adopting SR Policy, state in the network devices can be greatly reduced, which ultimately evolves the network into a stateless fabric. However, it also brings compatibility challenges on the legacy devices. In particular, the legacy devices need to upgrade software and/or hardware in order to support the processing of SRH.

Furthermore, as the segments in the segment list increase the SR Policy incrementally expands, the encapsulation header overhead increases, which imposes high performance requirements on the performance of hardware forwarding (i.e. the capability of the chipset).

This section identifies the challenges for legacy devices imposed by SRv6 in the following SPRING use cases.

3.1. Fast Reroute (FRR)

FRR is deployed to cope with link or node failures by precomputing backup paths. By relying on SR, Topology Independent Loop-free Alternate Fast Re-route (TI-LFA) [I-D.ietf-rtgwg-segment-routing-ti-lfa] provides a local repair mechanism with the ability to activate the data plane switch-over on to a loop-free backup path irrespective of topologies prior and after the failure.

Using SR, there is no need to create state in the network in order to enforce FRR behavior. Correspondingly, the Point of Local Repair,

i.e. the protecting router, needs to insert a repair list at the head of the segment list in the SRH, encoding the explicit post-convergence path to the destination. This action will increase the length of the segment list in the SRH as shown in Figure 1.

3.2. Traffic Engineering (TE)

TE enables network operators to control specific traffic flows going through configured explicit paths. There are loose and strict options. With the loose option, only a small number of hops along the path is explicitly expressed, while the strict option specifies each individual hop in the explicit path, e.g. to encode a low latency path from one network node to another.

With SRv6, the strict source-routed explicit paths will result in a long segment list in the SRH as shown in Figure 1, which places high requirements on the devices.

3.3. Service Function Chaining (SFC)

The SR segments can also encode instructions, called service segments, for steering packets through services running on physical service appliances or virtual network functions (VNF) running in a virtual environment [I-D.ietf-spring-sr-service-programming]. These service segments can also be integrated in an SR policy along with node and adjacency segments. This feature of SR will further increase the length of the segment list in the SRH as shown in Figure 1.

In terms of SR awareness, there are two types of services, i.e. SR-aware and SR-unaware services, which both impose new requirements on the hardware. The SR-aware service needs to be fully capable of processing SR traffic, while for the SR-unaware services, an SR proxy function needs to be defined.

If the Network Service Header (NSH) based SFC [RFC8300] has already been deployed in the network, the compatibility with existing NSH is required.

3.4. IOAM

IOAM, i.e. "in-situ" Operations, Administration, and Maintenance (OAM), encodes telemetry and operational information within the data packets to complement other "out-of-band" OAM mechanisms, e.g. ICMP and active probing. The IOAM data fields, i.e. a node data list, hold the information collected as the packets traverse the IOAM domain [I-D.ietf-ippm-ioam-data], which is populated iteratively starting with the last entry of the list.

The IOAM data can be embedded into a variety of transports. To support the IOAM on the SRv6 data plane, the O-flag in the SRH is defined [I-D.ietf-6man-spring-srv6-oam], which implements the "punt a timestamped copy and forward" or "forward and punt a timestamped copy" behavior. The IOAM data fields, i.e. the node data list, are encapsulated in the IOAM TLV in SRH, which further increases the length of the SRH as shown in Figure 1.

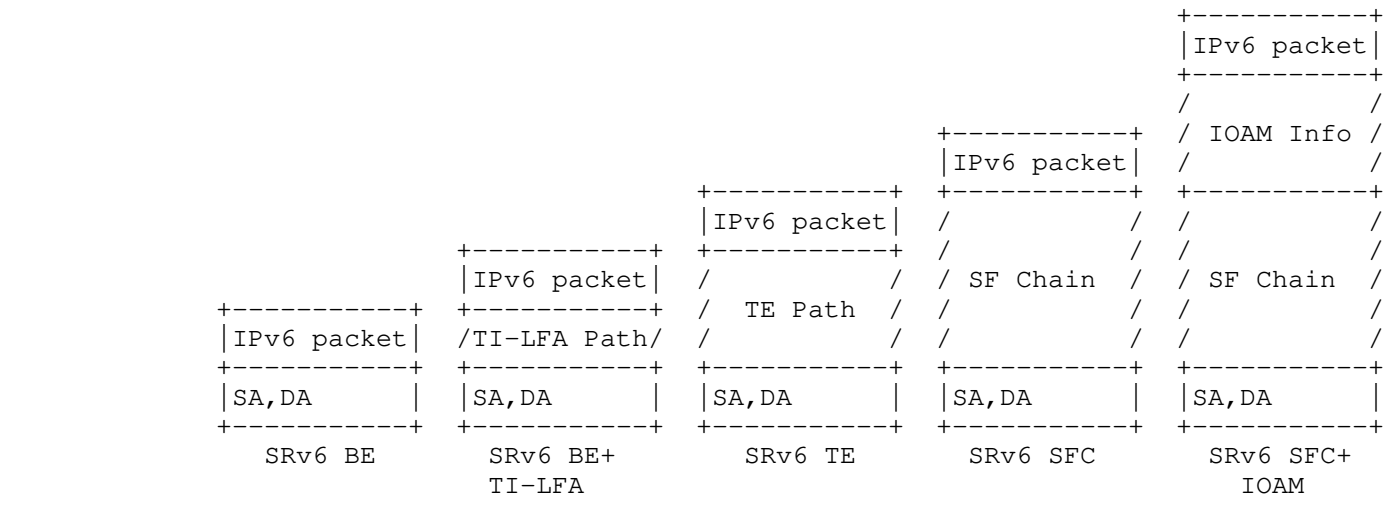


Figure 1. Evolution of SRv6 SRH

Compatibility challenges for legacy devices can be summarized as follows:

- Legacy devices need to upgrade software and/or hardware in order to support the processing of SRH
- As the SRH expands, the encapsulation overhead increases and correspondingly the effective payload decreases
- As the SRH expands, the hardware forwarding performance reduces which requires higher capabilities of the chipset

4. Solutions for mitigating the compatibility challenges

This section provides solutions to mitigate the challenges outlined in section 2.

4.1. Traffic Engineering

With strict traffic engineering, the resultant long SID list in the SRH raises high requirements on the hardware chipset, which can be mitigated by the following solutions.

4.1.1. Binding SID (BSID)

Binding SID [RFC8402] involves a list of SIDs and is bound to an SR Policy. The node(s) that imposes the bound policy needs to store the SID list. When a node receives a packet with its active segment as a BSID, the node will steer the packet in to the bound policy accordingly.

To reduce the long SID list of a strict TE explicit path, BSID can be used at selective nodes, maybe according to the processing capacity of the hardware chipset. BSID can also be used to impose the repair list in the TI-LFA as described in Section 2.1.

4.1.2. PCEP FlowSpec

When the SR architecture adopts a centralized model, the SDN controller (e.g. Path Computation Element (PCE)) only needs to apply the SR policy at the head-end. There is no state maintained at midpoints and tail-ends. Eliminating state in the network (midpoints and tail-points) is a key benefit of utilizing SR. However, it also leads to a long SID list for expressing a strict TE path.

PCEP FlowSpec [I-D.ietf-pce-pcep-flowspec] provides a trade-off solution. PCEP FlowSpec is able to disseminate Flow Specifications (i.e. filters and actions) to indicate how the classified traffic flows will be treated. In an SR-enabled network, PCEP FlowSpec can be applied at the midpoints to enforce traffic engineering policies where it is needed. In that case, state needs to be maintained at the corresponding midpoints of a TE explicit path, but the SID list can be shortened.

4.2. SFC

Currently two approaches are proposed to support SFC over SRv6, i.e. stateless SFC [I-D.ietf-spring-sr-service-programming] and stateful SFC [I-D.ietf-spring-nsh-sr].

4.2.1. Stateless SFC

A service can also be assigned an SRv6 SID which is integrated into an SR policy and used to steer traffic to it. In terms of the capability of processing the SR information in the received packets,

there are two types of services, i.e. SR-aware service and SR-unware service. An SR-aware service can process the SRH in the received packets. An SR-unaware service, i.e. legacy service, is not able to process the SR information in the traffic it receives, and may drop the received packets. In order to support such services in an SRv6 domain, the SR proxy is introduced to handle the processing of SRH on behalf of the SR-unware service. The service SID associated with the SR-unaware service is instantiated on the SR proxy, which is used to steer traffic to the service.

The SR proxy intercepts the SR traffic destined for the service via the locally instantiated service SID, removes the SR information, and sends the non-SR traffic out on a given interface to the service. When receiving the traffic coming back from the service, the SR proxy will restore the SR information and forwards it to the next segment in the segment list.

4.2.2. Stateful SFC

The NSH and SR can be integrated in order to support SFC in an efficient and cost-effective manner while maintaining separation of the service and transport planes.

In this NSH-SR integration solution, NSH and SR work jointly and complement each other. Specifically, SR is responsible for steering packets along a given Service Function Path (SFP) while NSH is for maintaining the SFC instance context, i.e. Service Path Identifier (SPI), Service Index (SI), and any associated metadata.

When a service chain is established, a packet associated with that chain will be first encapsulated with an NSH and then an SRH, and forwarded in the SR domain. When the packet arrives at an SFF and needs to be forwarded to an SF, the SFF performs a lookup based on the service SID associated with the SF to retrieve the next-hop context (a MAC address) between the SFF and SF. Then the SFF strips the SRH and forwards the packet with NSH carrying metadata to the SF where the packet will be processed as specified in [RFC8300]. In this case, the SF is not required to be capable of the SR operation, neither is the SR proxy. Meanwhile, the stripped SRH will be updated and stored in a cache in the SFF, indexed by the NSH SPI for the forwarding of the packet coming back from the SF.

4.3. Light Weight IOAM

In most cases, after the IPv6 Destination Address (DA) is updated according to the active segment in the SRH, the SID in the SRH will not be used again. However, the entire SID list in the SRH will

still be carried in the packet along the path till a PSP/USP is enforced.

The light weight IOAM method [I-D.li-spring-passive-pm-for-srv6-np] makes use of the used segments in the SRH to carry the IOAM information, which saves the extra space in the SRH and mitigate the requirements on the hardware.

4.4. Postcard Telemetry

Existing in-situ OAM techniques incur encapsulation and header overhead issues as described in section 2. Postcard-based Telemetry with Packet Marking for SRv6 on-path OAM[I-D.song-ippm-postcard-based-telemetry], provides a solution that avoids the extra overhead for encapsulating telemetry-related instruction and metadata in SRv6 packets.

5. Design Guidance for SRv6 Network

5.1. Locator and Address Planning

Address Planning is a very important factor for s successful network design, especially an IPv6 network, which will directly affect the design of routing, tunnel, and security. A good address plan can bring big benefit for service deployment and network operation.

If a network has already deployed IPv6 and set up IPv6 subnets, one of the subnets can be selected for the SRv6 Locator planning, and the existing IPv6 address plan will not be impacted.

If a network has not yet deployed IPv6 and there has not been an address plan, it needs to perform the IPv6 address planning first taking the following steps,

1. to decide the IPv6 address planning principles

2. to choose the IPv6 address assignment methods

3. to assign the IPv6 address in a hierarchical manner

For an SRv6 network, in the first step for IPv6 address planning, the following principles are suggested to follow,

1. Unification: all the IPv6 addresses SHOULD be planned altogether, including service addresses for end users, platform addresses (for IPTV, DHCP servers), and network addresses for network devices interconnection.

2. Uniqueness: every single address SHOULD be unique.

3. Separation: service addresses and network addresses SHOULD be planned separately; the SRv6 Locator subnet, the Loopback interface addresses and the link addresses SHOULD be planned separately.

4. Aggregatability: when being distributed across IGP/BGP domains, the addresses in the preassigned subnets (e.g. SRv6 Locator subnet, the Loopback interface subnet) SHOULD be aggregatable, which will make the routing easier.

5. Security: fast tracability of the assigned addresses SHOULD be facilitated, which will make the traffic filtering easier.

6. Evolvablity: enough address space SHOULD be reserved for each subset for future service development.

Considering the above-mentioned IPv6 address planning principles, it has been adopted in some deployment cases to set Locator length 96bits, function length 20bits, and args 12bits.

5.2. PSP

When Locator is imported in ISIS, the system will automatically assign END SID with Flavors such as PSP (Penultimate Segment Pop) and distribute the Locator subnet route through ISIS.

The Flavor PSP, that is, SRH is popped at penultimate segment, provides the following benefits,

1. Reduce the load of ultimate segment endpoint. Ultimate segment endpoint tends to have heavy load since it needs to handle the inner IP/IPv6/Ethernet payload and demultiplex the packet to the right overlay service.

2. Support of incremental deployment on existing network where the ultimate segment endpoint is low-end device that is not fully capable of handling SRH.

6. Incremental Deployment Guidance for SRv6 Migration

Incremental deployment is the key for a smooth network migration to SRv6. In order to quickly launch SRv6 network services and enjoy the benefits brought by SRv6, the recommended incremental SRv6 deployment steps are given as follows. These are based on practical deployment experience earned from the use cases described in [I-D.matsushima-spring-srv6-deployment-status].

The referenced network topology is shown in Figure 5.

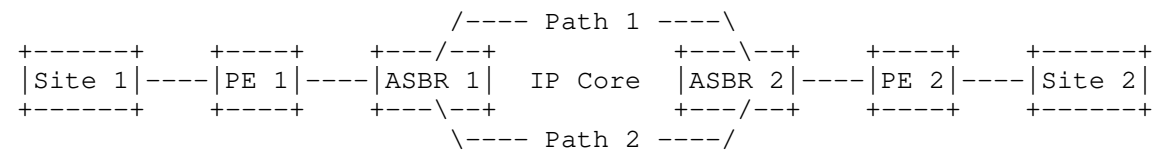


Figure 5. Reference Network Topology

Step1. All the network devices are upgraded to support IPv6.

Step 2. According to service demands, only a set of selected PE devices are upgraded to support SRv6 in order to immediately deploy SRv6 overlay VPN services. For instance, in Figure 3, PE1 and PE2 are SRv6-enabled.

Step 3. Besides the PE devices, some P devices are upgraded to support SRv6 in order to deploy loose TE which enables network path adjustment and optimization. SFC is also a possible service provided by upgrading some of the network devices.

Step 4. All the network devices are upgraded to support SRv6. In this case, it is now possible to deploy strict TE, which enables the deterministic networking and other strict security inspection.

7. Migration Guidance for SRv6/SR-MPLS Co-existence Scenario

As the network migration to SRv6 is progressing, in most cases SRv6-based services and SR-MPLS-based services will coexist.

As shown in Figure 6, in the Non-Standalone (NSA) case specified by 3GPP Release 15, 5G networks will be supported by existing 4G infrastructure. 4G eNB connects to CSG 2, 5G gNB connects to CSG 1, and EPC connects to RSG 1.

To support the 4G services, network services need to be provided between CSG 2 and RSG 1 for interconnecting 4G eNB and EPC, while for the 5G services, network services need to be deployed between CSG 1 and RSG 1 for interconnecting 5G gNB and EPC. Meanwhile, to support X2 interface between the eNB and gNB, network services also need to be deployed between the CSG 1 and CSG 2.

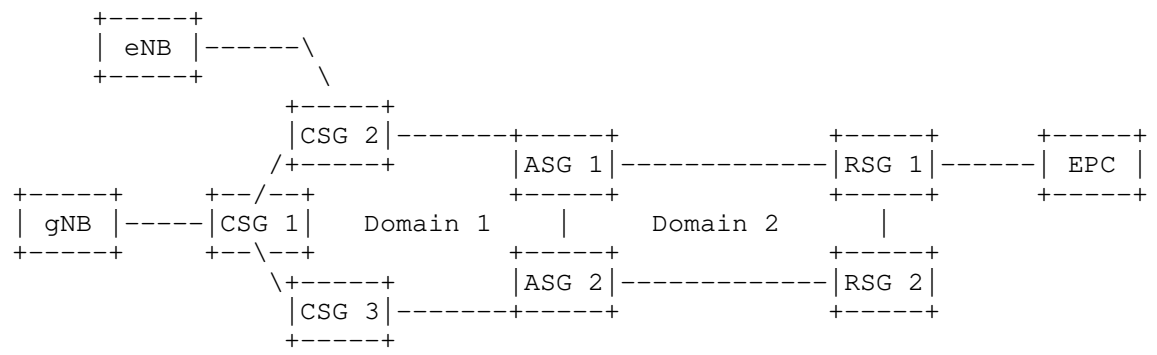


Figure 6. A 3GPP Non-Standalone deployment case

As shown in Figure 6, in most of the current network deployments, MPLS-based network services may have already existed between CSG 2 and RSG 1 for interconnecting 4G eNB and EPC for 4G services.

When 5G services are to be supported, more stringent network services are required, e.g. low latency and high bandwidth. SRv6-based network services could be deployed between CSG 1 and RSG 1 for interconnecting 5G gNB and EPC.

In order to perform smooth network migration, a dual-stack solution can be adopted which deploys both SRv6 and MPLS stack in one node.

With the dual-stack solution, only CSG 1 and RSG 1 need to be upgraded with SRv6/MPLS dual stack. In this case, CSG 1 can immediately start SRv6-based network services to RSG 1 for support of 5G services, but continue to use MPLS-based services to CSG 2 for X2 interface communications. The upgrade at CSG 1 will not affect the existing 4G services supported by the MPLS-based network services between CSG 2 and RSG 1. RSG1 can provide MPLS services to CSG2 for 4G services as well as SRv6 services to CSG 1 for 5G services.

8. Deployment cases

With the current network, the launch of leased line service is slow, the network operation and maintainence is complex, and the configuration points are many. SRv6 can solve the issues above. There have already been several successful SRv6 deployments following the incremental deployment guidance shown in Section 3.

8.1. China Telecom Si'chuan

China Telecom Si'chuan (Si'chuan Telecom) has enabled SRv6 at the PE node of the Magic-Mirror DC in Mei'shan, Cheng'du, Pan'zhihua and other cities. The SRv6 BE tunnel has been deployed through the 163 backbone network which has the IPv6 capability. It enables the fast launch of the Magic-Mirror video service, the interconnection of the DCs in various cities, and the isolation of video services. The deployment case is shown in Figure 7.

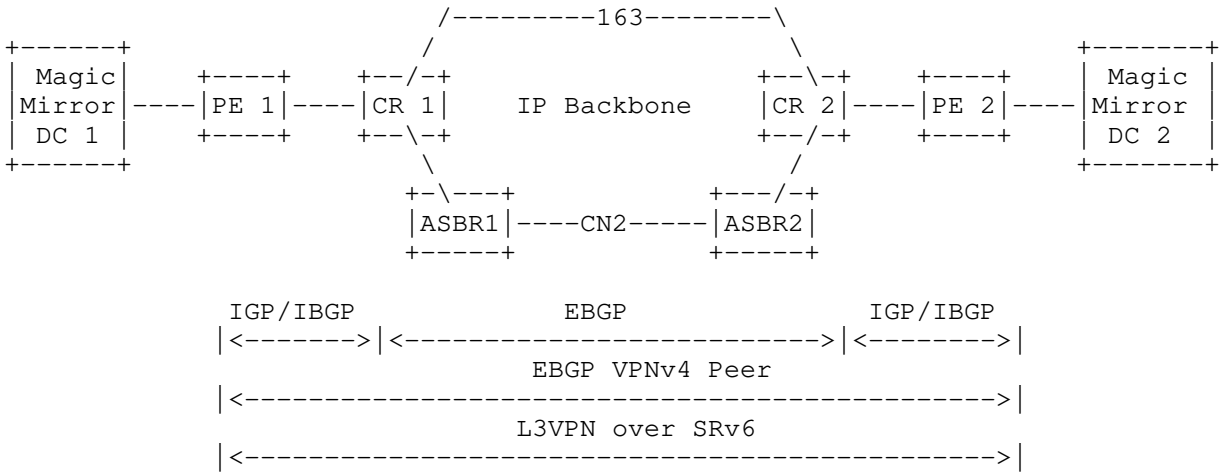


Figure 7. China Telecom Si'chuan deployment case

As shown in Figure 7, IGP (some cities such as Chengdu deploy ISIS, while other cities such as Panzhihua deploy OSPF) and IBGP are deployed between PE and CR, and EBGP is deployed between CRs of cities in order to advertise the aggregation route. EBGP VPNv4 peers are set up between PEs in different cities to deliver VPN private network routes.

The packet enters the SRv6 BE tunnel from the egress PE of DC, and the packet is forwarded according to the Native IP of the 163 backbone network. When the packet reaches the peer PE, the SRH is decapsulated, and then the IP packet is forwarded in the VRF according to the service SID (for example, End.DT4).

In order to further implement the path selection, ASBRs can be upgraded to support SRv6. Different SRv6 policies are configured on the DC egress PE so that different VPN traffic reaches the peer PE

through the 163 backbone network and the CN2 backbone network respectively.

8.2. China Unicom

China Unicom has deployed SRv6 L3VPN over 169 IPv6 backbone network from Guangzhou to Beijing to provide inter-domain Cloud VPN service. The deployment case is shown in Figure 8.

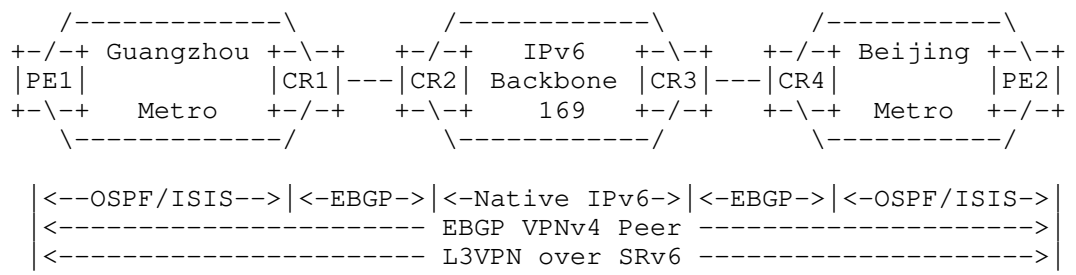


Figure 8. China Unicom SRv6 L3VPN case

In Guangzhou and Beijing metro networks, routers exchange basic routing information using IGP(OSPF/ISIS). The prefixes of IPv6 loopback address and SRv6 locator of routers are different, and both of them need to be imported into the IGP. The 169 backbone is a native IPv6 network. Between metro and backbone, the border routers establish EBGP peer with each other, e.g. CR1 with CR2, CR3 with CR4, to form basic connectivity. All of these constitute the foundation of overlay services, and have not been changed.

PE1 and PE2 establish EBGP peer and advertise VPNv4 routes with each other. If one site connects to two PEs, metro network will use multi RD, community and local preference rules to choose one best route and one backup.

After basic routing among networks and VPN routes between the two PEs are all ready, two PEs encapsulate and forward VPN traffic within SRv6 tunnel. The tunnel is SRv6 best effort (BE) tunnel. It introduces only outer IPv6 header but not SRH header into traffic packets. After encapsulation, the packet is treated as common IPv6 packet and forwarded to the egress PE, which performs decapsulation and forwards the VPN traffic according to specific VRF.

Guangdong Unicom has also lauched the SRv6 L3VPN among Guangzhou, Shenzhen, and Dongguan, which has passed the interop test between different vendors.

With SRv6 enabled at the PE devices, the VPN service can be launched very quickly without impact on the existing traffic. With SRv6 TE further deployed, more benefits of using SRv6 can be exploited.

8.3. MTN Uganda

MTN Uganda has enabled SRv6 at the MPBN PE/P nodes. The SRv6 BE tunnel has been deployed through the MPBN network which has the IPv6 capability. It enables the fast service provisoning for mobile service, enterprise service and internal IT services, and also improves service SLA such as service monitoring and availability. The deployment case is shown in Figure 9.

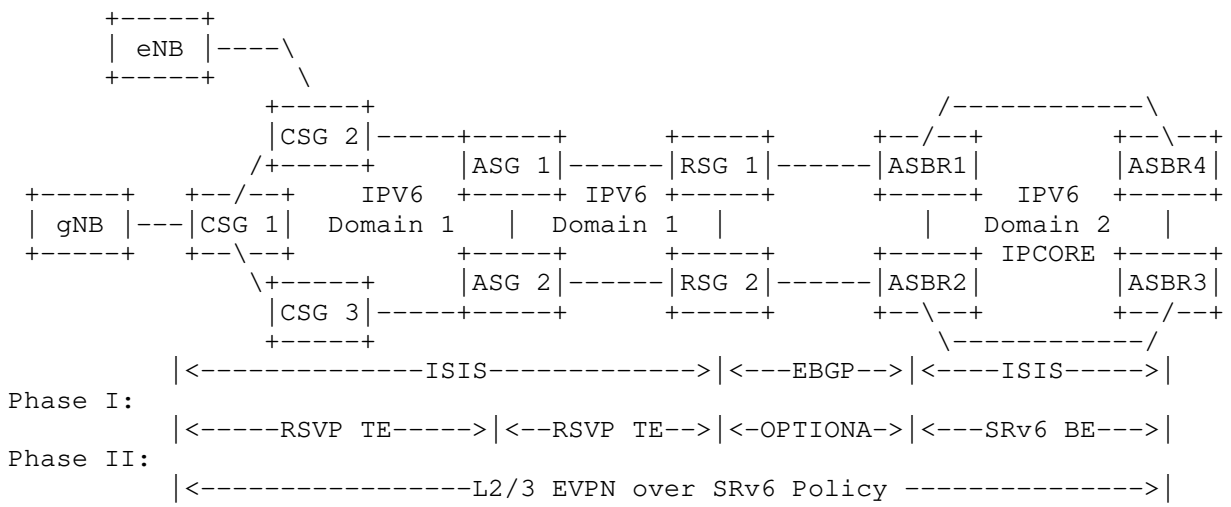


Figure 9. MTN Uganda Deployment Case

As shown in the Figure 9,

In the phase I, SRv6 BE was deployed in MPBN network. All services in the MPBN will be carried through SRv6 BE in the core network. The Option A is deployed between the IPRAN network and Core network.

In the phase II, SRv6 Policy will be deployed E2E from IPRAN to Core. Cross-domain path selection is available for mobile and enterprise services. The service will be carried in SRv6 Policy through the entire MPBN network.

L3VPN and L2VPN services will evolve to EVPN to simplify the network operation and management.

9. IANA Considerations

There are no IANA considerations in this document.

10. Security Considerations

TBD.

11. Acknowledgement

The section on the PSP use cases is inspired from the discussions over the mailing list. The authors would like to acknowledge the constructive discussions from Daniel Voyer, Jingrong Xie, etc..

12. Contributors

Hailong Bai
China Unicom
China

Email:

Jichun Ma
China Unicom
China

Email:

Huizhi Wen
Huawei Technologies
China

Email: wenhuizhi@huawei.com

Ruizhao Hu
Huawei Technologies
China

Email: huruizhao@huawei.com

Jianwei Mao
Huawei
China

Email: maojianwei@huawei.com

13. References

13.1. Normative References

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-16 (work in progress), June 2020.

[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>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC5659] Bocci, M. and S. Bryant, "An Architecture for Multi-Segment Pseudowire Emulation Edge-to-Edge", RFC 5659, DOI 10.17487/RFC5659, October 2009, <https://www.rfc-editor.org/info/rfc5659>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

13.2. Informative References

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-26 (work in progress), October 2019.

[I-D.ietf-6man-spring-srv6-oam]
Ali, Z., Filsfils, C., Matsushima, S., Voyer, D., and M. Chen, "Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)", draft-ietf-6man-spring-srv6-oam-05 (work in progress), June 2020.

[I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., remy@barefootnetworks.com, r., daniel.bernier@bell.ca, d., and J. Lemon, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-09 (work in progress), March 2020.

[I-D.ietf-pce-pcep-flowspec]
Dhody, D., Farrel, A., and Z. Li, "PCEP Extension for Flow Specification", draft-ietf-pce-pcep-flowspec-09 (work in progress), June 2020.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Decraene, B., Francois, P., Voyer, D., Clad, F., and P. Camarillo, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-03 (work in progress), March 2020.

[I-D.ietf-spring-nsh-sr]
Guichard, J., Song, H., Tantsura, J., Halpern, J., Henderickx, W., Boucadair, M., and S. Hassan, "Network Service Header (NSH) and Segment Routing Integration for Service Function Chaining (SFC)", draft-ietf-spring-nsh-sr-02 (work in progress), April 2020.

[I-D.ietf-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca, d., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", draft-ietf-spring-sr-service-programming-02 (work in progress), March 2020.

[I-D.li-spring-passive-pm-for-srv6-np]
Li, C. and M. Chen, "Passive Performance Measurement for SRv6 Network Programming", draft-li-spring-passive-pm-for-srv6-np-00 (work in progress), March 2018.

[I-D.matsushima-spring-srv6-deployment-status]
Matsushima, S., Filsfils, C., Ali, Z., Li, Z., and K. Rajaraman, "SRv6 Implementation and Deployment Status", draft-matsushima-spring-srv6-deployment-status-07 (work in progress), April 2020.

[I-D.song-ippm-postcard-based-telemetry]
Song, H., Zhou, T., Li, Z., Shin, J., and K. Lee, "Postcard-based On-Path Flow Data Telemetry", draft-song-ippm-postcard-based-telemetry-07 (work in progress), April 2020.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

Authors' Addresses

Hui Tian
CAICT
China

Email: tianhui@caict.ac.cn

Feng Zhao
CAICT
China

Email: zhaofeng@caict.ac.cn

Chongfeng Xie
China Telecom
China

Email: xiechf.bri@chinatelecom.cn

Tong Li
China Unicom
China

Email: litong@chinaunicom.cn

Jichun Ma
China Unicom
China

Email: majc16@chinaunicom.cn

Robbins Mwehaire
MTN Uganda Ltd.
Uganda

Email: Robbins.Mwehair@mtn.com

Edmore Chingwena
MTN Group Limited
South Africa

Email: Edmore.Chingwena@mtn.com

Shuping Peng
Huawei Technologies
China

Email: pengshuping@huawei.com

Zhenbin Li
Huawei Technologies
China

Email: lizhenbin@huawei.com

Yaqun Xiao
Huawei Technologies
China

Email: xiaoyaqun@huawei.com

RTGWG
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2021

Q. Xiong
ZTE Corporation
P. Liu
China Mobile

November 1, 2020

The Problem Statement for Precise Transport Networking
draft-xiong-rtgwg-precise-tn-problem-statement-01

Abstract

As described in [I-D.xiong-rtgwg-precise-tn-requirements], the deterministic networks not only need to offer the Service Level Agreements (SLA) guarantees such as low latency and jitter, low packet loss and high reliability, but also need to support the precise services such as flexible resource allocation and service isolation so as to the Precise Transport Networking. However, under the existing IP network architecture with statistical multiplexing characteristics, the existing deterministic technologies are facing long-distance transmission, queue scheduling, dynamic flows and per-flow state maintenance and other controversial issues especially in Wide Area Network (WAN) applications.

This document analyses the problems in existing deterministic technologies to provide precise services in various industries such as 5G networks.

Table of Contents

1. Introduction

1.1. Overview

5G network is oriented to the internet of everything. In addition to the Enhanced Mobile Broadband (eMBB) and Massive Machine Type Communications(mMTC) services, it also supports the Ultra-reliable Low Latency Communications (uRLLC) services. The uRLLC services cover the industries such as intelligent electrical network, intelligent factory, internet of vehicles, industry automation and other industrial internet scenarios, which is the key demand of digital transformation of vertical domains. These uRLLC services

demand SLA guarantees such as low latency and high reliability and other deterministic and precise properties.

For the intelligent electrical network, there are deterministic requirements for communication delay, jitter and packet loss rate. For example, in the electrical current difference model, a delay of 3~10ms and a jitter variation is no more than 100us are required. The isolation requirement is also important. For example, the automatic operation, control of a process, isochronous data and low priority service need to meet the requirements of hard isolation. In addition to the requirements of delay and jitter, the differential protection (DP) service needs to be isolated from other services.

The industrial internet is the key infrastructure that coordinate various units of work over various system components, e.g. people, machines and things in the industrial environment including big data, cloud computing, Internet of Things (IOT), Augment Reality (AR), industrial robots, Artificial Intelligence (AI) and other basic technologies. For example, automation control is one of the basic application and the the core is closed-loop control system. The control process cycle is as low as millisecond level, so the system communication delay needs to reach millisecond level or even lower to ensure the realization of precise control. There are three levels of real-time requirements for industrial interconnection: factory level is about 1s, and process level is 10~100ms, and the highest real-time requirement is motion control, which requires less than 1ms.

1.2. Motivation

The applications in 5G networks demand much more deterministic and precise properties. But traditional Ethernet, IP and MPLS networks which is based on statistical multiplexing provides best-effort packet service and offers no delivery and SLA guarantee. The deterministic forwarding can only apply to flows with such well-defined characteristics as periodicity and burstiness.

Technologies to provide deterministic service has been proposed to provide bounded latency and jitter based on a best-effort packet network. IEEE 802.1 Time-Sensitive Networking (TSN) has been proposed to provide bounded latency and jitter in L2 LAN networks. According to [RFC8655], Deterministic Networking (DetNet) operates at the IP layer and delivers service which provides extremely low data loss rates and bounded latency within a network domain. However, the existing mechanisms are not sufficient for precise performance such as precise latency, jitter variation, packet loss and more other precise and deterministic properties.

As described in [xiong-rtgwg-precise-networking-requirements], the deterministic networks not only need to offer the Service Level Agreements (SLA) guarantees such as low latency and jitter, low packet loss and high reliability, but also need to support the precise services such as flexible resource allocation and service isolation so as to the Precise Transport Networking. However, under the existing IP network architecture with statistical multiplexing characteristics, the existing deterministic technologies are facing long-distance transmission, traffic scheduling, dynamic flows, per-flow state maintenance and other controversial issues especially in Wide Area Network (WAN) applications.

This document analyses the problems in existing deterministic technologies to provide precise services in various industries such as 5G networks.

2. Conventions used in this document

2.1. Terminology

The terminology is defined as [RFC8655] and [xiong-rtgwg-precise-networking-requirements].

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. Problem Statement

3.1. Problem with Traffic Scheduling Mechanisms

As described in [RFC8655], the primary means by which DetNet achieves its QoS assurances in IP networks is to eliminate the latency and packet loss by the provision of sufficient resource at each node. But only the resource itself is not sufficient, the traffic scheduling mechanisms such as queuing, shaping, and scheduling functions must be applied in L3 networks.

The congestion control, queue scheduling and other traffic mechanisms which have been proposed in IEEE 802.1 TSN. But most of them are based on the time synchronization and time cycle, such as IEEE802.1Qbv, IEEE802.1Qch and so on. It will be difficult to achieve precise time synchronization with all network nodes due to deployment and cost reasons. And the shaping and queuing methods

which are not based on time synchronization such as IEEE802.1Qav and IEEE802.1Qcr might not be suitable or available for some L3 networks such as WAN application where multiple dynamic traffic flows may be existed.

Moreover, the requirements of the all nodes in WAN networks to apply the time synchronization and traffic scheduling mechanisms will also lead to the difficulty of network scalability and deployment.

3.2. Problem with Long-distance Transmission Delay and Jitter

In WAN application, long-distance transmission will lead to uncertainties, such as increasing transmission delay, jitter and loss. The link delay of transmission is variable and can not be ignored, and it must be considered in the end-to-end deterministic forwarding mechanisms which are based on time synchronous. So the following problems should be considered.

Precise measurement of the link delay.

The symmetry of bidirectional forwarding of the link delay.

Time cycle alignment in flows aggregation scenario.

3.3. Problem with SLA Guarantees of Dynamic Flows

As described in [RFC8557], deterministic forwarding can only apply to flows with such well-defined characteristics as periodicity and burstiness. As defined in DetNet architecture [RFC8655], the traffic characteristics of an App-flow can be CBR (constant bit rate) or VBR (variable bit rate) of L1, L2 and L3 layers (VBR takes the maximum value when reserving resources). But the current scenarios and technical solutions only consider CBR flow, without considering the coexistence of VBR and CBR, the burst and aperiodicity of flows. The operations such as shaping or scheduling have not been specified. Even TSN mechanisms are based on a constant and forecastable traffic characteristics.

It will be more complicated in WAN applications where much more flows coexist and the traffic characteristics is more dynamic. It is required to offer reliable delivery and SLA guarantee for dynamic flows. For example, periodic flow and aperiodic flow (including micro burst flow, etc.), CBR and VBR flow, flow with different periods or phases, etc. When the network needs to forward these deterministic flows at the same time, it must solve the problems of time window selection, queue processing and aggregation of multiple flows.

Moreover, the existing solutions do not consider the characteristics analysis of service requirements, including the impact of dynamic characteristics analysis on the network, mainly about how to ensure the certainty in the case of dynamic flows.

3.4. Problem with Service Isolation

In some scenarios, such as intelligent electrical network, the isolation requirements are very important. For example, the automatic operation or control of a process or isochronous data and low priority service need to meet the requirements of hard isolation. In addition to the requirements of delay and jitter, the differential protection (DP) service needs to be isolated from other services and hard isolated tunnel is required.

The resource reservation in DetNet can only ensure the statistical reuse of bandwidth resources, but it can not guarantee the precise isolation and control of instantaneous burst and can not realize the hard isolation of each flow. The existing solutions cannot achieve the requirements of service isolation.

3.5. Problem with Precise Resource Allocation

As described in [RFC8655], the primary means by which DetNet achieves its QoS assurances is to reduce, or even completely eliminate, packet loss by the provision of sufficient buffer storage at each node. But it can not be achieved by not enough resource which can be allocated due to practical and cost reason. The existing solutions can not achieve the precise resource allocation.

4. Security Considerations

TBA

5. Acknowledgements

TBA

6. IANA Considerations

TBA

7. Normative References

[I-D.xiong-rtgwg-precise-tn-requirements]
Xiong, Q. and P. Liu, "The Requirements for Precise Transport Networking", draft-xiong-rtgwg-precise-tn-requirements-00 (work in progress), April 2020.

[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>.

[RFC8557] Finn, N. and P. Thubert, "Deterministic Networking Problem Statement", RFC 8557, DOI 10.17487/RFC8557, May 2019, <https://www.rfc-editor.org/info/rfc8557>.

[RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <https://www.rfc-editor.org/info/rfc8655>.

Authors' Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd
Wuhan, Hubei 430223
China

Email: xiong.quan@zte.com.cn

Peng Liu
China Mobile
Beijing 100053
China

Email: liupengyjy@chinamobile.com

RTGWG
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2021

Q. Xiong
ZTE Corporation
P. Liu
China Mobile

November 1, 2020

The Requirements for Precise Transport Networking
draft-xiong-rtgwg-precise-tn-requirements-01

Abstract

The future networks not only need to offer the Service Level Agreements (SLA) guarantees such as low lantency and jitter, low packet loss and high reliability, but also need to support the precise services such as flexible resource allocation and service isolation. This document proposes a set of performance requirements and precise properties for Precise Transport Networking in various industries such as 5G networks.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at 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 May 5, 2021.

Table of Contents

1. Introduction

5G network is oriented to the internet of everything. In addition to the Enhanced Mobile Broadband (eMBB) and Massive Machine Type Communications(mMTC) services, it also supports the Ultra-reliable Low Latency Communications (uRLLC) services. The uRLLC services cover the industries such as intelligent electrical network, intelligent factory, internet of vehicles, industry automation and other industrial internet scenarios, which is the key demand of digital transformation of vertical domains. These uRLLC services demand SLA guarantees such as low latency and high reliability and other deterministic and precise properties.

For the intelligent electrical network, there are deterministic requirements for communication delay, jitter and packet loss rate. For example, in the electrical current difference model, a delay of 3~10ms and a jitter variation is no more than 100us are required. The isolation requirement is also important. For example, the automatic operation, control of a process, isochronous data and low priority service need to meet the requirements of hard isolation. In addition to the requirements of delay and jitter, the differential protection (DP) service needs to be isolated from other services.

The industrial internet is the key infrastructure that coordinate various units of work over various system components, e.g. people,

machines and things in the industrial environment including big data, cloud computing, Internet of Things (IOT), Augment Reality (AR), industrial robots, Artificial Intelligence (AI) and other basic technologies. For example, automation control is one of the basic application and the the core is closed-loop control system. The control process cycle is as low as millisecond level, so the system communication delay needs to reach millisecond level or even lower to ensure the realization of precise control. There are three levels of real-time requirements for industrial interconnection: factory level is about 1s, and process level is 10~100ms, and the highest real-time requirement is motion control, which requires less than 1ms.

The future networks not only need to offer the Service Level Agreements (SLA) guarantees such as low latency and jitter, low packet loss and high reliability, but also need to support the precise services such as flexible resource allocation and service isolation. This document proposes a set of performance requirements and precise properties for Precise Transport Networking in various industries such as 5G networks.

2. Conventions used in this document

2.1. Terminology

The terminology is defined as [RFC8655].

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. Terms of Precise Transport Networking

IEEE 802.1 Time-Sensitive Networking (TSN) has been proposed to provide bounded latency and jitter in L2 LAN networks. According to [RFC8655], Deterministic Networking (DetNet) operates at the IP layer and delivers service with extremely low data loss rates and bounded latency.

However, under the existing IP network architecture with statistical multiplexing characteristics, the existing deterministic technologies are facing long-distance transmission, queue scheduling, dynamic flows and other controversial issues as described in [xiong-rtgwg-precise-tn-problem-statement]. And besides precise latency, jitter, and packet loss, more other precise and deterministic properties and

performances should be provided such as flexible resource allocation and service isolation and so on.

Precise Transport Networking is defined to provide precise SLA guarantees such as latency, jitter, packet loss rate, reliability, and precise control such as flexible resource allocation and service isolation and more other precise services intelligently and dynamically. The purpose of the Precise Transport Networking is based on the hierarchical structure of the transport network, taking advantage of the existing technologies including the flexible precise tunnels technology and the deterministic mechanisms, to support the end-to-end precise service through the characteristics of slicing pieces, hard isolation and preemption characteristics, so as to achieve the high-precision of the future networks.

4. Requirements of Precise Transport Networking

4.1. Precise Latency, Jitter, and Packet Loss

It is required to provide precise Latency, jitter and packet loss dynamically and flexibly in all scenarios for each characterizd flow.

The precise requirements of latency includes bounded latency and low latency. The precise requirements of jitter includes bounded jitter and low jitter. So the precise requirements of latency and jitter may be the combination of latency and jitter, typically including bounded latency and low jitter, low latency and bounded latency, and so on.

4.2. Precise SLA Guarantees for Converged Networks

It is required to provide precise SLA guarantees for converged networks including computing and network convergence, lossless and network convergence, etc.

In some scenarios, such as MEC, it is required to provide precise computing for Controlized CFN/APN. Other resources such as computing resources, energy consumption should be considered. And the utilization and optimization of network resources are extremely important.

4.3. Precise Resource Allocation

As described in [RFC8655], the primary means by which DetNet achieves its QoS assurances is to reduce, or even completely eliminate, packet loss by the provision of sufficient buffer storage at each node. But it can not be achieved by not sufficient resource which can be

allocated due to practical and cost reason. The existing solutions can not achieve the precise resource allocation.

Precise resource allocation is required along with the explicit path with more SLA guarantee parameters like bandwidth, latency, packet loss and so on. The existing technologies such as FlexE and SR tunnels should be taken into consideration.

4.4. Precise Service Isolation

It is required to provide precise service isolation for every flow. In some scenarios, such as intelligent electrical network, the isolation requirements are very important. For example, the automatic operation or control of a process or isochronous data and service with different priorities need to meet the requirements of hard isolation. In addition to the requirements of delay and jitter, the differential protection (DP) service needs to be isolated from other services and hard isolated tunnel is required.

4.5. Precise OAM

It is required to consider precise service performance detection and perception, service support and recovery mechanisms, such as millisecond level service monitoring, 0.0001% packet loss awareness, etc. The existing solutions also do not consider the statistics, analysis and reporting of service performance.

Precise OAM is required including service monitoring, perception, performance statistics, precise service support and recovery mechanism, etc. The OAM mechanisms should be taken into consideration such as In-band OAM, iOAM and so on.

5. Security Considerations

TBA

6. Acknowledgements

TBA

7. IANA Considerations

TBA

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>.

[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>.

[RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <https://www.rfc-editor.org/info/rfc8655>.

Authors' Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd
Wuhan, Hubei 430223
China

Email: xiong.quan@zte.com.cn

Peng Liu
China Mobile
Beijing 100053
China

Email: liupengyjy@chinamobile.com