Layer 3 VPN Network Model
draft-aguado-opsawg-l3sm-l3nm-01

Abstract

RFC 8299 [RFC8299] defines a L3VPN Service Model (L3SM) YANG data model that can be used for communication between customers and network operators. It assumes that there is a monolithic management system with full control of transport resources. This approach (that is valid for the customer to network operator conversation) limits the usage of the model to the role of a Customer Service Model, according to the terminology defined in RFC 8309 [RFC8309].

There is a need for a YANG model for use between the entity that interacts directly with the customer (service orchestrator) and the entity in charge of network orchestration and control which, according to RFC 8309 [RFC8309], can be referred as Service Delivery Model. In some cases, the control of the network is further expanded into per-domain control.

This document uses the L3SM model defined in RFC 8299 [RFC8299], and extends it to facilitate communication between the service orchestrator and transport orchestrator (MSDC), and an MDSC and domain controllers. The resulting model is called the L3VPN Network Model (L3NM).

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/.
1. Introduction

RFC 8299 [RFC8299] defines a L3VPN Service Model (L3SM) YANG data model that can be used for communication between customers and network operators. Although the intention to provide an abstracted view of the customer’s requested services is clear, the assumption is that the model is applied at the top of a monolithic management system with full control of transport resources. That assumption substantially limits the usage of the L3SM to the role of a Customer Service Model, according to the terminology defined in RFC 8309 [RFC8309].

The yang data model defined in this document is called the L3VPN Network Model (L3NM). It enables further capabilities, such as resource management or to serve as a multi-domain orchestration interface, where transport resources must be synchronized. The proposed yang module has been built with a Prune and extend approach, taking as a starting points the YANG model described in RFC 8299 [RFC8299].

This document does not obsolete, but complements, the definitions in RFC 8299 [RFC8299]. It aims to provide a different scope for the L3SM, but does not attempt to address all deployment cases especially those where the L3VPN connectivity is supported through the coordination of different VPNs in different underlying networks. More complex deployment scenarios involving the coordination of different VPN instances and different technologies to provide end-to-end VPN connectivity is out of scope of this document, but is discussed in [I-D.evenwu-opsawg-yang-composed-vpn].

1.1. Terminology

This document assumes that the reader is familiar with the contents of RFC 6241 [RFC6241], RFC 7950 [RFC7950], RFC 8299 [RFC8299], RFC 8309 [RFC8309], and [RFC8453] and uses terminology from those documents. Tree diagrams used in this document follow the notation defined in [RFC8340].

1.2. Requirements Language

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

Figure 1 shows where the L3NM is used in a management stack. The figure is an expansion of the architecture presented in Section 5 of RFC 8299 [RFC8299] and decomposes the box marked "orchestration" in that figure into three separate functional components called "Service Orchestration", "Network Orchestration", and "Domain Orchestration".

At the same time, terminology from RFC 8309 [RFC8309] is introduced to show the distinction between the "Customer Service Model", the "Service Delivery Model", the "Network Configuration Model", and the "Device Configuration Model". In that context, the "Domain Orchestration" and "Config Manager" roles may be performed by "Controllers".
Figure 1: L3SM and L3NM
The L3SM and L3NM may also be set in the context of the ACTN architecture [RFC8453]. Figure 2 shows the Customer Network Controller (CNC), the Multi-Domain Service Coordinator (MDSC), and the Provisioning Network Controller (PNC). It also shows the interfaces between these functional units: the CNC-MDSC Interface (CMI), the MDSC-PNC Interface (MPI), and the Southbound Interface (SBI).
Figure 2: L3SM and L3NM in the Context of ACTN

3. Yang model explanation

The scenarios covered include: the integration of ethernet and encapsulation parameters, the extension for transport resources (e.g. RTs and RDs) to be orchestrated from the management system, far-end
configuration of PEs not managed by the management system and the definition for PE identification.

3.1. Structure of the model

The YANG module is divided into three main containers: "vpn-services", "sites" and "vpn-profiles".

3.2. sites and bearers

A site, as per RFC 8299 [RFC8299], represents a connection of a customer office to one or more VPN services. As this Yang module is the network view, each site is associated with a list of bearers. A bearer is the layer two connection with the site. In the module it is asumened that the bearer has been allocated by the Service Provider (e.g. by the service orchestrator). The bearer is associated to a network element and a port. Hence, a bearer is not just a bearer-reference, but also a true reference to given port in the service provider network.

3.3. Bearer ethernet Encapsulation

The definition of a L3 VPN is commonly defined not only at the IP layer, but also requires to identify parameters at the Ethernet layer, such as encapsulation (e.g. VLAN, QinQ, QinAny, VxLAN, etc). This specification is not supported in [RFC8299], whilst it suggests that any extension on this direction shall be implemented via augmentation of the bearer container. The extension defined to cope with these parameters uses the connection container inside the site-network-access defined by the the [RFC8466]. This container defines protocol parameters to enable connectivity at Layer 2. In the context of L3SM, the augmentation includes only mandatory parameters for the service configuration, which are mainly related to the interface encapsulation. Other definitions from L2SM connection container are left aside. For example, LAG information is not required and it shall be configured prior to the service configuration, being the aggregated interface identified in the model as the bearer-reference, as discussed later in Section 4.4.

3.4. Multi-Domain Resource Management

The implementation of L3 VPN services which spans across administratively separated domains (i.e. that under the administration of different management systems or controllers) requires some network resources to be synchronised between systems. Particularly, there are two resources that must be orchestrated and synchronised to avoid asymmetric (non-functional) configuration, or the usage of unavailable resources. For example, RTs shall be
synchronised between PEs. When every PE is controlled by the same management system, RT allocation can be performed by the system. In cases where the service spans across multiple management systems, this task shall be synchronised and, therefore, the service model must allow this specification. In addition, RDs must be also synchronised to avoid collisions in RD allocation between separated systems. A incorrect allocation might lead into same RD and IP prefixes being exported by different PE routers.

3.5. Remote Far-End Configuration

Depending on the control plane implementation, different network scenarios might require additional information for the L3 VPN service to be configured and active. For example, an L3 VPN Option C service, if no reflection of IPv4 VPN routes is configured via ASBR or route reflector, may require additional configuration (e.g. a new BGP neighbour) to be coordinated between both management systems. This definition requires for every management system participant on the VPN to receive not just their own sites and site-network-accesses, but also to receive information about external ones, identified as an external site-network-access-type. In addition, this particular site-network-access is augmented to include the loopback address of the far-end (remote/external) PE router.

3.6. Provide Edge Identification Point

RFC8299 states that The "bearer-reference" parameter is used in cases where the customer has already ordered a network connection to the SP apart from the IP VPN site and wants to reuse this connection. The string used is an internal reference from the SP and describe the already-available connection. Oftenly, a client interface (either a customer one or an interface used by the SP) is already in place and connected, although it has not being used previously. In some other cases (e.g. for stitching purposes), the termination of a VPN service is done over logical terminations within a PE router.

The bearer-reference must serve as a strict unequivocal parameters to identify the connection between a PE and a client (CE). This means that, despite the type is maintained as a string and there is no restriction in the way this data is formed, the bearer-reference must serve as the unique way to identify the PE router and the client interface. This, together with the encapsulation augments proposed in 4.1, serves as the way to identify the client interface and configure L2 specific parameters.
4. Design of the data model

The augments defined in this document are organised per scenario, as per defined in Section 4. The case described 4.4 does not need any further extension of the data model and only requires a more restricted definition on how the data model is used for PE router and client port identification, so no augment is implemented for this scenario.

The augments implemented are distributed as follows. The first augment implements the extensions for RT and RD definition for the L3 VPN, following the YANG definitions from BESS-L3VPN. The second augment copes with the information from a remote PE not directly under the management system supervision. This augment does not follow any previously defined model and includes the loopback IP address of the external router. The last augment includes information below layer 3 that is required for the service. In particular, we include information related to clients interface encapsulation and aggregation.

The high-level model structure proposed by this document is as shown below:

```
|------------------------ EXAMPLE ------------------------|

module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
    +--rw vpn-profiles
      | +--rw valid-provider-identifiers
      | | +--rw cloud-identifier* [id] {cloud-access}?
      | | | +--rw id string
      | | +--rw encryption-profile-identifier* [id]
      | | | +--rw id string
      | | +--rw qos-profile-identifier* [id]
      | | | +--rw id string
      | | +--rw bfd-profile-identifier* [id]
      | | | +--rw id string
      | | +--rw routing-profile-identifier* [id]
      | | | +--rw id string
      +--rw vpn-services
        +--rw vpn-service* [vpn-id]
          | +--rw vpn-id svc-id
          | | +--rw customer-name? string
          | +--rw vpn-service-topology? identityref
          | +--rw description? string
          +--rw ie-profiles
```
++rw ie-profile* [ie-profile-id]
  ++rw ie-profile-id      string
  ++rw rd?               rt-types:route-distinguisher
  ++rw vpn-targets
    ++rw vpn-target* [route-target]
      ++rw route-target   rt-types:route-target
      ++rw route-target-type  rt-types:route-target-type
  ++rw vpn-nodes
    ++rw vpn-node* [vpn-node-id ne-id]
      ++rw vpn-node-id          string
      ++rw description?         string
      ++rw ne-id                string
      ++rw router-id?           inet:ipv4-address
      ++rw autonomous-system?   uint32
      ++rw node-role?           identityref
      ++rw status
        ++rw admin-enabled?   boolean
        ++ro oper-status?     operational-type
      ++rw maximum-routes
        ++rw address-family* [af]
          ++rw af           address-family
          ++rw maximum-routes?  uint32
      ++rw site-attachments
        ++rw site-attachment* [site-id]
          ++rw site-id       -> /l3vpn-ntw/sites/site/
      ++rw cloud-accesses (cloud-access)?
        ++rw cloud-access* [cloud-identifier]
          ++rw cloud-identifier  -> /l3vpn-ntw/vpn-profiles/valid-provider-identifiers/cloud-identifier/id
      ++rw (list-flavor)?
        ++:(permit-any)
          ++rw permit-any?  empty
        ++:(deny-any-except)
          ++rw permit-site*  -> /l3vpn-ntw/sites/site/site-id
          ++:(permit-any-except)
            ++rw deny-site*    -> /l3vpn-ntw/sites/site/site-id
        ++rw address-translation
          ++rw nat44
            ++rw enabled?            boolean
            ++rw nat44-customer-address?  inet:ipv4-address
        ++rw multicast (multicast)?
          ++rw enabled?            boolean
          ++rw customer-tree-flavors
            ++rw tree-flavor* identityref
          ++rw rp
            ++rw rp-group-mappings
              ++rw rp-group-mapping* [id]
---rw id                        uint16
---rw provider-managed
  ---rw enabled?                   boolean
  ---rw rp-redundancy?             boolean
  ---rw optimal-traffic-delivery?  boolean
---rw rp-address                 inet:ip-address
---rw groups
  ---rw group* [id]
    ---rw id                        uint16
    ---rw (group-format)
      +++:(singleaddress)
        ---rw group-address?          inet:ip-address
      +++:(startend)
        ---rw group-start?           inet:ip-address
        ---rw group-end?            inet:ip-address
  ---rw rp-discovery
    ---rw rp-discovery-type?        identityref
    ---rw bsr-candidates
      ---rw bsr-candidate-address*    inet:ip-address
  ---rw carrierscarrier?
    boolean (carrierscarrier)?
---rw extranet-vpns {extranet-vpn}?
  ---rw extranet-vpn* [vpn-id]
    ---rw vpn-id                    svc-id
  ---rw local-sites-role?          identityref
---rw sites
  ---rw site* [site-id]
    ---rw site-id                   svc-id
    ---rw description?              string
    ---rw requested-site-start?      yang:date-and-time
    ---rw requested-site-stop?       yang:date-and-time
  ---rw locations
    ---rw location* [location-id]
      ---rw location-id             svc-id
      ---rw address?                string
      ---rw postal-code?            string
      ---rw state?                  string
      ---rw city?                   string
      ---rw country-code?           string
  ---rw devices
    ---rw device* [device-id]
      ---rw device-id               svc-id
      ---rw location                -> ../../../locations/location/location-id
    ---rw management
      ---rw address-family?         address-family
      ---rw address                 inet:ip-address
  ---rw site-diversity {site-diversity}?
    ---rw groups
      ---rw group* [group-id]
---rw group-id   string
---rw management
  |  ---rw type     identityref
  ---rw vpn-policies
    |   ---rw vpn-policy* [ vpn-policy-id ]
    |       ---rw vpn-policy-id   svc-id
    |       ---rw entries* [ id ]
    |     |       ---rw id         svc-id
    |     |       ---rw filters
    |     |         ---rw filter* [ type ]
    |     |           |       ---rw type     identityref
    |     |           |       ---rw lan-tag*   string { lan-tag }?
    |     |           |       ---rw ipv4-lan-prefix* inet:ipv4-prefix { ipv4 }?
    |     |           |       ---rw ipv6-lan-prefix* inet:ipv6-prefix { ipv6 }?
    |     |       ---rw vpn* [ vpn-id ]
    |     |       ---rw vpn-id        -> /l3vpn-ntw/vpn-services/vpn-service
    / vpn-id
    |       ---rw site-role?   identityref
    ---rw site-vpn-flavor?   identityref
  ---rw maximum-routes
    |       ---rw address-family* [ af ]
    |     |       ---rw af         address-family
    |     |       ---rw maximum-routes?    uint32
  ---rw security
    |       ---rw encryption { encryption }?
    |       ---rw enabled?    boolean
    |       ---rw layer?      enumeration
    |       --- rw encryption-profile
    |         --- rw ( profile )?
    |           |         ---:( provider-profile )
    |           |             |       --- rw profile-name?    -> /l3vpn-ntw/vpn-profile
    |           |         ---:( customer-profile )
    |           |             |       --- rw algorithm?    string
    |           |             |       --- rw ( key-type )?
    |           |             |         ---:( psk )
    |           |             |             |       --- rw preshared-key?    string
  --- rw service
    |       --- rw qos { qos }?
    |       --- rw qos-classification-policy
    |         --- rw rule* [ id ]
    |         --- rw ( match-type )?
    |         |         ---:( match-flow )
    |         |             |       --- rw dscp?    inet:dscp
    |         |             |       --- rw dot1p?    uint8
    |         |             |       --- rw ipv4-src-prefix?    inet:ipv4-prefix
    |         |             |       --- rw ipv6-src-prefix?    inet:ipv6-prefix
Internet-Draft                    l3nm                         July 2019

| | | | | | | +--rw ipv4-dst-prefix?     inet:ipv4-prefix
| | | | | | | +--rw ipv6-dst-prefix?     inet:ipv6-prefix
| | | | | | | +--rw l4-src-port?         inet:port-number
| | | | | | | +--rw target-sites*        svc-id {target-sites}
?  | | | | | | +--rw l4-src-port-range
     | | | | | | | +--rw lower-port?   inet:port-number
     | | | | | | | +--rw upper-port?   inet:port-number
     | | | | | | +--rw l4-dst-port?         inet:port-number
     | | | | | | | +--rw l4-dst-port-range
     | | | | | | | | +--rw lower-port?   inet:port-number
     | | | | | | | | +--rw upper-port?   inet:port-number
     | | | | | | +--rw protocol-field?      union
     | | | | |   +--:(match-application)
         | | | | +--rw match-application?   identityref
         | | | +--rw target-class-id?           string
         | | +--rw qos-profile
         | +--rw (qos-profile)?
         |     +--:(standard)
         | | +--rw profile?   -> /l3vpn-ntw/vpn-profiles/valid-prov
         | +--rw (qos-profile)?
         |     +--:(custom)
         | | +--rw classes {qos-custom}?
         | | | +--rw class* [class-id]
         | | | | +--rw class-id      string
         | | | +--rw direction?    identityref
         | | | +--rw rate-limit?   decimal64
         | | | +--rw latency
         | | | | +--rw (flavor)?
         | | | | | +--:(lowest)
         | | | | | | +--rw use-lowest-latency?   empty
         | | | | | +--:(boundary)
         | | | | | +--rw latency-boundary?     uint16
         | | | +--rw jitter
         | | | | +--rw (flavor)?
         | | | | | +--:(lowest)
         | | | | | | +--rw use-lowest-jitter?   empty
         | | | | | +--:(boundary)
         | | | | | +--rw latency-boundary?     uint32
         | | | +--rw bandwidth
         | | | | +--rw guaranteed-bw-percent    decimal64
         | | | | +--rw end-to-end?              empty
         | +--rw carrierscarrier {carrierscarrier}?
         | | +--rw signalling-type?   enumeration
         | | +--rw multicast {multicast}?
         | | | +--rw multicast-site-type?    enumeration
         | | | +--rw multicast-address-family
         | | | | +--rw ipv4?   boolean {ipv4}?
         | | | | +--rw ipv6?   boolean {ipv6}?
         | | | +--rw protocol-type?    enumeration

++--rw traffic-protection {fast-reroute}?
  |  +--rw enabled?  boolean
++--rw routing-protocols
  +--rw routing-protocol* [type]
    +--rw type                identityref
    +--rw routing-profiles* [id]
      |  +--rw id       -> /l3vpn-ntw/vpn-profiles/valid-provider-identifiers/routing-profile-identifier/id
      |  +--rw type?    ie-type
        +--rw ospf {rtg-ospf}?
          |  +--rw address-family*   address-family
          +--rw area-address       yang:dotted-quad
          +--rw metric?            uint16
          +--rw mtu?               uint16
          +--rw security
            |  +--rw auth-key?  string
          +--rw sham-links {rtg-ospf-sham-link}?
            |  +--rw sham-link* [target-site]
              |    +--rw target-site  svc-id
              |    +--rw metric?     uint16
            +--rw bgp {rtg-bgp}?
              |  +--rw autonomous-system  uint32
              +--rw address-family*   address-family
              +--rw neighbor?         inet:ip-address
              +--rw multihop?         uint8
              +--rw security
                |  +--rw auth-key?  string
            +--rw static
              |  +--rw cascaded-lan-prefixes
              |    +--rw ipv4-lan-prefixes* [lan next-hop] (ipv4)?
              |      |  +--rw lan        inet:ipv4-prefix
              |      +--rw lan-tag?  string
              |    +--rw ipv6-lan-prefixes* [lan next-hop] (ipv6)?
              |      |  +--rw lan        inet:ipv6-prefix
              |      +--rw lan-tag?  string
              +--rw rip {rtg-rip}?
                |  +--rw address-family*   address-family
              +--rw vrrp {rtg-vrrp}?
                |  +--rw address-family*   address-family
                |  +--ro actual-site-start?  yang:date-and-time
                +--ro actual-site-stop?   yang:date-and-time
++--rw site-bearers
  +--rw bearer* [bearer-id]
    |  +--rw bearer-id  string
    +--rw ne-id?      string
    +--rw port-id?    string
++--rw site-network-accesses
++-rw site-network-access* [site-network-access-id]
  +--rw site-network-access-id svc-id
  +--rw description? string
  +--rw status
    |  +--rw admin-enabled? boolean
    |  +--ro oper-status? operational-type
  +--rw site-network-access-type? identityref
  +--rw (location-flavor)
    |  +--:(location)
      |    +--rw location-reference? -> ../../../locations/location-id
  +--:(device)
    |    +--rw device-reference? -> ../../../devices/device/device-id
  +--rw access-diversity {site-diversity}?
    +--rw groups
      |    +--rw group* [group-id]
      |    +--rw group-id string
    +--rw constraints
      +--rw constraint* [constraint-type]
        +--rw constraint-type identityref
        +--rw target
          +--rw (target-flavor)?
            +--:(id)
              |    +--rw group* [group-id]
              |    +--rw group-id string
            +--:(all-accesses)
              |    +--rw all-other-accesses? empty
            +--:(all-groups)
              |    +--rw all-other-groups? empty
  +--rw bearer
    +--rw requested-type {requested-type}?
      |    +--rw requested-type? string
      |    +--rw strict? boolean {always-on}?
    +--rw always-on? boolean {always-on}?
    +--rw bearer-reference? string {bearer-reference}?
    +--rw connection
      |    +--rw encapsulation-type? identityref
      |    +--rw eth-inf-type? identityref
      +--rw tagged-interface
        |    +--rw type? identityref
        |    +--rw dot1q-vlan-tagged {dot1q}?
        |      +--rw tg-type? identityref
        |      +--rw cvlan-id uint16
        +--rw priority-tagged
          |    +--rw tag-type? identityref
          +--rw qinq {qinq}?
            |    +--rw tag-type? identityref
            +--rw svlan-id uint16
            +--rw cvlan-id uint16
Internet-Draft  13nm  July 2019

---rw qinany (qinany)?
  |  ---rw tag-type? identityref
  |  ---rw svlan-id uint16
  +--rw vxlan (vxlan)?
    |  ---rw vni-id uint32
    |  ---rw peer-mode? identityref
    |  ---rw peer-list* [peer-ip]
    +--rw peer-ip inet:ip-address
  +--rw pseudowire
    +--rw vcid? uint32
  +--rw ip-connection
  +--rw ipv4 {ipv4}?*
    |  ---rw address-allocation-type? identityref
    |  ---rw provider-dhcp
    |     |  ---rw provider-address? inet:ipv4-address
    |  ---rw prefix-length? uint8
    |  ---rw (address-assign)?
    |     |  ++:(number)
    |     |     |  ---rw number-of-dynamic-address? uint16
    |     |  ++:(explicit)
    |     |  ---rw customer-addresses
    |     |     |  ---rw address-group* [group-id]
    |     |     |  ---rw group-id string
    |     |     |  ---rw start-address? inet:ipv4-address
    |     |     |  ---rw end-address? inet:ipv4-address
    +--rw dhcp-relay
    |  ---rw provider-address? inet:ipv4-address
    |  ---rw prefix-length? uint8
    |  ---rw customer-dhcp-servers
    |     |  ---rw server-ip-address* inet:ipv4-address
    +--rw addresses
    +--rw provider-address? inet:ipv4-address
    +--rw customer-address? inet:ipv4-address
    +--rw prefix-length? uint8
  +--rw ipv6 {ipv6}?*
    |  ---rw address-allocation-type? identityref
    |  ---rw provider-dhcp
    |     |  ---rw provider-address? inet:ipv6-address
    |  ---rw prefix-length? uint8
    |  ---rw (address-assign)?
    |     |  ++:(number)
    |     |     |  ---rw number-of-dynamic-address? uint16
    |     |  ++:(explicit)
    |     |  ---rw customer-addresses
    |     |     |  ---rw address-group* [group-id]
    |     |     |  ---rw group-id string
    |     |     |  ---rw start-address? inet:ipv6-address
    |     |     |  ---rw end-address? inet:ipv6-address

| ++-rw dhcp-relay |
|---+--rw provider-address?      inet:ipv6-address |
|    | --rw prefix-length?           uint8 |
|    | --rw customer-dhcp-servers |
|    |     | --rw server-ip-address*   inet:ipv6-address |
|    | --rw addresses |
|    |     | --rw provider-address?      inet:ipv6-address |
|    |     | --rw customer-address?      inet:ipv6-address |
|    |     | --rw prefix-length?           uint8 |
|    |     | --rw addresses |
|    |     |     | --rw provider-address?      inet:ipv6-address |
|    |     |     | --rw customer-address?      inet:ipv6-address |
|    |     |     | --rw prefix-length?           uint8 |
|    |     | --rw oam |
|    |     | --rw bfd {bfd}? |
|    |     |     | --rw enabled?              boolean |
|    |     |     | --rw (holdtime)? |
|    |     |     |     | --rw fixed-value?    uint32 |
|    |     | --rw security |
|    |     |     | --rw authentication |
|    |     |     |     | --rw encryption {encryption}? |
|    |     |     |     |     | --rw enabled?              boolean |
|    |     |     |     |     | --rw layer?                enumeration |
|    |     |     |     | --rw encryption-profile |
|    |     |     |     |     | --rw (profile)? |
|    |     |     |     |     |     | --:(provider-profile) |
|    |     |     |     |     | --:(customer-profile) |
|    |     |     |     |     |     | --rw algorithm?            string |
|    |     |     |     |     |     | --rw (key-type)? |
|    |     |     |     |     |     |     | --:(psk) |
|    |     |     |     |     |     |     | +--rw preshared-key?   string |
|    |     | --rw service |
|    |     |     | --rw svc-input-bandwidth     uint64 |
|    |     |     | --rw svc-output-bandwidth    uint64 |
|    |     |     | --rw svc-mtu                 uint16 |
|    |     | --rw qos {qos}? |
|    |     |     | --rw qos-classification-policy |
|    |     |     |     | --rw rule* [id] |
|    |     |     |     |     | --rw id                   string |
|    |     |     |     |     | --rw (match-type)? |
|    |     |     |     |     |     | --:(match-flow) |
|    |     |     |     |     |     |     | --rw match-flow |
|    |     |     |     |     |     |     |     | --rw dscp?           inet:dscp |
|    |     |     |     |     |     |     |     | --rw dot1p?               uint8 |
|    |     |     |     |     |     |     |     | --rw ipv4-src-prefix?     inet:ipv4-prefix |
|    |     |     |     |     |     |     |     | --rw ipv4-dst-prefix?     inet:ipv4-prefix |
|    |     |     |     |     |     |     |     | --rw ipv6-src-prefix?    inet:ipv6-prefix |
|    |     |     |     |     |     |     |     | --rw ipv6-dst-prefix?    inet:ipv6-prefix |
5. Yang module

```yaml
<CODE BEGINS>file "ietf-l3vpn-ntw@2019-07-04.yang"
module ietf-l3vpn-ntw {
    yang-version 1.1;
}
```
prefix l3vpn-ntw;
import ietf-inet-types {
  prefix inet;
}
import ietf-yang-types {
  prefix yang;
}
import ietf-netconf-acm {
  prefix nacm;
}
import ietf-routing-types {
  prefix rt-types;
}
organization "DRAFT Proposal";
contact
"WG List: draft proposal
Editor:
  draft proposal
Chairs:
";
description "This YANG module defines a generic network-oriented model for the configuration of Layer 3 VPNs. This model is common across all vendor implementations.

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

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

This version of this YANG module is based on RFC 8299; see the RFC itself for full legal notices.";
revision 2019-07-04 {
  description
  "Initial document. The document as a whole is based on L3SM module, defined in RFC 8299, modified to fit the requirements of the platforms at the network layer."
  reference
  "RFC 8049.";
/* Features */
feature cloud-access {
    description
    "Allows the VPN to connect to a CSP.";
}
feature multicast {
    description
    "Enables multicast capabilities in a VPN.";
}
feature ipv4 {
    description
    "Enables IPv4 support in a VPN.";
}
feature ipv6 {
    description
    "Enables IPv6 support in a VPN.";
}
feature lan-tag {
    description
    "Enables LAN Tag support in a VPN Policy filter.";
}
feature carrierscarrier {
    description
    "Enables support of CsC.";
}
feature extranet-vpn {
    description
    "Enables support of extranet VPNs.";
}
feature site-diversity {
    description
    "Enables support of site diversity constraints.";
}
feature encryption {
    description
    "Enables support of encryption.";
}
feature qos {
    description
    "Enables support of classes of services.";
}
feature qos-custom {
    description
    "Enables support of the custom QoS profile.";
}
feature rtg-bgp {
    description
}
"Enables support of the BGP routing protocol.";
}
feature rtg-rip {
  description
  "Enables support of the RIP routing protocol.";
}
feature rtg-ospf {
  description
  "Enables support of the OSPF routing protocol.";
}
feature rtg-ospf-sham-link {
  description
  "Enables support of OSPF sham links.";
}
feature rtg-vrrp {
  description
  "Enables support of the VRRP routing protocol.";
}
feature fast-reroute {
  description
  "Enables support of Fast Reroute.";
}
feature bfd {
  description
  "Enables support of BFD.";
}
feature always-on {
  description
  "Enables support of the 'always-on' access constraint.";
}
feature requested-type {
  description
  "Enables support of the 'requested-type' access constraint.";
}
feature bearer-reference {
  description
  "Enables support of the 'bearer-reference' access constraint.";
}
feature target-sites {
  description
  "Enables support of the 'target-sites' match flow parameter.";
}
/* Typedefs */
typedef svc-id {
  type string;
  description
  "Defines a type of service component identifier.";
}
typedef template-id {
    type string;
    description
    "Defines a type of service template identifier.";
}
typedef address-family {
    type enumeration {
        enum ipv4 {
            description
            "IPv4 address family.";
        }
        enum ipv6 {
            description
            "IPv6 address family.";
        }
    }
    description
    "Defines a type for the address family.";
}
typedef ie-type {
    type enumeration {
        enum "import" {
            value 0;
            description "Import routing profile.";
        }
        enum "export" {
            value 1;
            description "Export routing profile";
        }
        enum "both" {
            value 2;
            description "Import/Export routing profile";
        }
    }
}
typedef operational-type {
    type enumeration {
        enum "up" {
            value 0;
            description "Operational status UP.";
        }
        enum "down" {
            value 1;
            description "Operational status DOWN";
        }
        enum "unknown" {
value 2;
    description "Operational status UNKNOWN";
  }
}

/* Identities */
identity site-network-access-type {
  description "Base identity for site-network-access type.";
}
identity point-to-point {
  base site-network-access-type;
  description "Identity for point-to-point connection.";
}

/* Extension */
identity pseudowire {
  base site-network-access-type;
  description "Identity for pseudowire connection.";
}
identity loopback {
  base site-network-access-type;
  description "Identity for an internal loopback interface.";
}

/* End of Extension */
identity multipoint {
  base site-network-access-type;
  description "Identity for multipoint connection.
Example: Ethernet broadcast segment.";
}
identity placement-diversity {
  description "Base identity for site placement constraints.";
}
identity bearer-diverse {
  base placement-diversity;
  description "Identity for bearer diversity.
The bearers should not use common elements.";
}
identity pe-diverse {
  base placement-diversity;
  description "Identity for PE diversity.";

identity pop-diverse {
  base placement-diversity;
  description "Identity for POP diversity.";
}

identity linecard-diverse {
  base placement-diversity;
  description "Identity for linecard diversity.";
}

identity same-pe {
  base placement-diversity;
  description "Identity for having sites connected on the same PE.";
}

identity same-bearer {
  base placement-diversity;
  description "Identity for having sites connected using the same bearer.";
}

identity customer-application {
  description "Base identity for customer application.";
}

identity web {
  base customer-application;
  description "Identity for Web application (e.g., HTTP, HTTPS).";
}

identity mail {
  base customer-application;
  description "Identity for mail application.";
}

identity file-transfer {
  base customer-application;
  description "Identity for file transfer application (e.g., FTP, SFTP).";
}

identity database {
  base customer-application;
  description "Identity for database application.";
}

identity social {
  base customer-application;
  description
Internet-Draft                    l3nm                         July 2019

"Identity for social-network application.";

identity games {
  base customer-application;
  description
  "Identity for gaming application.";
}

identity p2p {
  base customer-application;
  description
  "Identity for peer-to-peer application.";
}

identity network-management {
  base customer-application;
  description
  "Identity for management application (e.g., Telnet, syslog, SNMP).";
}

identity voice {
  base customer-application;
  description
  "Identity for voice application.";
}

identity video {
  base customer-application;
  description
  "Identity for video conference application.";
}

identity embb {
  base customer-application;
  description
  "Identity for an enhanced Mobile Broadband (eMBB) application. Note that an eMBB application demands network performance with a wide variety of characteristics, such as data rate, latency, loss rate, reliability, and many other parameters.";
}

identity urllc {
  base customer-application;
  description
  "Identity for an Ultra-Reliable and Low Latency Communications (URLLC) application. Note that a URLLC application demands network performance with a wide variety of characteristics, such as latency, reliability, and many other parameters.";
}

identity mmtc {
  base customer-application;
description
"Identity for a massive Machine Type Communications (mMTC) application. Note that an mMTC application demands network performance with a wide variety of characteristics, such as data rate, latency, loss rate, reliability, and many other parameters.";
}
identity site-vpn-flavor {
  description
  "Base identity for the site VPN service flavor."
}
identity site-vpn-flavor-single {
  base site-vpn-flavor;
  description
  "Base identity for the site VPN service flavor. Used when the site belongs to only one VPN."
}
identity site-vpn-flavor-multi {
  base site-vpn-flavor;
  description
  "Base identity for the site VPN service flavor. Used when a logical connection of a site belongs to multiple VPNs."
}
identity site-vpn-flavor-sub {
  base site-vpn-flavor;
  description
  "Base identity for the site VPN service flavor. Used when a site has multiple logical connections. Each connection may belong to different multiple VPNs."
}
identity site-vpn-flavor-nni {
  base site-vpn-flavor;
  description
  "Base identity for the site VPN service flavor. Used to describe an NNI option A connection."
}
identity management {
  description
  "Base identity for site management scheme."
}
identity co-managed {
  base management;
  description
  "Base identity for co-managed site."
}
identity customer-managed {

base management;
description
"Base identity for customer-managed site.";
}
identity provider-managed {
base management;
description
"Base identity for provider-managed site.";
}
identity address-allocation-type {
  description
  "Base identity for address-allocation-type for PE-CE link.";
}
identity provider-dhcp {
  base address-allocation-type;
description
  "Provider network provides DHCP service to customer.";
}
identity provider-dhcp-relay {
  base address-allocation-type;
description
  "Provider network provides DHCP relay service to customer.";
}
identity provider-dhcp-slaac {
  base address-allocation-type;
description
  "Provider network provides DHCP service to customer, as well as SLAAC.";
}
identity static-address {
  base address-allocation-type;
description
  "Provider-to-customer addressing is static.";
}
identity slaac {
  base address-allocation-type;
description
  "Use IPv6 SLAAC.";
}
identity site-role {
  description
  "Base identity for site type.";
}
identity any-to-any-role {
  base site-role;
description
  "Site in an any-to-any IP VPN.";
}
identity spoke-role {
    base site-role;
    description
    "Spoke site in a Hub-and-Spoke IP VPN.";
}

identity hub-role {
    base site-role;
    description
    "Hub site in a Hub-and-Spoke IP VPN.";
}

identity vpn-topology {
    description
    "Base identity for VPN topology.";
}

identity any-to-any {
    base vpn-topology;
    description
    "Identity for any-to-any VPN topology.";
}

identity hub-spoke {
    base vpn-topology;
    description
    "Identity for Hub-and-Spoke VPN topology.";
}

identity hub-spoke-disjoint {
    base vpn-topology;
    description
    "Identity for Hub-and-Spoke VPN topology where Hubs cannot communicate with each other.";
}

identity multicast-tree-type {
    description
    "Base identity for multicast tree type.";
}

identity ssm-tree-type {
    base multicast-tree-type;
    description
    "Identity for SSM tree type.";
}

identity asm-tree-type {
    base multicast-tree-type;
    description
    "Identity for ASM tree type.";
}

identity bidir-tree-type {
    base multicast-tree-type;
    description
    "Identity for bidirectional tree type.";
identity multicast-rp-discovery-type {
    description
    "Base identity for RP discovery type.";
}

identity auto-rp {
    base multicast-rp-discovery-type;
    description
    "Base identity for Auto-RP discovery type.";
}

identity static-rp {
    base multicast-rp-discovery-type;
    description
    "Base identity for static type.";
}

identity bsr-rp {
    base multicast-rp-discovery-type;
    description
    "Base identity for BSR discovery type.";
}

identity routing-protocol-type {
    description
    "Base identity for routing protocol type.";
}

identity ospf {
    base routing-protocol-type;
    description
    "Identity for OSPF protocol type.";
}

identity bgp {
    base routing-protocol-type;
    description
    "Identity for BGP protocol type.";
}

identity static {
    base routing-protocol-type;
    description
    "Identity for static routing protocol type.";
}

identity rip {
    base routing-protocol-type;
    description
    "Identity for RIP protocol type.";
}

identity vrrp {
    base routing-protocol-type;
    description
    "Identity for VRRP protocol type.";
This is to be used when LANs are directly connected to PE routers.

identity direct {
    base routing-protocol-type;
    description
    "Identity for direct protocol type."
}

identity protocol-type {
    description
    "Base identity for protocol field type."
}

identity tcp {
    base protocol-type;
    description
    "TCP protocol type."
}

identity udp {
    base protocol-type;
    description
    "UDP protocol type."
}

identity icmp {
    base protocol-type;
    description
    "ICMP protocol type."
}

identity icmp6 {
    base protocol-type;
    description
    "ICMPv6 protocol type."
}

identity gre {
    base protocol-type;
    description
    "GRE protocol type."
}

identity ipip {
    base protocol-type;
    description
    "IP-in-IP protocol type."
}

identity hop-by-hop {
    base protocol-type;
    description
    "Hop-by-Hop IPv6 header type."
}
identity routing {
    base protocol-type;
    description
    "Routing IPv6 header type.";
}

identity esp {
    base protocol-type;
    description
    "ESP header type.";
}

identity ah {
    base protocol-type;
    description
    "AH header type.";
}

identity vpn-policy-filter-type {
    description
    "Base identity for VPN Policy filter type.";
}

identity ipv4 {
    base vpn-policy-filter-type;
    description
    "Identity for IPv4 Prefix filter type.";
}

identity ipv6 {
    base vpn-policy-filter-type;
    description
    "Identity for IPv6 Prefix filter type.";
}

identity lan {
    base vpn-policy-filter-type;
    description
    "Identity for LAN Tag filter type.";
}

identity qos-profile-direction {
    description
    "Base identity for QoS profile direction.";
}

identity site-to-wan {
    base qos-profile-direction;
    description
    "Identity for Site-to-WAN direction.";
}

identity wan-to-site {
    base qos-profile-direction;
    description
}
"Identity for WAN-to-Site direction."

identity both {
  base qos-profile-direction;
  description
  "Identity for both WAN-to-Site direction and Site-to-WAN direction."
}

/* Extended Identities */

identity encapsulation-type {
  description
  "Identity for the encapsulation type."
}

identity ethernet {
  base encapsulation-type;
  description
  "Identity for Ethernet type."
}

identity vlan {
  base encapsulation-type;
  description
  "Identity for the VLAN type."
}

identity eth-inf-type {
  description
  "Identity of the Ethernet interface type."
}

identity tagged {
  base eth-inf-type;
  description
  "Identity of the tagged interface type."
}

identity untagged {
  base eth-inf-type;
  description
  "Identity of the untagged interface type."
}

identity lag {
  base eth-inf-type;
  description
"Identity of the LAG interface type."
}

identity tagged-inf-type {
  description
    "Identity for the tagged interface type.";
}

identity priority-tagged {
  base tagged-inf-type;
  description
    "Identity for the priority-tagged interface.";
}

identity qinq {
  base tagged-inf-type;
  description
    "Identity for the QinQ tagged interface.";
}

identity dot1q {
  base tagged-inf-type;
  description
    "Identity for the dot1Q VLAN tagged interface.";
}

identity qinany {
  base tagged-inf-type;
  description
    "Identity for the QinAny tagged interface.";
}

identity vxlan {
  base tagged-inf-type;
  description
    "Identity for the VXLAN tagged interface.";
}

identity tag-type {
  description
    "Base identity from which all tag types are derived.";
}

identity c-vlan {
  base tag-type;
  description
    "A CVLAN tag, normally using the 0x8100 Ethertype.";
}

identity s-vlan {
    base tag-type;
    description
    "An SVLAN tag.";
}

identity c-s-vlan {
    base tag-type;
    description
    "Using both a CVLAN tag and an SVLAN tag.";
}

identity vxlan-peer-mode {
    description
    "Base identity for the VXLAN peer mode.";
}

identity static-mode {
    base vxlan-peer-mode;
    description
    "Identity for VXLAN access in the static mode.";
}

identity bgp-mode {
    base vxlan-peer-mode;
    description
    "Identity for VXLAN access by BGP EVPN learning.";
}

identity bw-direction {
    description
    "Identity for the bandwidth direction.";
}

identity input-bw {
    base bw-direction;
    description
    "Identity for the input bandwidth.";
}

identity output-bw {
    base bw-direction;
    description
    "Identity for the output bandwidth.";
}

identity bw-type {
    description
"Identity of the bandwidth type."
}

identity bw-per-cos {
    base bw-type;
    description
        "Bandwidth is per CoS.";
}

identity bw-per-port {
    base bw-type;
    description
        "Bandwidth is per site network access.";
}

identity bw-per-site {
    base bw-type;
    description
        "Bandwidth is per site. It is applicable to all the site network accesses within the site.";
}

identity bw-per-svc {
    base bw-type;
    description
        "Bandwidth is per VPN service.";
}

/* Groupings */
grouping vpn-service-cloud-access {
    container cloud-accesses {
        if-feature cloud-access;
        list cloud-access {
            key cloud-identifier;
            leaf cloud-identifier {
                type leafref {
                    path "/l3vpn-ntw/vpn-profiles/valid-provider-identifiers/cloud-identifier/id";
                }
                description
                    "Identification of cloud service. Local administration meaning.";
            }
            choice list-flavor {
                case permit-any {
                    leaf permit-any {
                        type empty;
                        description
                            "permit-any flavor.
                            Local administration meaning.";
                    }
                }
            }
        } // cloud-access
    } // cloud-accesses
} // vpn-service-cloud-access
"Allows all sites.");
}

case deny-any-except {
  leaf-list permit-site {
    type leafref {
      path "/l3vpn-ntw/sites/site/site-id";
    }
    description "Site ID to be authorized.");
  }
}

case permit-any-except {
  leaf-list deny-site {
    type leafref {
      path "/l3vpn-ntw/sites/site/site-id";
    }
    description "Site ID to be denied.");
  }
}

description "Choice for cloud access policy. By default, all sites in the IP VPN MUST be authorized to access the cloud.";
}

container address-translation {
  container nat44 {
    leaf enabled {
      type boolean;
      default false;
      description "Controls whether or not Network address translation from IPv4 to IPv4 (NAT44) [RFC3022] is required.";
    }
    leaf nat44-customer-address {
      type inet:ipv4-address;
      description "Address to be used for network address translation from IPv4 to IPv4. This is to be used if the customer is providing the IPv4 address. If the customer address is not set, the model assumes that the provider will allocate the address.";
    }
    description "IPv4-to-IPv4 translation.";
  }
}
description
"Container for NAT.";
)
description
"Cloud access configuration.";
)
description
"Container for cloud access configurations.";
)
description
"Grouping for VPN cloud definition.";
)
grouping multicast-rp-group-cfg {
  choice group-format {
    mandatory true;
    case singleaddress {
      leaf group-address {
        type inet:ip-address;
        description
        "A single multicast group address.";
      }
    }
    case startend {
      leaf group-start {
        type inet:ip-address;
        description
        "The first multicast group address in
        the multicast group address range.";
      }
      leaf group-end {
        type inet:ip-address;
        description
        "The last multicast group address in
        the multicast group address range.";
      }
    }
  }
  description
  "Choice for multicast group format.";
}
description
"This grouping defines multicast group or
multicast groups for RP-to-group mapping.";
)
grouping vpn-service-multicast {
  container multicast {
    if-feature multicast;
    leaf enabled {

type boolean;
default false;
description
"Enables multicast.";
}
container customer-tree-flavors {
leaf-list tree-flavor {
type identityref {
    base multicast-tree-type;
}
description
"Type of tree to be used.";
}
description
"Type of trees used by customer.";
}
container rp {
container rp-group-mappings {
list rp-group-mapping {
    key id;
    leaf id {
        type uint16;
        description
        "Unique identifier for the mapping.";
    }
    container provider-managed {
        leaf enabled {
            type boolean;
            default false;
            description
            "Set to true if the Rendezvous Point (RP) must be a provider-managed node. Set to false if it is a customer-managed node.";
        }
        leaf rp-redundancy {
            type boolean;
            default false;
            description
            "If true, a redundancy mechanism for the RP is required.";
        }
        leaf optimal-traffic-delivery {
            type boolean;
            default false;
            description
            "If true, the SP must ensure that traffic uses an optimal path. An SP may use Anycast RP or RP-tree-to-SPT switchover";
        }
    }
}

leaf rp-address {
  when ".../provider-managed/enabled = 'false'" {
    description
    "Relevant when the RP is not provider-managed.";
  }
  type inet:ip-address;
  mandatory true;
  description
  "Defines the address of the RP. Used if the RP is customer-managed.";
}
container groups {
  list group {
    key id;
    leaf id {
      type uint16;
      description
      "Identifier for the group.";
    }
    uses multicast-rp-group-cfg;
    description
    "List of multicast groups.";
  }
  description
  "Multicast groups associated with the RP.";
}
container rp-discovery {
  leaf rp-discovery-type {
    type identityref {
      base multicast-rp-discovery-type;
    }
    default static-rp;
    description
    "Type of RP discovery used.";
  }
  container bsr-candidates {
    when "derived-from-or-self("../.rp-discovery-type", "+
"l3vpn-ntw:bsr-rp" {
  description
  "Only applicable if discovery type
  is BSR-RP.";
} leaf-list bsr-candidate-address {
  type inet:ip-address;
  description
  "Address of BSR candidate.";
} description
  "Container for List of Customer
  BSR candidate’s addresses.";
} description
  "RP discovery parameters.";
} description
  "RP parameters.";
} description
  "Multicast global parameters for the VPN service.";
} description
  "Grouping for multicast VPN definition.";
} grouping vpn-service-mpls {
  leaf carrierscarrier {
    if-feature carrierscarrier;
    type boolean;
    default false;
    description
    "The VPN is using CsC, and so MPLS is required.";
  } description
    "Grouping for MPLS CsC definition.";
} grouping customer-location-info {
  container locations {
    list location {
      key location-id;
      leaf location-id {
        type svc-id;
        description
        "Identifier for a particular location.";
      } leaf address {
        type string;
      }
    }
  }
description
"Address (number and street) of the site.";
}
leaf postal-code {
  type string;
  description
  "Postal code of the site.";
}
leaf state {
  type string;
  description
  "State of the site. This leaf can also be
  used to describe a region for a country that
  does not have states.";
}
leaf city {
  type string;
  description
  "City of the site.";
}
leaf country-code {
  type string {
    pattern '^[A-Z]{2}$';
  }
  description
  "Country of the site.
  Expressed as ISO ALPHA-2 code.";
}
description
"Location of the site.";
}
description
"List of locations for the site.";
}
description
"This grouping defines customer location parameters.";
}
grouping site-group {
  container groups {
    list group {
      key group-id;
      leaf group-id {
        type string;
        description
        "Group-id the site belongs to.";
      }
      description
      "List of group-ids.";
    }
  }
}


} description
"Groups the site or site-network-access belongs to.";
} description
"Grouping definition to assign
group-ids to site or site-network-access.";
} grouping site-diversity {
  container site-diversity {
    if-feature site-diversity;
    uses site-group;
    description
    "Diversity constraint type. All
    site-network-accesses will inherit
    the group values defined here.";
  } description
  "This grouping defines site
diversity parameters.";
} grouping access-diversity {
  container access-diversity {
    if-feature site-diversity;
    uses site-group;
    container constraints {
      list constraint {
        key constraint-type;
        leaf constraint-type {
          type identityref {
            base placement-diversity;
          }
          description
          "Diversity constraint type.";
        }
      }
      container target {
        choice target-flavor {
          default id;
          case id {
            list group {
              key group-id;
              leaf group-id {
                type string;
                description
                "The constraint will be applied against
                this particular group-id for this site
                network access level.";
              }
            }
          }
        }
description
"List of group-ids associated with one specific
constraint for this site network access level.";
}
)
case all-accesses {
leaf all-other-accesses {
  type empty;
  description
  "The constraint will be applied against
  all other site network accesses of this site.";
}
)
case all-groups {
leaf all-other-groups {
  type empty;
  description
  "The constraint will be applied against
  all other groups managed by the customer.";
}
}

description
"Choice for the target flavor definition.";
)
description
"The constraint will be applied against a
Specific target, and the target can be a list
of group-ids, all other site network accesses of
this site, or all other groups managed by the
customer.";
}

description
"List of constraints.";
)
description
"Placement constraints for this site network access.";
}
description
"Diversity parameters.";
)
description
"This grouping defines access diversity parameters.";
)
grouping operational-requirements {
  leaf requested-site-start {
    type yang:date-and-time;
    description
    "Optional leaf indicating requested date and
leaf requested-site-stop {
  type yang:date-and-time;
  description "Optional leaf indicating requested date and time when the service at a particular site is expected to stop.";
}

description "This grouping defines some operational parameters.";
}

grouping operational-requirements-ops {
  leaf actual-site-start {
    type yang:date-and-time;
    config false;
    description "Optional leaf indicating actual date and time when the service at a particular site actually started.";
  }
  leaf actual-site-stop {
    type yang:date-and-time;
    config false;
    description "Optional leaf indicating actual date and time when the service at a particular site actually stopped.";
  }
}
description "This grouping defines some operational parameters.";
}

grouping flow-definition {
  container match-flow {
    leaf dscp {
      type inet:dscp;
      description "DSCP value.";
    }
    leaf dot1p {
      type uint8 {
        range "0..7";
      }
      description
    }
  }
}

"802.1p matching."
}
leaf ipv4-src-prefix {
  type inet:ipv4-prefix;
  description
  "Match on IPv4 src address.";
}
leaf ipv6-src-prefix {
  type inet:ipv6-prefix;
  description
  "Match on IPv6 src address.";
}
leaf ipv4-dst-prefix {
  type inet:ipv4-prefix;
  description
  "Match on IPv4 dst address.";
}
leaf ipv6-dst-prefix {
  type inet:ipv6-prefix;
  description
  "Match on IPv6 dst address.";
}
leaf l4-src-port {
  type inet:port-number;
  must "current() < ../l4-src-port-range/lower-port or current() > ../l4-src-port-range/upper-port" {
    description
    "If l4-src-port and l4-src-port-range/lower-port and upper-port are set at the same time, l4-src-port
    should not overlap with l4-src-port-range.";
  }
  description
  "Match on Layer 4 src port.";
}
leaf-list target-sites {
  if-feature target-sites;
  type svc-id;
  description
  "Identify a site as traffic destination.";
}
container l4-src-port-range {
  leaf lower-port {
    type inet:port-number;
    description
    "Lower boundary for port.";
  }
  leaf upper-port {
    type inet:port-number;
  }
must ". >= ../lower-port" {
  description
  "Upper boundary for port. If it exists, the upper boundary must be higher than the lower boundary.";
} description
"Upper boundary for port."
} description
"Match on Layer 4 src port range. When only the lower-port is present, it represents a single port. When both the lower-port and upper-port are specified, it implies a range inclusive of both values."
} leaf l4-dst-port {
type inet:port-number;
  must "current() < ../l4-dst-port-range/lower-port or " + "current() > ../l4-dst-port-range/upper-port" {
    description
    "If l4-dst-port and l4-dst-port-range/lower-port and upper-port are set at the same time, l4-dst-port should not overlap with l4-src-port-range."
  } description
  "Match on Layer 4 dst port."
} container l4-dst-port-range {
  leaf lower-port {
    type inet:port-number;
    description
    "Lower boundary for port."
  }
  leaf upper-port {
    type inet:port-number;
    must ". >= ../lower-port" {
      description
      "Upper boundary must be higher than lower boundary."
    } description
    "Upper boundary for port. If it exists, upper boundary must be higher than lower boundary."
  } description
"Match on Layer 4 dst port range. When only the lower-port is present, it represents a single port. When both the lower-port and upper-port are specified, it implies a range inclusive of both values.";
}
"Match on Layer 4 dst port range. When only lower-port is present, it represents a single port. When both lower-port and upper-port are specified, it implies a range inclusive of both values."

leaf protocol-field {
    type union {
        type uint8;
        type identityref {
            base protocol-type;
        }
    }
    description
    "Match on IPv4 protocol or IPv6 Next Header field."
}

description
"Describes flow-matching criteria."

description
"Flow definition based on criteria."

grouping site-service-basic {
    leaf svc-input-bandwidth {
        type uint64;
        units bps;
        mandatory true;
        description
        "From the customer site’s perspective, the service input bandwidth of the connection or download bandwidth from the SP to the site."
    }
    leaf svc-output-bandwidth {
        type uint64;
        units bps;
        mandatory true;
        description
        "From the customer site’s perspective, the service output bandwidth of the connection or upload bandwidth from the site to the SP."
    }
    leaf svc-mtu {
        type uint16;
        units bytes;
        mandatory true;
        description
        "MTU at service level. If the service is IP, it refers to the IP MTU. If CsC is enabled,
the requested 'svc-mtu' leaf will refer to the
MPLS MTU and not to the IP MTU.
}
description
"Defines basic service parameters for a site."
}
grouping site-protection {
  container traffic-protection {
    if-feature fast-reroute;
    leaf enabled {
      type boolean;
      default false;
      description
      "Enables traffic protection of access link.";
    }
    description
    "Fast Reroute service parameters for the site."
  }
  description
  "Defines protection service parameters for a site."
}
grouping site-service-mpls {
  container carrierscarrier {
    if-feature carrierscarrier;
    leaf signalling-type {
      type enumeration {
        enum ldp {
          description
          "Use LDP as the signalling protocol
          between the PE and the CE. In this case,
          an IGP routing protocol must also be activated.";
        }
        enum bgp {
          description
          "Use BGP (as per RFC 8277) as the signalling protocol
          between the PE and the CE.
          In this case, BGP must also be configured as
          the routing protocol.";
        }
      }
      default bgp;
      description
      "MPLS signalling type.";
    }
    description
    "This container is used when the customer provides
    MPLS-based services. This is only used in the case
    of CsC (i.e., a customer builds an MPLS service using
an IP VPN to carry its traffic)."
}
description
"Defines MPLS service parameters for a site.";
grouping site-service-qos-profile {
container qos {
  if-feature qos;
  container qos-classification-policy {
    list rule {
      key id;
      ordered-by user;
      leaf id {
        type string;
        description
        "A description identifying the
        qos-classification-policy rule.";
      }
      choice match-type {
        default match-flow;
        case match-flow {
          uses flow-definition;
        }
        case match-application {
          leaf match-application {
            type identityref {
              base customer-application;
            }
            description
            "Defines the application to match.";
          }
        }
        description
        "Choice for classification.";
      }
      leaf target-class-id {
        type string;
        description
        "Identification of the class of service.
        This identifier is internal to the administration.";
      }
      description
      "List of marking rules.";
      description
      "Configuration of the traffic classification policy.";
    }
  }
}
container qos-profile {

choice qos-profile {
  description "Choice for QoS profile.
  Can be standard profile or customized profile."
  case standard {
    description "Standard QoS profile."
    leaf profile {
      type leafref {
        path "/l3vpn-ntw/vpn-profiles/valid-provider-identifiers" +
        "/qos-profile-identifier/id"
      }
      description "QoS profile to be used."
    }
  }
  case custom {
    description "Customized QoS profile."
    container classes {
      if-feature qos-custom;
      list class {
        key class-id;
        leaf class-id {
          type string;
          description "Identification of the class of service.
            This identifier is internal to the
            administration."
        }
        leaf direction {
          type identityref {
            base qos-profile-direction;
          }".default both;
          description "The direction to which the QoS profile
            is applied."
        }
        leaf rate-limit {
          type decimal64 {
            fraction-digits 5;
            range "0..100";
          }
          units percent;
          description "To be used if the class must be rate-limited.
            Expressed as percentage of the service"
bandwidth.
}

container latency {
  choice flavor {
    case lowest {
      leaf use-lowest-latency {
        type empty;
        description
          "The traffic class should use the path with the
          lowest latency.";
      }
    }
    case boundary {
      leaf latency-boundary {
        type uint16;
        units msec;
        default 400;
        description
          "The traffic class should use a path with a
          defined maximum latency.";
      }
    }
  }
  description
    "Latency constraint on the traffic class.";
}

container jitter {
  choice flavor {
    case lowest {
      leaf use-lowest-jitter {
        type empty;
        description
          "The traffic class should use the path with the
          lowest jitter.";
      }
    }
    case boundary {
      leaf latency-boundary {
        type uint32;
        units usec;
        default 40000;
        description
          "The traffic class should use a path with a
          defined maximum jitter.";
      }
    }
  }
  description
    "Latency constraint on the traffic class.";
}
description
"Jitter constraint on the traffic class."
} description
"Jitter constraint on the traffic class."
} container bandwidth {
  leaf guaranteed-bw-percent {
    type decimal64 {
      fraction-digits 5;
      range "0..100";
    } units percent;
    mandatory true;
    description
    "To be used to define the guaranteed bandwidth as a percentage of the available service bandwidth.";
  } leaf end-to-end {
    type empty;
    description
    "Used if the bandwidth reservation must be done on the MPLS network too.";
  } description
  "Bandwidth constraint on the traffic class.";
} description
  "List of classes of services.";
} description
  "Container for list of classes of services.";
} }
} description
  "QoS profile configuration.";
} description
  "QoS configuration.";
} description
  "This grouping defines QoS parameters for a site.";
}
grouping site-security-authentication {
  container authentication {
    description

"Authentication parameters."
}
description
"This grouping defines authentication parameters for a site.";
}
grouping site-security-encryption {
  container encryption {
    if-feature encryption;
    leaf enabled {
      type boolean;
      default false;
      description
        "If true, traffic encryption on the connection is required.";
    }
    leaf layer {
      when "./.enabled = 'true'" {
        description
          "Require a value for layer when enabled is true.";
      }
      type enumeration {
        enum layer2 {
          description
            "Encryption will occur at Layer 2.";
        }
        enum layer3 {
          description
            "Encryption will occur at Layer 3. For example, IPsec may be used when
              a customer requests Layer 3 encryption.";
        }
      }
      description
        "Layer on which encryption is applied.";
    }
  }
  container encryption-profile {
    choice profile {
      case provider-profile {
        leaf profile-name {
          type leafref {
            path "l3vpn-ntw/vpn-profiles/valid-provider-identifiers"+
              "/encryption-profile-identifier/id";
          }
          description
            "Name of the SP profile to be applied.";
        }
      }
      case customer-profile {
        leaf algorithm {

type string;
    description
        "Encryption algorithm to be used.";
}
choice key-type {
    default psk;
    case psk {
        leaf preshared-key {
            type string;
            description
                "Pre-Shared Key (PSK) coming from the customer.";
        }
    }
    description
        "Type of keys to be used.";
}
description
    "Choice of encryption profile. The encryption profile can be the provider profile or customer profile.";
} description
    "Profile of encryption to be applied.";
} description
    "Encryption parameters.";
} description
    "This grouping defines encryption parameters for a site.";
} grouping site-attachment-bearer {
    container bearer {
        container requested-type {
            if-feature requested-type;
            leaf requested-type {
                type string;
                description
                    "Type of requested bearer: Ethernet, DSL, Wireless, etc. Operator specific.";
            }
            leaf strict {
                type boolean;
                default false;
                description
                    "Defines whether requested-type is a preference or a strict requirement.";
            }
        }
    }
}
"Container for requested-type."
}
leaf always-on {
  if-feature always-on;
  type boolean;
  default true;
  description
    "Request for an always-on access type.
    For example, this could mean no dial access type.";
}

/* TODO: to be modified */
leaf bearer-reference {
  if-feature bearer-reference;
  type string;
  description
    "This is an internal reference for the SP.";
}

/* TODO: to be modified */

/* TODO: Verify the path ../site-network-access-type */

/* TODO: Verify the path ../site-network-access-type */

description
  "Bearer-specific parameters.
  To be augmented.";

uses ethernet-params;

uses pseudowire-params {
  when ".../site-network-access-type='pseudowire'" {
    description
      "Parameters associated to a pseudowire
      site-network-access";
  }
}

description
  "Defines physical properties of a site attachment.";
}

grouping site-routing {
  container routing-protocols {
    list routing-protocol {
      key type;
      leaf type {
        type identityref {
          base routing-protocol-type;
          };
        description
          "Type of routing protocol.";
        }
    }
  }
}
list routing-profiles {
    key "id";

    leaf id {
        type leafref {
            path "/l3vpn-ntw/vpn-profiles/valid-provider-identifiers"+
                   "/routing-profile-identifier/id";
        }
        description
        "Routing profile to be used.";
    }

    leaf type {
        type ie-type;
        description
        "Import, export or both.";
    }
}

container ospf {
    when "derived-from-or-self(../type, 'l3vpn-ntw:ospf')" {
        description
        "Only applies when protocol is OSPF.";
    }
    if-feature rtg-ospf;
    leaf-list address-family {
        type address-family;
        min-elements "1";
        description
        "If OSPF is used on this site, this node contains a configured value. This node contains at least one address family to be activated.";
    }

    leaf area-address {
        type yang:dotted-quad;
        mandatory true;
        description
        "Area address.";
    }

    leaf metric {
        type uint16;
        default 1;
        description
        "Metric of the PE-CE link. It is used in the routing state calculation and
path selection.
}

/* Extension */

leaf mtu {
  type uint16;
  description "Maximum transmission unit for a given
  OSPF link."
}

uses security-params;

/* End of Extension */

container sham-links {
  if-feature rtg-ospf-sham-link;
  list sham-link {
    key target-site;
    leaf target-site {
      type svc-id;
      description
      "Target site for the sham link connection. The site
      is referred to by its ID."
    }
    leaf metric {
      type uint16;
      default 1;
      description
      "Metric of the sham link. It is used in the routing
      state calculation and path selection. The default
      value is set to 1."
    }
    description
    "Creates a sham link with another site."
  }
  description
  "List of sham links."
}

description
"OSPF-specific configuration."
}

container bgp {
  when "derived-from-or-self(../type, 'l3vpn-ntw:bgp')" {
    description
    "Only applies when protocol is BGP."
  }
}
if-feature rtg-bgp;
leaf autonomous-system {
  type uint32;
  mandatory true;
  description
    "Customer AS number in case the customer
    requests BGP routing."
}
leaf-list address-family {
  type address-family;
  min-elements "1";
  description
    "If BGP is used on this site, this node
    contains a configured value. This node
    contains at least one address family
to be activated."
}
/* Extension */
leaf neighbor {
  type inet:ip-address;
  description
    "IP address of the BGP neighbor."
}
leaf multihop {
  type uint8;
  mandatory false;
  description
    "Describes the number of hops allowed between the
given BGP neighbor and the PE router."
}
uses security-params;
description
  "BGP-specific configuration."
}
container static {
  when "derived-from-or-self(../type, 'l3vpn-ntw:static')" {
    description
      "Only applies when protocol is static. BGP
      activation requires the SP to know the address of
      the customer peer. When BGP is enabled, the 'static-address'
      allocation type for the IP connection MUST be used.";
  }
}
container cascaded-lan-prefixes {
  list ipv4-lan-prefixes {
    if-feature ipv4;
    key "lan next-hop";
    leaf lan {
      type inet:ipv4-prefix;
      description
      "LAN prefixes.";
    }
    leaf lan-tag {
      type string;
      description
      "Internal tag to be used in VPN policies.";
    }
    leaf next-hop {
      type inet:ipv4-address;
      description
      "Next-hop address to use on the customer side.";
    }
  }
  description
  "List of LAN prefixes for the site.";
}
list ipv6-lan-prefixes {
  if-feature ipv6;
  key "lan next-hop";
  leaf lan {
    type inet:ipv6-prefix;
    description
    "LAN prefixes.";
  }
  leaf lan-tag {
    type string;
    description
    "Internal tag to be used in VPN policies.";
  }
  leaf next-hop {
    type inet:ipv6-address;
    description
    "Next-hop address to use on the customer side.";
  }
  description
  "List of LAN prefixes for the site.";
  description
  "LAN prefixes from the customer.";
  description
  "Configuration specific to static routing.";
container rip {
    when "derived-from-or-self(../type, 'l3vpn-ntw:rip')" {
        description
        "Only applies when the protocol is RIP. For IPv4, the model assumes that RIP version 2 is used.";
    }
    if-feature rtg-rip;
    leaf-list address-family {
        type address-family;
        min-elements "1";
        description
        "If RIP is used on this site, this node contains a configured value. This node contains at least one address family to be activated."
    }
    description
    "Configuration specific to RIP routing."
}
} container vrrp {
    when "derived-from-or-self(../type, 'l3vpn-ntw:vrrp')" {
        description
        "Only applies when protocol is VRRP."
    }
    if-feature rtg-vrrp;
    leaf-list address-family {
        type address-family;
        min-elements "1";
        description
        "If VRRP is used on this site, this node contains a configured value. This node contains at least one address family to be activated."
    }
    description
    "Configuration specific to VRRP routing."
}
} description
"List of routing protocols used on the site. This list can be augmented."
} description
"Defines routing protocols."
} description
"Grouping for routing protocols."
} grouping site-attachment-ip-connection {
container ip-connection {
  container ipv4 {
    if-feature ipv4;
    leaf address-allocation-type {
      type identityref {
        base address-allocation-type;
      }
      must "not(derived-from-or-self(current(), 'l3vpn-ntw:slaac') or "+
        "derived-from-or-self(current(), "+
          "'l3vpn-ntw:provider-dhcp-slaac'))" {
        error-message "SLAAC is only applicable to IPv6";
      }
      description
      "Defines how addresses are allocated. If there is no value for the address
       allocation type, then IPv4 is not enabled.";
    }
    container provider-dhcp {
      when "derived-from-or-self(../address-allocation-type, "+
        "'l3vpn-ntw:provider-dhcp')" {
        description
        "Only applies when addresses are allocated by DHCP.";
    }
    leaf provider-address {
      type inet:ipv4-address;
      description
      "Address of provider side. If provider-address is not
       specified, then prefix length should not be specified either. It also implies
       provider-dhcp allocation is not enabled. If provider-address is specified, then
       the prefix length may or may not be specified.";
    }
    leaf prefix-length {
      type uint8 {
        range "0..32";
        must "(.\provider-address)" {
          error-message
          "If the prefix length is specified, provider-address
           must also be specified.";
          description
          "If the prefix length is specified, provider-address
           must also be specified.";
        }
      }
      description
      "Subnet prefix length expressed in bits. If not specified, or specified as zero,
       this means the customer leaves the actual
prefix length value to the provider."
}
choice address-assign {
  default number;
  case number {
    leaf number-of-dynamic-address {
      type uint16;
      default 1;
      description
        "Describes the number of IP addresses the customer requires.";
    }
  }
  case explicit {
    container customer-addresses {
      list address-group {
        key "group-id";
        leaf group-id {
          type string;
          description
            "Group-id for the address range from start-address to end-address.";
        }
        leaf start-address {
          type inet:ipv4-address;
          description
            "First address.";
        }
        leaf end-address {
          type inet:ipv4-address;
          description
            "Last address.";
        }
        description
          "Describes IP addresses allocated by DHCP. When only start-address or only end-address
          is present, it represents a single address. When both start-address and end-address are specified, it implies a range inclusive of both addresses. If no address is specified, it implies customer addresses group is not supported.";
      }
      description
        "Container for customer addresses is allocated by DHCP.";
    }
    description
      "Choice for the way to assign addresses.";
  }
}
DHCP allocated addresses related parameters.

container dhcp-relay {
  when "derived-from-or-self(../address-allocation-type, "+"  
  
  'l3vpn-ntw:provider-dhcp-relay')" {
    description
    "Only applies when provider is required to implement
    DHCP relay function.";
  }
  leaf provider-address {
    type inet:ipv4-address;
    description
    "Address of provider side. If provider-address is not
    specified, then prefix length should not be specified
    either. It also implies provider-dhcp allocation is
    not enabled. If provider-address is specified, then
    prefix length may or may not be specified.";
  }
  leaf prefix-length {
    type uint8 {
      range "0..32";
    }
    must "(.//provider-address)" {
      error-message
      "If prefix length is specified, provider-address
      must also be specified.";
      description
      "If prefix length is specified, provider-address
      must also be specified.";
    }
    description
    "Subnet prefix length expressed in bits. If not
    specified, or specified as zero, this means the
    customer leaves the actual prefix length value
    to the provider.";
  }
  container customer-dhcp-servers {
    leaf-list server-ip-address {
      type inet:ipv4-address;
      description
      "IP address of customer DHCP server.";
    }
    description
    "Container for list of customer DHCP servers.";
  }
  description
  "DHCP allocated addresses related parameters.";
"DHCP relay provided by operator."
}
}

container addresses {
    when "derived-from-or-self(../address-allocation-type, "+
    ", 'l3vpn-ntw:static-address')" {
        description
        "Only applies when protocol allocation type is static."
    }
    leaf provider-address {
        type inet:ipv4-address;
        description
        "IPv4 Address List of the provider side.
        When the protocol allocation type is static,
        the provider address must be configured."
    }
    leaf customer-address {
        type inet:ipv4-address;
        description
        "IPv4 Address of customer side."
    }
    leaf prefix-length {
        type uint8 {
            range "0..32"
        }
        description
        "Subnet prefix length expressed in bits. It is applied to both provider-address
        and customer-address."
    }
    description
    "Describes IPv4 addresses used."
}

description
"IPv4-specific parameters."
}

container ipv6 {
    if-feature ipv6;
    leaf address-allocation-type {
        type identityref {
            base address-allocation-type;
        }
        description
        "Defines how addresses are allocated.
        If there is no value for the address allocation type, then IPv6 is
        not enabled.";
    }
}
container provider-dhcp {
  when "derived-from-or-self(../address-allocation-type, "+  
  
  
  "'l3vpn-ntw:provider-dhcp') "+  
  
  "or derived-from-or-self(../address-allocation-type, "+  
  
  "'l3vpn-ntw:provider-dhcp-slaac')" {  
    description  
    "Only applies when addresses are allocated by DHCP.";
  }

  leaf provider-address {
    type inet:ipv6-address;
    description  
    "Address of the provider side. If provider-address  
    is not specified, then prefix length should not be  
    specified either. It also implies provider-dhcp  
    allocation is not enabled. If provider-address is  
    specified, then prefix length may or may  
    not be specified.";
  }

  leaf prefix-length {
    type uint8 {
      range "0..128";
    }
    must "(.provider-address)" {  
      error-message  
      "If prefix length is specified, provider-address  
      must also be specified.";
      description  
      "If prefix length is specified, provider-address  
      must also be specified.";
    }
    description  
    "Subnet prefix length expressed in bits. If not  
    specified, or specified as zero, this means the  
    customer leaves the actual prefix length value  
    to the provider.";
  }

  choice address-assign {
    default number;
    case number {
      leaf number-of-dynamic-address {
        type uint16;
        default 1;
        description  
        "Describes the number of IP addresses the customer  
        requires.";
      }
    }
    case explicit {

container customer-addresses {
  list address-group {
    key "group-id";
    leaf group-id {
      type string;
      description
      "Group-id for the address range from start-address to end-address.";
    }
    leaf start-address {
      type inet:ipv6-address;
      description
      "First address.";
    }
    leaf end-address {
      type inet:ipv6-address;
      description
      "Last address.";
    }
    description
    "Describes IP addresses allocated by DHCP. When only start-address or only end-address is present, it represents a single address. When both start-address and end-address are specified, it implies a range inclusive of both addresses. If no address is specified, it implies customer addresses group is not supported.";
  }
  description
  "Container for customer addresses allocated by DHCP.";
}
}
description
"Choice for the way to assign addresses.";
}
description
"DHCP allocated addresses related parameters.";
}
container dhcp-relay {
  when "derived-from-or-self(../address-allocation-type, "+ "l3vpn-ntw:provider-dhcp-relay")" {
    description
    "Only applies when the provider is required to implement DHCP relay function.";
  }
  leaf provider-address {
    type inet:ipv6-address;
    description
  }
}
"Address of the provider side. If provider-address is not specified, then prefix length should not be specified either. It also implies provider-dhcp allocation is not enabled. If provider address is specified, then prefix length may or may not be specified.";
}
leaf prefix-length {
  type uint8 {
    range "0..128";
  }
  must ".(/provider-address)" {
    error-message
    "If prefix length is specified, provider-address must also be specified.";
    description
    "If prefix length is specified, provider-address must also be specified.";
  }
  description
  "Subnet prefix length expressed in bits. If not specified, or specified as zero, this means the customer leaves the actual prefix length value to the provider.";
}
container customer-dhcp-servers {
  leaf-list server-ip-address {
    type inet:ipv6-address;
    description
    "This node contains the IP address of the customer DHCP server. If the DHCP relay function is implemented by the provider, this node contains the configured value.";
  }
  description
  "Container for list of customer DHCP servers.";
}
container addresses {
  when "derived-from-or-self(.address-allocation-type, "+ "l3vpn-ntw:static-address")" {
    description
    "Only applies when protocol allocation type is static.";
  }
  leaf provider-address {
type inet:ipv6-address;
description
"IPv6 Address of the provider side. When the protocol
allocation type is static, the provider address
must be configured."
}
leaf customer-address {
  type inet:ipv6-address;
  description
  "The IPv6 Address of the customer side."
}
leaf prefix-length {
  type uint8 {
    range "0..128";
  }
  description
  "Subnet prefix length expressed in bits. It is applied to both provider-address and
customer-address."
}
description
"Describes IPv6 addresses used.";
}
description
"IPv6-specific parameters.";
}
container oam {
  container bfd {
    if-feature bfd;
    leaf enabled {
      type boolean;
      default false;
      description
      "If true, BFD activation is required."
    }
    choice holdtime {
      default fixed;
      case fixed {
        leaf fixed-value {
          type uint32;
          units msec;
          description
          "Expected BFD holdtime expressed in msec. The customer
          may impose some fixed values for the holdtime period
          if the provider allows the customer use this function.
          If the provider doesn't allow the customer to use this
          function, the fixed-value will not be set."
        }
      }
    }
  }
}
case profile {
  leaf profile-name {
    type leafref {
      path "/l3vpn-ntw/vpn-profiles/valid-provider-identifiers/"+
        "bfd-profile-identifier/id";
    }
    description "Well-known SP profile name. The provider can propose
      some profiles to the customer, depending on the service
      level the customer wants to achieve. Profile names
      must be communicated to the customer."
  }
  description "Well-known SP profile."
}

description "Choice for holdtime flavor."

description "Container for BFD."

description "Defines the Operations, Administration, and Maintenance (OAM)
mechanisms used on the connection. BFD is set as a fault
detection mechanism, but the 'oam' container can easily
be augmented by other mechanisms"

description "Defines connection parameters."

description "This grouping defines IP connection parameters."

grouping site-service-multicast {
  container multicast {
    if-feature multicast;
    leaf multicast-site-type {
      type enumeration {
        enum receiver-only {
          description "The site only has receivers.";
        }
        enum source-only {
          description "The site only has sources.";
        }
        enum source-receiver {

description
  "The site has both sources and receivers."
}
}
default source-receiver;
description
  "Type of multicast site."
}
container multicast-address-family {
  leaf ipv4 {
    if-feature ipv4;
    type boolean;
    default false;
    description
      "Enables IPv4 multicast."
  }
  leaf ipv6 {
    if-feature ipv6;
    type boolean;
    default false;
    description
      "Enables IPv6 multicast."
  }
  description
    "Defines protocol to carry multicast."
}
leaf protocol-type {
  type enumeration {
    enum host {
      description
        "Hosts are directly connected to the provider network.
        Host protocols such as IGMP or MLD are required."
    }
    enum router {
      description
        "Hosts are behind a customer router.
        PIM will be implemented."
    }
    enum both {
      description
        "Some hosts are behind a customer router, and
        some others are directly connected to the
        provider network. Both host and routing protocols
        must be used. Typically, IGMP and PIM will be
        implemented."
    }
  }
  default "both";
description
"Multicast protocol type to be used with the customer site."
}
description
"Multicast parameters for the site."
}
description
"Multicast parameters for the site."
}
grouping site-management {
container management {
leaf type {
  type identityref {
    base management;
  }
  mandatory true;
  description
    "Management type of the connection."
}
}
}

description
"Management configuration."
}

description
"Management parameters for the site."
}

grouping site-devices {
container devices {
when "derived-from-or-self(./management/type, "+
"'l3vpn-ntw:provider-managed') or "+
"derived-from-or-self(./management/type, 'l3vpn-ntw:co-managed')" {

description
"Applicable only for provider-managed or
co-managed device."
}
}
list device {
  key device-id;
  leaf device-id {
    type svc-id;
    description
      "Identifier for the device."
  }
}
leaf location {
  type leafref {
    path "../.../locations/"+
"location/location-id"
  }
  mandatory true;
  description
"
"Location of the device."
}

container management {
when "derived-from-or-self(../../../management/type,"+
"/l3vpn-ntw:co-managed")" {
  description
  "Applicable only for co-managed device."
}

leaf address-family {
  type address-family;
  description
  "Address family used for management."
}

leaf address {
  when "(../../../address-family)"
  description
  "If address-family is specified, then address should also be specified. If address-family is not specified, then address should also not be specified."
  type inet:ip-address;
  mandatory true;
  description
  "Management address."
}

description
"Management configuration. Applicable only for co-managed device."
}

description
"List of devices requested by customer."
}

description
"Device configuration."
}

description
"Grouping for device allocation."
}

grouping site-vpn-flavor {
leaf site-vpn-flavor {
  type identityref {
    base site-vpn-flavor;
  }
  default site-vpn-flavor-single;
  description
  "Defines the way the VPN multiplexing is done, e.g., whether the site belongs to a single VPN site or a multiVPN; or, in the case of a multiVPN, whether the logical accesses of the sites belong"
to the same set of VPNs or each logical access maps to
different VPNs.";
}
description
"Grouping for site VPN flavor.";
}
grouping site-vpn-policy {
  container vpn-policies {
    list vpn-policy {
      key vpn-policy-id;
      leaf vpn-policy-id {
        type svc-id;
        description
        "Unique identifier for the VPN policy.";
      }
      list entries {
        key id;
        leaf id {
          type svc-id;
          description
          "Unique identifier for the policy entry.";
        }
        container filters {
          list filter {
            key type;
            ordered-by user;
            leaf type {
              type identityref {
                base vpn-policy-filter-type;
              }
              description
              "Type of VPN Policy filter.";
            }
            leaf-list lan-tag {
              when "derived-from-or-self(.\../type, 'l3vpn-ntw:lan')" {
                description
                "Only applies when the VPN Policy filter is a
LAN Tag filter.";
              }
              if-feature lan-tag;
              type string;
              description
              "List of 'lan-tag' items to be matched. LAN Tag
is an Internal tag to be used in VPN policies ";
            }
            leaf-list ipv4-lan-prefix {
              when "derived-from-or-self(.\../type, 'l3vpn-ntw:ipv4')" {
                description
                "List of 'lan-tag' items to be matched. LAN Tag
is an Internal tag to be used in VPN policies ";
              }
            }
          }
        }
      }
    }
  }
}
"Only applies when VPN Policy filter is IPv4 Prefix filter.";
} if-feature ipv4;
type inet:ipv4-prefix;
description
"List of IPv4 prefixes as LAN Prefixes to be matched.";
} leaf-list ipv4-lan-prefix {
when "derived-from-or-self(../type, '13vpn-ntw:ipv4')" {
  description
  "Only applies when VPN Policy filter is IPv6 Prefix filter.";
} if-feature ipv6;
type inet:ipv6-prefix;
description
"List of IPv6 prefixes as LAN prefixes to be matched.";
} description
"List of filters used on the site. This list can
be augmented.";
}

description
"If a more-granular VPN attachment is necessary, filtering can
be used. If used, it permits the splitting of site LANs among
multiple VPNs. The Site LAN can be split based on either LAN
Tag or LAN prefix. If no filter is used, all the LANs will be
part of the same VPNs with the same role.";
}
list vpn {
key vpn-id;
leaf vpn-id {
  type leafref {
    path "/l3vpn-ntw/vpn-services/"+
    "vpn-service/vpn-id";
  }
  mandatory true;
  description
  "Reference to an IP VPN.";
}
leaf site-role {
  type identityref {
    base site-role;
  }
  default any-to-any-role;
  description
  "Role of the site in the IP VPN.";
} description
"List of VPNs the LAN is associated with.";
}
description
"List of entries for export policy.";
}
description
"List of VPN policies.";
}
description
"VPN policy.";
}
description
"VPN policy parameters for the site.";
}
grouping site-maximum-routes {
  container maximum-routes {
    list address-family {
      key af;
      leaf af {
        type address-family;
        description
        "Address family.";
      }
      leaf maximum-routes {
        type uint32;
        description
        "Maximum prefixes the VRF can accept
         for this address family.";
      }
      description
      "List of address families.";
    }
    description
    "Defines ‘maximum-routes’ for the VRF.";
  }
  description
  "Defines ‘maximum-routes’ for the site.";
}
grouping site-security {
  container security {
    uses site-security-authentication;
    uses site-security-encryption;
    description
    "Site-specific security parameters.";
  }
  description
  "Grouping for security parameters.";
}
grouping site-service {
  container service {
    uses site-service-qos-profile;
    uses site-service-mpls;
    uses site-service-multicast;
    description
      "Service parameters on the attachment.";
  }
  description
    "Grouping for service parameters.";
}
grouping site-network-access-service {
  container service {
    uses site-service-basic;
    uses site-service-qos-profile;
    uses site-service-mpls;
    uses site-service-multicast;
    description
      "Service parameters on the attachment.";
  }
  description
    "Grouping for service parameters.";
}
grouping vpn-extranet {
  container extranet-vpns {
    if-feature extranet-vpn;
    list extranet-vpn {
      key vpn-id;
      leaf vpn-id {
        type svc-id;
        description
          "Identifies the target VPN the local VPN want to access.";
      }
      leaf local-sites-role {
        type identityref {
          base site-role;
        }
        default any-to-any-role;
        description
          "This describes the role of the local sites in the target VPN topology. In the any-to-any VPN service topology, the local sites must have the same role, which will be 'any-to-any-role'. In the Hub-and-Spoke VPN service topology or the Hub-and-Spoke disjoint VPN service topology, the local sites must have a Hub role or a Spoke role.";
    }
  }
}
"List of extranet VPNs or target VPNs the local VPN is attached to."

"Container for extranet VPN configuration."

"Grouping for extranet VPN configuration. This provides an easy way to interconnect all sites from two VPNs."

"Grouping site-attachment-availability {
  container availability {
    leaf access-priority {
      type uint32;
      default 100;
      description
      "Defines the priority for the access. The higher the access-priority value, the higher the preference of the access will be.";
    }
    description
    "Availability parameters (used for multihoming).";
  }
  description
  "Defines availability parameters for a site."
}

"Grouping access-vpn-policy {
  container vpn-attachment {
    choice attachment-flavor {
      case vpn-policy-id {
        leaf vpn-policy-id {
          type leafref {
            path "../../../vpn-policies/vpn-policy/"+"vpn-policy-id";
          }
          description
          "Reference to a VPN policy. When referencing VPN policy for attachment, the vpn-policy-id must be configured.";
        }
      }
      case vpn-id {
        leaf vpn-id {

type leafref {
  path "/l3vpn-ntw/vpn-services"+
    "/vpn-service/vpn-id";
}

description
"Reference to an IP VPN. Referencing a vpn-id provides
an easy way to attach a particular logical access to
a VPN. In this case, vpn-id must be configured."
}

leaf site-role {
  type identityref {
    base site-role;
  }
  default any-to-any-role;
  description
"Role of the site in the IP VPN. When referencing a vpn-id,
the site-role setting must be added to express the role of
the site in the target VPN service topology."
}

mandatory true;

description
"Choice for VPN attachment flavor. A choice is implemented
to allow the user to choose the flavor that provides the
best fit."

description
"Defines VPN attachment of a site."

description
"Defines the VPN attachment rules for
a site’s logical access."

grouping vpn-profile-cfg {
  container valid-provider-identifiers {
    list cloud-identifier {
      if-feature cloud-access;
      key id;
      leaf id {
        type string;
        description
"Identification of cloud service.
Local administration meaning.";
      }
      description
"List for Cloud Identifiers."
    }
    list encryption-profile-identifier {

key id;
leaf id {
  type string;
  description
  "Identification of the SP encryption profile to be used. Local administration meaning.";
}
description
"List for encryption profile identifiers.";
}
list qos-profile-identifier {
  key id;
  leaf id {
    type string;
    description
    "Identification of the QoS Profile to be used. Local administration meaning.";
  }
description
"List for QoS Profile Identifiers.";
}
list bfd-profile-identifier {
  key id;
  leaf id {
    type string;
    description
    "Identification of the SP BFD Profile to be used. Local administration meaning.";
  }
description
"List for BFD Profile identifiers.";
}
list routing-profile-identifier {
  key id;
  leaf id {
    type string;
    description
    "Identification of the routing Profile to be used by the routing-protocols within sites and site-network-accesses. Local administration meaning.";
  }
description
"List for Routing Profile Identifiers.";
}

nacm:default-deny-write;
description
"Container for Valid Provider Identifies."
}
description
"Grouping for VPN Profile configuration."
}
grouping vpn-svc-cfg {
leaf vpn-id {
  type svc-id;
  description
  "VPN identifier. Local administration meaning."
}
leaf customer-name {
  type string;
  description
  "Name of the customer that actually uses the VPN service. In the case that any intermediary (e.g., Tier-2 provider or partner) sells the VPN service to their end user on behalf of the original service provider (e.g., Tier-1 provider), the original service provider may require the customer name to provide smooth activation/commissioning and operation for the service."
}
leaf vpn-service-topology {
  type identityref {
    base vpn-topology;
  }
  default any-to-any;
  description
  "VPN service topology."
}
leaf description {
  type string;
  description
  "Textual description of a VPN service."
}
uses ie-profiles-params;
uses vpn-nodes-params;
uses vpn-service-cloud-access;
uses vpn-service-multicast;
uses vpn-service-mpls;
uses vpn-extranet;

description
"Grouping for VPN service configuration."
}
grouping site-top-level-cfg {

uses operational-requirements;
uses customer-location-info;
uses site-devices;
uses site-diversity;
uses site-management;
uses site-vmn-policy;
uses site-vmn-flavor;
uses site-maximum-routes;
uses site-security;
uses site-service;
uses site-protection;
uses site-routing;

description
"Grouping for site top-level configuration.";
}
grouping site-network-access-top-level-cfg {

/* Extension */

uses status-params;

/* End of Extension */

leaf site-network-access-type {
  type identityref {
    base site-network-access-type;
  }
  default point-to-point;
  
  description
  "Describes the type of connection, e.g.,
  point-to-point or multipoint.";
}
choice location-flavor {

case location { 
  when "derived-from-or-self(../management/type, "+ 
  "'13vpn-ntw:customer-managed")" { 
    description 
    "Applicable only for customer-managed device.";
  }
  leaf location-reference {
    type leafref {
      path "../locations/location/location-id";
    }
    description 
    "Location of the site-network-access.";
  }
}

  case device {

when "derived-from-or-self(../../management/type, "+ "'l3vpn-ntw:provider-managed') or "+ "derived-from-or-self(../../management/type, "+ "'l3vpn-ntw:co-managed')" {  
  description
  "Applicable only for provider-managed or co-managed device.";
}
leaf device-reference {
  type leafref {
    path "../../../devices/device/device-id";
  }
  description
  "Identifier of CE to use."
}
}
mandatory true;

description
  "Choice of how to describe the site’s location."
} 
uses access-diversity;
uses site-attachment-bearer;
uses site-attachment-ip-connection;
uses site-security;
uses site-network-access-service;
uses site-routing;
uses site-attachment-availability;
/*uses access-vpn-policy;*/

description
  "Grouping for site network access top-level configuration."
}

/* Extensions */

/* Bearers in a site */
grouping site-bearer-params {

description "Container that encloses all the bearers connected to a site. A bearer is mapped one to one to a port on the PE router.";

container site-bearers {
  list bearer {
    key "bearer-id";
      
      leaf bearer-id {
        description "Unique identifier for a bearer. This identifies shall be mapped to the bearer-reference
on a site-network-access.
  type string;
}

leaf ne-id {
  description "Unique identifier for a network
  element. This identifier may be a string, a UUID,
  an IP address, etc.";
  type string;
}

leaf port-id {
  description "Port of the PE router for the given
  bearer.";
  type string;
}

/* UNUSED */
grouping svc-bandwidth-params {
  container svc-bandwidth {
    if-feature "input-bw";
    list bandwidth {
      key "direction type";
      leaf direction {
        type identityref {
          base bw-direction;
        }
        description
        "Indicates the bandwidth direction. It can be
        the bandwidth download direction from the SP to
        the site or the bandwidth upload direction from
        the site to the SP.";
      }
      leaf type {
        type identityref {
          base bw-type;
        }
        description
        "Bandwidth type. By default, the bandwidth type
        is set to 'bw-per-cos'.";
      }
      leaf cos-id {
        when ",derived-from-or-self(../type, " + "'l3vpn-ntw:bw-per-cos')" {
          description
          "";
        }
      }
    }
  }
}
"Relevant when the bandwidth type is set to 'bw-per-cos'."
}

type uint8;

description
"Identifier of the CoS, indicated by DSCP or a
CE-VLAN CoS (802.1p) value in the service frame.
If the bandwidth type is set to 'bw-per-cos',
the CoS ID MUST also be specified."
}

leaf vpn-id {
when "derived-from-or-self(../type, "
+ "l3vpn-ntw:bw-per-svc")" {

description
"Relevant when the bandwidth type is
set as bandwidth per VPN service."
}

type svc-id;

description
"Identifies the target VPN. If the bandwidth
type is set as bandwidth per VPN service, the
vpn-id MUST be specified."
}

leaf cir {

type uint64;

units "bps";

mandatory true;

description
"Committed Information Rate. The maximum number
of bits that a port can receive or send over
an interface in one second."
}

leaf cbs {

type uint64;

units "bps";

mandatory true;

description
"Committed Burst Size (CBS). Controls the bursty
nature of the traffic. Traffic that does not
use the configured Committed Information Rate
(CIR) accumulates credits until the credits
reach the configured CBS."
}

leaf eir {

type uint64;

units "bps";

description
"Excess Information Rate (EIR), i.e., excess frame
delivery allowed that is not subject to an SLA. The traffic rate can be limited by the EIR."

leaf ebs {
    type uint64;
    units "bps";
    description "Excess Burst Size (EBS). The bandwidth available for burst traffic from the EBS is subject to the amount of bandwidth that is accumulated during periods when traffic allocated by the EIR policy is not used.";
}

leaf pir {
    type uint64;
    units "bps";
    description "Peak Information Rate, i.e., maximum frame delivery allowed. It is equal to or less than the sum of the CIR and the EIR.";
}

leaf pbs {
    type uint64;
    units "bps";
    description "Peak Burst Size. It is measured in bytes per second.";
}

description "List of bandwidth values (e.g., per CoS, per vpn-id)."

description "From the customer site’s perspective, the service input/output bandwidth of the connection or download/upload bandwidth from the SP/site to the site/SP."
}
}


grouping status-params {
    container status {
        description "Operational and administrative status for different elements in the model.";
        leaf admin-enabled {
            description "True is the entity is administratively enabled.";
        }
    }
}
/* Parameters related to vpn-nodes (VRF config.) */
grouping vpn-nodes-params {
  description "Grouping to define VRF-specific configuration.";
  container vpn-nodes {
    description "Container that defines VRF-specific configuration.";
    list vpn-node {
      key "vpn-node-id ne-id";
      leaf vpn-node-id {
        description "Identifier of the VPN node. It can be identified or mapped as the VRF name. As it may not be globally unique, the ne-id is also needed.";
        type string;
      }
      leaf description {
        type string;
        description "Textual description of a VPN node.";
      }
      leaf ne-id {
        description "Unique identifier for a network element where to instantiate the VRF. This identifier may be a string, a UUID, an IP address, etc.";
        type string;
      }
      leaf router-id {
        description "In case of being necessary, it defines the IP address to identify the VRF. If not specified, the IP of the loopback interface within the base routing instance will be used.";
        type inet:ipv4-address;
      }
    }
  }
}
leaf autonomous-system {
    type uint32;
    description "AS number of the VRF.";
}

leaf node-role {
    type identityref {
        base site-role;
    }
    default any-to-any-role;
    description "Role of the vpn-node in the IP VPN."
}

uses status-params;

/* Here we use the name given to the existing structure in sites */
uses site-maximum-routes;

leaf node-ie-profile {
    description "Reference to an import export profile defined within a VPN service.";
    type leafref {
        path "/l3vpn-ntw/vpn-services/"+
            "vpn-service/ie-profiles/ie-profile/ie-profile-id";
    }
}

container site-attachments {
    list site-attachment {
        key "site-id";
        description "List of attachments (site-network-accesses) that are connected to the VPN-node (VRF instance).";
        leaf site-id {
            description "Identifier of the site where the site-network-access is located.";
            type leafref{
                path "/l3vpn-ntw/sites/site/site-id";
            }
        }
        leaf-list site-network-access-id {
            type leafref {
                description "Identifier of the site-network-access to be
attached to the VPN node.

path "/l3vpn-ntw/sites/site/site-network-accesses/"+
"site-network-access/site-network-access-id";

/* Parameters related to import and export profiles (RTs RDs.) */
grouping ie-profiles-params {
  description "Grouping to specify rules for route import and export";

  container ie-profiles {
    list ie-profile {
      key "ie-profile-id";

      leaf ie-profile-id {
        type string;
        description
        "Unique identifier for an import/export profile defined
         within a VPN node.";
      }

      leaf rd {
        type rt-types:route-distinguisher;
        description
        "Route distinguisher.";
      }

      container vpn-targets {
        description
        "Set of route-targets to match for import and export routes
to/from VRF";
        uses rt-types:vpn-route-targets;
      }
    }
  }
}

grouping pseudowire-params {
  container pseudowire {

/*leaf far-end {*/
/*  description "IP of the remote peer of the pseudowire.";*/
/*  type inet:ip-address;*/
/*}*/

leaf vcid {
  description "PW or virtual circuit identifier.";
  type uint32;
}

grouping security-params {
  container security {
    description "Container for aggregating any security parameter for routing sessions between a PE and a CE."
    leaf auth-key {
      type string;
      description "MD5 authentication password for the connection towards the customer edge."
    }
  }
}

grouping ethernet-params {
  container connection {
    leaf encapsulation-type {
      type identityref {
        base encapsulation-type;
      }
      default "ethernet";
      description "Encapsulation type. By default, the encapsulation type is set to 'ethernet'."
    }
    leaf eth-inf-type {
      type identityref {
        base eth-inf-type;
      }
      default "untagged";
      description "Ethernet interface type. By default, the Ethernet interface type is set to 'untagged'."
    }
  }
}
container tagged-interface {
    leaf type {
        type identityref {
            base tagged-inf-type;
        } default "priority-tagged";
        description "Tagged interface type. By default, the type of the tagged interface is 'priority-tagged'.";
    }
    container dot1q-vlan-tagged {
        when "derived-from-or-self(../type, " + "l3vpn-ntw:dot1q")" {
            description "Only applies when the type of the tagged interface is 'dot1q'.";
        }
        if-feature "dot1q";
        leaf tg-type {
            type identityref {
                base tag-type;
            } default "c-vlan";
            description "Tag type. By default, the tag type is 'c-vlan'.";
        }
        leaf cvlan-id {
            type uint16;
            mandatory true;
            description "VLAN identifier.";
        }
        description "Tagged interface.";
    }
    container priority-tagged {
        when "derived-from-or-self(../type, " + "l3vpn-ntw:priority-tagged")" {
            description "Only applies when the type of the tagged interface is 'priority-tagged'.";
        }
        leaf tag-type {
            type identityref {
                base tag-type;
            }
        }
    }
}
default "c-vlan";

description
"Tag type. By default, the tag type is 'c-vlan'."

description
"Priority tagged."

} container qinq {
when "derived-from-or-self(.//type, " + "l3vpn-ntw:qinq")" {

description
"Only applies when the type of the tagged interface is 'qinq'."

} if-feature "qinq";

leaf tag-type {

type identityref {

base tag-type;

} default "c-s-vlan";

description
"Tag type. By default, the tag type is 'c-s-vlan'."

} leaf svlan-id {

type uint16;

mandatory true;

description
"SVLAN identifier."

} leaf cvlan-id {

type uint16;

mandatory true;

description
"CVLAN identifier."

} description
"QinQ."

} container qinany {
when "derived-from-or-self(.//type, " + "l3vpn-ntw:qinany")" {

description
"Only applies when the type of the tagged interface is 'qinany'."

}
if-feature "qinany";
leaf tag-type {
    type identityref {
        base tag-type;
    }
    default "s-vlan";
    description
        "Tag type. By default, the tag type is 's-vlan'.";
}
leaf svlan-id {
    type uint16;
    mandatory true;
    description
        "SVLAN ID."
}
description
    "Container for QinAny."
}
container vxlan {
    when "derived-from-or-self(../type, " + "'l3vpn-ntw:vxlan')" {
        description
            "Only applies when the type of the tagged interface is 'vxlan'."
    }
if-feature "vxlan";
leaf vni-id {
    type uint32;
    mandatory true;
    description
        "VXLAN Network Identifier (VNI)."
}
leaf peer-mode {
    type identityref {
        base vxlan-peer-mode;
    }
    default "static-mode";
    description
        "Specifies the VXLAN access mode. By default, the peer mode is set to 'static-mode'."
}
list peer-list {
    key "peer-ip";
    leaf peer-ip {
        type inet:ip-address;
        description
            "Peer IP."
    }
}
/* Main blocks */
container l3vpn-ntw {
    container vpn-profiles {
        uses vpn-profile-cfg;
        description
            "Container for VPN Profiles.";
    }
    container vpn-services {
        list vpn-service {
            key vpn-id;
            uses vpn-svc-cfg;
            description
                "List of VPN services.";
        }
        description
            "Top-level container for the VPN services.";
    }
    container sites {
        list site {
            key site-id;
            leaf site-id {
                type svc-id;
                description
                    "Identifier of the site.";
            }
            leaf description {
                type string;
                description
                    "Textual description of a site.";
            }
            uses site-top-level-cfg;
            uses operational-requirements-ops;
            uses site-bearer-params;
            container site-network-accesses {

list site-network-access {
    key site-network-access-id;
    leaf site-network-access-id {
        type svc-id;
        description "Identifier for the access.";
    }
    leaf description {
        type string;
        description "Textual description of a VPN service.";
    }
    uses site-network-access-top-level-cfg;
    description "List of accesses for a site.";
}

description "List of accesses for a site.";

description "List of sites.";

description "Container for sites.";

description "Main container for L3VPN service configuration.";
}

Figure 4

6.  IANA Considerations

This memo includes no request to IANA.

7.  Security Considerations

All the security considerations of RFC 8299 [RFC8299] apply to this document. Subsequent versions will provide additional security considerations.

8.  Implementation Status

This section will be used to track the status of the implementations of the model. It is aimed at being removed if the document becomes RFC.
9. Acknowledgements

Thanks to Adrian Farrel and Miguel Cros for the suggestions on the document. Lots of thanks for the discussions on opsawg mailing list. Some of the comments will be addressed in next versions.

10. Contributors

Daniel King
Old Dog Consulting
Email: daniel@olddog.co.uk

Samier Barguil
Telefonica
Email: samier.barguilgiraldo.ext@telefonica.com

Luay Jalil
Verizon
Email: luay.jalil@verizon.com

Qin Wu
Huawei
Email: bill.wu@huawei.com

11. References

11.1. Normative References


11.2. Informative References


Authors’ Addresses

Alejandro Aguado
Telefonica
Madrid
ES

Email: alejandro.aguadomartin.ext@telefonica.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
ES

Email: oscar.gonzalezdedios@telefonica.com

Victor Lopez
Telefonica
Madrid
ES

Email: victor.lopezalvarez@telefonica.com

Daniel Voyer
Bell Canada
CA

Email: daniel.voyer@bell.ca

Luis Angel Munoz
Vodafone
ES

Email: luis-angel.munoz@vodafone.com
Changes in the Internet Threat Model
draft-arkko-arch-internet-threat-model-01

Abstract

Communications security has been at the center of many security improvements in the Internet. The goal has been to ensure that communications are protected against outside observers and attackers.

This memo suggests that the existing threat model, while important and still valid, is no longer alone sufficient to cater for the pressing security issues in the Internet. For instance, it is also necessary to protect systems against endpoints that are compromised, malicious, or whose interests simply do not align with the interests of the users. While such protection is difficult, there are some measures that can be taken.

It is particularly important to ensure that as we continue to develop Internet technology, non-communications security related threats are properly understood. While the consideration of these issues is relatively new in the IETF, this memo provides some initial ideas about potential broader threat models to consider when designing protocols for the Internet or when trying to defend against pervasive monitoring. Further down the road, updated threat models could result in changes in RFC 3552 (guidelines for writing security considerations) and RFC 7258 (pervasive monitoring), to include proper consideration of non-communications security threats. It may also be necessary to have dedicated guidance on how systems design and architecture affects security.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as evidence before any Internet Engineering Task Force, IETF, or working group meeting. They are only appropriate when reviewed in a working group at a meeting. They should be treated as work in progress.
Communications security has been at the center of many security improvements in the Internet. The goal has been to ensure that communications are protected against outside observers and attackers. At the IETF, this approach has been formalized in BCP 72 [RFC3552], which defined the Internet threat model in 2003.
The purpose of a threat model is to outline what threats exist in order to assist the protocol designer. But RFC 3552 also ruled some threats to be in scope and of primary interest, and some threats out of scope [RFC3552]:

The Internet environment has a fairly well understood threat model. In general, we assume that the end-systems engaging in a protocol exchange have not themselves been compromised. Protecting against an attack when one of the end-systems has been compromised is extraordinarily difficult. It is, however, possible to design protocols which minimize the extent of the damage done under these circumstances.

By contrast, we assume that the attacker has nearly complete control of the communications channel over which the end-systems communicate. This means that the attacker can read any PDU (Protocol Data Unit) on the network and undetectably remove, change, or inject forged packets onto the wire.

However, the communications-security-only threat model is becoming outdated. This is due to three factors:

- Advances in protecting most of our communications with strong cryptographic means. This has resulted in much improved communications security, but also highlights the need for addressing other, remaining issues. This is not to say that communications security is not important, it still is: improvements are still needed. Not all communications have been protected, and even out of the already protected communications, not all of their aspects have been fully protected. Fortunately, there are ongoing projects working on improvements.

- Adversaries have increased their pressure against other avenues of attack, from compromising devices to legal coercion of centralized endpoints in conversations.

- New adversaries and risks have arisen, e.g., due to creation of large centralized information sources.

In short, attacks are migrating towards the currently easier targets, which no longer necessarily include direct attacks on traffic flows. In addition, trading information about users and ability to influence them has become a common practice for many Internet services, often without consent of the users.

This memo suggests that the existing threat model, while important and still valid, is no longer alone sufficient to cater for the pressing security issues in the Internet. For instance, while it
continues to be very important to protect Internet communications against outsiders, it is also necessary to protect systems against endpoints that are compromised, malicious, or whose interests simply do not align with the interests of the users.

Of course, there are many trade-offs in the Internet on who one chooses to interact with and why or how. It is not the role of this memo to dictate those choices. But it is important that we understand the implications of different practices. It is also important that when it comes to basic Internet infrastructure, our chosen technologies lead to minimal exposure with respect to the non-communications threats.

It is particularly important to ensure that non-communications security related threats are properly understood for any new Internet technology. While the consideration of these issues is relatively new in the IETF, this memo provides some initial ideas about potential broader threat models to consider when designing protocols for the Internet or when trying to defend against pervasive monitoring. Further down the road, updated threat models could result in changes in BCP 72 [RFC3552] (guidelines for writing security considerations) and BCP 188 [RFC7258] (pervasive monitoring), to include proper consideration of non-communications security threats.

It may also be necessary to have dedicated guidance on how systems design and architecture affects security. The sole consideration of communications security aspects in designing Internet protocols may lead to accidental or increased impact of security issues elsewhere. For instance, allowing a participant to unnecessarily collect or receive information may be lead to a similar effect as described in [RFC8546] for protocols: over time, unnecessary information will get used with all the associated downsides, regardless of what deployment expectations there were during protocol design.

The rest of this memo is organized as follows. Section 2 and Section 3 outline the situation with respect to communications security and beyond it. Section 4.1 discusses how the author believes the Internet threat model should evolve, and what types of threats should be seen as critical ones and in-scope. Section 5 will also discuss high-level guidance to addressing these threats.

Section 6 outlines the author’s suggested future changes to RFC 3552 and RFC 7258 and the need for guidance on the impacts of system design and architecture on security. Comments are solicited on these and other aspects of this document. The best place for discussion is on the arch-discuss list (https://www.ietf.org/mailman/listinfo/Architecture-discuss). This memo acts also as an input for the IAB
retreat discussion on threat models, and it is a submission for the IAB DEDR workshop (https://www.iab.org/activities/workshops/dedr-workshop/).

Finally, Section 7 highlights other discussions in this problem space and Section 8 draws some conclusions for next steps.

2. Improvements in Communications Security

The fraction of Internet traffic that is cryptographically protected has grown tremendously in the last few years. Several factors have contributed to this change, from Snowden revelations to business reasons and to better available technology such as HTTP/2 [RFC7540], TLS 1.3 [RFC8446], QUIC [I-D.ietf-quic-transport].

In many networks, the majority of traffic has flipped from being cleartext to being encrypted. Reaching the level of (almost) all traffic being encrypted is no longer something unthinkable but rather a likely outcome in a few years.

At the same time, technology developments and policy choices have driven the scope of cryptographic protection from protecting only the pure payload to protecting much of the rest as well, including far more header and meta-data information than was protected before. For instance, efforts are ongoing in the IETF to assist encrypting transport headers [I-D.ietf-quic-transport], server domain name information in TLS [I-D.ietf-tls-esni], and domain name queries [RFC8484].

There has also been improvements to ensure that the security protocols that are in use actually have suitable credentials and that those credentials have not been compromised, see, for instance, Let’s Encrypt [RFC8555], HSTS [RFC6797], HPKP [RFC7469], and Expect-CT [I-D.ietf-httpbis-expect-ct].

This is not to say that all problems in communications security have been resolved - far from it. But the situation is definitely different from what it was a few years ago. Remaining issues will be and are worked on; the fight between defense and attack will also continue. Communications security will stay at the top of the agenda in any Internet technology development.

3. Issues in Security Beyond Communications Security

There are, however, significant issues beyond communications security in the Internet. To begin with, it is not necessarily clear that one can trust all the endpoints.
Of course, the endpoints were never trusted, but the pressures against endpoints issues seem to be mounting. For instance, the users may not be in as much control over their own devices as they used to be due to manufacturer-controlled operating system installations and locked device ecosystems. And within those ecosystems, even the applications that are available tend to have features that users by themselves would most likely not desire to have, such as excessive rights to media, location, and peripherals. There are also designated efforts by various authorities to hack end-user devices as a means of intercepting data about the user.

The situation is different, but not necessarily better on the side of servers. The pattern of communications in today’s Internet is almost always via a third party that has at least as much information than the other parties have. For instance, these third parties are typically endpoints for any transport layer security connections, and able to see any communications or other messaging in cleartext. There are some exceptions, of course, e.g., messaging applications with end-to-end protection.

With the growth of trading users’ information by many of these third parties, it becomes necessary to take precautions against endpoints that are compromised, malicious, or whose interests simply do not align with the interests of the users.

Specifically, the following issues need attention:

- Security of users’ devices and the ability of the user to control their own equipment.
- Leaks and attacks related to data at rest.
- Coercion of some endpoints to reveal information to authorities or surveillance organizations, sometimes even in an extra-territorial fashion.
- Application design patterns that result in cleartext information passing through a third party or the application owner.
- Involvement of entities that have no direct need for involvement for the sake of providing the service that the user is after.
- Network and application architectures that result in a lot of information collected in a (logically) central location.
- Leverage and control points outside the hands of the users or end-user device owners.
For instance, while e-mail transport security [RFC7817] has become much more widely distributed in recent years, progress in securing e-mail messages between users has been much slower. This has lead to a situation where e-mail content is considered a critical resource by mail providers who use it for machine learning, advertisement targeting, and other purposes.

The Domain Name System (DNS) shows signs of ageing but due to the legacy of deployed systems, has changed very slowly. Newer technology [RFC8484] developed at the IETF enables DNS queries to be performed confidentially, but its deployment is happening mostly in browsers that use global DNS resolver services, such as Cloudflare’s 1.1.1.1 or Google’s 8.8.8.8. This results in faster evolution and better security for end users.

However, if one steps back and considers the overall security effects of these developments, the resulting effects can be different. While the security of the actual protocol exchanges improves with the introduction of this new technology, at the same time this implies a move from using a worldwide distributed set of DNS resolvers into more centralised global resolvers. While these resolvers are very well maintained (and a great service), they are potential high-value targets for pervasive monitoring and Denial-of-Service (DoS) attacks. In 2016, for example, DoS attacks were launched against Dyn, one of the largest DNS providers, leading to some outages. It is difficult to imagine that DNS resolvers wouldn’t be a target in many future attacks or pervasive monitoring projects.

Unfortunately, there is little that even large service providers can do to refuse authority-sanctioned pervasive monitoring. As a result it seems that the only reasonable course of defense is to ensure that no such information or control point exists.

There are other examples about the perils of centralised solutions in Internet infrastructure. The DNS example involved an interesting combination of information flows (who is asking for what domain names) as well as a potential ability to exert control (what domains will actually resolve to an address). Routing systems are primarily about control. While there are intra-domain centralized routing solutions (such as PCE [RFC4655]), a control within a single administrative domain is usually not the kind of centralization that we would be worried about. Global centralization would be much more concerning. Fortunately, global Internet routing is performed a among peers. However, controls could be introduced even in this global, distributed system. To secure some of the control exchanges, the Resource Public Key Infrastructure (RPKI) system ([RFC6480]) allows selected Certification Authorities (CAs) to help drive decisions about which participants in the routing infrastructure can
make what claims. If this system were globally centralized, it would be a concern, but again, fortunately, current designs involve at least regional distribution.

In general, many recent attacks relate more to information than communications. For instance, personal information leaks typically happen via information stored on a compromised server rather than capturing communications. There is little hope that such attacks can be prevented entirely. Again, the best course of action seems to be avoid the disclosure of information in the first place, or at least to not perform that in a manner that makes it possible that others can readily use the information.

4. Impacts

4.1. The Role of End-to-end

[RFC1958] notes that "end-to-end functions can best be realised by end-to-end protocols":

The basic argument is that, as a first principle, certain required end-to-end functions can only be performed correctly by the end-systems themselves. A specific case is that any network, however carefully designed, will be subject to failures of transmission at some statistically determined rate. The best way to cope with this is to accept it, and give responsibility for the integrity of communication to the end systems. Another specific case is end-to-end security.

The "end-to-end argument" was originally described by Saltzer et al [Saltzer]. They said:

The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible.

These functional arguments align with other, practical arguments about the evolution of the Internet under the end-to-end model. The endpoints evolve quickly, often with simply having one party change the necessary software on both ends. Whereas waiting for network upgrades would involve potentially a large number of parties from application owners to multiple network operators.

The end-to-end model supports permissionless innovation where new innovation can flourish in the Internet without excessive wait for other parties to act.
But the details matter. What is considered an endpoint? What characteristics of Internet are we trying to optimize? This memo makes the argument that, for security purposes, there is a significant distinction between actual endpoints from a user’s interaction perspective (e.g., another user) and from a system perspective (e.g., a third party relaying a message).

This memo proposes to focus on the distinction between "real ends" and other endpoints to guide the development of protocols. A conversation between one "real end" to another "real end" has necessarily different security needs than a conversation between, say, one of the "real ends" and a component in a larger system. The end-to-end argument is used primarily for the design of one protocol. The security of the system, however, depends on the entire system and potentially multiple storage, compute, and communication protocol aspects. All have to work properly together to obtain security.

For instance, a transport connection between two components of a system is not an end-to-end connection even if it encompasses all the protocol layers up to the application layer. It is not end-to-end, if the information or control function it carries actually extends beyond those components. For instance, just because an e-mail server can read the contents of an e-mail message does not make it a legitimate recipient of the e-mail.

This memo also proposes to focus on the "need to know" aspect in systems. Information should not be disclosed, stored, or routed in cleartext through parties that do not absolutely need to have that information.

The proposed argument about real ends is as follows:

Application functions are best realised by the entities directly serving the users, and when more than one entity is involved, by end-to-end protocols. The role and authority of any additional entities necessary to carry out a function should match their part of the function. No information or control roles should be provided to these additional entities unless it is required by the function they provide.

For instance, a particular piece of information may be necessary for the other real endpoint, such as message contents for another user. The same piece of information may not be necessary for any additional parties, unless the information had to do with, say, routing information for the message to reach the other user. When information is only needed by the actual other endpoint, it should be protected and be only relayed to the actual other endpoint. Protocol
design should ensure that the additional parties do not have access to the information.

Note that it may well be that the easiest design approach is to send all information to a third party and have majority of actual functionality reside in that third party. But this is a case of a clear tradeoff between ease of change by evolving that third party vs. providing reasonable security against misuse of information.

Note that the above "real ends" argument is not limited to communication systems. Even an application that does not communicate with anyone else than its user may be implemented on top of a distributed system where some information about the user is exposed to untrusted parties.

The implications of the system security also extend beyond information and control aspects. For instance, poorly design component protocols can become DoS vectors which are then used to attack other parts of the system. Availability is an important aspect to consider in the analysis along other aspects.

4.2. Trusted networks

Some systems are thought of as being deployed only in a closed setting, where all the relevant nodes are under direct control of the network administrators. Technologies developed for such networks tend to be optimized, at least initially, for these environments, and may lack security features necessary for different types of deployments.

It is well known that many such systems evolve over time, grow, and get used and connected in new ways. For instance, the collaboration and mergers between organizations, and new services for customers may change the system or its environment. A system that used to be truly within an administrative domain may suddenly need to cross network boundaries or even run over the Internet. As a result, it is also well known that it is good to ensure that underlying technologies used in such systems can cope with that evolution, for instance, by having the necessary security capabilities to operate in different environments.

In general, the outside vs. inside security model is outdated for most situations, due to the complex and evolving networks and the need to support mixture of devices from different sources (e.g., BYOD networks). Network virtualization also implies that previously clear notions of local area networks and physical proximity may create an entirely different reality from what appears from a simple notion of a local network.
4.2.1. Even closed networks can have compromised nodes

This memo argues that the situation is even more dire than what was explained above. It is impossible to ensure that all components in a network are actually trusted. Even in a closed network with carefully managed components there may be compromised components, and this should be factored into the design of the system and protocols used in the system.

For instance, during the Snowden revelations it was reported that internal communication flows of large content providers were compromised in an effort to acquire information from large number of end users. This shows the need to protect not just communications targeted to go over the Internet, but in many cases also internal and control communications.

Furthermore, there is a danger of compromised nodes, so communications security alone will be insufficient to protect against this. The defences against this include limiting information within networks to the parties that have a need to know, as well as limiting control capabilities. This is necessary even when all the nodes are under the control of the same network manager; the network manager needs to assume that some nodes and communications will be compromised, and build a system to mitigate or minimise attacks even under that assumption.

Even airgapped networks can have these issues, as evidenced, for instance, by the Stuxnet worm. The Internet is not the only form of connectivity, as most systems include, for instance, USB ports that proved to be the achilles heel of the targets in the Stuxnet case. More commonly, every system runs large amount of software, and it is often not practical or even possible to black the software to prevent compromised code even in a high-security setting, let alone in commercial or private networks. Installation media, physical ports, both open source and proprietary programs, firmware, or even innocent-looking components on a circuit board can be suspect. In addition, complex underlying computing platforms, such as modern CPUs with underlying security and management tools are prone for problems.

In general, this means that one cannot entirely trust even a closed system where you picked all the components yourself. Analysis for the security of many interesting real-world systems now commonly needs to include cross-component attacks, e.g., the use of car radios and other externally communicating devices as part of attacks launched against the control components such as breaks in a car [Savage].
4.3. Balancing Threats

Note that not all information needs to be protected, and not all threats can be protected against. But it is important that the main threats are understood and protected against.

Sometimes there are higher-level mechanisms that provide safeguards for failures. For instance, it is very difficult in general to protect against denial-of-service against compromised nodes on a communications path. However, it may be possible to detect that a service has failed.

Another example is from packet-carrying networks. Payload traffic that has been properly protected with encryption does not provide much value to an attacker. As a result, it does not always make sense, for instance, encrypt every packet transmission in a packet-carrying system where the traffic is already encrypted at other layers. But it almost always makes sense to protect control communications and to understand the impacts of compromised nodes, particularly control nodes.

5. Guidelines

As [RFC3935] says:

We embrace technical concepts such as decentralized control, edge-user empowerment and sharing of resources, because those concepts resonate with the core values of the IETF community.

To be more specific, this memo suggests the following guidelines for protocol designers:

1. Consider first principles in protecting information and systems, rather than following a specific pattern such as protecting information in a particular way or at a particular protocol layer. It is necessary to understand what components can be compromised, where interests may or may not be aligned, and what parties have a legitimate role in being a party to a specific information or a control task.

2. Minimize information passed to others: Information passed to another party in a protocol exchange should be minimized to guard against the potential compromise of that party.

3. Perform end-to-end protection via other parties: Information passed via another party who does not intrinsically need the information to perform its function should be protected end-to-end to its intended recipient. This guideline is general, and
holds equally for sending TCP/IP packets, TLS connections, or application-layer interactions. As [I-D.iab-wire-image] notes, it is a useful design rule to avoid "accidental invariance" (the deployment of on-path devices that over-time start to make assumptions about protocols). However, it is also a necessary security design rule to avoid "accidental disclosure" where information originally thought to be benign and untapped over-time becomes a significant information leak. This guideline can also be applied for different aspects of security, e.g., confidentiality and integrity protection, depending on what the specific need for information is in the other parties.

4. Minimize passing of control functions to others: Any passing of control functions to other parties should be minimized to guard against the potential misuse of those control functions. This applies to both technical (e.g., nodes that assign resources) and process control functions (e.g., the ability to allocate number or develop extensions). Control functions can also become a matter of contest and power struggle, even in cases where their function as such is minimal, as we saw with the IANA transition debates.

5. Avoid centralized resources: While centralized components, resources, and function provide usually a useful function, there are grave issues associated with them. Protocol and network design should balance the benefits of centralized resources or control points against the threats arising from them. The general guideline is to avoid such centralized resources when possible. And if it is not possible, find a way to allow the centralized resources to be selectable, depending on context and user settings.

6. Have explicit agreements: When users and their devices provide information to network entities, it would be beneficial to have an opportunity for the users to state their requirements regarding the use of the information provided in this way. While the actual use of such requirements and the willingness of network entities to agree to them remains to be seen, at the moment even the technical means of doing this are limited. For instance, it would be beneficial to be able to embed usage requirements within popular data formats.

7. Treat parties that your equipment connects to with suspicion, even if the communications are encrypted. The other endpoint may misuse any information or control opportunity in the communication. Similarly, even parties within your own system need to be treated with suspicion, as some nodes may become compromised.
8. Do not take any of this as blanket reason to provide no information to anyone, encrypt everything to everyone, or other extreme measures. However, the designers of a system need to be aware of the different threats facing their system, and deal with the most serious ones (of which there are typically many). Similarly, users should be aware of the choices made in a particular design, and avoid designs or products that protect against some threats but are wide open to other serious issues.

6. Potential Changes in IETF Analysis of Protocols

6.1. Changes in RFC 3552

This memo suggests that changes maybe necessary in RFC 3552. One initial, draft proposal for such changes would be this:

OLD:

In general, we assume that the end-systems engaging in a protocol exchange have not themselves been compromised. Protecting against an attack when one of the end-systems has been compromised is extraordinarily difficult. It is, however, possible to design protocols which minimize the extent of the damage done under these circumstances.

NEW:

In general, we assume that the end-system engaging in a protocol exchange has not itself been compromised. Protecting against an attack of a protocol implementation itself is extraordinarily difficult. It is, however, possible to design protocols which minimize the extent of the damage done when the other parties in a protocol become compromised or do not act in the best interests the end-system implementing a protocol.

In addition, the following new section could be added to discuss the capabilities required to mount an attack:

NEW:

3.x. Other endpoint compromise

In this attack, the other endpoints in the protocol become compromised. As a result, they can, for instance, misuse any information that the end-system implementing a protocol has sent to the compromised endpoint.
6.2. Changes in RFC 7258

This memo also suggests that additional guidelines may be necessary in RFC 7258. An initial, draft suggestion for starting point of those changes could be adding the following paragraph after the 2nd paragraph in Section 2:

NEW:

PM attacks include those cases where information collected by a legitimate protocol participant is misused for PM purposes. The attacks also include those cases where a protocol or network architecture results in centralized data storage or control functions relating to many users, raising the risk of said misuse.

6.3. System and Architecture Aspects

This definitely needs more attention from Internet technology developers and standards organizations. Here is one possible

The design of any Internet technology should start from an understanding of the participants in a system, their roles, and the extent to which they should have access to information and ability to control other participants.

7. Other Work

See, for instance, [I-D.farrell-etc].

8. Conclusions

More work is needed in this area. To start with, Internet technology developers need to be better aware of the issues beyond communications security, and consider them in design. At the IETF it would be beneficial to include some of these considerations in the usual systematic security analysis of technologies under development.

In particular, when the IETF develops infrastructure technology for the Internet (such as routing or naming systems), considering the impacts of data generated by those technologies is important. Minimising data collection from users, minimising the parties who get exposed to user data, and protecting data that is relayed or stored in systems should be a priority.

A key focus area at the IETF has been the security of transport protocols, and how transport layer security can be best used to provide the right security for various applications. However, more work is needed in equivalently broadly deployed tools for minimising
or obfuscating information provided by users to other entities, and the use of end-to-end security through entities that are involved in the protocol exchange but who do not need to know everything that is being passed through them.

Comments on the issues discussed in this memo are gladly taken either privately or on the architecture-discuss mailing list.

9. Acknowledgements

The author would like to thank John Mattsson, Mirja Kuehlewind, Alissa Cooper, Stephen Farrell, Eric Rescorla, Simone Ferlin, Kathleen Moriarty, Brian Trammell, Mark Nottingham, Christian Huitema, Karl Norrman, Ted Hardie, Mohit Sethi, Phillip Hallam-Baker, Goran Eriksson and the IAB for interesting discussions in this problem space. The author would also like to thank all members of the 2019 Design Expectations vs. Deployment Reality (DEDR) IAB workshop held in Kirkkonummi, Finland.

10. Informative References

[I-D.farrell-etm]
Farrell, S., "We're gonna need a bigger threat model", draft-farrell-etm-03 (work in progress), July 2019.

[I-D.iab-wire-image]

[I-D.ietf-httpbis-expect-ct]
estark@google.com, e., "Expect-CT Extension for HTTP", draft-ietf-httpbis-expect-ct-08 (work in progress), December 2018.

[I-D.ietf-quic-transport]

[I-D.ietf-tls-esni]

[I-D.nottingham-for-the-users]
Nottingham, M., "The Internet is for End Users", draft-nottingham-for-the-users-08 (work in progress), June 2019.


Author’s Address

Jari Arkko
Ericsson

Email: jari.arkko@piuha.net
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 6, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

This document defines a base YANG [RFC6020] [RFC7950] data module for Quality of Service (QoS) configuration parameters. Differentiated Services (DiffServ) module is an augmentation of the base QoS model. Remote Procedure Calls (RPC) or notification definition is not part of this document. QoS base modules define a basic building blocks to define a classifier, policy, action and target. The base modules have been augmented to include packet match fields and action parameters to define the DiffServ module. Queues and schedulers are stitched as part of diffserv policy itself or separate modules are defined for creating Queue policy and Scheduling policy. The DiffServ model is based on DiffServ architecture, and various references have been made to available standard architecture documents.
DiffServ is a preferred approach for network service providers to offer services to different customers based on their network Quality-of-Service (QoS) objectives. The traffic streams are differentiated based on DiffServ Code Points (DSCP) carried in the IP header of each packet. The DSCP markings are applied by upstream node or by the edge router on entry to the DiffServ network.

Editorial Note: (To be removed by RFC Editor)

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

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

1.1. Tree Diagrams

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

2. Terminology

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

3. QoS Model Design

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

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

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

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

4. DiffServ Model Design

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

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

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

5. Modules Tree Structure

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

```yang
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.

```yang
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
                  }
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.

```yang
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
```

++-:(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
| ---: (meter-reference) {action:meter-reference-feature}? |
| | ---: (traffic-group-marking) {action:traffic-group-feature}? |
| | ---: (child-policy) {action:child-policy-feature}? |
| | ---: (count) {action:count-feature}? |
| | ---: (named-count) {action:named-counter-feature}? |
| | ---: (queue-inline) {diffserv-queue-inline-support}? |
| | +--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)
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)?
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;
    prefix classifier;

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

    description "This module contains a collection of YANG definitions for configuring qos specification implementations. Copyright (c) 2019 IETF Trust and the persons identified as the document authors. 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.";
    }

Choudhary, et al. Expires January 6, 2020
feature classifier-template-feature {
  description
    "This feature allows classifier as template configuration in a policy.";
}

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

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

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

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

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

grouping filters {
  description
    "Filters types in a Classifier entry";
  leaf filter-type {
    type identityref {
      base filter-type;
    }
    description
      "This leaf defines type of the filter";
  }
  leaf filter-logical-not {
    type boolean;
    description
      "classifier entry filter logical not operation";
  }
}
This is logical-not operator for a filter. When true, it indicates filter looks for absence of a pattern defined by the 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";
    }
  }
}

6.2. IETF-QOS-POLICY

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

This module contains a collection of YANG definitions for configuring qos specification implementations. Copyright (c) 2019 IETF Trust and the persons identified as the document authors. 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 {

6.3. IETF-QOS-ACTION

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

feature meter-inline-feature {
    description "This feature allows support of meter-inline configuration.";
}

feature meter-reference-feature {
    description "This feature allows support of meter by reference configuration.";
}

feature queue-action-support {
    description "This feature allows support of queue action configuration in policy.";
}

feature scheduler-action-support {
    description "This feature allows support of scheduler configuration in policy.";
}

feature child-policy-feature {
    description "This feature allows configuration of hierarchical policy.";
}

feature count-feature {
    description "This feature allows action configuration to enable counter in a classifier";
}

feature named-counter-feature {
    description "This feature allows action configuration to enable named counter in a classifier";
}
feature traffic-group-feature {
    description
    "traffic-group action support";
}
feature burst-time-unit-support {
    description
    "This feature allows burst unit to be configured as
    time duration.";
}

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

identity mega-bytes {
    base burst-unit-type;
    description
    "mega bytes";
}

identity millisecond {
    base burst-unit-type;
    if-feature burst-time-unit-support;
    description
    "milli seconds";
}

identity microsecond {
    base burst-unit-type;
    if-feature burst-time-unit-support;
    description
    "micro seconds";
}

identity dscp-marking {
    base policy:action-type;
    description
    "dscp marking action type";
}

identity meter-inline {
    base policy:action-type;
    if-feature meter-inline-feature;
    description
    "meter-inline action type";
}

identity meter-reference {
    base policy:action-type;
    if-feature meter-reference-feature;
    description
    "meter reference action type";
}

identity queue {
    base policy:action-type;
    if-feature queue-action-support;
    description
    "queue action type";
}

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

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

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

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

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

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

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

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

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

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

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


grouping threshold {
  description
  "Threshold Parameters";
  container threshold {
    description
    "threshold";
    choice threshold-type {
      case size {
        leaf threshold-size {
          type uint64;
          units "bytes";
          description
          "Threshold size";
        }
      }
      case interval {
        leaf threshold-interval {
          type uint64;
          units "microsecond";
          description
          "Threshold interval";
        }
        description
        "Choice of threshold type";
      }
    }
  }
}
grouping drop {
   container drop-cfg {
      leaf drop-action {
         type empty;
         description
         "always drop algorithm";
      }
      description
      "the drop action";
   }
   description
   "always drop grouping";
}

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

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

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

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

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

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

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

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

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

grouping one-rate-tri-color-meter {
  container one-rate-tri-color-meter {
    description
    "single rate three color meter";
    reference
    "RFC2697: A Single Rate Three Color Marker";
    leaf committed-rate-value {
      type uint64;
      description
      "meter rate";
    }
    leaf committed-rate-unit {
      type identityref {
        base rate-unit-type;
      }
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 "committed 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 "committed 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
        "two-rate-tri-color-meter attributes";
}
"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";
  }
}
grouping min-rate {
    container min-rate-cfg {
        uses rate-value-unit;
        description "min guaranteed bandwidth";
        reference "RFC3289, section 3.5.3";
    }
    description "minimum rate grouping";
}

grouping dscp-marking {
    container dscp-cfg {
        leaf dscp {
            type inet:dscp;
            description "dscp marking";
        }
        description "dscp marking container";
    }
    description "dscp marking grouping";
}

grouping traffic-group-marking {
    container traffic-group-cfg {
        leaf traffic-group {
            type string;
            description "traffic group marking";
        }
        description "traffic group marking container";
    }
    description "traffic group marking grouping";
}

grouping child-policy {
    container child-policy-cfg {
        if-feature child-policy-feature;
        leaf policy-name {
            type string;
            description "Hierarchical Policy";
        }
    }
    description "child policy grouping";
}
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";
}

6.4. IETF-QOS-TARGET

<CODE BEGINS>file "ietf-qos-target@2019-03-13.yang"
module ietf-qos-target {
   yang-version 1.1;
   prefix target;

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

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

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

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

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

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

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

"Direction of 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";
}
}

6.5. IETF-DIFFSERV

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

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

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

Choudhary, et al. Expires January 6, 2020
This module contains a collection of YANG definitions for configuring diffserv specification implementations. Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved. Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info). This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.

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

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

feature diffserv-scheduler-inline-support {
  description
    "scheduler inline support in diffserv policy";
}

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

identity ipv4-diffserv-policy-type {
  base policy:policy-type;
  description
    "This defines ipv4 policy-type";
}
identity ipv6-diffserv-policy-type {
    base policy:policy-type;
    description
        "This defines ipv6 policy-type";
}

identity dscp {
    base classifier:filter-type;
    description
        "Differentiated services code point filter-type";
}

identity source-ipv4-address {
    base classifier:filter-type;
    description
        "source ipv4 address filter-type";
}

identity destination-ipv4-address {
    base classifier:filter-type;
    description
        "destination ipv4 address filter-type";
}

identity source-ipv6-address {
    base classifier:filter-type;
    description
        "source ipv6 address filter-type";
}

identity destination-ipv6-address {
    base classifier:filter-type;
    description
        "destination ipv6 address filter-type";
}

identity source-port {
    base classifier:filter-type;
    description
        "source port filter-type";
}

identity destination-port {
    base classifier:filter-type;
    description
        "destination port filter-type";
}

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

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

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

identity one-rate-two-color-meter-type {
  base meter-type;
  description
    "one rate two color meter type";
}

identity one-rate-tri-color-meter-type {
  base meter-type;
  description
    "one rate three color meter type";
}

identity two-rate-tri-color-meter-type {
  base meter-type;
  description
    "two rate three color meter action type";
}

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

grouping source-ipv4-address-cfg {
  list source-ipv4-address-cfg {
    key "source-ipv4-addr";
    description
      "list of source ipv4 address";
  }
}
leaf source-ipv4-addr {
    type inet:ipv4-prefix;
    description
    "source ipv4 prefix";
}
}
description
"Filter grouping containing list of source ipv4 addresses";
}
grouping destination-ipv4-address-cfg {
    list destination-ipv4-address-cfg {
        key "destination-ipv4-addr";
        description
        "list of destination ipv4 address";
        leaf destination-ipv4-addr {
            type inet:ipv4-prefix;
            description
            "destination ipv4 prefix";
        }
    }
}
description
"Filter grouping containing list of destination ipv4 address";
}
grouping source-ipv6-address-cfg {
    list source-ipv6-address-cfg {
        key "source-ipv6-addr";
        description
        "list of source ipv6 address";
        leaf source-ipv6-addr {
            type inet:ipv6-prefix;
            description
            "source ipv6 prefix";
        }
    }
}
description
"Filter grouping containing list of source ipv6 addresses";
}
grouping destination-ipv6-address-cfg {
    list destination-ipv6-address-cfg {
        key "destination-ipv6-addr";
        description
        "list of destination ipv4 or ipv6 address";
        leaf destination-ipv6-addr {
            type inet:ipv6-prefix;
            description
            "destination ipv6 prefix";
        }
    }
}
description
 "Filter grouping containing list of destination ipv6 address";
}
grouping source-port-cfg {
  list source-port-cfg {
    key "source-port-min source-port-max";
    description
    "list of ranges of source port";
    leaf source-port-min {
      type inet:port-number;
      description
      "minimum value of source port range";
    }
    leaf source-port-max {
      type inet:port-number;
      description
      "maximum value of source port range";
    }
  }
  description
  "Filter grouping containing list of source port ranges";
}
grouping destination-port-cfg {
  list destination-port-cfg {
    key "destination-port-min destination-port-max";
    description
    "list of ranges of destination port";
    leaf destination-port-min {
      type inet:port-number;
      description
      "minimum value of destination port range";
    }
    leaf destination-port-max {
      type inet:port-number;
      description
      "maximum value of destination port range";
    }
  }
  description
  "Filter grouping containing list of destination port ranges";
}
grouping protocol-cfg {
  list protocol-cfg {
    key "protocol-min protocol-max";
    description
    "list of ranges of protocol values";
    leaf protocol-min {
      type uint8
range "0..255";
}  

description  
"minimum value of protocol range";
}
leaf protocol-max {  

type uint8 {  

range "0..255";
}  

description  
"maximum value of protocol range";
}
}

description  
"Filter grouping containing list of Protocol ranges";
}
grouping traffic-group-cfg {  

container traffic-group-cfg {  

leaf traffic-group-name {  

type string ;  

description  
"This leaf defines name of the traffic group referenced";
}

description  
"traffic group container";
}

description  
"traffic group grouping";
}

augment "/classifier:classifiers/classifier:classifier-entry" +  
"/classifier:filter-entry" {  

 choice filter-param {  

 description  
 "Choice of filter types";
 case dscp {  

 uses dscp-cfg;  

 description  
 "Filter containing list of dscp ranges";
 }
 case source-ipv4-address {  

 uses source-ipv4-address-cfg;  

 description  
 "Filter containing list of source ipv4 addresses";
 }
 case destination-ipv4-address {  

 uses destination-ipv4-address-cfg;  

 description

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

  import ietf-qos-policy {
    prefix policy;
    reference "RFC XXXX: YANG Model for QoS";
  }
  import ietf-qos-action {
    prefix action;
    reference "RFC XXXX: YANG Model for QoS";
  }
  import ietf-diffserv {
    prefix diffserv;
    reference "RFC XXXX: YANG Model for QoS";
  }
}
organization "IETF RTG (Routing Area) Working Group";
contact
"WG Web: <http://tools.ietf.org/wg/rtgwg/>
WG List: <mailto:rtgwg@ietf.org>
WG Chair: Chris Bowers
   <mailto:cbowers@juniper.net>
WG Chair: Jeff Tantsura
   <mailto:jefftant.ietf@gmail.com>
Editor: Aseem Choudhary
   <mailto:asechoud@cisco.com>
Editor: Mahesh Jethanandani
   <mailto:mjethanandani@gmail.com>
Editor: Norm Strahle
   <mailto:nstrahle@juniper.net>"

description
"This module contains a collection of YANG definitions for
configuring diffserv specification implementations.
Copyright (c) 2019 IETF Trust and the persons identified as
the document authors. 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-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";
}

6.7. IETF-SCHEDULER-POLICY

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

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

description
"This module contains a collection of YANG definitions for
configuring diffserv specification implementations.
Copyright (c) 2019 IETF Trust and the persons identified as
the document authors.  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;
}
augment "/policy:policies/policy:policy-entry" + "/policy:classifier-entry" + "/policy:classifier-action-entry-cfg" + "/policy:action-cfg-params" { 
  when "../..//policy:policy-type = 
      'scheduler-policy:scheduler-policy-type'" { 
    description 
      "Only when policy type is scheduler-policy"; 
  } 
  case scheduler { 
    uses action:scheduler; 
  } 
  case queue-policy-name { 
    uses queue-policy-name-cfg; 
  } 
  description 
    "augments scheduler template reference to scheduler policy"; 
} 
</CODE ENDS>

7. IANA Considerations

TBD
8. Security Considerations

9. Acknowledgement

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

10. References

10.1. Normative References


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.

```yang
mODULE example-compa-diffserv 
  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.";
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: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: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" {
"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;
    }
}

      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;
      }
    }

"/difserv:meter-type" +
"/difserv:two-rate-tri-color-meter-type" +
"/difserv:two-rate-tri-color-meter" +
"/difserv:exceed-action" +
"/difserv:exceed-3color-meter-action-params" +
"/difserv: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" +
"/difserv:meter-inline" +
"/difserv:meter-type" +
"/difserv:two-rate-tri-color-meter-type" +
"/difserv:two-rate-tri-color-meter" +
"/difserv:violate-action" +
"/difserv:violate-3color-meter-action-params" +
"/difserv: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: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: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";
  }
}
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 multiple 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
    the document authors. 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" +
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:one-rate-two-color-meter-type" +
  "/action:one-rate-two-color-meter" {
leaf two-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";
  import ietf-qos-classifier {
    prefix classifier;
    reference "RFC XXXX: YANG Model for QoS";
  }
  import ietf-qos-policy {
    prefix policy;
  }
}
This module defines a queue policy. The classification is based on a forwarding class, and the actions are queues. Copyright (c) 2019 IETF Trust and the persons identified as the document authors. 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;
  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-porportion {
        type uint32 {
            range "1..1000";
            description
                "excess-rate-porportion";
        }
    }
    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 guaranteed rate) traffic";
}
container buffer-size {
    choice buffer-size-type {
        case percent {
            leaf buffer-size-percent {
                type uint32 {
                    range "1..100";
                }
            }
            description
                "Excess rate value";
        }
    }
    description
        "Choice of excess-rate type";
}

"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 mdrq-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 */

declaration "/pol:polices/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-queuing-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";
   }
}  
}
  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 mdrq-queuing-policy { 
  when "../../../../sub-type = 'mdrr-queuing-policy'" { 
    description 
    "MDRR queue handling properties for the traffic " + 
    "classified into current queue";
  }
  leaf mdrq-queue-weight { 
    type uint8 { 
      range "20..100";
    }
    units percentage;
  }
}
case pwfq-queuing-policy { 
  when "../../../../sub-type = 'pwfq-queuing-policy'" { 
    description 
    "PWFQ queue handling properties for traffic " + 
    "classified into current queue";
  }
  leaf pwfq-queue-weight { 
    type uint8 { 
      range "20..100";
    }
    units percentage;
  }
}
leaf pwfq-queue-priority {
  type uint8;
}
leaf pwfq-queue-rate {
  type uint8;
}

case forwarding-policy {
  when "../../../sub-type = 'forwarding-policy'" {
    description
    "Forward policy handling properties for traffic " +
    "in this classifier";
  }
  uses redirect-action-grp;
  description
  "Add the classify action configuration";
}

Authors’ Addresses

Aseem Choudhary
Cisco Systems
170 W. Tasman Drive
San Jose, CA  95134
US
Email: asechoud@cisco.com

Mahesh Jethanandani
Cisco Systems
170 W. Tasman Drive
San Jose, CA  95134
US
Email: mjethanandani@gmail.com
Preferred Path Loop-Free Alternate (pLFA)
draft-bryant-rtgwglpfa-00

Abstract

Fast re-route (FRR) is a technique that allows productive forwarding to continue in a network after a failure has occurred, but before the network has has time to re-converge. This is achieved by forwarding a packet on an alternate path that will not result in the packet looping. Preferred Path Routing (PPR) provides a method of injecting explicit paths into the routing protocol. The use of PPR to support FRR has a number of advantages. This document describes the advantages of using PPR to provide a loop-free alternate FRR path, and provides a framework for its use in this application.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
1. Introduction

Preferred Path Routing (PPR)
[I-D.chunduri-lsr-isis-preferred-path-routing] is a method of introducing explicit paths to a network. Such a path may be any loop-free path between two points in the network that satisfies the need for which the path was created. The PPR path is not constrained to be the shortest path between any points in the network, although...
the use of shortest path segments is provided for in order to compress the size of the path description flooded by the routing protocol. The advantages of PPR over alternate methods of creating such paths is described in [I-D.chunduri-lsr-isis-preferred-path-routing].

A packet is carried over the network in an appropriate form using the Preferred Path Routing Identifier (PPR-ID) as the data plane identifier to map the packet to the PPR path, and hence the resources and next-hop (NH). One way of adding a PPR-ID to a packet would be to encapsulate it, but PPR does not restrict the user to the use of encapsulation. How the PPR-ID is carried in the general case is outside the scope of this document. Various methods of adding the PPR-ID to a packet for the purposes of Fast-Reroute (FRR) are described in Section 9.

IP Fast Re-route (IPFRR) Section 1.1 and the methods known at the time of its writing is described in [RFC5714]. A number of later methods are described in [RFC6981], [RFC7490], [RFC7812] and [I-D.ietf-rtgwg-segment-routing-ti-lfa].

This document is a framework describing various methods whereby PPR can be used to provide IP Fast Reroute (IPFRR) paths. PPR can provide IPFRR in a number of ways.

- Signaling pre-computed preferred alternatives for the primary path
- Signaling individual segments on the repair path.
- Selective overriding of locally computed Loop Free Alternates (LFA) for the NH failure.
- Local repair to a Traffic Engineered paths avoiding the need for multi-hop Bidirectional Forwarding Detection (BFD) [RFC5880].
- Micro-loop elimination [RFC5715].

These are described in more detail within this memo.

1.1. A Note on the term IPFRR

The term IP fast re-route (IPFRR) was adopted by the IETF as the general name for best-effort Fast Re-route (FRR) in best effort IP and MPLS networks. This was to distinguish this new work from the then established FRR as described in [RFC4090] which uses RSVP Traffic Engineered (RSVP-TE) MPLS paths [RFC3936].

Within this document the terms IPFRR and FRR are used interchangeably.
2. PPR Overview

PPR works by injecting into the network a path or a graph and a corresponding forwarding identifier (PPR-ID). A node examines each PPR path description and if it is on the path it inserts into the Forwarding Information Database (FIB) an entry for the PPR-ID with the next hop as either the next entry along the PPR path, or if a loose path is specified, the next hop on the shortest path to the next hop along the PPR path. This is described in [I-D.chunduri-lsr-isis-preferred-path-routing].

PPR also has the ability to inject into a network a tree rooted at a node identified a PPR-ID. This is described in [I-D.ce-lsr-ppr-graph]. This graph mechanism provides a compact representation of a set of paths to a given PPR-ID. This works in a similar manner to the linear path case, in which a node on the graph inserts a FIB entry for the PPR-ID with the next hop as either the next node in the graph, or the next hop on the shortest path to the next node in the graph. Clearly the graph needs to be a spanning tree and must not contain a cycle.

In the description of the FRR methods provided in the text, the term encapsulation (and decapsulation) is frequently used in connection with the addition (and removal) of a PPR-ID to be used by the forwarding later to identify to the forwarders the PPR path that the packet needs to traverse to follow the repair path. Encapsulation is only one of a number of methods that can be used and is used in this memo as a convenience without loss of generality. For more information see Section 9.

3. Preferred Path LFA (pLFA) Deployment Advantages

PPR allows the construction of arbitrary engineered backup paths. In this respect it is like similar to RSVP-TE and Topology Independent Loop-Free Alternates (TI-LFA) [I-D.ietf-rtgwg-segment-routing-ti-lfa]. However, unlike those approaches PPR is applicable to any forwarding plane. For example, it is possible to support MPLS, both IPv4 and IPv6 and Ethernet.

Like Segment Routing (SR) [RFC8402], PPR uses extensions to the existing IGP, however, unlike SR, PPR requires no extension to the data plane. Again, unlike SR, which requires a Segment Identifier (SID) in the network layer header for every non-shortest path forwarding instruction, an arbitrary path does not require expansion of the user data packet beyond that needed for the initial insertion of the PPR-ID. This mitigates the MTU stress that SR introduces to the network.
PPR based IPFRR supports 100% failure coverage similar to RSVP-TE
[RFC4090], TI-LFA, Maximally Redundant Trees (MRT) [RFC7812] and Not-
Via [RFC6981]. It does not have the coverage restrictions that apply
to Loop-Free Alternate (LFA) [RFC5286] and Remote LFA (RLFA)
[RFC7490].

Shared Risk Link Groups (SRLGs) make it more difficult find repairs
in LFA and RLFA reducing repair coverage. TI-LFA can address this,
but only at a cost of expanding the number of SIDs and hence the
packet size.

Supporting multiple concurrent failures is difficult in all of the
IPRFF approaches except MRT, which can repair two concurrent
failures. However unlike MRT, which is constrained by its network
wide algorithm, PPR allows individual, arbitrary repair paths to
instantiated, for any failure.

In the current TI-LFA design, priority is given to repairing
connectivity rather than conforming to the operator traffic policy.
A PPR based FFR approach can apply policy to the repaired traffic,
including, if required multiple policies to an individual failure.

One of the main advantages of TI-LFA compared to other IPFRR
approaches is that it creates repair paths that are congruent with
the post convergence path from the Point of Local Repair (PLR)
[RFC4090] to the destination. These paths, which may be longer than
strictly necessary to reach Q-space [I-D.bryant-ipfrr-tunnels], stop
micro-loops from forming along the repair path during re-convergence.
PPR can also create these congruent paths without the need to
introduce SR into the network.

One of the limitations in TI-LFA, RLFA and LFA is that they do not
have a method of selectively creating alternative next-hops or indeed
full repair paths based on policy, or traffic engineering information
known to the operator. PPR provides a simple way to inject arbitrary
paths. It may therefore be used to enhance an existing LFA/RLFA/TI-
LFA IPFRR enabled network by selectively injecting paths to provide a
repair for business critical links with a policy in the PLR that
where provided a PPR path should be preferred over a local calculated
LFA based paths.

PPR is applicable to both centralized and PLR computed repair paths
each of which has advantages in different circumstances. A centrally
computed repair path only requires interaction with one network node
which then floods the instruction. This differs from the normal SDN
approach which requires interaction with all of the nodes along the
path and RSVP-TE which requires interaction with at least one end-
point of every repair.
Like TI-LFA, pLFA is based on a small extension to the IGP. It uses the IGP flooding mechanism and in-built state maintenance and consistency checks. This contrasts with RSVP-TE which needs its own separate Signaling and soft-state maintenance method.

The requirement that the pLFA solution addresses is thus the ability to construct repair paths that conform to operator policy without data-plane changes or significant MTU increase, and without introducing any control plane changes other than a small addition to the existing IGP.

A more detailed technical comparison between pLFA and the existing solutions is provided in the technical description of pLFA that follows.

4. Simple Repair Using pLFA

4.1. Link Repair

In this, the most basic, scenario Figure 1 we assume that we have a path A-B-C-D that the packet must traverse. This may be a normal best effort path or a traffic engineered path.

```
A---B---/---C---D
|      |    
E---F---G
```

Figure 1: Simple IPFRR Using pLFA

PPR is used to inject the repair path B->E->F->G->C into the network with a PPR-ID of c’. B is monitoring the health of link B->C, for example looking for loss-of-light, or using Bidirectional Forwarding Detection (BFD) [RFC5880]. When B detects a failure it encapsulates the packet to C by adding to the packet the PPR-ID for c’ and sending it to E. At C the packet is decapsulated and sent to D. The path C->E->F->G->C may be a traffic engineered path or it may be a best effort path. B may have at its disposal multiple paths to C with different properties for different traffic classes. In this case each path to be used would require its own PPR-ID (c’, c’’ etc).

In some circumstances, the repair path may be terminated at another point in Q-space or at a node between C and D. For example, in Figure 1 if all costs are 1, F is in Q-space with respect to a B->C failure (F->G->C cost = 2, whilst F->E->B->C cost = 3) and thus the packet can safely be encapsulated and send to F with a PPR-ID of f’. Releasing the packet early in Q-space has two advantages, firstly the
packet can take a shorter path to its destination if one is available rather than traveling to the far side of the failure and then back tracking.

Releasing a packet in Q-space also reduces the size of the PPR path that needs to be advertised, and potentially allows a repair path to be shared among a number of failures. For example in Figure 2 G with PPR-ID g’ via B->E->F->G can be used to provide an IPFRR path for the failure of both B->C and B->H.

```
  A---B--//--C---D
   |     |
   H--+
  E---F--G
     g'
```

Figure 2: Simple IPFRR Using pLFA With Shared Repair

Shared paths are useful in reducing the number of PPR paths that need to be flooded to support FRR.

Note that where the packet takes the shortest path to the point in Q-space that is closest to the destination, it will be taking a path that is congruent with the post convergence path from the PLR to the destination. This is the path that TI-LFA chooses to avoid its loop-free convergence. However this is not the only loop-free strategy available to a pLFA based solution.

4.2. Node Repair

Consider the network fragment shown in Figure 3 taken from [I-D.bryant-ipfrr-tunnels], and consider that node A needs to deal with the possible failure of node E.
Node S needs the use of four repair paths to address the failure of node E, one repair to each of E’s neighbours for which E is on the path to that neighbour. In Figure 3 there are three of these next-next-hop repairs, noted as Repair S-Sx in the figure. In addition a repair to E (Repair S->E) using a path other than along the path S-E> should be installed for traffic to E on the basis that the problem may be a failure of link S->E rather than a failure of the node.

The three repair paths to the next-next-hops of E can be installed as PPR path S-Sx with a PPR-ID of sx’. The link repair for E a PPR path to E which avoids link S-E with a PPR-ID of e’.

### 4.3. Shared Risk Link Groups

A shared risk link group (SRLG) is a set of links that are believed to have some systematic connection such that when one fails there is a high probability of all of them failing. This occurs, for example, where all of the members of the group run in a common cable duct. Where this relationship is known, and the simultaneous failure does not partition the network, PPR can install paths such that all members of the SRLG are avoided. pLFA has fewer constraints than other methods in constructing arbitrary repair paths in the network. [RFC6981] Section 6.1 describes the SRLG problem as it applies to IPFRR. pLFA can address all of the cases described in [RFC6981].

SRLG avoiding IPFRR paths can be complex. Since a packet can be attracted towards the failure whenever it is released from a strict path, the repair path may need a number of segments to steer it safely into Q-space. If this is done in the data-plane this can
stress the MTU. pLFA creates the path in the control plane and its 
encapsulation is invariant with respect to the complexity of the 
path. Furthermore, if the need to reduce the data-plane 
encapsulation side means that the repair path needs to use a sequence 
of loose hops it is necessary to determine the behaviour of each 
router on the chosen path. This contrasts with pLFA which can 
determine the path using whatever metrics and policy is appropriate, 
and then simply impose it without any data-plane overhead beyond that 
needed for a simple repair.

4.4. Local Area Networks

LANs are a special type of SRLG and are solved using the SRLG 
mechanisms outlined above. With all SRLGs, there is a trade-off 
between the sophistication of the fault detection and the size of the 
SRLG. [RFC6981] Section 6.2 describes the LAN problem as it applies 
to IPFRR. pLFA can address all of the cases described in [RFC6981].

4.5. Multiple Independent Failures

The Multiple Independent Failure cases described in [RFC6981] 
Section 6.3 will be analyzed in a future version of this document.

4.6. Multi-homed Prefixes

The Multi-Homed Prefix (MHP) problem is described in [RFC5286] 
Section 6.1, [RFC6981] Section 5.3 and [RFC8518]. MHP will be 
addressed in a future version of this document.

4.7. ECMP

Equal Cost Multi-Path (ECMP) is a consideration in any IPFRR method 
that does not use strict paths, and can be both an opportunity and 
threat. It is an opportunity in that it allows for the repair 
traffic to be distributed over a number of alternative paths to 
minimize congestion. If a loose pLFA path is injected into the 
network, then any available ECMP paths that fulfill the PPR path 
constraints can be installed following the same procedure used in 
normal IGP path computation.

However, care must be taken that a packet is not in a position where 
it is released from a repair at an ECMP point such that one of the 
ECMP paths is back via the failure. This can never happen if the 
correct definition of Q-space [RFC7490] is used in calculating the 
repair path.
5. Repair To A Traffic Engineered Alternate Path

In this approach there are two traffic engineered paths from A to D (Figure 4).

\[ A' \rightarrow B \rightarrow C' \rightarrow D \]

\[ E \rightarrow F \rightarrow G \rightarrow D \]

Figure 4: Traffic Engineered IPFRR Using pLFA

The primary path A->B->C->D is protected by a traffic engineered path A->F->G->D (PPR-ID d’) with traffic engineered connectors from B (B->F) and C (C->G). The path A->F->G->D and its connectors can be created and injected by any node with access to the IGP, but it is more likely to be created by a traffic engineering controller.

If link B->C fails, B re-routes packets destined for D to the traffic engineered path A->F->G->D via connector B->F. It does this by encapsulating the packet with a PPR-ID of d’.

Clearly there is nothing special needed to get the packet from B to F as they are adjacent but if there is a node say X on the path from B to F an explicit path needs to be created from B to F via X. Normally the repair would be created as a single PPR path (i.e. B->F->G->D) with a PPR-ID of d’. In this approach the repair from A would be A->F->G->D with a PPR-ID of d’ also. Similarly C-G-D would again share the PPR-ID d’.

If preferred the repair path could also be constructed using double encapsulation or using an SR approach in which the first segment was B-F with a PPR-ID/SID f’ and the second segment was F-D with PPR-ID of d’.

In the example shown in Figure 4 the proposed B-//->C protection path was B->F->G->D. This is node protecting on C since the repair path avoids C. Although link failures tend to be more common than node failures some critical applications would prefer node protection where possible. Node avoidance may not be possible within the network, and may come at a cost of increased path repair path length. However, whether to include node protection and as what cost to accept its inclusion is a matter of network operator policy.

The repair constructed in this section required the inclusion of a set of PPR defined links to construct the repair. PPR has the ability to construct graphs [I-D.ce-1sr-ppr-graph] which can simplify
the specification of the required repair topology. This is discussed in Section 6.

6. Use of a Repair Graph

PPR has the ability to inject graphs into a network as well as linear paths [I-D.ce-lsr-ppr-graph]. PPR graphs specify the paths from a set of nodes to a single node, and are a compact method of representing a set of paths to that destination with shared properties.

6.1. Single Repair Graph

In [I-D.ce-lsr-ppr-graph] the S bit in the PPR Path Description Element (PDE) specifies that a network node is a Source and a D bit specifies that it is a destination. A graph with all S bits set on the leaves and a D bit on the root is a unidirectional tree.

```
    d'
   S  S  S   D
A-??-B--??--C--??-D
|    |      |     |
E----F------G-----+
```

Figure 5: pLFA using PPR Graphs

Consider the network fragment shown in Figure 5. A graph with a PPR-ID d’ is constructed attaching each of the nodes A, B, and C to D. Should any of the nodes A, B or C fail the packet can be forwarded on the PPR graph to D with the PPR-ID of d’. In the unidirectional repair graph A, B, and C are all sources (signaled with the S bit set), and D is the only destination (signaled with the D bit set).

6.2. Multiple Disjoint Graphs

Consider Figure 1 from [RFC7812] which illustrates the problem of IPFRR in a network that is 2-connected.
Figure 6: A 2-Connected Network

Figure 6(a) is the full network, and Figure 6(b) and (b) are two corresponding redundant trees from [RFC7812]. Using the Red and Blue trees towards R every node has at least two paths to R. We give R a PPR-ID of r’ in the Blue tree and a PPR-ID of r” in the Red tree. R is the only destination in the PPR graph (D bit set), but all other nodes are sources (S bit set). For clarity this bit setting is not shown in Figure 6.

It is worth noting what happens at nodes B and D in Figure 6(b). B is an ECMP to D via F and C. What happens at node B is a matter of implementation and operator preference. Either B can choose one of the next-hops, or it use them as an ECMP pair. It can also use the availability of the pair to protect against B->F or B->C being an unexpected SRLG with respect to link A->R. D is a merge point for traffic destined for r’ arriving from F and from C. It simply forwards the traffic to r’ as normal. Similarly in Figure 6(c) D can sent traffic to r” via F or C.

Whilst in this example the Red and Blue trees use exactly the same links and nodes used by the main topology, a repair graph could use available nodes and links outside this network fragment.

Now consider Figure 2 from [RFC7812] which illustrates the problem of IPFRR in a network that is not 2-connected.
(a) a graph that is not 2-connected

\[
\begin{array}{c}
\text{[E]} \rightarrow \text{[D]} \rightarrow \text{[J]} \\
\text{[R]} \rightarrow \text{[F]} \rightarrow \text{[C]} \rightarrow \text{[G]} \\
\text{[A]} \rightarrow \text{[B]} \rightarrow \text{[H]}
\end{array}
\]

(b) Blue Tree towards R

(c) Red Tree towards R

Figure 7: A Network That Is Not 2-Connected

Again there are two paths (with PPR-IDs $r'$ and $r''$) to R from all nodes except that G, J and H all depend on link G->C and node C which is a single point of failure in the network.

Again note that B in the Blue tree and D in the Red tree has two paths to $r'$ and $r''$ respectively that it may use according to configuration or preference.

7. Centralized and Decentralized Approaches

pLFA paths can be established through both centralized and decentralized approaches.

A centralized system has a more holistic view of the network and its policies, its resource constraints and resource usage. A decentralized system is inherently more resilient to failure and is a good fit where the network is a simple best effort network as is commonly deployed.

A centralized system gathers the network state, just as any SDN system does, and computes the FRR paths needed. However, unlike normal SDN operation where the controller needs to individually instruct every entity on the path for every path, in a PPR network it is only necessary to inject the PPR path at one point. In practice, for reliability, it would inject the PPR paths in a small number of places, and the naturally reliability of the IGP would ensure the
complete distribution of the paths. Furthermore, the system collecting the network state would naturally send the PPR LSPs back to the SDN controller providing quality assurance that the FRR paths had been distributed.

In a decentralized approach the pLFA path is computed within the network, normally by the PLR. Further details of this approach will be provided in a future version of this document.

8. Independence of operation

Each PPR path is independent of all other paths in the network. This means that there is no constraint on how the path is calculated, and a different algorithm can be used on every path. Some of the other FRR approaches have this property, but not all. For example LFA is constrained by the properties of the base IGP as to a large degree is RLFA. PPR can incorporate best effort segments if required, but from a data-plane perspective there is no advantage in doing so. In this case there is a dependence on the path choice in the base routing protocol.

MRT and Not-Via can use any algorithm to calculate the repair, but it needs to be common across the network, although the expectation in the case of Not-Via is that the algorithm would be a Dijkstra based SPF calculation. In both these cases to change the algorithm would require turning off FRR for the whole network, re-configuring and then restarting FRR.

RSVP-TE based FRR can specify any path, but at the cost of maintaining the soft-state.

A PLR in a TI-LFA or any SR based approach can also compute paths independent of each other, but they tend to need to do this as a concatenation of a series of shortest paths in order to reduce the number of SIDs they need to form the path. TI-LFA is thus highly dependent on the underlying best effort paths.

pLFA can be used as a method of converting classic LFA or RLFA to full coverage by providing the paths that these methods are unable to support, or to provide any the sub-paths needed to reduce the number of TI-LFA SIDs.

9. Data-plane Considerations

This section is a survey of a number of data-planes in each case considering how a PPR-ID could added to map the packet to required FRR path.
9.1. Traditional IP

Where the data-plane is "traditional" IP the user packet needs to be encapsulated such that the outer IP address is the PPR-ID. Any preferred encapsulation can be used such as: IP in IP, IP in GRE, or IP in UDP.

The tunnel capabilities of a node can be advertised using the method described in [I-D.xu-isis-encapsulation-cap] allowing different tunnel types to be used for different PPR paths, depending on the capability of the various nodes in the network.

A common operational issue with this type of encapsulation for IPFRR has been the shortage of IP addresses. However this is not an issue in an IPv6 network.

9.2. Segment Routing over an IPv6 Data Plane (SRv6)

Where the data-plane is SRv6 [I-D.ietf-6man-segment-routing-header] pLFA would be used to steer a packet towards the next segment endpoint. Clearly an extra level of IP encapsulation could be used Section 9.1, but that expands the packet by adding at least 36 octets.

Where the packet is a "traditional" IP packet, and the repair endpoint is SRv6 capable, an alternative to the methods described in Section 9.1 is to insert an SRH into the IP packet setting the SID in the SRH to the original packet DA and replacing the outer DA with the PPR-ID. If this method is used the semantics of the PPR-ID must include the reconstruction of the packet, by replacing the DA with the original DA retrieved from the SRH and the removal of the SRH.

9.3. MPLS

Where the data-plane is MPLS any encapsulation needed is tiny (a label push), but the exact action depends on the repair strategy, and there is the usual FRR problem of the setting of the new value for the top label prior to pushing the PPR-ID label.

Where the FRR path terminates at an MPLS node other than the network egress provider edge (PE) in the type of pLFA repair described in Section 4, the original top label needed to be set to the label the node was expecting.

Consider the network fragment shown in Figure 1. This is straight forward case because node B swaps the top label the label it would have used without the failure and then pushes the label that corresponds to c'. If the repair strategy had been to exit Q-space...
at the earliest opportunity for example at F, then B would have needed to know what label F required to reach the destination. A very similar problem occurs when a node repair is undertaken Figure 3, where S needs to know the label that the next-next-hops (S1, S2 and S3) need to reach the destination.

Where the traffic is being moved to a new path terminating at the egress PE as shown in Figure 4, the problem much simpler and only requires the swapping of the top label with the label that represents d’.

10. Loop Free Convergence

Whilst IPFRR puts in place a temporary network repair, eventually the network needs to re-converge around the surviving network components. During this phase there is a danger that micro-loops will form and disrupt the traffic flowing across the network. A similar problem can occur when the failed component returns to service, or when a new component is introduced into the network. [RFC5715] describes the problem of loop-free convergence in detail and examines the methods known at the time of its writing. Since that time [RFC8333] has proposed a timer based loop mitigation (but not elimination) process, and [I-D.ietf-rtgwg-segment-routing-ti-lfa] has proposed that by making the IPFRR path congruent with the post convergence path loops can be eliminated along the repair path. However whilst these mitigation techniques address component failure, neither are targeted at the repair/new component case.

These problems only effect best effort paths and path segments, fully defined paths do not have this problem.

A network using pLFA is compatible with all of the know loop-free convergence and loop mitigation approaches.

11. OAM Considerations

PPR may also be used in a way that provides an alternative to running multi-hop BFD from ingress on a traffic engineered (TE) path with reducing the complexities that arise from echo reply false alarms. In this use case pLFA works by locally detecting the failure and transferring the traffic to preferred TE backups which are in time replaced by the newly computed TE paths to the same PPR-ID.

12. Privacy Considerations

As noted in Section 13 pLFA paths are constrained by the routing domain and thus the traffic will be no more subject to observation than it would in normal operation. Indeed PPR has the capability to
13. Security Considerations

The security considerations of PPR are discussed in [I-D.chunduri-lsr-isis-preferred-path-routing] which in turn refers the reader to the security considerations of the underlying routing protocol and the data-plane in use. The pLFA application of PPR to IPFRR introduces no additional security regarding PPR itself.

General IPFRR security considerations are discussed in [RFC5714] and these apply to this solution.

One further consideration, is the whether policy that applied to the original path needs to be applied to the repair path. The decision is operator and application specific, however pLFA is better than some other IPFRR solution in that it is possible to precisely choose the repair path.

IPFRR is deployed within the scope of the routing protocol that underpins it which limits the security vulnerability. Furthermore it is unlikely that IPFRR would be deployed outside a well managed network. These restrictions in-turn significantly mitigate any security threat.

14. IANA Considerations

This document makes no IANA requests.

15. References

15.1. Normative References

[I-D.ce-lsr-ppr-graph]

[I-D.chunduri-lsr-isis-preferred-path-routing]
15.2. Informative References

[I-D.bryant-ipfrr-tunnels]

[I-D.ietf-6man-segment-routing-header]

[I-D.ietf-rtgwg-segment-routing-ti-lfa]

[I-D.xu-isis-encapsulation-cap]


Internet-Draft                    pLFA                         July 2019


Authors' Addresses

Stewart Bryant
Futurewei Technologies Inc

Email: stewart.bryant@gmail.com

Uma Chunduri
Futurewei Technologies Inc

Email: uchunduri@futurewei.com
Abstract

This document specifies the Simple Control Plane (CP) and User Plane (UP) Separation Broadband Network Gateway (BNG) control channel Protocol (S-CUSP) for communications between a CP and a UP. S-CUSP is designed to be flexible and extensible so as to easily allow for the addition of further messages and data items to meet future requirements.

Status of This Memo

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

Distribution of this document is unlimited. Comments should be sent to the authors or the RGTWG working group mailing list: rtgwg@ietf.org.

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."
# Table of Contents

1. Introduction............................................6

2. Terminology.............................................7
   2.1 Implementation Requirement Keywords..............7
   2.2 Terms................................................7

3. BNG CUPS Overview......................................10
   3.1 BNG CUPS Motivation....................................10
   3.2 BNG CUPS Architecture Overview.....................10
   3.3 BNG CUPS Interfaces...................................12
       3.3.1 Service Interface................................13
       3.3.2 Control Interface................................14
       3.3.3 Management Interface............................14
   3.4 BNG CUPS Procedure Overview.........................14

4. S-CUSP Protocol Overview...............................18
   4.1 Control Channel Related Procedures...............18
       4.1.1 S-CUSP Session Establishment....................18
       4.1.2 Keep Alive........................................19
   4.2 Node Related Procedures.............................20
       4.2.1 UP Resource Report................................20
       4.2.2 Update BAS Function on Access Interface........21
       4.2.3 Update Network Routing............................21
   4.2.4 CGN Public IP Address Allocation..................22
   4.2.5 Data Synchronization between the CP and UP.........23
   4.3 Subscriber Session Related Procedures...............24
       4.3.1 Create Subscriber Session..........................25
       4.3.2 Update Subscriber Session..........................26
       4.3.3 Delete Subscriber Session..........................27
       4.3.4 Subscriber Session Events Report................27

5. S-CUSP Call Flows......................................29
   5.1 IPoE................................................29
       5.1.1 DHCPv4 Access....................................29
       5.1.2 DHCPv6 Access....................................30
       5.1.3 IPv6 SLAAC Access................................32
       5.1.4 DHCPv6 + SLAAC Access.............................33
       5.1.5 DHCP Dual Stack Access............................35
   5.1.6 L2 Static Subscriber Access.......................37
   5.2 PPPoE...............................................40
       5.2.1 IPv4 PPPoE Access..................................40
       5.2.2 IPv6 PPPoE Access..................................41
       5.2.3 PPPoE Dual Stack Access...........................43
   5.3 WLAN Access...........................................45
   5.4 L2TP................................................47
       5.4.1 L2TP LAC Access...................................47
       5.4.2 L2TP LNS IPv4 Access...............................49
       5.4.3 L2TP LNS IPv6 Access...............................51
   5.5 CGN (Carrier Grade NAT)..............................54
Table of Contents (continued)

5.6 L3 Leased Line Access......................................55
5.6.1 Web Authentication........................................55
5.6.2 User Traffic Trigger.......................................57
5.7 Multicast Access.............................................58

6. S-CUSP Message Formats......................................60
6.1 Common Message Header......................................60
6.2 Control Messages.............................................61
6.2.1 Hello Message...............................................61
6.2.2 Keepalive Message..........................................62
6.2.3 Sync_Request Message.....................................62
6.2.4 Sync_Begin Message........................................62
6.2.5 Sync_Data Message..........................................63
6.2.6 Sync_End Message...........................................63
6.2.7 Update_Request Message...................................64
6.2.8 Update_Response Message...................................64
6.3 Event Message..................................................65
6.4 Report Message................................................66
6.5 CGN Messages....................................................66
6.5.1 Addr_Allocation_Req Message..............................66
6.5.2 Addr_Allocation_Ack Message..............................66
6.5.3 Addr_Renew_Req Message....................................67
6.5.4 Addr_Renew_Ack Message....................................67
6.5.5 Addr_Release_Req Message..................................67
6.5.6 Addr_Release_Ack Message..................................67
6.6 Vendor Message................................................67
6.7 Error Message..................................................68

7. S-CUSP TLVs and Sub-TLVs....................................69
7.1 Common TLV Header.............................................69
7.2 Basic Data Fields.............................................70
7.3 Sub-TLV Format and Sub-TLVs...............................71
7.3.1 Name sub-TLVs.............................................71
7.3.2 Ingress-CAR sub-TLV........................................72
7.3.3 Egress-CAR sub-TLV.........................................72
7.3.4 If-Desc sub-TLV............................................73
7.3.5 IPv6 Address List sub-TLV.................................75
7.3.6 Vendor sub-TLV.............................................75
7.4 The Hello TLV..................................................77
7.5 The Keep Alive TLV............................................78
7.6 The Error Information TLV....................................79
7.7 BAS Function TLV.............................................79
7.8 Routing TLVs..................................................82
7.8.1 IPv4 Routing TLV...........................................82
7.8.2 IPv6 Routing TLV...........................................84
7.9 Subscriber TLVs...............................................85
7.9.1 Basic Subscriber TLV.......................................86
7.9.2 PPP Subscriber TLV.........................................88
7.9.3 IPv4 Subscriber TLV........................................89
Table of Contents (continued)

7.9.4 IPv6 Subscriber TLV........................................90
7.9.5 IPv4 Static Subscriber Detect TLV........................91
7.9.6 IPv6 Static Subscriber Detect TLV........................93
7.9.7 L2TP-LAC Subscriber TLV..................................94
7.9.8 L2TP-LNS Subscriber TLV..................................95
7.9.9 L2TP-LAC Tunnel TLV.....................................95
7.9.10 L2TP-LNS Tunnel TLV.....................................96
7.9.11 Update Response TLV.....................................97
7.9.12 Subscriber Policy TLV.....................................98
7.9.13 Subscriber CGN Port Range TLV..........................100
7.10 Device Status TLVs..........................................100
7.10.1 Interface Status TLV.......................................101
7.10.2 Board Status TLV..........................................101
7.11 CGN TLVs....................................................102
7.11.1 Address Allocation Request TLV..........................102
7.11.2 Address Allocation Response TLV........................103
7.11.3 Address Renewal Request TLV.............................104
7.11.4 The Address Renewal Response TLV.......................105
7.11.5 Address Release Request TLV.............................106
7.11.6 The Address Release Response TLV.......................106
7.12 Event TLVs................................................107
7.12.1 Subscriber Traffic Statistics TLV........................108
7.12.2 Subscriber Detection Result TLV........................109
7.13 Vendor TLV................................................110

8. Implementation Status........................................112
8.1 Implementations.............................................112
8.1.1 Huawei Technologies.....................................112
8.1.2 ZTE................................................................113
8.1.3 H3C..........................................................113
8.2 Hackathon......................................................113
8.3 EANTC Testing................................................114

9. IANA Considerations............................................115
9.1 Message Types.................................................115
9.2 TLV Types.......................................................115
9.3 TLV Operation Codes..........................................117
9.4 Sub-TLV Types................................................118
9.5 Error Codes....................................................118

10. Security Considerations......................................120

Contributors........................................................121

Normative References.............................................122
Informative References..........................................123

Authors’ Addresses...............................................125
1. Introduction

A fixed network Broadband Network Gateway (BNG) is an Ethernet-centric IP edge router, and the aggregation point for user traffic. To provide centralized session management, flexible address allocation, high scalability for subscriber management capacity, and cost-efficient redundancy, the Control/User (CU) separated BNG framework is described in [TR-384]. The CU separated service Control Plane (CP), which is responsible for user access authentication and setting forwarding entries in User Planes (UPs), can be virtualized and centralized. The routing control and forwarding plane, i.e. the BNG user plane (local), can be distributed across the infrastructure. Other structures can also be supported such as both CP and UP being virtual or both being physical.

This document specifies the Simple CU Separation BNG control channel Protocol (S-CUSP) for communications between a BNG Control Plane (CP) and a set of User Planes (UPs). S-CUSP is designed to be flexible and extensible so as to easily allow for additional messages and data items, should further requirements be expressed in the future.
2. Terminology

This section specifies implementation requirement keywords and terms used in this document. S-CUSP messages are described in this document using Routing Backus-Naur Form (RBNF) as defined in [RFC5511].

2.1 Implementation Requirement Keywords

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

2.2 Terms

This section specifies terms used in this document.


ACK: Acknowledgement message.

BAS: Broadband Access Server (BRAS, BNG).

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

BRAS: BRoadband Access Server (BNG).

CAR: Committed Access Rate.

CBS: Committed Burst Size.

CGN: Carrier Grade NAT.

Ci: Control Interface.

CIR: Committed Information Rate.

CoA: Change of Authorization.
CP: Control Plane.

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

CPE: Customer Premises Equipment.

CU: Control-plane / User-plane.

CUSP: Control and User plane Separation Protocol.

DEI: Drop Eligibility Indicator. A bit in a VLAN tag after the priority and before the VLAN ID. (This bit was formerly the CFI (Canonical Format Indicator).) [802.1Q]

DHCP: Dynamic Host Configuration Protocol [RFC2131].

dial-up: This refers to the initial connection messages when a new user appears. The name is left over from when users literally dialed up on a modem equipped phone line but herein is applied to other initial connection techniques. Initial connection is frequently indicated by the receipt of packets over PPPoE [RFC2516] or IPoE.

EMS: Element Management System.

IPoE: IP over Ethernet.

L2TP: Layer 2 Tunneling Protocol [RFC2661].

LAC: L2TP Access Concentrator.

LNS: L2TP Network Server.

MAC: 48-bit Media Access Control address [RFC7042].

MANO: Management and Orchestration.

Mi: Management Interface.

MSS: Maximum Segment Size.

MRU: Maximum Receive Unit.

NAT: Network Address Translation [RFC3022].

ND: Neighbor Discovery.

NFV: Network Function Virtualization.
NFVI: NFV Infrastructure
PBS: Peak Burst Size.
PD: Prefix Delegation.
PIR: Peak Information Rate.
PPP: Point to Point Protocol [RFC1661].
PPPoE: PPP over Ethernet [RFC2516].
RBNF: Routing Backus-Naur Form [RFC5511].
RG: Residential Gateway.
Si: Service Interface.
TLV: Type, Length, Value. See Sections 7.1 and 7.3.
UP: User Plane. UP is a network edge and user policy implementation component. The traditional router’s Control Plane and Forwarding Plane are both preserved on BNG devices in the form of a user plane.
URPF: Unicast Reverse Path Forwarding.
User: Equivalent to "customer" or "subscriber".
VRF: Virtual Routing and Forwarding.
3. BNG CUPS Overview

3.1 BNG CUPS Motivation

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

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

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

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

To address these challenges for fixed networks, the framework for a cloud-based BNG with Control Plane and User Plane (CU) separation is described in [TR-384]. The main idea of CU separation is to extract and centralize the user management functions of multiple BNG devices, forming a unified and centralized Control Plane (CP). And the traditional router’s Control Plane and Forwarding Plane are both preserved on BNG devices in the form of a User Plane (UP).

3.2 BNG CUPS Architecture Overview

The functions in a traditional BNG can be divided into two parts: one is the user access management function, the other is the router function. The user management function can be centralized and deployed as a concentrated module or device, called the BNG Control Plane (BNG-CP). The other functions, such as the router function and forwarding engine, can be deployed in the form of the BNG User Plane (BNG-UP).
The following figure shows the architecture of CU separated BNG:

![Figure 1: Architecture of CU Separated BNG](image)

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

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

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

The Control Plane is responsible for the following:

1. Address management: unified address pool management and CGN subscriber address traceability management.

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

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

4. Access management: process user dial-up packets, such as PPPoE, DHCP, L2TP, etc.

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

The User Plane is responsible for the following:

1. Routing control functions: responsible for constructing routing forwarding plane (e.g., routing, multicast, MPLS, etc.).

2. Routing and Service Forwarding plane functions: responsible including traffic forwarding, QoS and traffic statistics collection.

Subscriber detection: responsible for detecting whether a subscriber is still online.

3.3 BNG CUPS Interfaces

To support the communication between the Control Plane and User Plane, three interfaces are assumed. These are referred to as the Service Interface (Si), Control Interface (Ci), and Management Interface (Mi) as shown in Figure 2.
3.3.1 Service Interface

For a traditional BNG (without CU separation), the user dial-up signals are terminated and processed by the control plane of a BNG. When the CP and UP of a BNG are separated, there needs to be a way to relay these signals between the CP and the UP.

The Service Interface (Si) is used to establish tunnels between the CP and UP. The tunnels are responsible for relaying the PPPoE, IPoE, and L2TP related control packets that are received from a Residential Gateway (RG) over those tunnels. An appropriate tunnel type is VXLAN [RFC7348].

The detailed definition of Si is out of scope for this document.
3.3.2 Control Interface

The CP uses the Control Interface to deliver subscriber session states, network routing entries, etc. to the UP (see Section 6.2.7). The UP uses this interface to report subscriber service statistics, subscriber detection results, etc. to the CP (see Sections 6.3 and 6.4). A carrying protocol for this interface is specified in this document.

3.3.3 Management Interface

NETCONF [RFC6241] is the protocol used on the Management Interface between a CP and UP. It is used to configure the parameters of the Control Interface, Service Interface, the Access interfaces and QoS/ACL Templates. It is expected that implementations will make use of existing YANG models where possible, but that new YANG models specific to S-CUSP will need to be defined. The definitions of the parameters are out of scope for this document.

3.4 BNG CUPS Procedure Overview

The following numbered sequences (Figure 3) gives a high level view of the main BNG CUPS procedures.

<table>
<thead>
<tr>
<th>RG</th>
<th>Online Req</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>------------&gt;</td>
<td>Establish S-CUSP Channel</td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Report Board</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>interface</td>
<td>information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>------to CP via Ci------&gt;</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>-----to CP via Si------&gt;</td>
<td>Relay the Online Req</td>
<td></td>
</tr>
<tr>
<td></td>
<td>------on UP via Ci------&gt;</td>
<td>Update BAS function request / response</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;&lt;------ via Ci--------&gt;</td>
<td>Update network routing request / response</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>&lt;&lt;-------- via Ci--------&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>&lt;-------- on UP via Ci------&gt;</td>
<td>Update BAS function request / response</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>------to CP via Ci------&gt;</td>
<td>Report Board interface information</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>------------------------&gt;</td>
<td>Establish S-CUSP Channel</td>
<td></td>
</tr>
</tbody>
</table>
Figure 3: BNG CUPS Procedures Overview

1. S-CUSP session establishment: This is the first step of BNG CUPS procedures. Once the Control Interface parameters are configured on a UP, it will start to setup S-CUSP sessions with the specified CPs. The detailed definition of S-CUSP session establishment can be found in Section 4.1.1.

2. Board and interface report: Once the S-CUSP session is established between the UP and a CP, the UP will report status information on the boards and subscriber side interfaces of this UP to the CP. A board can also be called a Line/Service Process.
Unit (LPU/SPU) card. The subscriber side interfaces refer to the interfaces that connect the Access Network nodes (e.g., OLT: Optical Line Terminal, DSLAM: Digital Subscriber Line Access Multiplexer, etc.). The CP can use this information to enable the Broadband Access Service (BAS) function (e.g., IPoE, PPoE, etc.) on the specified interfaces. See Sections 4.2.1 and 7.10 for more details on Resource reporting.

3. BAS (Broadband Access Service) function enable: To enable the BAS function on the specified interfaces of a UP.

4. Subscriber network route advertisement: The CP will allocate one or more IP address blocks to a UP. Each address block contains a series of IP addresses. Those IP addresses will be allocated to subscribers who are dialing up from the UP. To enable other nodes in the network to learn how to reach the subscribers, the CP needs to notify the UP to advertise to the network the routes that can reach those IP addresses.

5. 5.1-5.6 is a complete call flow of a subscriber dial-up process. When a UP receives a dial-up request, it will relay the request packet to a CP through the Service Interface. The CP will parse the request. If everything is OK, it will send an authentication request to the AAA server to authenticate the subscriber. Once the subscriber passes the authentication, the AAA server will return a positive response to the CP. Then the CP will send the dial-up response packet to the UP and the UP will forward the response packet to the subscriber (RG). At the same time, the CP will create a subscriber session on the UP, which enables the subscriber to access the network. For different access types, the process may be a bit different. But the high-level process is similar. For each access type, the detail process can be found in Section 5.

6. 6.1-6.3 is the sequence when updating an existing subscriber session. The AAA server initiates a Change of Authorization (CoA) and sends the CoA to the CP. The CP will then update the session according to the CoA. See Section 4.3.2 for more detail on CP messages updating UP tables.

7. 7.1-7.5 is the sequence for deleting an existing subscriber session. When a UP receives an offline request, it will relay the request to a CP through the Service Interface. The CP will send back a response to the UP through the Service Interface. The UP will then forward the offline response to the subscriber. Then the CP will delete the session on the UP through the Control Interface.
8. Event reports include the following two parts (more detail can be found in Section 4.3.4) Both are reported using the Event message.

8.1 Subscriber Traffic Statistics Report
8.2 Subscriber Detection Result Report

9. Data synchronization: See Sections 4.2.5 for more detail on CP and UP Synchronization.

10. CGN address allocation: See Sections 4.2.4 for more detail on CGN Address Allocation.
4. S-CUSP Protocol Overview

4.1 Control Channel Related Procedures

4.1.1 S-CUSP Session Establishment

A UP is associated with a CP and is controlled by that CP. In the case of a hot-standby or cold-standby, a UP is associated with two CPs, one called the Master CP and the other called the Standby CP. The association between a UP and its CPs is implemented by dynamic configuration.

Once a UP knows its CPs, the UP starts to establish S-CUSP sessions with those CPs as shown in Figure 4.

```
UP                                 CP
<p>| |
|                                 |
| TCP Session Establishment       |
|--------------------------------&gt;|-------------------------|</p>
<table>
<thead>
<tr>
<th>HELLO (version, capability)</th>
<th>HELLO (version, capability)</th>
</tr>
</thead>
</table>
```

Figure 4: S-CUSP Session Establishment

The S-CUSP session establishment consists of two successive steps:

1. Establishment of a TCP [RFC793] connection (3-way handshake) between the CP and the UP using a configured port from the dynamic port range (49152-65535).

2. Establishment of a S-CUSP session over the TCP connection.

Once the TCP connection is established, the CP and the UP initialize the S-CUSP session during which the version and Keepalive timers are negotiated.

The version information (Hello TLV, see Section 7.4) is carried within Hello messages (see Section 6.2.1). A CP can support multiple versions, but a UP can only support one version. So, the version negotiation is based on whether a version can be support by both the CP and the UP. For a CP or UP, if a Hello message is received that
does not indicate a version supported by both, a subsequent Hello message with an Error Information TLV will be sent to the peer to notify the peer of the "Version-Mismatch" error and the session establishment phase fails.

Keepalive negotiation is performed by carrying a Keepalive TLV in the Hello message. The Keepalive TLV includes a Keepalive timer and Dead Timer field. The CP and UP have to agree on the Keepalive Timer and Dead Timer. Otherwise, a subsequent Hello message with an Error Information TLV will be sent to its peer and the session establishment phase fails.

The S-CUSP session establishment phase fails if the CP or UP disagree on the version and keepalive parameters or if one of the CP or UP does not answer after the expiration of the Establishment timer. When the S-CUSP session establishment fails, the TCP connection is promptly closed. Successive retries are permitted but an implementation SHOULD make use of an exponential back-off session establishment retry procedure.

The S-CUSP session timer values that need to be configured are summarized in the table below.

<table>
<thead>
<tr>
<th>Timer Name</th>
<th>Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establishment</td>
<td>1-32767</td>
<td>45</td>
</tr>
<tr>
<td>Keepalive</td>
<td>0-255</td>
<td>30</td>
</tr>
<tr>
<td>DeadTimer</td>
<td>1-32767</td>
<td>4 * Keepalive</td>
</tr>
</tbody>
</table>

4.1.2 Keep Alive

Once an S-CUSP session has been established, a UP or CP may want to know that its S-CUSP peer is still available for use.

Each end of a S-CUSP session runs a Keepalive timer. It restarts the timer every time it sends a message on the session. When the timer expires, it sends a Keepalive message.

The ends of the S-CUSP session also run DeadTimers, and they restart the timers whenever a message is received on the session. If one end of the session receives no message after the DeadTimer expires, it declares the session dead. The session will be closed.

The minimum value of the Keepalive timer is 1 second, and it is specified in units of 1 second. The RECOMMENDED default value is 30 seconds. The timer may be disabled by setting it to zero.
The recommended default for the DeadTimer is 4 times the value of the Keepalive timer used by the remote peer. This implies there is essentially no risk of TCP congestion due to excessive Keepalive messages.

The Keepalive timer and DeadTimer are initially negotiated through the Keepalive TLV carried in the Hello Message.

4.2 Node Related Procedures

4.2.1 UP Resource Report

Once an S-CUSP session has been established between a CP and an UP. The UP reports the information of the Boards and access side interfaces on this UP to the CP as shown in Figure 5. Report messages are unacknowledged and are assumed to be delivered because the session runs over TCP.

The CP can use that information to activate/enable the Broadband Access Service (BAS) functions (e.g., IPoE, PPPoE, etc.) on the specified interfaces.

In addition, the UP resource report may trigger a UP warm-standby process. In the case of warm-standby, a failure on an UP may trigger the CP to start a warm-standby process, by moving the on-line subscriber sessions to a standby UP and then direct the affected subscribers to access the Internet through the standby UP.

```
<table>
<thead>
<tr>
<th>UP</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report Board Status</td>
<td>------to CP via Ci------&gt;</td>
</tr>
<tr>
<td>Report Interface Status</td>
<td>------to CP via Ci------&gt;</td>
</tr>
</tbody>
</table>
```

Figure 5: UP Board and Interface Report

Board status information is carried in the Board Status TLV (Section 7.10.2) and Interface status information is carried in Interface Status TLV (Section 7.10.1). Both Board and Interface Status TLVs are carried in the Report Message (Section 6.4).
4.2.2 Update BAS Function on Access Interface

Once the CP collects the interface status of a UP, it will activate/de-activate/modify the BAS functions on specified interfaces through the Update_Request and Update_Response message (Section 6.2) exchanges carrying the BAS Function TLV (Section 7.7).

```
<table>
<thead>
<tr>
<th>UP</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update BAS function</td>
<td></td>
</tr>
<tr>
<td>Request</td>
<td></td>
</tr>
<tr>
<td>&lt;-----on UP via Ci------</td>
<td></td>
</tr>
<tr>
<td>Update BAS function</td>
<td></td>
</tr>
<tr>
<td>Response</td>
<td></td>
</tr>
<tr>
<td>-----on UP via Ci-------</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 6: Update BAS Function

4.2.3 Update Network Routing

The CP will allocate one or more address blocks to a UP. Each address block contains a series of IP addresses. Those IP addresses will be allocated to subscribers who are dialing up to the UP. To enable the other nodes in the network to learn how to reach the subscribers, the CP needs to install the routes on the UP and notify the UP to advertise the routes to the network.

```
<table>
<thead>
<tr>
<th>UP</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscriber network route</td>
<td></td>
</tr>
<tr>
<td>update request</td>
<td></td>
</tr>
<tr>
<td>&lt;-------- via Ci--------</td>
<td></td>
</tr>
<tr>
<td>Subscriber network route</td>
<td></td>
</tr>
<tr>
<td>update response</td>
<td></td>
</tr>
<tr>
<td>-----via Ci-------------</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 7: Update Network Routing

The subscriber network routing update request and response are achieved through the Update Request and Response Message exchanges by carrying the IPv4/IPv6 Routing Information TLVs (Section 7.8).
4.2.4 CGN Public IP Address Allocation

The following sequences describe the CGN address management related procedures. Three independent procedures are defined, one each for CGN address allocation request/response, CGN address renewal request/response, and CGN address release request/response.

CGN address allocation/renew/release procedures are designed for the case where the CGN function is running on the UP. The UP has to map the subscriber private IP addresses to a public IP addresses, and such mapping is performed by the UP locally when a subscriber dials-up. That means the UP has to ask for public IPv4 address blocks for CGN subscribers from the CP.

In addition, when a public IP address is allocated to a UP, there will be a lease time (e.g., one day). Before the lease time expires, the UP can ask for renewal of the IP address lease from the CP. It is achieved by the exchange of the Addr_Renew_Req and Addr_Renew_Ack messages.

If the public IP address will not be used anymore, the UP SHOULD release the address by sending an Addr_Release_Req message to the CP.

If the CP wishes to withdraw addresses that it has previously leased to a UP, it uses the same procedures as above. The "Oper" code in the IPv4/IPv6 Routing TLV (see Section 7.1) determines whether the request is an update or withdraw.

The relevant messages are defined in Section 6.5.
4.2.5 Data Synchronization between the CP and UP

For a CU separated BNG, the UP will continue to function using the state that has been installed in it even if the CP fails or the session between the UP and CP fails.

Under some circumstances it is necessary to synchronize state between the CP and UP, for example if a CP fails and the UP is switched to a different CP.

Synchronization includes two directions. One direction is from UP to CP; in that case, the synchronization information is mainly about the board/interface status of the UP. The other direction is from CP to UP; in that case, the subscriber sessions, subscriber network routes, L2TP tunnels, etc. will be synchronized to the UP.

The synchronization is triggered by a Sync_Request message, to which the receiver will (1) reply with a Sync_Begin message to notify the requester that synchronization will begin, and (2) then start the synchronization using the Sync_Data message. When synchronization finished, a Sync_End message will be sent.
The following figure shows the process of data synchronization between a UP and a CP.

1) Synchronization from UP to CP

2) Synchronization from CP to UP

Figure 9: Data Synchronization

4.3 Subscriber Session Related Procedures

A subscriber session consists of a set of forwarding states, policies, and security rules that are applied to the subscriber. It is used for forwarding subscriber traffic in a UP. To initialize a session on a UP, a set of hardware resource have to be allocated (e.g., NP, TCAM etc.) to a session.

Subscriber session related procedures include subscriber session
create, update, delete, and statistics report. The following subsections give a high level view of the procedures.

4.3.1 Create Subscriber Session

The below sequence describes the DHCP IPv4 dial-up process, it is an example that shows how a subscriber session is created. (An example for IPv6 appears in Section 5.1.2.)

The request starts from an Online Request message (step 1) from the RG (for example, a DHCP Discovery packet). When the UP receives the Online Request from the RG, it will tunnel the Online Request to the CP through the Service Interface (Step 2). The Service Interface is implemented by a tunneling technology.

When the CP receives the Online Request from the UP, it will send an authentication request to the AAA server to authenticate and authorize the subscriber (step 3). When a positive reply is received from the AAA server, the CP starts to create a subscriber session for the request. Relevant resources (e.g., IP address, bandwidth, etc.)
will be allocated to the subscriber, policies and security rules will be generated for the subscriber. Then the CP sends a session create request to the UP through the Control Interface (Ci) (step 4), and a response is expected from the UP to confirm the creation (step 5).

Finally, the CP will notify the AAA server to start accounting (step 6). At the same time, an Online Response message (for example, a DHCP Ack packet) will be sent to the UP through the Si (step 7). And the UP will forward the Online Response to the RG (step 8).

This completes the subscriber online process.

4.3.2 Update Subscriber Session

The following numbered sequence shows the process of subscriber session update.

<table>
<thead>
<tr>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>COA Request</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Session update Request</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&lt;--------via Ci--------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Session update Response</td>
<td>&lt;---------via Ci--------&gt;</td>
</tr>
<tr>
<td>3</td>
<td>-------via Ci--------&gt;</td>
<td>COA Response</td>
</tr>
<tr>
<td>4</td>
<td>-----------&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Figure 11: Subscriber Session Update

When a subscriber session has been created on a UP, there may be requirements to update the session with new parameters (e.g., Bandwidth, QoS, policies, etc.).

This procedure is triggered by a Change of Authorization (COA) request message sent by the AAA server. The CP will update the session on the UP according to the new parameters through the Control Interface.
4.3.3 Delete Subscriber Session

The below call flow shows generally how S-CUPS deals with a subscriber offline request.

```
RG               UP                       CP
|                |                        |
|Offline Request |                        |
1|--------------->|                        |
|                |    Relay the Offline   |
|                |        Request         |
|               2|------to CP via Si-----|
|                |                        |
|                |    Send the Offline    |
|                |        Response        |
|               3|<-----to UP via Si------|
|Offline Response|                        |
4|<---------------|                        |
|                |     Session delete     |
|                |        Request         |
|                |<--------via Ci---------|
|                |     Session delete     |
|                |       Response         |
|                |---------via Ci-------->|
```

Figure 12: Subscriber Session Delete

Similar to the session creation process, when a UP receives an offline request from a RG, it will tunnel the request to a CP through the Si.

When the CP receives the offline request, it will withdraw/release the resources (e.g., IP address, bandwidth) that have been allocated to the subscriber. Then, it sends a reply to the UP through the Service Interface and the UP will forward the reply to the RG. At the same time, it will delete all the status of the session on the UP through the Ci.

4.3.4 Subscriber Session Events Report

```
UP                       CP
<p>| |
|                        |</p>
<table>
<thead>
<tr>
<th>Statistic/Detect report</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;--------via Ci--------&gt;</td>
</tr>
</tbody>
</table>
```

Figure 13: Events Report
When a session is created on an UP, the UP will periodically report statistics information and detect results of the session to the CP.
5. S-CUSP Call Flows

The subsections below give an overview of various "dial-up" interactions over the Service Interface followed by an overview of the setting of various information in the UP by the CP using S-CUSP over the Control Interface.

S-CUSP messages are described in this document using Routing Backus Naur Form (RBNF) as defined in [RFC5511].

5.1 IPoE

5.1.1 DHCPv4 Access

The following sequence shows detailed procedures for DHCPv4 access.

<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DHCP Discovery</td>
<td>Relay the DHCP Discovery</td>
<td>AAA</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>to CP via Si----&gt;</td>
<td>Req/Rep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send the DHCP Offer</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DHCP Request</td>
<td>to UP via Si----</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DHCP Offer</td>
<td>to UP via Si----</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>DHCP Request</td>
<td>Relay the DHCP Request</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>to CP via Si----</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create subscriber session Request</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>via Ci---------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create subscriber session Response</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>via Ci---------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Accounting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>to UP via Si----</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hu, et al
Step 8 and 9 are implemented by the S-CUSP protocol.

When a subscriber is authenticated and authorized by the AAA server, the CP will create a subscriber session on the UP. This is achieved by sending an Update_Request message to the UP.

The format of the Update_Request message is shown as follows using RBNF:

```
<Update_Request Message> ::= <Common Header>
    <Basic Subscriber TLV>
    <IPv4 Subscriber TLV>
    <IPv4 Routing TLV>
    [<Subscriber Policy TLV>]
```

The UP will reply with an Update_Response message, the format of the Update_Response message is as follows:

```
<Update_Response Message> ::= <Common Header>
    <Update Response TLV>
    [<Subscriber CGN Port Range TLV>]
```

5.1.2 DHCPv6 Access

The following sequence shows detailed procedures for DHCPv6 access.

```
<table>
<thead>
<tr>
<th>RG</th>
<th>Solicit</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-------------</td>
<td>-------</td>
<td>--------</td>
<td>--------------</td>
</tr>
<tr>
<td>2</td>
<td>Relay the Solicit to CP via Si</td>
<td>AAA Req/Rep</td>
<td>&lt;---------------</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Send the Advertise &lt;---to UP vis Si</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Advertise</td>
<td>Request</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Figure 15: DHCPv6 Access

Steps 1-7 are a standard DHCP IPv6 access process. The subscriber creation is triggered by a DHCP IPv6 request message. When this message is received, it means that the subscriber has passed the AAA authentication and authorization. Then the CP will create a subscriber session on the UP. This is achieved by sending an Update_Request message to the UP (Step 8).

The format of the Update_Request message is as follows:

<Update_Request Message> ::= <Common Header>  
<Basic Subscriber TLV>  
<IPv6 Subscriber TLV>  
<IPv6 Routing TLV>  
[<Subscriber Policy TLV>]

The UP will reply with an Update_Response message (Step 9). The format of the Update_Response message is as follows:

<Update_Response Message> ::= <Common Header>  
<Update Response TLV>
5.1.3 IPv6 SLAAC Access

The following flow shows the IPv6 SLAAC access process.

<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
<td>--------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>1</td>
<td>&lt;--------------------</td>
<td>Relay the Router</td>
<td>AAA Req/Rep</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Solicit (RS)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-----to CP via Si-</td>
<td>--&lt;-----------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create subscriber</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>session Request</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>&lt;-------------via Ci--------&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create subscriber</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>session Response</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;-----------------via Ci--------&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Send Router Advertise (RA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6&lt;-----to UP vis Si------&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>&lt;-----------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS</td>
<td>--------------------</td>
<td>Relay the Neighbor</td>
<td>Accounting</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Solicit (NS)</td>
<td>&lt;-----------------</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>-----to CP via Si-</td>
<td>--&lt;-----------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send a Neighbor</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Advertise (NA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>&lt;-----to UP via Si------&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>&lt;-------------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 16: IPv6 SLAAC Access**

It starts with a Router Solicit (RS) request from an RG that is tunneled to the CP by the UP. After the AAA authentication and authorization, the CP will create a subscriber session on the UP.

Hu, et al
This is achieved by sending an Update_Request message to the UP (step 4).

The format of the Update_Request message is as follows:

<Update_Request Message> ::= <Common Header>
    <Basic Subscriber TLV>
    <IPv6 Subscriber TLV>
    <IPv6 Routing TLV>
    [<Subscriber Policy TLV>]

The UP will reply with an Update_Response message (step 5), the format of the Update_Response message is as follows:

<Update_Response Message> ::= <Common Header>
    <Update Response TLV>

5.1.4 DHCPv6 + SLAAC Access

The following call flow shows the DHCP IPv6 and SLAAC access process.

<table>
<thead>
<tr>
<th></th>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>AAA</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. RS
2. Relay the Router Solicit (RA) -----to CP via Si------> AAA Req/Rep
3. AAA
4. Create subscriber session Request <--------via Ci-------->
5. Create subscriber session Response <--------via Ci-------->
6. Send Router Advertise (RA) <--to UP vis Si------>
7. RA
8. DHCPv6 Solicit
     ----------------->
     Relay DHCPv6 Solicit

Hu, et al
When a subscriber passes AAA authentication, the CP will create a subscriber session on the UP. This is achieved by sending an Update_Request message to the UP (step 4).

```
<Update_Request Message> ::= <Common Header>
  <Basic Subscriber TLV>
  <IPv6 Subscriber TLV>
  <IPv6 Routing TLV>
  [<Subscriber Policy TLV>]
```

The UP will reply with an Update_Response message (step 5). The format of the Update_Response is as follows:

```
<Update_Response Message> ::= <Common Header>
  <Update Response TLV>
```

After receiving a DHCPv6 Solicit, the CP will update the subscriber session by sending an Update_Request message with new parameters to the UP (Step 10).

The format of the Update_Request message is as follows:

```
<Update_Request Message> ::= <Common Header>
  <Basic Subscriber TLV>
  <IPv6 Subscriber TLV>
  <IPv6 Routing TLV>
  [<Subscriber Policy TLV>]
```
The UP will reply with an Update_Response message (step 11). The format of the Update_Response is as follows:

\[ \text{<Update_Response Message>} ::= \text{<Common Header>}
\text{<Update Response TLV>} \]

5.1.5 DHCP Dual Stack Access

The following sequence is a combination of DHCP IPv4 and DHCP IPv6 access processes.

<table>
<thead>
<tr>
<th></th>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DHCP Discovery</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>AAA Req/Resp</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>Send the DHCP Offer</td>
</tr>
<tr>
<td>5</td>
<td>DHCP Offer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>DHCP Request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>Relay the DHCP Request</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Create subscriber session Request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td>Create subscriber session Response</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>Accounting</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td>Send DHCP ACK</td>
</tr>
<tr>
<td>12</td>
<td>DHCP ACK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>RS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td>Relay the Router Solicit (RA)</td>
</tr>
</tbody>
</table>

Hu, et al
The DHCP dual stack access includes three sets of Update_Request / Update_Response exchanges to create/update DHCPv4/v6 subscriber session.

1. Create DHCPv4 session (step 8 and 9)

<Update_Request Message> ::= <Common Header>
  <Basic Subscriber TLV>
  <IPv4 Subscriber TLV>
  <IPv4 Routing TLV>
  [<Subscriber Policy TLV>]
<Update_Response Message> ::= <Common Header>
<Update Response TLV>
[<Subscriber CGN Port Range TLV>]

2. Create DHCPv6 session (step 16 and 17)

<Update_Request Message> ::= <Common Header>
<Basic Subscriber TLV>
<IPv6 Subscriber TLV>
<IPv6 Routing TLV>
[<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
<Update Response TLV>

3. Update DHCPv6 session (step 22 and 23)

<Update_Request Message> ::= <Common Header>
<Basic Subscriber TLV>
<IPv6 Subscriber TLV>
<IPv6 Routing TLV>
[<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
<Update Response TLV>

5.1.6 L2 Static Subscriber Access

L2 static subscriber access processes are as follows:

<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Static Subscriber Detection Req.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>&lt;-----to UP via Ci---------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Static Subscriber Detection Rep.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>------to UP via Ci---------&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>ARP/ND(REQ)</td>
<td>ARP/ND(ACK)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;---------------</td>
<td>-----------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td></td>
<td>Relay the ARP/ND</td>
<td>AAA</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>-----to CP via Si-------------</td>
<td>Req/Rep</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>WAY to CP via Si-------------</td>
<td>AAA</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Create subscriber session Request</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hu, et al                                                        [Page 37]
For L2 static subscriber access, the process starts with a CP installing a static subscriber detection list on an UP. The list determines which subscribers will be detected. This is implemented by exchanging Update_Request and Update_Response messages between CP and UP. The format of the messages are as follows:

Figure 19: L2 Static Subscriber Access
<Update_Request Message> ::= <Common Header>
    <IPv4 Static Subscriber Detect TLVs>
    <IPv6 Static Subscriber Detect TLVs>

<Update_Response Message> ::= <Common Header>
    <Update Response TLV>

For L2 Static subscriber access, there are three ways to trigger the
access process:

1. Triggered by UP (3.1-3.6): This assumes that the UP knows the IP
   address, the access interface, and VLAN of the RG. The UP will
   actively trigger the access flow by sending an ARP/ND packet to
   the RG. If the RG is online, it will reply with an ARP/ND to the
   UP. The UP will tunnel the ARP/ND to the CP through the Si. The
   CP then triggers the authentication process. If the
   authentication result is positive. The CP will create a
   corresponding subscriber session on the UP.

2. Triggered by RG ARP/ND (4.1-4.6): Most of the process is same as
   option 1 (triggered by UP). The difference is that the RG will
   actively send the ARP/ND to trigger the process.

3. Triggered by RG IP traffic (5.1-5.7): This is for the case where
   the RG has the ARP/ND information, but the subscriber session on
   the UP is lost (e.g., due to failure on the UP, or the UP
   restarted). That means the RG may keep sending IP packets to the
   UP. The packets will trigger the UP to start a new access
   process.

From a subscriber session point of view, the procedures and the
message formats for the above three cases are the same, as follows:

IPv4 Case:
<Update_Request Message> ::= <Common Header>
    <Basic Subscriber TLV>
    <IPv4 Subscriber TLV>
    <IPv4 Routing TLV>
    [<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
    <Update Response TLV>
    [<Subscriber CGN Port Range TLV>]

IPv6 Case:
<Update_Request Message> ::= <Common Header>
    <Basic Subscriber TLV>
    <IPv6 Subscriber TLV>
    <IPv6 Routing TLV>
    [<Subscriber Policy TLV>]

Hu, et al

[Page 39]
5.2 PPPoE

5.2.1 IPv4 PPPoE Access

The following figure shows the IPv4 PPPoE access call flow.

<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPPoE Disc</td>
<td>&lt;--------via Si-----------</td>
<td>PPPoE Disc</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>PPP LCP</td>
<td>PPP LCP</td>
<td>AAA</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&lt;--------via Si-----------</td>
<td>PPP PAP/CHAP</td>
<td></td>
</tr>
<tr>
<td>PPP PAP/CHAP</td>
<td>PPP IPCP</td>
<td>AAA</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>&lt;--------via Si-----------</td>
<td>Create subscriber session Request</td>
<td></td>
</tr>
<tr>
<td>PPP IPCP</td>
<td>Create subscriber session Response</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>&lt;--------via Ci-----------</td>
<td>Accounting</td>
<td></td>
</tr>
</tbody>
</table>

Figure 20: IPv4 PPPoE Access

From the above sequence, step 1-4 are the standard PPPoE call flow. The UP is responsible for redirecting the PPPoE control packets to the CP or RG. The PPPoE control packets are transmitted between the CP and UP through the Si.

After the PPPoE call flow, if the subscriber passed the AAA authentication and authorization, the CP will create a corresponding session on the UP through the Ci. The formats of the messages are as follows:
5.2.2 IPv6 PPPoE Access

The following figure describes the IPv6 PPPoE access call flow.

<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPPoE Disc</td>
<td>PPPoE Disc</td>
<td>PPPoE Disc</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>&lt;-------------</td>
<td>&lt;-------------via Si--</td>
<td></td>
</tr>
<tr>
<td>PPP LCP</td>
<td>PPP LCP</td>
<td>PPP LCP</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&lt;-------------</td>
<td>&lt;-------------via Si--</td>
<td>AAA</td>
</tr>
<tr>
<td>PPP PAP/CHAP</td>
<td>PPP PAP/CHAP</td>
<td>PPP PAP/CHAP</td>
<td>Req/Rep</td>
</tr>
<tr>
<td>3</td>
<td>&lt;-------------</td>
<td>&lt;-------------via Si--</td>
<td></td>
</tr>
<tr>
<td>PPP IP6CP</td>
<td>PPP IP6CP</td>
<td>PPP IP6CP</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>&lt;-------------</td>
<td>&lt;-------------via Si--</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create subscriber</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>session Request</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 &lt;-------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;-------------via Ci--</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create subscriber</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>session Response</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 &lt;-------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;-------------via Ci--</td>
<td></td>
</tr>
<tr>
<td>ND Negotiation</td>
<td>ND Negotiation</td>
<td>ND Negotiation</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>&lt;-------------</td>
<td>&lt;-------------via Si--</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update subscriber</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>session Request</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 &lt;-------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;-------------via Ci--</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update subscriber</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>session Response</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 &lt;-------------</td>
<td></td>
</tr>
<tr>
<td>DHCPv6 Negotiation 7’</td>
<td>DHCPv6 Negotiation 10’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHCPv6</td>
<td>DHCPv6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negotiation</td>
<td>Negotiation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;---------via Si-----</td>
<td>&lt;-------------&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update subscriber</td>
<td>Update subscriber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>session Request</td>
<td>session Response</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8’</td>
<td>9’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;---------via Ci------</td>
<td>&lt;---------via Ci--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accounting</td>
<td>Accounting</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 21: IPv6 PPPoE Access

From the above sequence, steps 1-4 are the standard PPPoE call flow. The UP is responsible for redirecting the PPPoE control packets to the CP or RG. The PPPoE control packets are transmitted between the CP and UP through the Si.

After the PPPoE call flow, if the subscriber passed the AAA authentication and authorization, the CP will create a corresponding session on the UP through the Ci. The formats of the messages are as follows:

```
<Update_Request Message> ::= <Common Header>
   <Basic Subscriber TLV>
   <PPP Subscriber TLV>
   <IPv6 Subscriber TLV>
   <IPv6 Routing TLV>
   [<Subscriber Policy TLV>]
```

```
<Update_Response Message> ::= <Common Header>
   <Update Response TLV>
```

Then, the RG will initialize a ND/DHCPv6 negotiation process with the CP (see step 7 and 7’), after that, it will trigger an update (8-9, 8’-9’) to the subscriber session. The formats of the update messages are as follows:
<Update_Request Message> ::= <Common Header>  
               <Basic Subscriber TLV>  
               <PPP Subscriber TLV>  
               <IPv6 Subscriber TLV>  
               <IPv6 Routing TLV>  
               [<Subscriber Policy TLV>]  

<Update_Response Message> ::= <Common Header>  
                           <Update Response TLV>  

5.2.3 PPPoE Dual Stack Access

The following figure shows a combination of IPv4 and IPv6 PPPoE access call flow.

<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPPoE Discovery</td>
<td>PPPoE Discovery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 &lt;---------------&gt;</td>
<td>&lt;-------------via Si---&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPP LCP</td>
<td>PPP LCP</td>
<td></td>
<td>AAA Req/Rep</td>
</tr>
<tr>
<td>2 &lt;---------------&gt;</td>
<td>&lt;-------------via Si---&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPP PAP/CHAP</td>
<td>PPP PAP/CHAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 &lt;---------------&gt;</td>
<td>&lt;-------------via Si---&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPP IPCP</td>
<td>PPP IPCP</td>
<td></td>
<td>Create v4 subscriber session Request</td>
</tr>
<tr>
<td>4 &lt;---------------&gt;</td>
<td>&lt;-------------via Si---&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create v4 subscriber session Response</td>
<td></td>
</tr>
<tr>
<td>5 &lt;---------------&gt;</td>
<td>&lt;-------------via Ci---&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accounting</td>
<td></td>
</tr>
<tr>
<td>6 &lt;---------------&gt;</td>
<td>&lt;-------------via Ci---&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPP IP6CP</td>
<td>PPP IP6CP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4’ &lt;---------------&gt;</td>
<td>&lt;-------------via Si---&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create v6 subscriber session Request</td>
<td></td>
</tr>
<tr>
<td>5’ &lt;---------------&gt;</td>
<td>&lt;-------------via Ci---&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create v6 subscriber session Response</td>
<td></td>
</tr>
<tr>
<td>6’ &lt;---------------&gt;</td>
<td>&lt;-------------via Ci---&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ND Negotiation</td>
<td>ND Negotiation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PPPv6 dual stack is a combination of IPv4 PPPvE and IPv6 PPPvE access. The process is as above. The formats of the messages are as follows:

1. Create an IPv4 PPPvE subscriber session (5-6)

   `<Update_Request Message> ::= <Common Header>`
   `<Basic Subscriber TLV>`
   `<PPP Subscriber TLV>`
   `<IPv4 Subscriber TLV>`
   `<IPv4 Routing TLV>`
   `[<Subscriber Policy TLV>]`

   `<Update_Response Message> ::= <Common Header>`
   `<Update Response TLV>`
   `[<Subscriber CGN Port Range TLV>]`

2. Create an IPv6 PPPvE subscriber session (5'-6')

   `<Update_Request Message> ::= <Common Header>`
   `<Basic Subscriber TLV>`
   `<PPP Subscriber TLV>`

Figure 22: PPPvE Dual Stack Access
3. Update the IPv6 PPPoE subscriber session (9-10, 9’-10’)

5.3 WLAN Access

The following figure shows the WLAN access call flow.

```
<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
<th>WEB Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DHCP Discovery</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DHCP Discovery</td>
<td></td>
<td>AAA</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DHCP Offer</td>
<td>AAA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DHCP Request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DHCP Request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>DHCP Request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Create session</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Create session</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Hu, et al  [Page 45]
WLAN access starts with the DHCP dial-up process (steps 1-6), after that the CP will create a subscriber session on the UP (steps 7-8). The formats of the session creation messages are as follows:

**IPv4 Case:**

\[
\text{<Update Request Message>} ::= \text{<Common Header>}
\text{<Basic Subscriber TLV>}
\text{<IPv4 Subscriber TLV>}
\text{<IPv4 Routing TLV>}
\text{[<Subscriber Policy TLV>]}\]
<Update_Response Message> ::= <Common Header>
  <Update Response TLV>
  [<Subscriber CGN Port Range TLV>]

IPv6 Case:
<Update_Request Message> ::= <Common Header>
  <Basic Subscriber TLV>
  <IPv6 Subscriber TLV>
  <IPv6 Routing TLV>
  [<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
  <Update Response TLV>

After step 10, the RG will be allocated an IP address and its first HTTP packet will be redirected to a WEB server for subscriber authentication (steps 11-17). After the WEB authentication, if the result is positive, the CP will update the subscriber session by using the following message exchanges:

IPv4 Case: <Update_Request Message> ::= <Common Header>
  <Basic Subscriber TLV>
  <IPv4 Subscriber TLV>
  <IPv4 Routing TLV>
  [<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
  <Update Response TLV>
  [<Subscriber CGN Port Range TLV>]

IPv6 Case: <Update_Request Message> ::= <Common Header>
  <Basic Subscriber TLV>
  <IPv6 Subscriber TLV>
  <IPv6 Routing TLV>
  [<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
  <Update Response TLV>

5.4 L2TP

5.4.1 L2TP LAC Access

<table>
<thead>
<tr>
<th>RG</th>
<th>UP(LAC)</th>
<th>CP(LAC)</th>
<th>AAA</th>
<th>LNS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PPPoE</td>
<td>PPPoE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hu, et al
<table>
<thead>
<tr>
<th>1</th>
<th>Discovery</th>
<th>Discovery</th>
<th>---via Si---</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PPP LCP</td>
<td>PPP LCP</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PPP PAP/CHAP</td>
<td>PPP PAP/CHAP</td>
<td>AAA Req/Rep</td>
</tr>
<tr>
<td>3</td>
<td>PPP IPCP</td>
<td>PPP IPCP</td>
<td>---via Si---</td>
</tr>
<tr>
<td>4</td>
<td>L2TP tunnel negotiation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCCRQ/</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCCRP/</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCCCN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>---via Si---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>L2TP session negotiation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ICRQ/</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ICRP/</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ICCN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>---via Si---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Create subscriber session Request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Create subscriber session Request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>---via Ci----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PAP/CHAP (Triggered by LNS) ---via routing?-------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Steps 1-4 are a standard PPPoE access process. After that the LAC-CP starts to negotiate an L2TP session and tunnel with the LNS. After the negotiation, the CP will create an L2TP LAC subscriber session on the UP through the following messages:

```
<Update_Request Message> ::= <Common Header>
    <Basic Subscriber TLV>
    <L2TP-LAC Subscriber TLV>
    <L2TP-LAC Tunnel TLV>
```

```
<Update_Response Message> ::= <Common Header>
    <Update Response TLV>
```

<table>
<thead>
<tr>
<th>RG</th>
<th>LAC</th>
<th>UP(LNS)</th>
<th>AAA</th>
<th>CP(LNS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPPoE</td>
<td>Discovery</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>&lt;---------------&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPP LCP</td>
<td></td>
<td></td>
<td>AAA</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&lt;---------------&gt;</td>
<td>AAA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPP PAP/CHAP</td>
<td></td>
<td>AAA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>&lt;---------------&gt;</td>
<td>AAA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.4.2 L2TP LNS IPv4 Access

1. **PPP LCP**
2. **PPP PAP/CHAP**
3. **L2TP tunnel negotiation**
   - SCCRQ/
   - SCCRP/
   - SCCCN
4. **Create**
   - subscriber session
5. **ICRQ/**
   - ICRP/
   - ICCN

 RG          LAC                        UP(LNS) | AAA | CP(LNS) W|
In this case, the BNG is running as an LNS and separated into LNS-CP and LNS-UP. Steps 1-5 finish the normal L2TP dial-up process. When the L2TP session and tunnel negotiations are finished, the LNS-CP will create an L2TP LNS subscriber session on the LNS-UP. The format of messages are as follows:

<Update_Request Message> ::= <Common Header> <L2TP-LNS Subscriber TLV> <Basic Subscriber TLV> <PPP Subscriber TLV> <IPv4 Subscriber TLV> <IPv4 Routing TLV> <L2TP-LNS Tunnel TLV> [<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header> <Update Response TLV> [<Subscriber CGN Port Range TLV>]
After that, the LNS-CP will trigger an AAA authentication. If the authentication result is positive, a PPP IPCP process will follow, then the CP will update the session with the following message exchanges:

\[<Update\_Request\_Message> ::= <Common\_Header>\]
\[<L2TP-LNS\_Subscriber\_TLV>\]
\[<Basic\_Subscriber\_TLV>\]
\[<PPP\_Subscriber\_TLV>\]
\[<IPv4\_Subscriber\_TLV>\]
\[<IPv4\_Routing\_TLV>\]
\[<L2TP-LNS\_Tunnel\_TLV>\]
\[[<Subscriber\_Policy\_TLV>]\]

\[<Update\_Response\_Message> ::= <Common\_Header>\]
\[<Update\_Response\_TLV>\]
\[[<Update\_Response\_CGN\_Port\_Range\_TLV>]\]

### 5.4.3 L2TP LNS IPv6 Access

<table>
<thead>
<tr>
<th>RG</th>
<th>LAC</th>
<th>UP(LNS)</th>
<th>AAA</th>
<th>CP (LNS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PPPoE Discovery</td>
<td>1 &lt;----------&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PPP LCP</td>
<td>2 &lt;----------&gt;</td>
<td>AAA</td>
<td>Reg/Rep</td>
</tr>
<tr>
<td></td>
<td>PPP PAP/CHAP</td>
<td>3 &lt;----------&gt;</td>
<td>&lt;-------------------&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L2TP tunnel negotiation</td>
<td>4 &lt;----------&gt;</td>
<td>&lt;--------via Si------&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCCRQ/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCCRP/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCCCN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>L2TP session negotiation</td>
<td>5 &lt;----------&gt;</td>
<td>&lt;--------via Si------&gt;</td>
<td>Create</td>
</tr>
<tr>
<td></td>
<td>ICRQ/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ICRP/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ICCN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 26: L2TP-LNS IPv6 Access
Steps 1-12 are the same as L2TP and LNS IPv4 Access. Steps 1-5 finish the normal L2TP dial-up process. When the L2TP session and tunnel negotiations are finished, the LNS-CP will create an L2TP LNS subscriber session on the LNS-UP. The format of the messages is as follows:

<Update_Request Message> ::= <Common Header>
  <L2TP-LNS Subscriber TLV>
  <Basic Subscriber TLV>
  <PPP Subscriber TLV>
  <IPv6 Subscriber TLV>
  <IPv6 Routing TLV>
  <L2TP-LNS Tunnel TLV>
  [<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
  <Update Response TLV>

After that, the LNS-CP will trigger a AAA authentication. If the authentication result is positive, a PPP IP6CP process will follow, then the CP will update the session with the following message exchanges:

<Update_Request Message> ::= <Common Header>
  <L2TP-LNS Subscriber TLV>
  <Basic Subscriber TLV>
  <PPP Subscriber TLV>
  <IPv6 Subscriber TLV>
  <IPv6 Routing TLV>
  <L2TP-LNS Tunnel TLV>
  [<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
  <Update Response TLV>

Then, an ND negotiation will be triggered by the RG. After the ND negotiation, the CP will update the session with the following message exchanges:

<Update_Request Message> ::= <Common Header>
  <L2TP-LNS Subscriber TLV>
  <Basic Subscriber TLV>
  <PPP Subscriber TLV>
  <IPv6 Subscriber TLV>
  <IPv6 Routing TLV>
  <L2TP-LNS Tunnel TLV>
  [<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
  <Update Response TLV>

Hu, et al
[Page 53]
5.5 CGN (Carrier Grade NAT)

<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Public Address Block Allocation Request</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Public Address Block Allocation Reply</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>--------via Ci--------&gt;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subscriber access request</th>
<th>Subscriber access reply</th>
<th>AAA Reg/Rep</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 ------------------------</td>
<td>4 ---------via Si--------&gt;</td>
<td></td>
</tr>
<tr>
<td>7 &lt;----------------------</td>
<td>5 &lt;----------------------</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subscriber access reply 6</th>
<th>Create subscriber session Request</th>
<th>Accounting with source information</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 &lt;----------------------</td>
<td>8 &lt;----------------------</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Create subscriber session Response (with NAT information)</th>
<th></th>
<th>Public IP + Port range to Private IP mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 &lt;----------------------</td>
<td>10 &lt;----------------------</td>
<td></td>
</tr>
</tbody>
</table>

Figure 27: CGN Access

The first steps allocate one or more CGN address blocks to the UP (steps 1-2). This is achieved by the following message exchanges between CP and UP.
<Addr_Allocation_Req Message> ::= <Common Header>
<Request Address Allocation TLV>

<Addr_Allocation_Ack Message> ::= <Common Header>
<Address Assignment Response TLV>

Steps 3-9 show the general dial-up process in the case of CGN mode. The specific processes (e.g., IPoE, PPPoE, L2TP, etc.) are defined in above sections.

If a subscriber is a CGN subscriber, once the subscriber session is created/updated, the UP will report the NAT information to the CP. This is achieved by carrying the "Subscriber CGN Port Range TLV" in the Update_Response message.

5.6 L3 Leased Line Access

5.6.1 Web Authentication

<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
<th>WEB Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User traffic</td>
<td></td>
<td>AAA Req/Rep</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>User traffic</td>
<td></td>
<td>3 &lt;--------&gt;</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AAA Req/Rep</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Create session Request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Create session Response</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>HTTP traffic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Redirect to Web URL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-----------------Redirected to Web server-----------&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hu, et al [Page 55]
In this case, IP traffic from the RG will trigger the CP to authenticate the RG by checking the source IP and the exchanges with the AAA server. Once the RG passed the authentication, the CP will create a corresponding subscriber session on the UP through the following message exchanges:

**IPv4 Case:**

```plaintext
<Update_Request Message> ::= <Common Header> 
    <Basic Subscriber TLV> 
    <IPv4 Subscriber TLV> 
    <IPv4 Routing TLV> 
    [<Subscriber Policy TLV>] 

<Update_Response Message> ::= <Common Header> 
    <Update Response TLV> 
    [<Subscriber CGN Port Range TLV>] 
```

**IPv6 Case:**

```plaintext
<Update_Request Message> ::= <Common Header> 
    <Basic Subscriber TLV> 
    <IPv6 Subscriber TLV> 
    <IPv6 Routing TLV> 
    [<Subscriber Policy TLV>] 

<Update_Response Message> ::= <Common Header> 
    <Update Response TLV> 
```

Figure 28: Web Authentication based L3 Leased Line Access
Then, the HTTP traffic from the RG will be redirected to a WEB server to finish the WEB authentication. Once the WEB authentication is passed, the CP will trigger another AAA authentication. After the AAA authentication, the CP will update the session with the following message exchanges:

**IPv4 Case:**

```plaintext
<Update_Request Message> ::= <Common Header>
<Basic Subscriber TLV>
<IPv4 Subscriber TLV>
<IPv4 Routing TLV>
[{Subscriber Policy TLV}]

<Update_Response Message> ::= <Common Header>
<Update Response TLV>
[{Subscriber CGN Port Range TLV}]
```

**IPv6 Case:**

```plaintext
<Update_Request Message> ::= <Common Header>
<Basic Subscriber TLV>
<IPv6 Subscriber TLV>
<IPv6 Routing TLV>
[{Subscriber Policy TLV}]

<Update_Response Message> ::= <Common Header>
<Update Response TLV>
```

5.6.2 User Traffic Trigger

<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L3 access</td>
<td>via Ci</td>
<td></td>
</tr>
<tr>
<td></td>
<td>control list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>User</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>traffic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-----via Si</td>
<td></td>
<td>AAA Req/Rep</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>&lt;---------&gt;</td>
</tr>
<tr>
<td>5</td>
<td>Create</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>session</td>
<td>Request</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AAA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create session</td>
<td></td>
</tr>
</tbody>
</table>

Hu, et al [Page 57]
In this user traffic triggered case, the CP must install an access control list on the UP, which is used by the UP to determine whether an RG is legal or not. If the traffic is from a legal RG, it will be redirected to the CP through the SI. The CP will trigger a AAA interchange with the AAA server. After that, the CP will create a corresponding subscriber session on the UP with the following message exchanges:

IPv4 Case:
<Update_Request Message> ::= <Common Header>
  <Basic Subscriber TLV>
  <IPv4 Subscriber TLV>
  <IPv4 Routing TLV>
  [<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
  <Update Response TLV>
  [<Subscriber CGN Port Range TLV>]

IPv6 Case:
<Update_Request Message> ::= <Common Header>
  <Basic Subscriber TLV>
  <IPv6 Subscriber TLV>
  <IPv6 Routing TLV>
  [<Subscriber Policy TLV>]

<Update_Response Message> ::= <Common Header>
  <Update Response TLV>

5.7 Multicast Access

---
<table>
<thead>
<tr>
<th>RG</th>
<th>UP</th>
<th>CP</th>
<th>AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>User access request 1</td>
<td>User access request &lt;-----via SI----&gt; Request User</td>
<td>AAA &lt;------ Req/Rep</td>
<td></td>
</tr>
<tr>
<td>Create session Request 2</td>
<td>&lt;-----via Ci-----&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
---

Hu, et al
Multicast access starts with an user access request from the RG. The request will be redirected to the CP by the Si. A follow-up AAA interchange between the CP and the AAA server will be triggered. After the authentication, the CP will create a multicast subscriber session on the UP through the following messages:

**IPv4 Case:**
```
<Update_Request Message> ::= <Common Header>
    <Multicast Group Information TLV>
    <Basic Subscriber TLV>
    <IPv4 Subscriber TLV>
    <IPv4 Routing TLV>
    [<Subscriber Policy TLV>]
```
```
<Update_Response Message> ::= <Common Header>
    <Update Response TLV>
    [<Subscriber CGN Port Range TLV>]
```

**IPv6 Case:**
```
<Update_Request Message> ::= <Common Header>
    <Multicast Group Information TLV>
    <Basic Subscriber TLV>
    <IPv6 Subscriber TLV>
    <IPv6 Routing TLV>
    [<Subscriber Policy TLV>]
```
```
<Update_Response Message> ::= <Common Header>
    <Update Response TLV>
```

---

**Figure 30: Multicast Access**

```
| 3 | Create session Response    |
|-----------------------------|
| 4 | Multicast negotiation      |
```

---
6. S-CUSP Message Formats

An S-CUSP message consists of a common header followed by a variable-length body consisting entirely of TLVs. Receiving an S-CUSP message with an unknown message type or missing mandatory TLV MUST trigger an Error message (see Section 6.7) or a response message with an Error Information TLV (see Section 7.6).

Conversely, if a TLV is optional, the TLV may or may not be present. Optional TLVs are indicated in the message formats shown in this document by being enclosed in square brackets.

This section specifies the format of the common S-CUSP message header and lists the defined messages.

Network byte order is used for all multi-byte fields.

6.1 Common Message Header

S-CUSP Common Message Header:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Ver  |  Resv | Message-Type  |        Message-Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Reserved           |        Transaction-ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                   Figure 6.1: S-CUSP Message Common Header
```

- **Ver (4 bits):** The major version of the protocol. This document specifies version 1. Different major versions of the protocol may have significantly different message structure and format except that the Ver field will always be in the same place at the beginning of each message. A successful S-CUSP session depends on the CP and the UP both using the same major version of the protocol.

- **Resv (4 bits):** Reserved. MUST be sent as zero and ignored on receipt.

- **Message-Type (8 bits):** The set of message types specified in this document is listed in Section 9.1.

- **Message-Length (16 bits):** Total length of the S-CUSP message including the common header, expressed in number of bytes as an unsigned integer.
o Transaction ID (16 bits): This field is used to identify requests. It is echoed back in any corresponding ACK / response / Error message. It is RECOMMENDED that a monotonically increasing value be used in successive message and that value wrap back to zero after 0xFFFF. The contents of this field is an opaque value that the receiver MUST NOT use for any purpose except to echo back in a corresponding response and, optionally, for logging.

6.2 Control Messages

This document defines the following control messages:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Notes and TLVs that can be carried</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hello</td>
<td>Hello TLV, Keep-Alive TLV.</td>
</tr>
<tr>
<td>2</td>
<td>Keepalive</td>
<td>A common header with the Keepalive message type.</td>
</tr>
<tr>
<td>3</td>
<td>Sync_Request</td>
<td>Synchronization request.</td>
</tr>
<tr>
<td>4</td>
<td>Sync_Begin</td>
<td>Synchronization starts.</td>
</tr>
<tr>
<td>5</td>
<td>Sync_Data</td>
<td>Synchronization data: TLVs specified in Section 5.</td>
</tr>
<tr>
<td>6</td>
<td>Sync_End</td>
<td>End synchronization.</td>
</tr>
<tr>
<td>7</td>
<td>Update_Request</td>
<td>TLVs specified in Sections 7.6-7.9.</td>
</tr>
<tr>
<td>8</td>
<td>Update_Response</td>
<td>TLVs specified in Sections 7.6-7.9.</td>
</tr>
</tbody>
</table>

6.2.1 Hello Message

Hello message is used for S-CUSP session establishment and version negotiation. The detail of S-CUSP session establishment and version negotiation can be found in Section 4.1.1.

The format of Hello message is as follows:

<Hello Message> ::= <Common Header>
<Hello TLV>
<Keepalive TLV>
[<Error Information TLV>]

The return code and negotiation result will be carried in the Error Information TLV. They are listed as follows:

0: Success, version negotiation success.
1: Failure, malformed message received.
2: One or more of the TLVs was not understood.

1001: The version negotiation fails. The S-CUSP session establishment phase fails.

1002: The keepalive negotiation fails. The S-CUSP session establishment phase fails.

1003: The establishment timer expires. Session establishment phase fails.

6.2.2 Keepalive Message

The Keepalive message is periodically sent by each end of an S-CUSP session. It is used to detect whether the peer end is still alive. The Keepalive procedures are defined in Section 4.1.2.

The format of the Keepalive message is as follows:

<Keepalive Message> ::= <Common Header>

6.2.3 Sync_Request Message

The Sync_Request message is used to request synchronization from an S-CUSP peer. Both CP and UP can request their peer to synchronize data.

The format of the Sync_Request message is as follows:

<Sync_Request Message> ::= <Common Header>

A Sync_Request message may result in a Sync_Begin message from its peer. The Sync_Begin message is defined in Section 6.2.4.

6.2.4 Sync_Begin Message

The Sync_Begin message is a reply to a Sync_Request message. It is used to notify the synchronization requester whether the synchronization can be started.

The format of Sync_Begin message is as follows:

<Sync_Begin Message> ::= <Common Header>
          <Error Information TLV>
The return codes are carried in the Error Information TLV. The codes are listed below:

0: Success, be ready to synchronize.
1: Failure, malformed message received.
2: One or more of the TLVs was not understood.
2001: Synch-NoReady. The data to be synchronized is not ready.
2002: Synch-Unsupport. The data synchronization is not supported.

6.2.5 Sync_Data Message

The Sync_Data message is used to send data being synchronized between the CP and UP. The Sync_Data message has the same function and format as the Update_Request message. The difference is that there is no ACK for a Sync_Data message. An error caused by the Sync_Data message will result in a Sync_End message.

There are two scenarios:

Synchronization from UP to CP: Synchronize the resource data to CP.

\[\text{Sync_Data Message} ::= \langle\text{Common Header} \rangle\]
\[\langle\text{Resource Reporting TLVs}\rangle\]

Synchronization from CP to UP: Synchronize all subscriber sessions to UP. As for which TLVs should be carried, it depends on the specific session data to be synchronized. This is equivalent to create the specific session. Refer to Section 5 to see more details.

\[\text{Sync_Data Message} ::= \langle\text{Common Header} \rangle\]
\[\langle\text{User Routing TLVs}\rangle\]
\[\langle\text{User Information TLVs}\rangle\]
\[\langle\text{L2TP Subscriber TLVs}\rangle\]
\[\langle\text{Subscriber CGN Port Range TLV}\rangle\]
\[\langle\text{Subscriber Policy TLV}\rangle\]

6.2.6 Sync_End Message

The Sync_End message is used to indicate the end of a synchronization process. The format of a Sync_End message is as follows:
<Sync_End Message> ::= <Common Header>
   <Error Information TLV>

The return/error codes are listed as follows:

0: Success, synchronization finished.
1: Failure, malformed message received.
2: One or more of the TLVs was not understood.

6.2.7 Update_Request Message

The Update_Request message is a multi-task message, it can be used to create, update, and delete subscriber sessions on a UP.

For session operations, the specific operation is controlled by the "Oper" field of the carried TLVs. As defined in Section 7.1, the "Oper" can be set to either "update" or "delete" when a TLV is carried in an Update_Request message.

When the "Oper" set to update, it means to create or update a subscriber session, if the "Oper" set to delete, it indicates to delete a corresponding session on an UP.

The format of Update_Request message is as follows:

<Update_Request Message> ::= <Common Header>
   [User Routing TLVs]
   [User Information TLVs]
   [L2TP Subscriber TLVs]
   [Subscriber CGN Port Range TLV]
   [Subscriber Policy TLV]

Each Update_Request message will result in an Update_Response message that is defined in Section 6.2.8.

6.2.8 Update_Response Message

The Update_Response message is a response to an Update_Request message. It is used to confirm the update request (or reject it in the case of an error). The format of an Update_Response message is as follows:
<Update_Response Message> ::= <Common Header>
    [<Subscriber CGN Port Range TLV>]
    <Error Information TLV>

The return/error codes are carried in the Error Information TLV. They are listed as follows:

0: Success.

1: Failure, malformed message received.

2: One or more of the TLVs was not understood.

3001 (Pool-Mismatch): The corresponding address pool cannot be found.

3002 (Pool-Full): The address pool is fully allocated and no address segment is available.

3003 (Subnet-Mismatch): The address pool subnet cannot be found.

3004 (Subnet-Conflict): Subnets in the address pool have been classified into other clients.

4001 (Update-Fail-No-Res): The forwarding table fails to be delivered because the forwarding resources are insufficient.

4002 (QoS-Update-Success): The QoS policy takes effect.

4003 (QoS-Update-Sq-Fail): Failed to process the queue in the QoS policy.

4004 (QoS-Update-CAR-Fail): Processing of the CAR in the QoS policy fails.

4005 (Statistic-Fail-No-Res): Statistics processing failed due to insufficient statistics resources.

6.3 Event Message

The Event message is used to report subscriber session traffic statistics and detection information. The format of Event message is as follows:

<Event Message> ::= <Common Header>
    [<User Traffic Statistics Report TLV>]
    [<User Detection Result Report TLV>]
6.4 Report Message

The Report message is used to report board and interface status on a UP. The format of Report message is as follows:

<Report Message> ::= <Common Header>

[<Board Status TLVs>]

[<Interface Status TLVs>]

6.5 CGN Messages

This document defines the following resource allocation messages:

<table>
<thead>
<tr>
<th>Type</th>
<th>Message Name</th>
<th>TLV that is carried</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Addr_Allocation_Req</td>
<td>Address Allocation Request</td>
</tr>
<tr>
<td>201</td>
<td>Addr_Allocation_Ack</td>
<td>Address Allocation Response</td>
</tr>
<tr>
<td>202</td>
<td>Addr_Renew_Req</td>
<td>Address Renewal Request</td>
</tr>
<tr>
<td>203</td>
<td>Addr_Renew_Ack</td>
<td>Address Renewal Response</td>
</tr>
<tr>
<td>204</td>
<td>Addr_Release_Req</td>
<td>Address Release Request</td>
</tr>
<tr>
<td>205</td>
<td>Addr_Release_Ack</td>
<td>Address Release Response</td>
</tr>
</tbody>
</table>

6.5.1 Addr_Allocation_Req Message

The Addr_Allocation_Req message is used to request CGN address allocation. The format of Addr_Allocation_Req message is as follows:

<Addr_Allocation_Req Message> ::= <Common Header>
    <Address Allocation Request TLV>

6.5.2 Addr_Allocation_Ack Message

The Addr_Allocation_Ack message is a response to an Addr_Allocation_Req message. The format of Addr_Allocation_Ack message is as follows:

<Addr_Allocation_Ack Message> ::= <Common Header>
    <Address Allocation Response TLV>
6.5.3 Addr_Renew_Req Message

The Addr_Renew_Req message is used to request address renewal. The format of Addr_Renew_Req message is as follows:

<Addr_Renew_Req Message> ::= <Common Header>
                             <Address Renewal Request TLV>

6.5.4 Addr_Renew_Ack Message

The Addr_Renew_Ack message is a response to an Addr_Renew_Req message. The format of Addr_Renew_Ack message is as follows:

<Addr_Renew_Ack Message> ::= <Common Header>
                             <Address Renewal Response TLV>

6.5.5 Addr_Release_Req Message

The Addr_Release_Req message is used to request address release. The format of Addr_Release_Req message is as follows:

<Addr_Release_Req Message> ::= <Common Header>
                             <Address Release Request TLV>

6.5.6 Addr_Release_Ack Message

The Addr_Release_Ack message is a response to an Addr_Release_Req message. The format of Addr_Release_Ack message is as follows:

<Addr_Release_Ack Message> ::= <Common Header>
                             <Address Release Response TLV>

6.6 Vendor Message

The Vendor message is, in conjunction with the vendor TLV and vendor sub-TLV, can be used by vendors to extend the S-CUSP protocol. It’s message type is 11. If the receiver does not recognize the message, an Error message will be returned to the sender.
The format of the Vendor message is as follows:

\[
\text{<Vendor Message>} ::= \text{<Common Header>}
\]
\[
\text{<Vendor TLV>}
\]
\[
[\text{<any other TLVs as specified by the vendor>}]
\]

6.7 Error Message

The Error message is defined to return some critical error information to the sender. If a receiver does not know the message type of a received message, it MUST return an Error message to the sender.

The format of the Error message is as below:

\[
\text{<Error Message>} ::= \text{<Common Header>}
\]
\[
\text{<Error Information TLV>}
\]
7. S-CUSP TLVs and Sub-TLVs

This section specifies the following:

- the format of the TLVs that appear in S-CUSP messages,
- the format of the sub-TLVs that appear within the values of some TLVs, and
- the format of some basic data fields that appear within TLVs or sub-TLVs.

See Section 9 for a list of all defined TLVs and sub-TLVs.

7.1 Common TLV Header

S-CUSP messages consist of the common header specified in Section 6.1 followed by TLVs formatted as specified in this section.

```
+---------------------------------------------+----------+
| Oper |      TLV-Type         |       TLV-Length              |
+---------------------------------------------+----------+
```

- Oper (4 bits): For Message-Types that indicate an operation on a data set, the Oper field is interpreted as Update, Delete, or Reserved as specified in Section 9.3. For all other Message-Types, the Oper field MUST be sent as zero and ignored on receipt.
- TLV-Type (12 bits): The Type of a TLV, that is the meaning and format of the Value part, are determined by the TLV-Type of the TLV. See Section 9.2.
- TLV-Length (2 bytes): The length of the Value portion of the TLV in bytes as an unsigned integer.
- Value (variable length): This is the value portion of the TLV whose size is given by TLV-Length. The value portion consists of fields, frequently using one of the basic data field types (see Section 7.2) and sub-TLVs (see Section 7.3).
7.2 Basic Data Fields

This section specifies the binary format of several standard basic data fields that are used within other data structures in this specification.

- **STRING**: 0 to 255 octets. Will be encoded as a sub-TLV (see Section 7.3) to provide the length. The use of this data type in S-CUSP is to provide convenient labels for use by network operators in configuring and debugging their networks and interpreting S-CUSP messages. These labels will not normally be seen by subscribers. They are normally interpreted as ASCII [RFC20].

- **MAC-Addr**: 6 octets. Ethernet MAC Address [RFC7042].

- **IPv4-Address**: 8 octets. 4 octets of the IPv4 address value followed by a 4 octet address mask in the format XXX.XXX.XXX.XXX.

- **IPv6-Address**: 20 octets. 16 octets of IPv6 address followed by a 4 octet integer n in the range of 0 to 128 which gives the address mask as the one’s complement of 2**(128-n) - 1.

- **VLAN ID**: 2 octets. As follows [802.1Q]:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-----------------------------+
| PRI |D|      VLAN-ID          |
+-----------------------------+
```


D: Drop Eligibility Indicator (DEI). Default value 0.

VLAN-ID: Unsigned integer in the range 1-4094. (0 and 4095 are not valid VLAN IDs [802.1Q].)
In some cases, the Value portion of a TLV, as specified above, can contain one or more Sub-TLVs formatted as follows:

```
+----------------------------------+
<table>
<thead>
<tr>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
</tr>
<tr>
<td>----------------------------------</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>----------------------------------</td>
</tr>
</tbody>
</table>
```

Figure 33: Sub-TLV Header

- Type (2 bytes): The Type of a Sub-TLV, that is the meaning and format of the Value part, are determined by the Type of the TLV. Sub-TLV Types numbers have the same meaning regardless of the TLV Type of the TLV within which the sub-TLV occurs. See Section 9.4.

- Length (2 bytes): The length of the Value portion of the sub-TLV in bytes as an unsigned integer.

- Value (variable length): This is the value portion of the sub-TLV whose size is given by Length.

The sub-TLVs currently specified are defined in the following subsections.

### 7.3.1 Name sub-TLVs

This document defines the following name sub-TLVs that are used to carry the name of the corresponding object. The length of each of these sub-TLV is variable from 1 to 255 octets. The value is of type STRING padded with zeros octets to 4-octet alignment.

<table>
<thead>
<tr>
<th>Type</th>
<th>Sub-TLV Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VRF-Name</td>
<td>The name of a VRF</td>
</tr>
<tr>
<td>2</td>
<td>Ingress-QoS-Profile</td>
<td>The name of an ingress QoS profile</td>
</tr>
<tr>
<td>3</td>
<td>Egress-QoS-Profile</td>
<td>The name of an egress QoS profile</td>
</tr>
<tr>
<td>4</td>
<td>User-ACL-Policy</td>
<td>The name of an ACL policy</td>
</tr>
<tr>
<td>5</td>
<td>Multicast-ProfileV4</td>
<td>The name of an IPv4 multicast profile</td>
</tr>
<tr>
<td>6</td>
<td>Multicast-ProfileV6</td>
<td>The name of an IPv6 multicast profile</td>
</tr>
<tr>
<td>7</td>
<td>NAT-Instance</td>
<td>The name of a NAT instance</td>
</tr>
<tr>
<td>8</td>
<td>Pool-Name</td>
<td>The name of an address pool</td>
</tr>
</tbody>
</table>
7.3.2 Ingress-CAR sub-TLV

The Ingress-CAR sub-TLV indicates the authorized upstream Committed Access Rate (CAR) parameters. The sub-TLV type of the Ingress-CAR sub-TLV is 9 and the sub-TLV length is 16. The format is as shown in Figure 34.

```
+---------------------------------------------------------------+
|                  CIR (Committed Information Rate)               |
+---------------------------------------------------------------+
|                  PIR (Peak Information Rate)                    |
+---------------------------------------------------------------+
|                  CBS (Committed Burst Size)                     |
+---------------------------------------------------------------+
|                  PBS (Peak Burst Size)                          |
```

Figure 34: Ingress-CAR sub-TLV

Where:

- CIR (4 bytes): Guaranteed rate in bits/second.
- PIR (4 bytes): Burst rate in bits/second.
- CBS (4 bytes): The token bucket in bytes.
- PBS (4 bytes): Burst token bucket in bytes.

These fields are unsigned integers. More details about CIR, PIR, CBS, and PBS can be found in [RFC2698].

7.3.3 Egress-CAR sub-TLV

The Egress-CAR sub-TLV indicates the authorized downstream Committed Access Rate (CAR) parameters. The sub-TLV type of the Egress-CAR sub-TLV is 10 and its sub-TLV length is 16 octets. The format of the value part is as defined below.
Figure 35: Egress-CAR sub-TLV

Where:

CIR (4 bytes): Guaranteed rate in bits/second.
PIR (4 bytes): Burst rate in bits/second.
CBS (4 bytes): The token bucket in bytes.
PBS (4 bytes): Burst token bucket in bytes.

These fields are unsigned integers. More details about CIR, PIR, CBS, and PBS can be found in [RFC2698].

7.3.4 If-Desc sub-TLV

The If-Desc sub-TLV is defined to designate an interface. It is an optional sub-TLV that may be carried in those TLVs that have an "if-index" or "out-if-index" field. The If-Desc sub-TLV is used as a local unique identifier within a BNG.

The sub-TLV type is 11 and the sub-TLV length is 12 octets. The format depends on the If-Type. The format of the value part is as follows:
If-Type: 8 bits in length, indicates the type of an interface. Following types are defined in this document:

0: Reserved  
1: Fast Ethernet (FE)  
2: GE  
3: 10GE  
4: 100GE  
5: Eth-Trunk  
6: Tunnel  
7: VE  
8-255: Reserved.

Chassis (8 bits): Identifies the chassis that the interface belongs to.

Slot (16 bits): Identifies the slot that the interface belongs to.

Sub-slot (16 bits): Identifies the sub-slot the interface belongs to.

Port Number (16 bits): An identifier of a physical port/interface (e.g., If-Type: 1-5). It is locally significant within the slot/sub-slot.
Sub-port Number (32 bits): An identifier of the sub-port. Locally significant within its "parent" port (physical or virtual).

Logic-ID (32 bits): An identifier of a virtual interface (e.g., If-Type: 6-7)

7.3.5 IPv6 Address List sub-TLV

The IPv6 Address List sub-TLV is used to convey one or more IPv6 addresses. It is carried in the IPv6 Subscriber TLV. The sub-TLV type is 12, the sub-TLV length is variable.

The format of IPv6 Addresses sub-TLV is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                        IPv6 Address                           ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                        IPv6 Address                           ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                            ...                                ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                        IPv6 Address                           ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 37: IPv6 Address List sub-TLV

Where:

IP Address (IPv6-Address): Each IP Address is an IP-Address type, carries an IPv6 address.

7.3.6 Vendor sub-TLV

The Vendor sub-TLV is intended to be used inside the value portion of the Vendor TLV (Section 7.13). It provides a Sub-Type that effectively extends the sub-TLV type in the sub-TLV header and provides for versioning of vendor sub-TLVs.

The value part of the Vendor sub-TLV is formatted as follows:

```
Hu, et al
```
Figure 38: Vendor sub-TLV

Where:

The sub-TLV type: 13.

The sub-TLV length: variable.

Vendor-ID (4 bytes): Vendor ID as defined in RADIUS [RFC2865].

Sub-Type (2 bytes): Used by the Vendor to distinguish multiple different sub-TLVs.

Sub-Type-Version (2 bytes): Used by the Vendor to distinguish different versions of a Vendor Defined sub-TLV sub-Type.

value: as specified by the vendor.

Since Vendor code will be handling the sub-TLV after the Vendor ID field is recognized, the remainder of the sub-TLV can be organized however the vendor wants. But it desirable for a vendor to be able to define multiple different vendor sub-TLVs and to keep track of different versions of its vendor defined sub-TLVs. Thus, it is RECOMMENDED that the vendor assign a Sub-Type value for each of that vendor’s sub-TLVs that is different from other Sub-Type values that vendor has used. Also, when modifying a vendor defined sub-TLV in a way potentially incompatible with a previous definition, the vendor SHOULD increase the value it is using in the Sub-Type-Version field.
7.4 The Hello TLV

The Hello TLV is defined to be carried in the Hello message for version and capabilities negotiation. It indicates the S-CUSP sub-version and capabilities supported. The format of Hello TLV is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          VerSupported                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Vendor ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Capabilities                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 39: Hello TLV

Where:

The TLV type is 100.

The TLV length is 12 octets.

The value field consists of three parts:

VerSupported: 32 bits in length. This is a bit map of the Sub-Versions of the S-CUSP protocol that the sender supports. This document specifies Sub-Version zero of Major Version 1, that is, Version 1.0. The VerSupported field MUST be non-zero. The VerSupported bits are numbered from 0 as the most significant bit. Bit 0 indicates support of Sub-Version zero, bit 1 indicates support of Sub-Version one, etc.

Vendor-ID: 4 bytes in length. Vendor ID, as defined in RADIUS [RFC2865].

Capabilities: 32 bits in length. Flags that indicate the support of particular capabilities by the sender of the Hello. No Capabilities are defined in this document and so implementations will set this field to zero. The Capabilities field of the Hello TLV MUST be checked before any other TLVs in the Hello because capabilities defined in the future might extend existing TLVs or permit new TLVs.

After the exchange of Hello messages, the CP and UP each perform a logical AND of the Sub-Version supported by the CP and the UP and separately perform a logical AND of the Capabilities bits fields for the CP and the UP.
If the result of the AND of the Sub-Versions supported is zero, then no session can be established and the connection is torn down. If the result of the AND of the Sub-Versions supported is non-zero, then the session uses the highest Sub-Version supported by both the CP and UP.

For example, if one side supports Sub-Versions 1, 3, 4, and 5 (VerSupported = 0x5C000000) and the other side supports 2, 3, and 4 (VerSupported = 0x38000000) then 3 and 4 are the Sub-Versions in common and 4 is the highest Sub-Version supported by both sides. So Sub-Version 4 is used for the session that has been negotiated.

The result of the logical AND of the Capabilities bits will show what additional capabilities both sides support. If this result is zero, there are no such capabilities so none can be used during the session. If this result is non-zero, it shows the additional capabilities that can be used during the session. The CP and the UP MUST NOT use a capability unless both advertise support.

7.5 The Keep Alive TLV

The Keep Alive TLV is defined to be carried in the Hello message. It provides timing information for the keep alive feature. The format of Hello TLV is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Keepalive   | DeadTimer     |            Reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 40: Keep Alive TLV

Where:

The TLV type: 102.

The value of length: 4 octets.

Keepalive (8 bits): Indicates the maximum period of time (in seconds) between two consecutive S-CUSP messages sent by the sender of the message containing this TLV as an unsigned integer. The minimum value for the Keepalive is 1 second. When set to 0, once the session is established, no further Keepalive messages are sent to the remote peer. A RECOMMENDED value for the Keepalive frequency is 30 seconds.

DeadTimer (8 bits in length): Specifies the amount of time as an unsigned integer number of seconds after the expiration of which...
the S-CUSP peer can declare the session with the sender of the Hello message to be down if no S-CUSP message has been received. The DeadTimer SHOULD be set to 0 and MUST be ignored if the Keepalive is set to 0. A RECOMMENDED value for the DeadTimer is 4 times the value of the Keepalive.

The Reserved bits MUST be sent as zero and ignored on receipt.

7.6 The Error Information TLV

The Error Information TLV is a common TLV that can be used in many Response (e.g., Update_Response message) and ACK messages (e.g., Addr_Allocation_Ack message, etc.). It is used to convey the information about an error in the received S-CUSP message. The format of the Error Information TLV is as follows:

```
+---------------------------------------------------------------+
| Message-Type | Reserved             | TLV-Type             |
+---------------------------------------------------------------+
|                           Error Code                          |
+---------------------------------------------------------------+
```

Figure 41: Error Information TLV

Where:

The TLV type: 101.

The value of length: 8 octets.

Message-Type(1 byte): This parameter is the message type of the message containing an error.

Reserved (1 byte): MUST be sent as zero and ignored on receipt.

TLV-Type (2 bytes): Indicates which TLV caused the error.

Error Code: 4 bytes in length. Indicate the specific Error Code (see Section 9.5).

7.7 BAS Function TLV

The BAS Function TLV is used by a CP to control the access mode, authentication methods, and other related functions of an interface.
The format of the BAS Function TLV value part is as follows:

```
+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          If-Index                             |
+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Access-Mode  |  Auth-Method4 |  Auth-Method6 |    Reserved   |
+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Flags                             |
+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         sub-TLVs (optional)                   |
+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 42: BAS Function TLV

Where:

The TLV type: 1.

The value of length: variable.

If-Index: 4 bytes in length, a unique identifier of an interface of a BNG.

Access-Mode: 1 byte in length, indicates the access mode of the interface. This document defines following values:

- 0: Layer 2 subscriber;
- 1: Layer 3 subscriber;
- 2: Layer 2 leased line;
- 3: Layer 3 leased line;
- 4-255: Reserved.

Auth-Method4: 1 byte in length, for IPv4 scenario, it indicates the authentication on this interface; this field is defined as a bitmap, this document defines following values (other bits are reserved and MUST be sent as zero and ignored on receipt):

- 0x1: PPPoE authentication;
- 0x2: DOT1X authentication;
- 0x4: Web authentication;
- 0x8: Web fast authentication;
- 0x10: Binding authentication.

Auth-Method6: 1 byte in length, for IPv6 scenario, it indicates the authentication on this interface; this field is defined as a bitmap, this document defines following values (other bits are
reserved and MUST be sent as zero and ignored on receipt):

0x1: PPPoE authentication;
0x2: DOT1X authentication;
0x4: Web authentication;
0x8: Web fast authentication;
0x10: Binding authentication;

sub-TLVs:
The IF-Desc sub-TLV can be carried.
If-Desc sub-TLV: carries the interface information.

The flags field is defined as follows:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                MBZ                            |Y|X|P|I|N|A|S|F|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 43: Interface Flags

Where:

F (IPv4 Trigger) bit: Indicates whether IPv4 packets can trigger a subscriber to go online. 1: enabled, 0: disabled.

S (IPv6 Trigger) bit: Indicates whether IPv6 packets can trigger a subscriber to go online. 1: enabled, 0: disabled.

A (ARP Trigger) bit: Indicates whether ARP packets can trigger a subscriber to go online. 1: enabled, 0: disabled.

N (ND Trigger) bit: Indicates whether ND packets can trigger a subscriber to go online. 1: enabled, 0: disabled.

I (IPoE-Flow-Check): Used for UP detection. IPoE 1: Enable traffic detection. 0: Disable traffic detection.

P (PPP-Flow-Check) bit: Used for UP detection. PPP 1: Enable traffic detection. 0: Disable traffic detection.

X (ARP-Proxy) bit: 1: The interface is enabled with ARP proxy and can process ARP requests across different Port+VLANs. 0: The ARP proxy is not enabled on the interface, and only the ARP requests of the same Port+VLAN are processed.
Y (ND-Proxy) bit: 1: The interface is enabled with ND proxy and can process ND requests across different Port+VLANs. 0: The ND proxy is not enabled on the interface, and only the ND requests of the same Port+VLAN are processed.

MBZ: Reserved bits that MUST be sent as zero and ignored on receipt.

7.8 Routing TLVs

Normally, after an S-CUSP session is established between a UP and a CP, the CP will allocate one or more blocks of IP addresses to the UP. Those IP addresses will be allocated to subscribers who will dial-up to the UP. In order to make sure that other nodes within the network learn how to reach those IP addresses, the CP needs to install one or more routes that can reach those IP addresses on the UP and notify the UP to advertise the routes to the network.

The Routing TLVs are used by a CP to notify a UP of the network routing. They can be carried in the Update_Request message and Sync_Data message.

7.8.1 IPv4 Routing TLV

The IPv4 Routing TLV is used to carry IPv4 network routing related information.
The format of the TLV value part is as below:

```
+---------------------------------+
<table>
<thead>
<tr>
<th>User ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dest-Address</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>Next-Hop</td>
</tr>
<tr>
<td>Out-If-Index</td>
</tr>
<tr>
<td>Cost</td>
</tr>
<tr>
<td>Tag</td>
</tr>
<tr>
<td>Route Type</td>
</tr>
<tr>
<td>sub-TLVs (optional)</td>
</tr>
</tbody>
</table>
+---------------------------------+
```

Figure 44: IPv4 Routing TLV

Where:

The TLV Type: 7

The TLV Length: Variable

User-ID: 4 bytes in length. Carries the user identifier. This field is filled with all Fs when a non-user route is delivered to the UP.

Dest-Address (IPv4-Address type): Identifies the destination address.

Next-Hop: (IPv4-Address type): Identifies the next hop address.

Out-If-Index (4 bytes): Indicates the interface index.

Cost (4 bytes): The cost value of the route.

Tag (4 bytes): The tag value of the route.

Route-Type (2 bytes): Indicates the route type. The options are as follows:
0: User host route
1: Radius authorization FrameRoute
2: Network segment route
3: Gateway route
4: Radius authorized IP route
5: L2TP LNS side user route

Advertise-Flag: 1 bit. Indicates whether the route should be advertised by the UP. Following flags are defined:
0: Not advertised,
1: advertised.

sub-TLVs: The VRF-Name and/or If-Desc sub-TLVs can be carried.
VRF-Name sub-TLV: indicates which VRF the route belongs to.
If-Desc sub-TLV: carries the interface information.

The Reserved field MUST be sent as zero and ignored on receipt.

7.8.2 IPv6 Routing TLV

The IPv6 Routing TLV is used to carry IPv6 network routing information.

The format of this TLV is as follows:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                          User ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
˜                          IPv6 Dest-Address                    ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
˜                          IPv6 Next-Hop                        ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                          Out-If-Index                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                          Cost                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                          Tag                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|        Route Type             |          Reserved           |A|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
˜                          sub-TLVs (optional)                  ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 45: IPv6 Routing TLV

Where:

Hu, et al [Page 84]
The TLV Type: 7

The TLV Length: Variable

User-ID: 4 bytes in length. Carries the user identifier. This field is filled with all Fs when a non-user route is delivered to the UP.

IPv6 Dest-Address (IPv6-Address type): Identifies the destination address.

IPv6 Next-Hop: (IPv6-Address type): Identifies the next hop address.

Out-If-Index (4 bytes): Indicates the interface index.

Cost (4 bytes): The cost value of the route.

Tag (4 bytes): The tag value of the route.

Route-Type: (2 bytes): Indicates the route type. The options are as follows:

0: User host route
1: Radius authorization FrameRoute
2: Network segment route
3: Gateway route
4: Radius authorized IP route
5: L2TP LNS side user route

Advertise-Flag: 1 bit. Indicates whether the route should be advertised by the UP. Following flags are defined:

0: Not advertised,
1: advertised.

sub-TLVs: If-Desc and VRF-Name sub-TLVs can be carried.

VRF-Name sub-TLV: Indicates the VRF to which the subscriber belongs.
If-Desc sub TLV: carries the interface information.

The Reserved field MUST be sent as zero and ignored on receipt.

7.9 Subscriber TLVs

The Subscriber TLVs are defined for a CP to send the basic information about a user to a UP.
7.9.1 Basic Subscriber TLV

The Basic Subscriber TLV is used to carry the basic common information for all kinds of access subscribers. It is carried in an Update_Request message.

The format of the Basic Subscriber TLV value part is as follows:

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Session ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User MAC                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| User MAC | Oper ID | Reserved |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Access Type | Sub-access Type | Account Type | Address Family |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| C-VID | P-VID |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Detect Times | Detect Interval |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| If-Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
- sub-TLVs (optional) -
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 46: Basic Subscriber TLV

Where:

- The TLV Type: 2.
- The TLV Length: variable in length.
- User-ID (4 bytes): The identifier of a subscriber.
- User-Mac (MAC-Addr type): The MAC Address of a subscriber.
- Oper-ID (1 byte): Indicates the ID of an operation performed by a user. This field is carried in the response from the UP.
- Reserved (1 byte): MUST be sent as zero and ignored on receipt.
Access-Type (1 byte):

1: PPP access (PPP [RFC1661])
2: PPP over Ethernet over ATM access (PPPoEoA)
3: PPP over ATM access (PPPoA [RFC3336])
4: PPP over Ethernet access (PPPoE [RFC2516])
5: PPPoE over VLAN access (PPPoEoVLAN [RFC2516])
6: PPP over LNS access (PPPoLNS)
7: IP over Ethernet DHCP access (IPoE_DHCP)
8: IP over Ethernet EAP authentication access (IPoE_EAP)
9: IP over Ethernet Layer 3 access (IPoE_L3)
10: IP over Ethernet Layer 2 Static access (IPoE_L2_STATIC)
11: Layer 2 Leased Line access (L2_Leased_Line)
12: Layer 2 VPN Leased Line access (L2VPN_Leased_Line)
13: Layer 3 Leased line access (L3_Leased_Line)
14: Layer 2 Leased line Sub-User access (L2_Leased_Line_SUB_USER)
15: L2TP LAC tunnel access (L2TP_LAC)
16: L2TP LNS tunnel access (L2TP_LNS)

Sub-Access-Type (1 byte): Indicates whether PPP termination or PPP relay is used.
0: Reserved
1: PPP Relay (for LAC)
2: PPP termination (for LNS)

Account-Type(1 byte):
0: Collects statistics on IPv4 and IPv6 traffic of terminals independently.
1: Collects statistics on IPv4 and IPv6 traffic of terminals.

Address Family (1 byte)
1: IPv4
2: IPv6
3: dual stack

C-VID (VLAN-ID): Indicates the inner VLAN ID. The value 0 indicates that the VLAN ID is invalid. The default value of PRI is 7, the value of DEI is 0, and the value of VID is 1˜4094. The PRI value can also be obtained by parsing terminal packets.

P-VID (VLAN-ID): Indicates the outer VLAN ID. The value 0 indicates that the VLAN ID is invalid. The format is the same as that for C-VID.

Detect-Times (2 bytes): Number of detection timeout times. The value 0 indicates that no detection is performed.

Detect-Interval (2 bytes): Detection interval in seconds.
If-Index (4 bytes): Interface index.

Sub-TLVs: VRF-Name sub-TLV and If-Desc sub-TLV can be carried.
VRF-Name sub-TLV: Indicates the VRF to which the subscriber belongs.
If-Desc sub-TLV: carries the interface information.

The Reserved field MUST be sent as zero and ignored on receipt.

7.9.2 PPP Subscriber TLV

The PPP Subscriber TLV is defined to carry PPP information of a User from a CP to a UP. It will be carried in an Update_Request message when PPPOE or L2TP access is used.

The format of the TLV value part is as follows:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        MSS                    |        Reserved             |M|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        MRU                    |        Reserved               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Magic Number                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Peer Magic Number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 47: PPP Subscriber TLV

Where:

The TLV type: 3.

The TLV length: 12 octets.

User-ID (4 bytes): The identifier of a subscriber.

MSS-Value (2 bytes): Indicates the MSS value (in bytes).

MSS-Enable (M) (1 bit): Indicates whether the MSS is enabled.
0: Disabled.
1: Enabled.

MRU (2 bytes): PPPOE local MRU (in bytes).
Magic-Number (4 bytes): Local magic number in PPP negotiation packets.

Peer-Magic-Number (4 bytes): Remote peer magic number.

The Reserved fields MUST be sent as zero and ignored on receipt.

7.9.3 IPv4 Subscriber TLV

The IPv4 Subscriber TLV is defined to carry IPv4 related information for a BNG user. It will be carried in an Update_Request message when IPv4 IPoE, or PPPoE access is used.

The format of the TLV value part is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User IPv4 Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Gateway IPv4 Address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          MTU                  |   Reserved            |U|E|W|P|                          VRF-Name sub-TLV                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                          VRF-Name sub-TLV                     ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 48: IPv4 Subscriber TLV

Where:

The TLV type: 4.

The TLV length: variable.

User-ID (4 bytes): The identifier of a subscriber.

User-IPv4 (IPv4-Address): The IPv4 address of the subscriber.

Gateway-IPv4 (IPv4-Address): The gateway address of the subscriber.

Portal Force (P) (1 bit): Push advertisement, 0: off, 1: on.

Echo-Enable (E) (1 bit): UP returns ARP Req or PPP Echo. 0: off, 1: on.

IPv4-URPF (U) (1 bit): User Unicast Reverse Path Forwarding (URPF) flag. 0: off, 1: on.

MTU 2 bytes MTU value. The default value is 1500.

VRF-Name Sub-TLV: Indicates the subscriber belongs to which VRF.

The Reserved field MUST be sent as zero and ignored on receipt.

7.9.4 IPv6 Subscriber TLV

The IPv6 Subscriber TLV is defined to carry IPv6 related information for a BNG user. It will be carried in an Update_Request message when IPv6 IPoE, or PPPoE access is used.

The format of the TLV value part is as follows:

```
<table>
<thead>
<tr>
<th>User ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>User PD-Address (IPv6 Address List sub-TLV)</td>
</tr>
<tr>
<td>Gateway ND-Address (IPv6 Address List sub-TLV)</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>User IANA Address</td>
</tr>
<tr>
<td>IPv6 Interface ID</td>
</tr>
<tr>
<td>IPv6 Interface ID (cont.)</td>
</tr>
<tr>
<td>MTU</td>
</tr>
<tr>
<td>VRF Name sub-TLV (optional)</td>
</tr>
</tbody>
</table>
```

Figure 49: IPv6 Subscriber TLV

Where:

The TLV type: 5.

The TLV length: variable.
User-ID (4 bytes): The identifier of a subscriber.

User PD-Addresses (IPv6 Address List): Carries a list of Prefix Delegation (PD) addresses of the subscriber.

User ND-Addresses (IPv6 Address List): Carries a list of Neighbor Discovery (ND) addresses of the subscriber.

User IANA-Address (IPv6-Address): The IANA address of the subscriber.

IPv6 Interface ID (8 bytes): The identifier of an IPv6 interface.

Portal Force 1 bit (P): Push advertisement, 0: off, 1: on.

Web-Force 1 bit (W): Push IPv6 Web, 0: off, 1: on.

Echo-Enable 1 bit (E): The UP returns ARP Req or PPP Echo. 0: off; 1: on.

IPv6-URPF 1 bit (U): User Reverse Path Forwarding (URPF) flag, 0: off; 1: on.

MTU (2 bytes): The MTU value. The default value is 1500.

VRF-Name Sub-TLV: Indicates the VRF to which the subscriber belongs.

The Reserved field MUST be sent as zero and ignored on receipt.

7.9.5 IPv4 Static Subscriber Detect TLV

The IPv4 Static Subscriber Detect TLV is defined to carry IPv4 related information for a static access subscriber. It will be carried in an Update_Request message when IPv4 static access on a UP needs to be enabled.
The format of the TLV value part is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          If-Index                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           C-VID               |           P-VID               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Gateway Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User MAC                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User MAC (cont.)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Reserved                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       sub-TLVs (optional)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 50: IPv4 Static Subscriber TLV

Where:

The TLV type: 6.

The TLV length: variable.

If-Index (4 bytes): The interface index of the interface from which the subscriber will dial-up.

C-VID (VLAN-ID): Indicates the inner VLAN ID. The value 0 indicates that the VLAN ID is invalid. A valid value is 1-4094.

P-VID (VLAN-ID): Indicates the outer VLAN ID. The value 0 indicates that the VLAN ID is invalid. The format is the same as that of the C-VID. A valid value is 1-4094. For a single-layer VLAN, set this parameter to PeVid.

User Address (IPv4-Addr): The user’s IPv4 address.

Gateway Address (IPv4-Addr): The gateway’s IPv4 Address.

User-MAC (MAC-Addr type): The MAC address of the subscriber.

Sub-TLVs: VRF-Name and If-Desc sub-TLVs may be carried.

VRF-Name sub-TLV: Indicates the VEF to which the subscriber belongs.

If-Desc sub-TLV: Carries the interface information.
The Reserved field MUST be sent as zero and ignored on receipt.

7.9.6 IPv6 Static Subscriber Detect TLV

The IPv6 Static Subscriber Detect TLV is defined to carry IPv6 related information for a static access subscriber. It will be carried in an Update_Request message when needed to enable IPv6 static subscriber detection on a UP.

The format of the TLV value part is as follows:

```
+---------------+-----------------+-----------------+
| If-Index      | C-VID           | P-VID           |
+---------------+-----------------+-----------------+
| User Address  |                 |                 |
+---------------+-----------------+-----------------+
| Gateway Address|                |                 |
+---------------+-----------------+-----------------+
| User MAC      |                 | Reserved        |
+---------------+-----------------+-----------------+
| User MAC (cont.)|              | sub-TLVs (optional) |
+---------------+-----------------+-----------------+
```

Figure 51: IPv6 Static Subscriber Detect TLV

Where:

The TLV type: 6.

The TLV length: variable.

If-Index (4 bytes): The interface index of the interface from which the subscriber will dial-up.

C-VID (VLAN-ID): Indicates the inner VLAN ID. The value 0 indicates that the VLAN ID is invalid. A valid value is 1-4094.

P-VID (VLAN-ID): Indicates the outer VLAN ID. The value 0 indicates that the VLAN ID is invalid. The format is the same as that of C-VID. A valid value is 1-4094. For a single-layer VLAN, set this parameter to PeVid.
User Address (IPv6-Address type): The subscriber’s IPv6 address.
Gateway Address (IPv6-Address type): The gateway’s IPv6 Address.
User-MAC (MAC-Addr type): The MAC address of the subscriber.

sub-TLVs: VRF-Name and If-Desc sub-TLVs may be carried
  VRF-Name Sub-TLV: Indicates the VRF to which the subscriber
  belongs.
  If-Desc sub-TLV: Carries the interface information.

The Reserved field MUST be sent as zero and ignored on receipt.

7.9.7 L2TP-LAC Subscriber TLV

The L2TP-LAC Subscriber TLV is defined to carry the related
information for a L2TP LAC access subscriber. It will be carried in
an Update_Request message when L2TP LAC access is used.

The format of the TLV value part is as follows:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Local Tunnel ID          |     Local Session ID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Remote Tunnel ID         |     Remote Session ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 52: L2TP-LAC Subscriber TLV

Where:

- The TLV type: 11.
- The TLV length: 12 octets.
- User-ID (4 bytes): The identifier of a user/subscriber.
- Local-Tunnel-ID (2 bytes): The local ID of the L2TP tunnel.
- Local-Session-ID (2 bytes): The local session ID with the L2TP
tunnel.
- Remote-Tunnel-ID (2 bytes): The remote ID of the L2TP tunnel.
Remote-Session-ID (2 bytes): The remote session ID of the L2TP tunnel.

7.9.8 L2TP-LNS Subscriber TLV

The L2TP-LNS Subscriber TLV is defined to carry the related information for a L2TP LNS access subscriber. It will be carried in an Update_Request message when L2TP LNS access is used.

The format of the TLV value part is as follows:

```
0                   1                   2                   3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Local Tunnel ID          |     Local Session ID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Remote Tunnel ID         |     Remote Session ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 53: L2TP-LNS Subscriber TLV

Where:

The TLV type: 12.

The TLV length: 12 octets.

User-ID (4 bytes): The identifier of a user/subscriber.

Local-Tunnel-ID (2 bytes): The local ID of the L2TP tunnel.

Local-Session-ID (2 bytes): The local session ID with the L2TP tunnel.

Remote-Tunnel-ID (2 bytes): The remote ID of the L2TP tunnel.

Remote-Session-ID (2 bytes): The remote session ID of the L2TP tunnel.

7.9.9 L2TP-LAC Tunnel TLV

The L2TP-LAC Tunnel TLV is defined to carry the L2TP LAC tunnel related information. It will be carried in the Update_Request message when L2TP LAC access is used.
The format of the TLV value part is as follows:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Local Tunnel ID         |       Remote Tunnel ID        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Source Port         |        Destination Port       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                            Tunnel Source Address
                            Tunnel Destination Address
                            VRF Name sub-TLV
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 54: L2TP-LAC Tunnel TLV

Where:

- **The TLV type**: 13.
- **The TLV length**: variable.
- **Local-Tunnel-ID (2 bytes)**: The local ID of the L2TP tunnel.
- **Remote-Tunnel-ID (2 bytes)**: The remote ID of the L2TP tunnel.
- **Source-Port (2 bytes)**: The source UDP port number of an L2TP subscriber.
- **Dest-Port (2 bytes)**: The destination UDP port number of an L2TP subscriber.
- **Source-IP (IPv4/v6)**: The source IP address of the tunnel.
- **Dest-IP (IPv4/v6)**: The destination IP address of the tunnel.
- **VRF-Name Sub-TLV**: The VRF name to which the L2TP subscriber tunnel belongs.

### 7.9.10 L2TP-LNS Tunnel TLV

The L2TP-LNS Tunnel TLV is defined to carry the L2TP LNS tunnel related information. It will be carried in the Update_Request message when L2TP LNS access is used.
The format of the TLV value part is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Local Tunnel ID        |       Remote Tunnel ID        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Source Port            |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                            Tunnel Source Address
                            Tunnel Destination Address
                            VRF Name sub-TLV
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 55: L2TP-LNS Tunnel TLV

Where:

- The TLV type: 14.
- The TLV length: variable.
- Local-Tunnel-ID (2 bytes): The local ID of the L2TP tunnel.
- Remote-Tunnel-ID (2 bytes): The remote ID of the L2TP tunnel.
- Source-Port (2 bytes): The source UDP port number of an L2TP subscriber.
- Dest-Port (2 bytes): The destination UDP port number of an L2TP subscriber.
- Source-IP (IPv4/v6): The source IP address of the tunnel.
- Dest-IP (IPv4/v6): The destination IP address of the tunnel.
- VRF-Name Sub-TLV: The VRF name to which the L2TP subscriber tunnel belongs.

### 7.9.11 Update Response TLV

The Update Response TLV is used to return the operation result of an update request. It is carried in the Update_Response message as a response to the Update_Request message.
The format of Update Response TLV is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User-ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| User-Trans-ID |   Oper-Code   |   Oper-Result |  Reserved     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Error-Code                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 56: Update Response TLV

Where:

- The TLV type: 302.
- The TLV length: 12.
- User-ID (4 bytes): A unique identifier of an user/subscriber.
- User-Trans-ID (1 byte): In the case of dual-stack access or when modifying a session, User-Trans-ID is used to identify a user operation transaction. It is used by the CP to correlate a response to a specific request.
- Oper-Result (1 byte): Operation Result. 0: Success, Others: Failure.
- Error-Code (4 bytes): Operation failure cause code. For details, see Section 9.5.
- The Reserved field MUST be sent as zero and ignored on receipt.

7.9.12 Subscriber Policy TLV

The Subscriber Policy TLV is used to carry the policies that will be applied to a subscriber. It is carried in the Update_Request message.
The format of the TLV value part is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   I-Priority  |   E-Priority  |   Reserved                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                          sub-TLVs                             ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 57: User QoS Policy Information TLV

Where:

The TLV type: 6.

The TLV length: variable.

User-ID (4 bytes): The identifier of a user/subscriber.

Ingress-Priority (1 byte): Indicates the upstream priority. The value range is 0~7.

Egress-Priority (1 byte): Indicates the downstream priority. The value range is 0~7.

sub-TLVs: The sub-TLVs that are present can occur in any order.

- Ingress-CAR sub-TLV: Upstream CAR.
- Egress-CAR sub-TLV: Downstream CAR.
- Ingress-QoS-Profile sub-TLV: Indicates the name of the QoS-Profile profile in the upstream direction.
- Egress-QoS-Profile Sub-TLV: Indicates the name of the QoS-Profile profile in the downstream direction.
- User-ACL-Policy Sub-TLV: All ACL user policies, including v4ACLIN, v4ACLOUT, v6ACLIN, v6ACLOUT, v4WEBACL, v6WEBACL, v4SpecialACL, and v6SpecialACL.
- Multicast-Profile4 Sub-TLV: IPv4 multicast policy name.
- Multicast-Profile6 Sub-TLV: IPv6 multicast policy name.
- NAT-Instance Sub-TLV: Indicates the instance ID of a NAT user.
The Reserved field MUST be sent as zero and ignored on receipt.

7.9.13 Subscriber CGN Port Range TLV

The Subscriber CGN Port Range TLV is used to carry the NAT public address and port range. It will be carried in the Update_Response message when CGN is used.

The format of this TLV is as follows:

```
+-------+-------+-------+-------+-------+-------+-------+-------+
|       |       |       |       |       |       |       |       |
| User ID |       |       |       | NAT-Port-Start | NAT-Port-End |       |       |
|       |       |       |       |       |       |       |       |
|       |       |       |       | NAT-Address |       |       |       |
```

Figure 58: Subscriber CGN Port Range TLV

Where:

- The TLV type: 15.
- The TLV length: 12 octets.
- User-ID (4 bytes): The identifier of a user/subscriber.
- NAT-Port-Start (2 bytes): The start port number.
- NAT-Port-End (2 bytes): The end port number.
- NAT-Address (4 bytes): The NAT public network address.

7.10 Device Status TLVs

The TLVs in this section are for reporting Interface and Board level information from the UP to the CP.
7.10.1 Interface Status TLV

The Interface Status TLV is used to carry the status information of an interface on a UP. It is carried in a Report message.

The format of the value part of this TLV is as follows:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          If-Index                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          MAC Address (upper part)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| MAC Address (lower part) | Phy-State | Reserved          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          MTU                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          sub-TLVs (optional)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 59: Interface Status TLV

Where:

- The TLV type: 200.
- The TLV length: variable.
- If-Index (4 bytes): Indicates the interface index.
- MAC-Address (MAC-Addr type): Interface MAC address.
- Phy-State (1 byte): Physical status of the interface. 0: down, 1: Up
- MTU (4 bytes): Interface MTU value.
- sub-TLVs: The If-Desc and VRF-Name sub-TLVs can be carried.
- The Reserved field MUST be sent as zero and ignored on receipt.

7.10.2 Board Status TLV

The Board Status TLV is used to carry the status information of a board on an UP. It is carried in a Report message.
The format of Board Status TLV is as follows:

```
<table>
<thead>
<tr>
<th>Board-Type</th>
<th>Board-State</th>
<th>Reserved</th>
<th>Chassis</th>
</tr>
</thead>
</table>
```

Where:

- The TLV type: 201.
- The TLV length: 8 octets.
- Chassis (1 byte): The chassis number of the Board.
- Slot (1 byte): The slot number of the Board.
- Sub-Slot (1 byte): The sub-slot number of the Board.
- Board-Type (1 byte):
  1: CGN Service Process Unit (SPU) board.
  2: Line Process Unit (LPU) Board.
- Board-State (1 byte):
  0: Normal.
  1: Abnormal.

The reserved field MUST be sent as zero and ignored on receipt.

7.11 CGN TLVs

7.11.1 Address Allocation Request TLV

The Address Allocation Request TLV is used to request address allocation from CP. An address Pool-Name sub-TLV is carried to indicate from which address pool to allocate addresses. The Address Allocation Request TLV is carried in the Addr_Allocation_Req message.
The format of the value part of this TLV is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                          Pool-Name sub-TLV                    ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 61: Address Allocation Request TLV

Where:

The TLV type: 400.

The TLV length: variable.

Pool-Name sub-TLV: Indicates from which Address pool to allocate address.

7.11.2 Address Allocation Response TLV

The Address Allocation Response TLV is used to return the address allocation result, it is carried the Addr_Allocation_Ack message.

The value part of the Address Allocation Response TLV is formatted as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Lease Time                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        IPv4 Addr and Mask                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        IPv4 Addr and Mask continued           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Error-Code                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                        Pool-Name sub-TLV                      ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 62: Address Assignment Response TLV

Where:

The TLV type: 401.

The TLV length: variable.
Lease Time (4 bytes): Duration of address lease in seconds.

IPv4 Addr and Mask (IPv4-Address type): The allocated IPv4 address.


  0: Success.

  1: Failure.

  3001 (Pool-Mismatch): The corresponding address pool cannot be found.

  3002 (Pool-Full): The address pool is fully allocated and no address segment is available.

Pool-Name sub-TLV: A Pool-Name sub-TLV to indicate from which Address pool the address is allocated.

7.11.3 Address Renewal Request TLV

The Address Renewal Request TLV is used to request address renewal from the CP. It is carried the Addr_Renew_Req message.

The format of this TLV value is as follows:

```
+---------------+---------------+---------------+---------------+
| IPv4 Addr and Mask | IPv4 Addr and Mask continued |
| Pool-Name sub-TLV |
+---------------+---------------+---------------+---------------+
```

Figure 63: Address Renewal Request TLV

Where:

The TLV type: 402.

The TLV length: variable.

IPv4 Addr and Mask (IPv4-Addr): The IPv4 address to be renewed.

Pool Name sub-TLV: A Pool-Name sub-TLV to indicate from which
Address pool to renew the address.

7.11.4 The Address Renewal Response TLV

The Address Renewal Response TLV is used to return the address renewal result. It is carried in the Addr_Renew_Ack message.

The format of this TLV value is as follows:

```
+---------------+---------------+---------------+---------------+
| IPv4 Address and Mask | IPv4 Address and Mask continued | Error-Code | Pool-Name sub-TLV |
+---------------+---------------+---------------+---------------+
```

Figure 64: Address Renewal Response TLV

Where:

The TLV type: 403.

The TLV length: variable.

Client-IP (IPv4-Address type): The renewed IPv4 address.

Error Code (4 bytes): Indicates success or an error:

0: Renew success.

1: Renew failed.

3001 (Pool-Mismatch): The corresponding address pool cannot be found.

3002 (Pool-Full): The address pool is fully allocated and no address segment is available.

3003 (Subnet-Mismatch): The address pool subnet cannot be found.

3004 (Subnet-Conflict): Subnets in the address pool have been assigned to other clients.
7.11.5 Address Release Request TLV

The Address Release Request TLV is used to release an IPv4 address. It is carried in the Addr_Release_Req message.

The value part of this TLV is formatted as follows:

```
+-------------+-------------+-------------+-------------+
| IPv4 Address and Mask | IPv4 Address and Mask continued |
+-------------+-------------+-------------+-------------+
| Pool-Name sub-TLV |                       |
+-------------+-------------+-------------+-------------+
```

Figure 65: Address Release Request TLV

Where:

- The TLV type: 404.
- The TLV length: variable.
- IPv4 Address and Mask (IPv4-Address type): The IPv4 address be released.
- Pool-Name sub-TLV: A Pool-Name Sub-TLV to indicate from which Address pool to release the address.

7.11.6 The Address Release Response TLV

The Address Release Response TLV is used to return the address release result. It is carried in the Addr_Release_Ack message.
The format of this TLV is as follows:

\[
\begin{array}{cccccccc}
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 \\
\hline
\end{array}
\]

Figure 66: Address Renewal Response TLV

Where:

The TLV type: 405.

The TLV length: variable.

Client-IP (IPv4-Address type): The released IPv4 address.


\begin{itemize}
  \item 0: Address release success.
  \item 1: Address release failed.
  \item 3001 (Pool-Mismatch): The corresponding address pool cannot be found.
  \item 3003 (Subnet-Mismatch): The address cannot be found.
  \item 3004 (Subnet-Conflict): The address has been allocated to another subscriber.
\end{itemize}

Pool-Name sub-TLV: A Pool-Name Sub-TLV to indicate from which address pool to release the address.

7.12 Event TLVs
7.12.1. Subscriber Traffic Statistics TLV

The Subscriber Traffic Statistics TLV is used to return the traffic statistics of a user/subscriber. The format of this TLV is as follows:

```
+----------------------------------+-
|                User-ID             |
+----------------------------------+-
|                Statistics Type     |
+----------------------------------+-
|                Ingress Packets    |
|    (upper part)                  |
+----------------------------------+-
|                Ingress Packets    |
|    (lower part)                  |
+----------------------------------+-
|                Ingress Bytes      |
|    (upper part)                  |
+----------------------------------+-
|                Ingress Bytes      |
|    (lower part)                  |
+----------------------------------+-
|                Ingress Loss       |
|    Packets (upper part)          |
+----------------------------------+-
|                Ingress Loss       |
|    Packets (lower part)          |
+----------------------------------+-
|                Ingress Loss Bytes|
|    (upper part)                  |
+----------------------------------+-
|                Ingress Loss Bytes|
|    (lower part)                  |
+----------------------------------+-
|                Egress Packets     |
|    (upper part)                  |
+----------------------------------+-
|                Egress Packets     |
|    (lower part)                  |
+----------------------------------+-
|                Egress Bytes       |
|    (upper part)                  |
+----------------------------------+-
|                Egress Bytes       |
|    (lower part)                  |
+----------------------------------+-
|                Egress Loss        |
|    Packets (upper part)          |
+----------------------------------+-
|                Egress Loss        |
|    Packets (lower part)          |
+----------------------------------+-
|                Egress Loss Bytes  |
|    (upper part)                  |
+----------------------------------+-
|                Egress Loss Bytes  |
|    (lower part)                  |
```

Figure 67: Subscriber Traffic Statistics TLV

Where:

Hu, et al                                      [Page 108]
The TLV type: 300.
The TLV length: 72 octets.
User-ID (4 bytes): The user/subscriber identifier.
Statistics-Type (4 bytes): Traffic type. It can be one of the following options:
  0: IPv4 traffic.
  1: IPv6 traffic.
  2: Dual stack traffic.
Ingress Packets (8 bytes): The number of the packets in upstream direction.
Ingress Bytes (8 bytes): The bytes of the upstream traffic.
Ingress Loss Packets (8 bytes): The number of the lost packets in upstream direction.
Ingress Loss Bytes (8 bytes): The bytes of the lost upstream packets.
Egress Packets (8 bytes): The number of the packets in downstream direction.
Egress Bytes (8 bytes): The bytes of the downstream traffic.
Egress Loss Packets (8 bytes): The number of the lost packets in downstream direction.
Egress Loss Bytes (8 bytes): The bytes of the lost downstream packets.

7.12.2 Subscriber Detection Result TLV

The Subscriber Detection Result TLV is used to return the detection result of a subscriber. Subscriber detection is a function to detect whether a subscriber is online or not. The result can be used by the CP to determine how to deal with the subscriber session. (e.g., delete the session if detection failed).
The format of this TLV value part is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          User-ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Detect Type   | Detect Result |          Reserved             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 68: Subscriber Detection Result TLV

Where:

The TLV type: 301.

The TLV length: 8 octets.

User-ID (4 bytes): A user/subscriber identifier.

Detect-Type (1 byte):

0: IPv4 detection.

1: IPv6 detection.

2: PPP detection.

Detect-Result (1 byte):

0: Indicates that the detection is successful.

1: Detection failure. The UP needs report only when the detection fails.

The Reserved field MUST be sent as zero and ignored on receipt.

7.13 Vendor TLV

The Vendor ID TLV occurs as the first TLV in the Vendor message (Section 6.6). It provides a Sub-Type that effectively extends the message type in the message header, provides for versioning of vendor TLVs, and can accommodate sub-TLVs.
The value part of the Vendor TLV is formatted as follows:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Vendor ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Sub-Type            |       Sub-Type-Version        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                      sub-TLVs (optional)                      ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 69: Vendor TLV

Where:

- The TLV type: 1024.
- The TLV length: variable.
- Vendor-ID (4 bytes): Vendor ID ass defined in RADIUS [RFC2865].
- Sub-Type (2 bytes): Used by the Vendor to distinguish multiple different vendor messages.
- Sub-Type-Version (2 bytes): Used by the Vendor to distinguish different versions of a Vendor Defined message sub-type.
- Sub-TLVs (variable): Sub-TLVs as specified by the vendor.

Since Vendor code will be handling the TLV after the Vendor ID field is recognized, the remainder of the TLV value can be organized however the vendor wants. But it is desirable for a vendor to be able to define multiple different vendor messages and to keep track of different versions of its vendor defined messages. Thus, it is RECOMMENDED that the vendor assign a Sub-Type value for each vendor message that it defines different from other Sub-Type values that vendor has used. Also, when modifying a vendor defined message in a way potentially incompatible with a previous definition, the vendor SHOULD increase the value it is using in the Sub-Type-Version field.
8. Implementation Status

RFC Editor: Please remove this section before publication.

This section discusses the status of implementations that have provided information and the testing of this protocol at the time of posting of this Internet-Draft, and is based on the proposal in [RFC7942] ("Improving Awareness of Running Code: The Implementation Status Section"). The description of implementations in this section is intended to assist in processing drafts to RFCs.

Please note that the listing of any individual implementation or test results here does not imply endorsement by the RFC Editor or the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their testing or features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers ... to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature."

8.1 Implementations

Information on three S-CUSP implementations appears below.

8.1.1 Huawei Technologies

Name: Cloud-based BNG.

Maturity: Production.

Coverage: According to S-CUSP protocol.

Contact information:
Zhouyi Yu: yuzhouyi@huawei.com

Date: 2018-11-01
8.1.2 ZTE

Name: ZXR10 V6000 vBRAS

Maturity: Production

Coverage: According to S-CUSP protocol.

Contact information:
Yong Chen: 10056167@zte.com.cn
Huaibin Wang: 10008729@zte.com.cn

Date: 2018-12-01

8.1.3 H3C

Name: CUSP protocol for BRAS Control Plane and User Plane Separation

Maturity: Research

Coverage: According to S-CUSP protocol

Contact information: mengdan@h3c.com; liuhanlei@h3c.com

Date: 2019-1-30

8.2 Hackathon

Successful use of the protocol at the IETF-102 Hackathon, Montreal, Quebec, in 2018.

Hackathon Project: Control Plane and User Plane Separation BNG control channel Protocol (CUSP)

Champions: Zhenqiang Li, Michael Wang

8.3 EANTC Testing

EANTC (European Advanced Networking Test Center (www.eantc.de)) tested the Huawei implementation. Their summary was as follows: "EANTC tested advanced aspects of the Cloud-based Broadband Network Gateway (vBNG) with a focus on performance, scalability and high availability with up to 20 Million emulated subscribers. The solution performed very well across all test scenarios."

See report at
9. IANA Considerations

IANA is requested to create an "S-CUSP Parameters" web page and include thereon the registries set up in the Sub-Sections below.

9.1 Message Types

IANA is requested to create an S-CUSP Message Types registry on the S-CUSP Parameters Web Page as follows:

Registry Name: S-CUSP Message Types
Registration Procedure: Expert Review
Reference: [this document]

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Section of [this document]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>- Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Hello</td>
<td>6.2.1.</td>
</tr>
<tr>
<td>2</td>
<td>Keepalive</td>
<td>6.2.2.</td>
</tr>
<tr>
<td>3</td>
<td>Sync_Request</td>
<td>6.2.3.</td>
</tr>
<tr>
<td>4</td>
<td>Sync_Begin</td>
<td>6.2.4.</td>
</tr>
<tr>
<td>5</td>
<td>Sync_Data</td>
<td>6.2.5.</td>
</tr>
<tr>
<td>6</td>
<td>Sync_End</td>
<td>6.2.6.</td>
</tr>
<tr>
<td>7</td>
<td>Update_Request</td>
<td>6.2.7.</td>
</tr>
<tr>
<td>8</td>
<td>Update_Response</td>
<td>6.2.8.</td>
</tr>
<tr>
<td>9</td>
<td>Report</td>
<td>6.4.</td>
</tr>
<tr>
<td>10</td>
<td>Event</td>
<td>6.3.</td>
</tr>
<tr>
<td>11</td>
<td>Vendor</td>
<td>6.6.</td>
</tr>
<tr>
<td>12</td>
<td>Error</td>
<td>6.7.</td>
</tr>
<tr>
<td>13-199</td>
<td>- Unassigned</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>Addr_Allocation_Req</td>
<td>6.5.1.</td>
</tr>
<tr>
<td>201</td>
<td>Addr_Allocation_Ack</td>
<td>6.5.2.</td>
</tr>
<tr>
<td>202</td>
<td>Addr_Renew_Req</td>
<td>6.5.3.</td>
</tr>
<tr>
<td>203</td>
<td>Addr_Renew_Ack</td>
<td>6.5.4.</td>
</tr>
<tr>
<td>204</td>
<td>Addr_Release_Req</td>
<td>6.5.5.</td>
</tr>
<tr>
<td>205</td>
<td>Addr_Release_Ack</td>
<td>6.5.6.</td>
</tr>
<tr>
<td>206-254</td>
<td>- Unassigned</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>- Reserved</td>
<td></td>
</tr>
</tbody>
</table>

9.2 TLV Types

IANA is requested to create an S-CUSP TLV Types registry on the S-CUSP Parameters Web Page as follows:
Registry Name: S-CUSP TLV Types
Registration Procedure: Expert Review
Reference: [this document]

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Usage Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>reserved</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>BAS Function</td>
<td>Carries the BNG access functions to be enabled or disabled on specified interfaces.</td>
</tr>
<tr>
<td>2</td>
<td>Basic Subscriber</td>
<td>Carries the basic information about a BNG subscriber.</td>
</tr>
<tr>
<td>3</td>
<td>PPP Subscriber</td>
<td>Carries the PPP information about a BNG subscriber.</td>
</tr>
<tr>
<td>4</td>
<td>IPv4 Subscriber</td>
<td>Carries the IPv4 address of a BNG subscriber.</td>
</tr>
<tr>
<td>5</td>
<td>IPv6 Subscriber</td>
<td>Carries the IPv6 address of a BNG subscriber.</td>
</tr>
<tr>
<td>6</td>
<td>Subscriber Policy</td>
<td>Carries the policy information applied to a BNG subscriber.</td>
</tr>
<tr>
<td>7</td>
<td>IPv4 Routing</td>
<td>Carries the IPv4 network routing information.</td>
</tr>
<tr>
<td>8</td>
<td>IPv6 Routing</td>
<td>Carries the IPv6 network routing information.</td>
</tr>
<tr>
<td>9</td>
<td>IPv4 Static Subscriber Detect</td>
<td>Carries the IPv4 static subscriber detect information.</td>
</tr>
<tr>
<td>10</td>
<td>IPv6 Static Subscriber Detect</td>
<td>Carries the IPv6 static subscriber detect information.</td>
</tr>
<tr>
<td>11</td>
<td>L2TP-LAC Subscriber</td>
<td>Carries the L2TP LAC subscriber information.</td>
</tr>
<tr>
<td>12</td>
<td>L2TP-LNS Subscriber</td>
<td>Carries the L2TP LNS subscriber information.</td>
</tr>
<tr>
<td>13</td>
<td>L2TP-LAC-Tunnel</td>
<td>Carries the L2TP LAC tunnel subscriber information.</td>
</tr>
<tr>
<td>14</td>
<td>L2TP-LNS-Tunnel</td>
<td>Carries the L2TP LNS tunnel subscriber information.</td>
</tr>
<tr>
<td>15</td>
<td>Subscriber CGN Port Range</td>
<td>Carries the public IPv4 address and related port range of a BNG subscriber.</td>
</tr>
<tr>
<td>16-99</td>
<td>unassigned</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>Hello</td>
<td>Used for version and Keep Alive timers negotiation.</td>
</tr>
<tr>
<td>101</td>
<td>Error Information</td>
<td>Carried in the Ack of the control message. Carries the processing result, success, or error.</td>
</tr>
</tbody>
</table>
| 102  | Keep Alive                     | Carried in the Hello message for Keep Alive timers negotiation. }
200  Interface Status  Interfaces status reported by the UP including physical interfaces, sub-interfaces, trunk interfaces, and tunnel interfaces.

201  Board Status  Board information reported by the UP including the board type and in-position status.

300  Subscriber Traffic Statistics  User traffic statistics.

301  Subscriber Detection Results  User detection information.

302  Update Response  The processing result of a subscriber session update.

400  Address Allocation Request  Request address allocation.

401  Address Allocation Response  Address allocation response.

402  Address Renewal Request  Request for address lease renewal.

403  Address Renewal Response  Response to a request for extending an IP address lease.

404  Address Release Request  Request to release the address.

405  Address Release Response  Ack of a message releasing an IP address.

9.3 TLV Operation Codes

IANA is requested to create an S-CUSP TLV Operation Codes registry on the S-CUSP Parameters Web Page as follows:

Registry Name: S-CUSP TLV Operation Codes
Registration Procedure: Expert Review
Reference: [this document]

<table>
<thead>
<tr>
<th>Code</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>- reserved</td>
</tr>
<tr>
<td>1</td>
<td>Update</td>
</tr>
<tr>
<td>2</td>
<td>Delete</td>
</tr>
<tr>
<td>3-15</td>
<td>- unassigned</td>
</tr>
</tbody>
</table>
9.4 Sub-TLV Types

IANA is requested to create an S-CUSP Sub-TLV Types registry on the S-CUSP Parameters Web Page as follows:

Registry Name: S-CUSP Sub-TLV Types
Registration Procedure: Expert Review
Reference: [this document]

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Section of [this document]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>VRF Name</td>
<td>7.3.1.</td>
</tr>
<tr>
<td>2</td>
<td>Ingress-QoS-Profile</td>
<td>7.3.1.</td>
</tr>
<tr>
<td>3</td>
<td>Egress-QoS-Profile</td>
<td>7.3.1.</td>
</tr>
<tr>
<td>4</td>
<td>User-ACL-Policy</td>
<td>7.3.1.</td>
</tr>
<tr>
<td>5</td>
<td>Multicast-ProfileV4</td>
<td>7.3.1.</td>
</tr>
<tr>
<td>6</td>
<td>Multicast-ProfileV6</td>
<td>7.3.1.</td>
</tr>
<tr>
<td>7</td>
<td>Ingress-CAR</td>
<td>7.3.2.</td>
</tr>
<tr>
<td>8</td>
<td>Egress-CAR</td>
<td>7.3.3.</td>
</tr>
<tr>
<td>9</td>
<td>NAT-Instance</td>
<td>7.3.1.</td>
</tr>
<tr>
<td>10</td>
<td>Pool-Name</td>
<td>7.3.1.</td>
</tr>
<tr>
<td>11</td>
<td>If-Desc</td>
<td>7.3.4.</td>
</tr>
<tr>
<td>12</td>
<td>I Pv6-Address List</td>
<td>7.3.5.</td>
</tr>
<tr>
<td>13</td>
<td>Vendor</td>
<td>7.3.6.</td>
</tr>
<tr>
<td>12-64534</td>
<td>- unassigned</td>
<td></td>
</tr>
<tr>
<td>65535</td>
<td>- reserved</td>
<td></td>
</tr>
</tbody>
</table>

9.5 Error Codes

IANA is requested to create an S-CUSP ERRID Codes registry on the S-CUSP Parameters Web Page as follows:

Registry Name: S-CUSP ERRID Codes
Registration Procedure: Expert Review
Reference: [this document]

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Fail</td>
<td>Malformed message received.</td>
</tr>
<tr>
<td>2</td>
<td>TLV-Unknown</td>
<td>One or more of the TLVs was not understood.</td>
</tr>
<tr>
<td>3</td>
<td>TLV-Length</td>
<td>The TLV length is abnormal.</td>
</tr>
<tr>
<td>4-999</td>
<td>- unassigned</td>
<td>Unassigned basic error codes.</td>
</tr>
<tr>
<td>1000</td>
<td>- reserved</td>
<td></td>
</tr>
</tbody>
</table>
The subsequent service processes corresponding to the UP are suspended.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1002</td>
<td>Keepalive Error</td>
</tr>
<tr>
<td>1003</td>
<td>Timer Expires</td>
</tr>
<tr>
<td>1004-1999</td>
<td>- unassigned</td>
</tr>
<tr>
<td>2000</td>
<td>- reserved</td>
</tr>
<tr>
<td>2001</td>
<td>Synch-NoReady</td>
</tr>
<tr>
<td>2002</td>
<td>Synch-Unsupport</td>
</tr>
<tr>
<td>2003-2999</td>
<td>- unassigned</td>
</tr>
<tr>
<td>3000</td>
<td>- reserved</td>
</tr>
<tr>
<td>3001</td>
<td>Pool-Mismatch</td>
</tr>
<tr>
<td>3002</td>
<td>Pool-Full</td>
</tr>
<tr>
<td>3003</td>
<td>Subnet-Mismatch</td>
</tr>
<tr>
<td>3004</td>
<td>Subnet-Conflict</td>
</tr>
<tr>
<td>3005-3999</td>
<td>- unassigned</td>
</tr>
<tr>
<td>4000</td>
<td>- reserved</td>
</tr>
<tr>
<td>4001</td>
<td>Update-Fail-No-Res</td>
</tr>
<tr>
<td>4002</td>
<td>QoS-Update-Success</td>
</tr>
<tr>
<td>4003</td>
<td>QoS-Update-Sq-Fail</td>
</tr>
<tr>
<td>4004</td>
<td>QoS-Update-CAR-Fail</td>
</tr>
<tr>
<td>4005</td>
<td>Statistic-Fail-No-Res</td>
</tr>
<tr>
<td>4006-4999</td>
<td>- unassigned forwarding table delivery error codes.</td>
</tr>
</tbody>
</table>

5000-4294967295 - reserved
10. Security Considerations

The Service, Control, and Management Interfaces between the CP and UP might be across the general Internet or other hostile environment. The ability of an adversary to block or corrupt messages or introduce spurious messages on any one or more of these interfaces would give the adversary the ability to stop subscribers from accessing network services, disrupt existing subscriber sessions, divert traffic, mess up accounting statistics, and generally cause havoc. Damage would not necessarily be limited to one or a few subscribers but could disrupt routing or deny service to one or more instances of the CP or otherwise cause extensive interference. If the adversary knows the details of the UP equipment and its forwarding rule capabilities, the adversary may be able to cause a copy of most or all user data to be sent to an address of the adversary’s choosing, thus enabling eavesdropping.

Thus, appropriate protections MUST be implemented to provide integrity, authenticity, and secrecy of traffic over those interfaces. Whether such protection is used is a network operator decision. See [RFC6241] for Management Interface / NETCONF security. Security on the Service Interface is dependent on the tunneling protocol used which is out of scope for this document. Security for the Control Interface, over which the S-CUSP protocol flows, is further discussed below.

S-CUSP messages do not provide security. Thus, if these messages are exchanged in an environment where security is a concern, that security MUST be provided by another protocol such as TLS 1.3 [RFC8446] or IPSEC. TLS 1.3 is the mandatory to implement protocol for interoperability. The use of a particular security protocol on the Control Interface is determined by configuration. Such security protocols need not always be used and lesser security precautions might be appropriate because, in some cases, the communication between the CP and UP is in a benign environment.
Contributors

Zhouyi Yu
Huawei Technologies
Email: yuzhouyi@huawei.com

Chengguang Niu
Huawei Technologies
Email: chengguang.niu@huawei.com

Zitao Wang
Huawei Technologies
Email: wangzitao@huawei.com

Jun Song
Huawei Technologies
Email: song.jun@huawei.com

Dan Meng
H3C Technologies
No.1 Lixing Center
No.8 guangxun south street, Chaoyang District,
Beijing, 100102
China
Email: mengdan@h3c.com

Hanlei Liu
H3C Technologies
No.1 Lixing Center
No.8 guangxun south street, Chaoyang District,
Beijing, 100102
China
Email: hanlei_liu@h3c.com
Normative References


Informative References


Authors’ Addresses

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

Email: hushujun@chinamobile.com

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

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

Mach (Guoyi) Chen
Huawei Technologies
Huawei Bldg., No. 156 Beiqing Road
Beijing 100095 China

Email: mach.chen@huawei.com

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

Email: qinfengwei@chinamobile.com

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

Email: lizhenqiang@chinamobile.com
Tee Mong Chua  
Singapore Telecommunications Limited  
31 Exeter Road, #05-04 Comcentre Podium Block  
Singapore City  239732  
Singapore  

Email: teemong@singtel.com

Daniel Huang  
ZTE  

Email: huang.guangping@zte.com.cn
Copyright, Disclaimer, and Additional IPR Provisions

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
Gap Analysis of Dynamic Networks to Hybrid Cloud DCs

draft-ietf-rtgwg-net2cloud-gap-analysis-02

Abstract

This document analyzes the technological gaps when using SD-WAN to
dynamically interconnect workloads and applications hosted in various third-party cloud data centers.

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 December 19, 2019.
1. Introduction

[Net2Cloud-Problem] describes the problems that enterprises face today in transitioning their IT infrastructure to support digital
economy, such as connecting enterprises’ branch offices to dynamic workloads in different Cloud DCs.

This document analyzes the technological gaps to interconnect dynamic workloads & apps hosted in cloud data centers that the enterprise’s VPN service provider may not own/operate or may be unable to provide the enterprise with the required connectivity to access such locations. When VPN service providers have insufficient bandwidth to reach a location, SD-WAN techniques can be used to aggregate bandwidth of multiple networks, such as MPLS VPNs or the Public Internet to achieve better performance. This document primarily focuses on the technological gaps raised by using SD-WAN techniques to connect enterprise premises to cloud data centers operated by third parties.

For the sake of readability, a SD-WAN edge, a SD-WAN endpoint, C-PE, or CPE are used interchangeably throughout this document. However, each term has some minor emphasis, especially when used in other related documents:

- SD-WAN Edge: could include multiple devices (virtual or physical);
- SD-WAN endpoint: to refer to a WAN port of SD-WAN devices or a single SD-WAN device;
- C-PE: more for provider owned SD-WAN edge, e.g. for SECURE-EVPN’s PE based VPN, when PE is the edge node of SD-WAN;
- CPE: more for enterprise owned SD-WAN edge.

2. Conventions used in this document

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

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

SD-WAN: Software Defined Wide Area Network, "SD-WAN" refers to the solutions of pooling WAN bandwidth from multiple underlay networks to get better WAN bandwidth management, visibility & control. When the underlay is a private network, traffic may be forwarded without any additional encryption; when the underlay networks are public, such as the Internet, some traffic needs to be encrypted when passing through (depending on user-provided policies).

3. Gap Analysis of C-PEs WAN Port Registration

SD-WAN technology has emerged as means to dynamically and securely interconnect the OnPrem branches with the workloads instantiated in Cloud DCs that do not have direct connectivity to BGP/MPLS VPN PEs or have very limited bandwidth.

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

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

- Interworking with the MPLS VPN control plane. A SD-WAN 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 before being sent.
- Scalability: NHRP/DSVPN/DMVPN works 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 C-PE’s owner to be informed of how/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 addresses, such as the NAT type, Private address, Public address, Private port, Public port, etc.

[BGP-SDWAN-PORT] describes how SD-WAN edge nodes use BGP to register their WAN ports properties to the SD-WAN controller, which then propagates the information to other SD-WAN edge nodes that are authenticated and authorized to communicate with them.

4. Aggregating VPN paths and Internet paths

Most likely, enterprises (especially the largest ones) already have their CPEs interconnected by providers’ VPNs, based upon VPN techniques such as EVPN, L2VPN, or L3VPN. The VPN can be PE-based or CPE-based. The commonly used PE-based VPNs have CPE directly attached to PEs, therefore the communication between CPEs and PEs is considered as secure. MP-BGP is used to learn & distribute routes among CPEs, even though sometimes routes among CPEs are statically configured on the CPEs.

To aggregate paths over the Internet and paths over the VPN, the C-PEs need to have some WAN ports connected to the PEs of the VPNs and other WAN ports connected to the Internet. It is necessary for the CPEs to use a protocol so that they can register the WAN port properties with their SD-WAN Controller(s): this information conditions the establishment and the maintenance of IPsec SA associations among relevant C-PEs.
When using NHRP for registration purposes, C-PEs need to run two separate control planes: EVPN&BGP for CPE-based VPNs, and NHRP & DSVPN/DMVPN for ports connected to the Internet. Two separate control planes not only add complexity to C-PEs, but also increase operational cost.

```
+---+
<table>
<thead>
<tr>
<th>RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untrusted</td>
</tr>
</tbody>
</table>
\   /  \
+---+  +--------------|RR |----------+
/  Untrusted    +-+-+           \
/                                 \
+----+  +---------+  packets encrypted over     +------+  +----+
| TN3|--| TN3|--| A1----- Untrusted ------ B1 | C-PE | C-PE | -----+
+----+  | A2--\ | C-PE | C-PE | -----+
|-----++---+  +---------+  packets          +-----++-----+
| TN2|--| TN2|--| Untrusted B2 | C-PE | PE----+-|---B3 | TN3|
+----+  | A3--\ | ---| PE+------------|--PE |-----|---B2 | C-PE |
|-----++---+  +---------+  trusted         +-----++-----+
|-----++---+  +---------+  packets          +-----++-----+
| TN1|--| TN1|--| TN1|--| C1--| PE go natively | PE | PE | D1 | TN1|
+----+  | C-PE | C2----+- without encry++- | C-PE | C-PE |-----+
|-----++---+  +---------+       +-----++-----+
| C----| C----| without encrypt over |-----+
|-----++---+  +---------+  Untrusted     ++---++-----+
| TN2|--| TN2|--| TN2|--| C4----- Untrusted ------D2 |-----+
+----+  +---------+  packets          +-----++-----+
|-----++---+  +---------+  packets          +-----++-----+
```

Figure 1: CPEs interconnected by VPN paths and Internet Paths

4.1. Key Control Plane Components of SD-WAN

As described in [BGP-SDWAN-Usage], the SD-WAN Overlay Control Plane has three distinct properties:

- SD-WAN node’s WAN Port Property registration to the SD-WAN Controller.
  - To inform the SD-WAN controller and authorized peers of the WAN port properties of the C-PE [SDWAN-Port]. When the WAN ports are assigned private addresses, this step can register the type of NAT that translates private addresses into public ones.

- Controller facilitated IPsec SA management and NAT information distribution
- Establishing and Managing the topology and reachability for services attached to the client ports of SD-WAN nodes.
  - This is for the overlay layer’s route distribution, so that a C-PE can populate its overlay routing table with entries that identify the next hop for reaching a specific route/service attached to remote nodes. [SECURE-EVPN] describes EVPN and other options.

4.2. Using BGP Tunnel-Encap

RFC5512 and [Tunnel-Encap] describe methods to construct BGP UPDATE messages that advertise endpoints’ tunnel encapsulation capability and the respective attached client routes, so that the peers that receive of the BGP UPDATE can establish appropriate tunnels with the endpoints for the aforementioned routes. RFC5512 uses the Endpoint Address subTLV, whereas [Tunnel-Encap] uses Remote Endpoint Address subTLV to indicates address of the tunnel endpoint which can be an IPv4 or an IPv6 address. There are Tunnel Encapsulation attribute subTLVs to indicate the supported encapsulation types, such as L2TPv3, GRE, VxLAN, IP-in-IP, etc.

[Tunnel-Encap] removed SAFI =7 (which was specified by RFC5512) for distributing encapsulation tunnel information. [Tunnel-Encap] requires that tunnels need to be associated with routes.

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

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

- [Tunnel-Encap] doesn’t have the functionality that would help the C-PE to register its WAN Port properties.
- A SD-WAN tunnel, e.g. IPsec-based, requires a negotiation between the tunnel’s end points for supported encryption algorithms and tunnel types before it can be properly established, whereas [Tunnel-Encap] only allow the announcement of one endpoint’s supported encapsulation capabilities for specific attached routes.
and no negotiation between tunnel end points is needed. The establishment of a SD-WAN tunnel can fail, e.g., in case the two endpoints support different encryption algorithms. That is why a SD-WAN tunnel needs to be established and maintained independently from advertising client routes attached to the edge node.

- [Tunnel-Encap] requires all tunnels updates are associated with routes. There can be many client routes associated with the SD-WAN IPsec tunnel between two C-PEs’ WAN ports; the corresponding destination prefixes (as announced by the aforementioned routes) may also be reached through the VPN underlay without any encryption. A more realistic approach to separate SD-WAN tunnel management from client routes association with the SD-WAN tunnels.

- When SD-WAN tunnel and clients routes are separate, the SD-WAN Tunnel establishment may not have routes associated. There is a suggestion on using a "Fake Route" for a SD-WAN node to use [Tunnel-Encap] to advertise its SD-WAN tunnel end-points properties. However, using "Fake Route" can raise some design complexity for large SD-WAN networks with many tunnels. For example, for a SD-WAN network with hundreds of nodes, with each node having many ports & many endpoints to establish SD-WAN tunnels with their corresponding peers, the node would need as many "fake addresses". For large SD-WAN networks (such as those comprised of more than 10000 nodes), each node might need 10’s thousands of "fake addresses", which is very difficult to manage and requires lots of configuration tasks to get the nodes properly set up.

- [Tunnel-Encap] does not have any field to carry detailed information about the remote C-PE, such as Site-ID, System-ID, Port-ID

- [Tunnel-Encap] Does not have any field to carry IPsec attributes for the SD-WAN edge nodes to establish IPsec Security Associations with others. It does not have any proper way for two peer CPEs to negotiate IPsec keys either, based on the configuration sent by the Controller.

- [Tunnel-Encap] does not have any field to indicate the UDP NAT private address <-> public address mapping

- C-PEs tend to communicate with a subset of the other C-PEs, not all the C-PEs need to be connected through a mesh topology. Without any BGP extension, many nodes can get dumped with too much
information coming from other nodes that they never need to communicate with.

4.3. SECURE-L3VPN/EVPN

[SECURE-L3VPN] describes how to extend the 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-protected packets are forwarded to the Internet or to other backbone network thereby eliminating the need for MPLS transport in the backbone.

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

[SECURE-EVPN] describes a solution where point-to-multipoint BGP signaling is used in the control plane for SDWAN Scenario #1. 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 without IKEv2 point-to-point signaling or any other direct peer-to-peer session establishment messages.

Both [SECURE-L3VPN] and [SECURE-EVPN] are useful, however, they both miss the aspects of aggregating VPN and Internet underlays. In summary:

- These documents do not address the scenario of C-PE having some ports facing VPN PEs and other ports facing the Internet.
- 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. In SD-WAN, Zero Touch Provisioning is expected. Manual configuration is not an option, as it contradicts the objectives of SD-WAN to automate configuration tasks.
- For RR communication with C-PEs, this draft only mentions IPsec. Missing TLS/DTLS.
- The draft assumes that C-PEs and RR are connected with an IPsec tunnel. With zero touch provisioning, we need an automatic way to synchronize the IPsec SAs between C-PEs and RR. The draft assumes:
A CPE must also be provisioned with whatever additional information is needed in order to set up an IPsec SA with each of the red RRs

- IPsec requires periodic refreshment of the keys. The draft 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. Let us assume that the RR is responsible for creating the key for all endpoints: When one endpoint is compromised, all other connections will be impacted.

4.4. 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 CPEs to support means to determine whether traffic sent by remote peers is legitimate to prevent spoofing attacks.

5. CPEs not directly connected to VPN PEs

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

However, SD-WAN paths established over the public Internet can have unpredictable performance, especially over long distances and across operators’ domains. Therefore, it is highly desirable to steer as much as possible the portion of SD-WAN paths over service provider VPN (e.g., enterprise’s existing VPN) that have guaranteed SLA to minimize the distance or the number of segments over the public Internet.
MEF Cloud Service Architecture [MEF-Cloud] also describes a use case of network operators that uses SD-WAN over LTE or the public Internet for last mile access where the VPN service providers cannot necessarily provide the required physical infrastructure.

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

When using SD-WAN to connect the enterprise’s existing sites with the workloads hosted in Cloud DCs, the corresponding CPEs have to be upgraded to support SD-WAN. 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 integrates SD-WAN with traditional MPLS-based VPNs, to extend the existing MPLS-based VPNs to the Cloud DC Workloads over the access paths that are not under the VPN provider’s control. To make it work properly, a small number of the PEs of the MPLS VPN can be designated to connect to the remote workloads via SD-WAN secure IPsec tunnels. Those designated PEs are shown as fPE (floating PE or smart PE) in Figure 3. Once the secure IPsec tunnels are established, the workloads hosted in Cloud DCs can be reached by the enterprise’s VPN without upgrading all of the enterprise’s existing CPEs. The only CPE that needs to support SD-WAN would be a virtualized CPE instantiated within the cloud DC.
In Figure 3, the optimal Cloud DC to host the workloads (as a function of the proximity, capacity, pricing, or other criteria chosen by the enterprises) does not have a direct connection to the PEs of the MPLS VPN that interconnects the enterprise’s existing sites.
5.1. Floating PEs to connect to Remote CPEs

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

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

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

5.2. NAT Traversal

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

A SD-WAN edge node can solicit a STUN (Session Traversal of UDP Through Network Address Translation RFC 3489) Server to get the NAT property, the public IP address and the Public Port number so that such information can be communicated to the relevant peers.

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

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

The path between the remote CPEs and VPN PEs that maintain VPN routes may very well traverse untrusted nodes.
EBGP Multi-hop design requires static configuration on both peers. To use EBGP between a PE and remote CPEs, the PE has to be manually configured with the "next-hop" set to the IP 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.

5.4. Designated Forwarder to the remote edges

Among the multiple floating PEs that are reachable from a remote CPE, 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.

Therefore, it is necessary to designate one floating PE as the CPE’s Designated Forwarder, similar to TRILL’s Appointed Forwarders [RFC6325].
MPLS VPNs do not have features like TRILL’s Appointed Forwarders.

5.5. Traffic Path Management

When there are multiple floating PEs that have established IPsec tunnels with the remote CPE, the remote CPE can forward outbound traffic to the Designated Forwarder PE, which in turn forwards traffic to egress PEs and then to the final destinations. However, it is not straightforward for the egress PE to send back the return traffic to the Designated Forwarder PE.

Example of Return Path management using Figure 3 above.

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

6. Manageability Considerations

Zero touch provisioning of SD-WAN edge nodes should be a major feature of SD-WAN deployments. It is necessary for a newly powered up SD-WAN edge node to establish a secure connection (by means of TLS, DTLS, etc.) with its controller.

7. Security Considerations

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

Several of these approaches have gaps in meeting enterprise security requirements when tunneling their traffic over the Internet, since this is the purpose of SD-WAN. See the individual sections above for further discussion of these security gaps.
8. IANA Considerations

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

9. References

9.1. Normative References


9.2. Informative References


[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:

[DSVPN] Dynamic Smart VPN:


10. Acknowledgments

Acknowledgements to xxx for his review and contributions.

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

Linda Dunbar
Futurewei
Email: ldunbar@futurewei.com

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

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

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

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

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

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 3, 2009.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction...................................................3
1.1. On the evolution of Cloud DC connectivity.................3
1.2. The role of SD-WAN techniques in Cloud DC connectivity....4
2. Definition of terms............................................4
3. Current Practices in Interconnecting Enterprise Sites with Cloud DCs...............................................................5
3.1. Multiple connection to workloads in a Cloud DC............5
3.2. Interconnect to Hybrid Cloud DCs..........................7
3.3. Connecting workloads among hybrid Cloud DCs................8
4. Desired Properties for Networks that interconnect Hybrid Clouds9
5. Problems with MPLS-based VPNs extending to Hybrid Cloud DCs...10
6. Problem with using IPsec tunnels to Cloud DCs.................11
6.1. Complexity of multi-point any-to-any interconnection.....12
6.2. Poor performance over long distance......................12
6.3. Scaling Issues with IPsec Tunnels.........................13
1. Introduction

1.1. On the evolution of Cloud DC connectivity

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

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

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

In addition, more enterprises are moving towards hybrid cloud DCs, i.e. owned or operated by different Cloud operators, to maximize the benefits of geographical proximity, elasticity and special features offered by different cloud DCs.

1.2. The role of SD-WAN techniques in Cloud DC connectivity

This document discusses the issues associated with connecting enterprise to their workloads/applications instantiated in multiple third-party data centers (a.k.a. Cloud DCs). Very often, the actual Cloud DCs that host the workloads/applications can be transient.

SD-WAN, initially launched to maximize bandwidths between locations by aggregating multiple paths managed by different service providers, has expanded to include flexible, on-demand, application-based connections established over any networks to access dynamic workloads in Cloud DCs.

As a consequence, this document discusses the use of SD-WAN techniques as a means to improve enterprise-to-cloud DC connectivity.

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).

SD-WAN: Software Defined Wide Area Network. In this document, "SD-WAN" refers to the solutions of pooling WAN bandwidth from multiple underlay networks to get better WAN bandwidth management, visibility & control. When the underlay networks are private networks, traffic can traverse without additional encryption; when the underlay networks are public, such as Internet, some traffic needs to be encrypted when traversing through (depending on user provided policies).

VPC: Virtual Private Cloud 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. Current Practices in Interconnecting Enterprise Sites with Cloud DCs

3.1. Multiple connection to workloads in a Cloud DC

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

- 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 (https://aws.amazon.com/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.

As an example, some branch offices of an enterprise can connect to over the Internet to reach AWS’s vGW via IPsec tunnels. Other branch offices of the same enterprise can connect to AWS DirectConnect via a private network (without any encryption). It is important for enterprises to be able to observe the specific behaviors when connected by different connections.

Figure below shows an example of some tenants’ 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. vR1 uses IPsec to establish secure tunnels over the Internet to avoid paying extra fees for the IPsec features provided by AWS vGW. Some tenants can deploy separate virtual routers to connect to internet traffic and to traffic from the secure channels from vGW and DirectConnect, e.g. vR1 & vR2. Others may have one virtual router connecting to both types of traffic. Customer Gateway can be customer owned router or ports physically connected to AWS Direct Connect GW.
3.2. Interconnect to Hybrid Cloud DCs

It is likely that hybrid designs will become the rule for cloud services, as more enterprises see the benefits of integrating public and private cloud infrastructures. However, enabling the growth of hybrid cloud deployments in the enterprise requires fast and safe interconnection between public and private cloud services. For an enterprise to connect to applications & workloads hosted in multiple Cloud DCs, the enterprise can use IPsec tunnels established over the Internet or a (virtualized) leased line service to connect its on-premises gateways to each of the Cloud DC’s gateways, virtual...
routers instantiated in the Cloud DCs, or any other suitable design (including a combination thereof).

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

3.3. Connecting workloads among hybrid Cloud DCs

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

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 have to traverse through the customer gateway.

c) Establish direct tunnels among different VPCs (Virtual Private Clouds) via client’s own virtual routers instantiated within Cloud DCs. DMVPN (Dynamic Multipoint Virtual Private Network) or DSVPN (Dynamic Smart VPN) techniques can be used to establish direct Multi-point-to-Point or multi-point-to multi-point tunnels among those client’s own virtual routers.

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. Therefore, DMVPN and/or DSVPN cannot be used directly for connecting workloads in hybrid Cloud DCs.

Other protocols such as BGP can be used, as described in [BGP-SDWAN].

4. Desired Properties for Networks that interconnect Hybrid Clouds
The networks that interconnect hybrid cloud DCs must address the following requirements:
- High availability to access all workloads in the desired cloud DCs.
  Many enterprises include cloud infrastructures in their disaster recovery strategy, e.g., by enforcing periodic backup policies within the cloud, or by running backup applications in the Cloud, etc. Therefore, the connection to the cloud DCs may not be permanent, but rather needs to be on-demand.

- Global reachability from different geographical zones, thereby facilitating the proximity of applications as a function of the end users’ location, to improve latency.

- Elasticity: prompt connection to newly instantiated applications at Cloud DCs when end-users’ usages increase and prompt release of connection after applications at locations being removed when demands change.
  Some enterprises have front-end web portals running in cloud DCs and database servers in their on-premises DCs. Those front-end web portals need to be reachable from the public Internet. The backend connection to the sensitive data in database servers hosted in the on-premises DCs might need secure connections.

- Scalable security management. IPsec is commonly used to interconnect cloud gateways with CPEs deployed in the enterprise premises. For enterprises with a large number or branch offices, managing the IPsec’s Security Associations among many nodes can be very difficult.
5. Problems with MPLS-based VPNs extending to Hybrid Cloud DCs

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

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

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

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

One of the main drivers for moving workloads into the cloud is the widely available cloud DCs at geographically diverse locations, where apps can be instantiated so that they can be as close to their end-users as possible. When the user base changes, the applications may be migrated to a new cloud DC location closest to the new user base.
- Most of the cloud DCs do not expose their internal networks, so the MPLS-based VPNs can only reach Cloud DC’s Gateways, not to the workloads hosted inside. Even with AWS DirectConnect, the connection only reaches the AWS DirectConnect Gateway. AWS DirectConnect Gateway uses BGP to exchange all routes with devices located behind the gateway, even including routes of applications that might be physically located in different geographical locations. There is no visibility on how the applications/workloads are interconnected within a Cloud DC or across multiple Cloud DCs.

- Extensive usage of Overlay by Cloud DCs:

Many cloud DCs use an overlay to connect their gateways to the workloads located inside the DC. There is currently no standard that specifies the interworking between the Cloud Overlay and the enterprise’ existing underlay networks. One of the characteristics of overlay networks is that some of the WAN ports of the edge nodes connect to third party networks. There is therefore a need to propagate WAN port information to remote authorized peers in third party network domains in addition to route propagation. Such an exchange cannot happen before communication between peers is properly secured.

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

6. Problem with using IPsec tunnels to Cloud DCs
As described in the previous section, many Cloud operators expose their gateways for external entities (which can be enterprises themselves) to directly establish IPsec tunnels. Enterprises can also instantiate virtual routers within Cloud DCs to connect to their on-premises devices via IPsec tunnels. If there is only one enterprise location that needs to reach the Cloud DC, an IPsec tunnel is a very convenient solution.
However, many medium-to-large enterprises usually have multiple sites and multiple data centers. For workloads and apps hosted in cloud DCs, multiple sites need to communicate securely with those cloud workloads and apps. This section documents some of the issues associated with using IPsec tunnels to connect enterprise premises with cloud gateways.

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

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

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

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

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

6.2. Poor performance over long distance

When enterprise CPEs or gateways are far away from cloud DC gateways or across country/continent boundaries, performance of IPsec tunnels over the public Internet can be problematic and unpredictable. Even though there are many monitoring tools available to measure delay and various performance characteristics of the network, the measurement for paths over the Internet is passive and past measurements may not represent future performance.
Many cloud providers can replicate workloads in different available zones. An App instantiated in a cloud DC closest to clients may have to cooperate with another App (or its mirror image) in another region or database server(s) in the on-premises DC. This kind of coordination requires predictable networking behavior/performance among those locations.

6.3. Scaling Issues with IPsec Tunnels

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

If there is only one enterprise location connected to the cloud gateway, a small number of IPsec tunnels can be configured on-demand between the on-premises DC and the Cloud DC, which is an easy and flexible solution.

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

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

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

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

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

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

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

- SD-WAN for interconnecting branch offices usually have two end-points (e.g., CPEs) controlled by one entity (e.g., a controller or management system operated by the enterprise).
- SD-WAN for Cloud DC interconnects may consider CPEs owned or managed by the enterprise, while remote end-points are being managed or controlled by Cloud DCs (For the ease of description, let’s call such CPEs asymmetrically-managed CPEs).
Cloud DCs may have different entry points (or devices) with one entry point that terminates a private direct connection (based upon a leased line for example) and other entry points being devices terminating the IPsec tunnels, as shown in Figure 2.

Therefore, the SD-WAN design becomes asymmetric.

Figure 2: Different Underlays to Reach Cloud DC
8. End-to-End Security Concerns for Data Flows

When IPsec tunnels established from enterprise on-premises CPEs are terminated at the Cloud DC gateway where the workloads or applications are hosted, some enterprises have concerns regarding traffic to/from their workload being exposed to others behind the data center gateway (e.g., exposed to other organizations that have workloads in the same data center).

To ensure that traffic to/from workloads is not exposed to unwanted entities, IPsec tunnels may go all the way to the workload (servers, or VMs) within the DC.

9. Requirements for Dynamic Cloud Data Center VPNs

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

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

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

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

11. IANA Considerations

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

12. References

12.1. Normative References

12.2. Informative References


13. Acknowledgments

Many thanks to Chris Bowers, 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
Futurewei
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
Abstract

For routing protocol troubleshooting, different approaches exhibit merits w.r.t. different situations. They can be generally divided into two categories, the distributive way and the centralized way. A very commonly used distributive approach is to log in possibly all related devices one by one to check massive data via CLI. Such approach provides very detailed device information, however it requires operators with high NOC (Network Operation Center) experience and suffers from low troubleshooting efficiency and high cost. The centralized approach is realized by collecting data from devices via approaches, like the streaming Telemetry or BMP (BGP Monitoring Protocol) RFC7854 [RFC7854], for the centralized server to analyze all gathered data. Such approach allows a comprehensive view of the whole network and facilitates automated troubleshooting, but is limited by the data collection boundary set by different management domains, as well as high network bandwidth and CPU computation costs.

This document proposes a semi-distributive and semi-centralized approach for fast routing protocol troubleshooting, localizing the target device and possibly the root cause, more precisely. It defines a new protocol, called the PAP (Protocol assisted Protocol), for devices to exchange protocol related information between each other in both active and on-demand manners. It allow devices to request specific information from other devices and receive replies to the requested data. It also allows actively transmission of information without request to inform other devices to better react w.r.t. network issues.

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 January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ........................................... 3
   1.1. Motivation ....................................... 3
   1.2. PAP Usage Use cases .............................. 5
       1.2.1. Use Case 1: BGP Route Oscillation ......... 5
       1.2.2. Use Case 2: RSVP-TE Set Up Failure ....... 5
2. Terminology ........................................... 6
3. PAP Overview .......................................... 6
4. PAP Message Format .................................... 8
   4.1. Common Header .................................. 8
   4.2. Negotiation Message ............................ 9
   4.3. Request Message ................................. 10
   4.4. Reply Message .................................. 10
   4.5. Notification Message ............................ 11
   4.6. ACK Message .................................... 12
1. Introduction

A healthy control plane, providing network connectivity, is the foundation of a well-functioning network. There have been rich routing and signaling protocols designed and used for IP networks, such as IGP (ISIS, OSPF), BGP, LDP, RSVP-TE and so on. The health issues of these protocols, such as neighbor/peer disconnect/set up failure, LSP set up failure, route flapping and so on, have been devoted with ongoing efforts for diagnosing and remediation.

1.1. Motivation

The distributive protocol troubleshooting approach is typically realized through manual per-device check. It’s both time- and labor-consuming, and requires NOC experience of the operators. Amongst all, localizing the target device is usually the most difficult and time-consuming part. For example, in the case of route loop, operators first log in a random device that reports TTL alarms, and then check the looped route in the Forwarding Information Base (FIB) and/or the Routing Information Base (RIB). It requires device by device check, as well as manual data correlation, to pin point to the exact responsible device, since the information retrieval and analysis of such distributive way is fragmented. In addition, the low efficiency and manual troubleshooting activities may further impact new network services and/or enlarge affected areas.

The centralized network OAM, by collecting network-wide data from devices, enables automatic routing protocol troubleshooting. Date collection protocols, such as SNMP (Simple Network Management Protocol) [RFC1157], NETCONF (Network Configuration Protocol) [RFC6241], and (BMP) [RFC7854], can provide various information retrieval, such as network states, routing data, configurations and so on. Such centralized way relies on the existence of a centralized server/controller, which is not supported by some legacy networks. What’s more, even with the existence of a centralized server/controller, it can only collect the data within its own management domain, while the cross-domain data are not available due to independent management of different ISPs. Thus, the lack of such
information may lead to troubleshooting failure. In addition, centralized approaches may suffer from high network bandwidth and CPU computation consumptions.

Another way of protocol troubleshooting is utilizing the protocol itself to convey diagnosing information. For example, some reason codes are carried in the Path-Err/ResvErr messages of RSVP-TE, so that to other nodes may know the why the tunnel fails to be set up. Such approaches is semi-distributive and semi-centralized. It does not rely on the deployment of a centralized server, but still gets partial global view of the network. However, there still requires non-trivial augmentation works to existing routing protocols in order to support troubleshooting. This then raises the question that whether such non-routing data is suitable to be carried in these routing protocols. The extra encapsulation, parsing and analyzing work for the non-routing data would further slow down the network convergence. Thus, it’s better to separate the routing and non-routing data transmission as well as data parsing. In addition, coexisting with legacy devices may cause interop issues. Thus, relying on augmenting existing routing protocols without network-wide upgrading may not only fail to provide the troubleshooting benefit, but further affect the operation of the existing routing system. What’s more, the failure of routing protocol instance would lead to the failure of diagnosing itself. All in all, it’s reasonable to separate the protocol diagnosing data generation/encapsulation/transmission/parsing from the protocol itself.

This document proposes a new protocol, called the PAP (Protocol assisted Protocol), for devices to exchange protocol related information between each other. It allows both active and on-demand data exchange. Considering that massiveness of protocol/routing related data, the intuitive of designing PAP is not to exchange the comprehensive protocol/routing status between devices, but to provide very specific information required for the contemporary troubleshooting. The benefits of such a semi-distributive and semi-centralized approach are summarized as follows:

1. It facilitates automatic troubleshooting without requiring manual device by device check.
2. It allows individual device to have a more global view by requesting data from other devices.
3. It does not rely on the deployment of a centralized server/controller.
4. It passes the data collection boundary set by different management domains by cross-domain data exchange between devices.

5. It relieves the bandwidth pressure of network-wide data collection, and the processing pressure of the centralized server.

6. It does not affect the running of existing routing protocols.

1.2. PAP Usage Use cases

PAP allows both data request/reply and data notification between devices. PAP speakers use the exchanged PAP data to help fast localize the network issues.

1.2.1. Use Case 1: BGP Route Oscillation

A BGP route oscillation can be caused by various reasons, and usually leaves network-wide impact. In order to find the root cause and take remediation actions, the first step is to localize the oscillation source. In this case, a BGP speaker can send a PAP Request Message to the next hop device of the oscillating route asking "Are you the oscillation source?". If the BGP speaker is the oscillation source, possibly knows by running a device diagnosing system, replies with a PAP Reply Message saying that "I'm the oscillation source!" to the device who sends the PAP Request Message. If the BGP speaker is not the oscillation source, it further asks the same question with a PAP Request Message to its next hop device of the oscillating route. This request and reply process continues until the request has reached the oscillation source. The source device then sends a PAP Reply Message to tell its upstream device along the PAP request path that "I am the oscillation source!", and then "xx is the oscillation source!" information is further sent back hop by hop to the device who originates the request.

1.2.2. Use Case 2: RSVP-TE Set Up Failure

The MPLS label switch path set up, either using RSVP-TE or LDP, may fail due to various reasons. Typical troubleshooting procedures are to log in the device, and then check if the failure lies on the configuration, or path computation error, or link failure. Sometimes, it requires the check of multiple devices along the tunnel. Certain reason codes can be carried in the Path-Err/ResvErr messages of RSVP-TE, while other data are currently not supported to be transmitted to the path ingress/egress node, such as the authentication failure. Using PAP, the device, which is responsible for the tunnel set up failure, can send the PAP Notification Message to the Ingress device, and possibly with some reason codes so that
the ingress device can not only localize the target device but also the root cause.

2. Terminology

IGP: Interior Gateway Protocol
IS-IS: Intermediate System to Intermediate System
OSPF: Open Shortest Path First
BGP: Boarder Gateway Protocol
BGP-LS: Boarder Gateway Protocol-Link State
MPLS: Multi-Protocol Label Switching
RSVP-TE: Resource Reservation Protocol-Traffic Engineering
LDP: Label Distribution Protocol
BMP: BGP Monitoring Protocol
LSP: Link State Packet
IPFIX: Internet Protocol Flow Information Export
PAP: Protocol assisted Protocol
UDP: User Datagram Protocol

3. PAP Overview

PAP is a non-routing protocol used for PAP speakers to exchange routing protocol related information. The primary function of PAP is to provide a unified tunnel for protocol diagnosing information exchange without augmenting each specific protocol.

PAP uses UDP as its transport protocol, which is connectionless. The reason that UDP is selected over TCP is because PAP is intended for on-demand communications. The PAP packet is defined as follows.

This document requires the assignment of a User Port registry for the UDP Destination Port.
This document uses PAP speakers to refer to two routing devices that support PAP and/or communicate with each other using PAP throughout. The communications between two PAP speakers should follow three major processes, i.e., the Capability Negotiation Process, the Request and Reply Process, and the Notification Process. This document defines 5 PAP Message types, i.e., Negotiation Message, Request Message, Reply Message, Notification Message, and ACK Message, which are used in the above PAP processes.

Although PAP is connectionless, there still requires a successful Capability Negotiation between two PAP speakers before the exchange of any PAP messages (except the Negotiation Message). The purpose of the Capability Negotiation process is to inform both PAP speakers of each other’s PAP capabilities. The capabilities of a PAP speaker refer to the support of diagnosing information exchange of specific protocols, while the generation of such data are possibly enabled by a local protocol diagnosing system. The negotiation can be initiated by either the local or remote PAP speaker through sending out a PAP Negotiation Message. The Negotiation Message may or may not require an ACK Message, as indicated in the Negotiation Message. A successful negotiation is achieved if both PAP speakers have correctly received the other speaker’s Negotiation Message regardless of the message content. A more detailed definition of a successful negotiation is defined in Section 5.1. After a successful negotiation, two PAP speakers can exchange any PAP Message on-demand.

The purpose of the Request and Reply process is to acquire information needed by a PAP speaker from other PAP speakers for diagnosing a specific protocol. The Request and Reply Messages can be customized for different protocols. The process starts by any PAP speaker sending a Request Message to another PAP speaker. The PAP receiver, after receiving the Request message, sends out a Reply Message to the request sender. The generation of both Request and Reply Messages is protocol-specific, and depends on possibly a local protocol diagnosing system. One Request Message received may further results in a new Request Message generated and sent out to a third PAP speaker, if the receiver does not have the answer to the sender. Both Request and Reply Messages may or may not require an ACK Message, as indicated in the Request/Reply Messages.

The Notification Process is used to serve active data notification. Any PAP speaker, having information to share with other PAP speakers,
may start to send Notification Messages to any target PAP speaker. This information sharing is also protocol-specific, and can be possibly generated by a local protocol diagnosing system. The Notification Message may or may not require an ACK Message, as indicated in the Notification Message.

4. PAP Message Format

4.1. Common Header

The common header is encapsulated in all PAP messages. It includes the following fields.

```
+----------------------+---------------------+
|    Sequence Number   |        Length       |
|-----------+----------+---------------------|
| Msg. Type |
|-----------+
```

Figure 2. PAP Common Header

- Sequence Number (2 byte): It is used to indicate the sequence of the PAP Message. It uses unsigned integers to distinguish the PAP Messages sent to the same remote device. The integer is set incrementally in order time.

- Length (2 bytes): Length of the message in bytes, including the Common Header and the following Message.

- Message Type (1 byte): This indicates the PAP message type. The following types are defined, and listed as follows.

  * Type = TBD1: Negotiation Message. It is used for two devices to inform each other of the capabilities they support and no longer support.

  * Type = TBD2: Request Message.

  * Type = TBD3: Reply Message.

  * Type = TBD4: Notification Message.

  * Type = TBD5: ACK Message. It is used to confirm to the local device that the remote device has received a previous sent PAP message, which can be either a Negotiation Message, a Request Message, a Reply Message or an Notification Message.
4.2. Negotiation Message

The Negotiation Message is used for both capability negotiation between the local PAP speaker and the remote PAP speaker. It is also used to indicate the enabling and disabling of any supported capability. The Negotiation Message is required to be exchanged before two PAP speakers can exchange any other PAP Message types.

```
+--------------------------------------------+
|       Version        |A|C|   Reserved      |
+--------------------------------------------+
|             Protocol Capability            |
+--------------------------------------------+
|             Optional Capability            |
+--------------------------------------------+
```

Figure 3. PAP Negotiation Message

- **Version (2 bytes)**: It indicates the PAP version. The current version is 0.

- **Flags (2 bytes)**: Two flag bits are currently defined.
  
  * The A bit is used to indicate if an ACK Message from the remote PAP speaker is required for each Negotiation Message sent. If an ACK is required, then the A bit SHOULD be set to "1", and "0" otherwise.
  
  * The C bit is used to indicate the enabling/disabling of the capabilities that carried in the Protocol Capability field. If the local device wants to inform the remote device of enabling one or more capabilities, the C bit SHOULD be set to "1". If the local device wants to inform the remote device of disabling one or more capabilities, the C bit SHOULD be set to "0".

- **Protocol Capability (4 bytes)**: It is 4-byte bit map that indicates the capability of information exchange regarding various protocols. Each bit represents one protocol. For example, the right most bit of the Protocol Capability represents the capability of exchanging IS-IS protocol related information. The format and definition of information exchanged, regarding each protocol, MUST follow the PAP Message format and definition.

- **Optional Capability (4 bytes)**: It is 4-byte bit map that indicates the capability of other information. It can be used to carry other application-specific information, which allows future extension. Its format and definition remains to be determined.
4.3. Request Message

The Request Message is used for the local device to request specific data regarding one specific protocol or application from the remote device. It MUST be sent after a successful Capability Negotiation Process (described in Section 5.1), and the requested protocol/application MUST be supported by both the local and remote devices, as indicated in the Negotiation Messages exchanged between the local and remote devices.

The Statistic TLV is defined as follows.

```
0                      15                   31
+----------------------+
<p>|A|     Reserved       |
|-------------------------------|</p>
<table>
<thead>
<tr>
<th>Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Request</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
</tbody>
</table>
```

Figure 4. PAP Request Message

- Flags (2 bytes): The A bit is used to indicate if an ACK Message from the remote PAP speaker is required for each Request Message sent. If an ACK is required, then the A bit SHOULD be set to "1", and "0" otherwise.

- Capability (4 bytes): It is 4-byte bit map that indicates the specific protocol/application the Request Message is requesting data for. The respective bit of this specific protocol/application MUST be set to "1", and the rest of the bits MUST be set to "0".

- Data Request (Variable): Specifies what data the local device is requesting. The specific format remains to be determined.

4.4. Reply Message

The Reply Message is used to carry the information that the local device requests from the remote device through the Request Message.
Flags (2 bytes): The A bit is used to indicate if an ACK Message from the remote PAP speaker is required for each Request Message sent. If an ACK is required, then the A bit SHOULD be set to "1", and "0" otherwise.

Capability (4 bytes): It is 4-byte bit map that indicates the specific protocol/application the Reply Message is replying for. The respective bit of this specific protocol/application MUST be set to "1", and the rest of the bits MUST be set to "0".

Data Reply (Variable): It contains the data that the local device requests from the remote device. The specific format remains to be determined.

4.5. Notification Message

The Notification Message is used to carry the information that the local device sends to the remote device.

Flags (2 bytes): The A bit is used to indicate if an ACK Message from the remote PAP speaker is required for each Request Message sent. If an ACK is required, then the A bit SHOULD be set to "1", and "0" otherwise.

Capability (4 bytes): It is 4-byte bit map that indicates the specific protocol/application the Notification Message is
notifying for. The respective bit of this specific protocol/application MUST be set to "1", and the rest of the bits MUST be set to "0".

- Data Notification (Variable): It contains the data that the local device actively sends to the remote device. The specific format remains to be determined.

4.6. ACK Message

The ACK Message is used to confirm that the remote device has received a PAP Message that set the A bit to "1". The ACK Message includes only the ACK sequence Number field, which MUST be set to the sequence number carried in the PAP Common Header of the received PAP message, which requires this ACK Message.

```
    +--------------------------------------------+
    |            ACK Sequence Number             |
    +--------------------------------------------+
```

Figure 7. PAP ACK Message

5. PAP Operations

The PAP operations include the following 3 processes, the Capability Negotiation Process, the Data Request and Reply Process, and the Data Notification Process.

5.1. Capability Negotiation Process

The Capability Negotiation process refers to the process that two devices inform each other of the capabilities they support and no longer support. A successful negotiation that inform each other of the supported capabilities between two devices MUST be achieved before the exchange of any other PAP Message.

The first Negotiation Message can be sent at any time per the local device's configuration. One or more Negotiation Messages can be sent at any time after the first Negotiation Message. Once a Negotiation Message is sent from a local device, an ACK Message from the remote device is required/or not per the indication of the "A" bit in the Negotiation Message. If the A bit is set to "1" (i.e., ACK is required), the local device SHUOULD wait for the ACK Message from the remote device for a certain time period before taking further actions, and if no ACK Message is received within this time frame, the local device SHOULD resend the Negotiation Message to the remote device. The waiting period can be configured locally. This send and
wait process CAN be repeated for at most 3 times before receiving a
ACK Message from the remote device. If after 3 times of resending
the Negotiation Message, still no ACK received, then this negotiation
is treated as unsuccessful. If the A bit is set to "0", then the
local device does not wait for any ACK Message. If no Negotiation
Message is received from the remote device within a time frame, the
local device can resend the Negotiation Message. This send and wait
process CAN be repeated for at most 3 times before receiving a
Negotiation Message from the remote device. If after 3 times of
resending the Negotiation Message, still no Negotiation Message
received, then this negotiation is treated as Unsuccessful. The
waiting period can be configured locally. Once a Negotiation Message
is received from the remote device, the negotiation between the two
devices is treated as successful regardless from the local device’s
perspective.

On the other hand, the remote device, after receiving the Negotiation
Message from the local device, SHOULD send out its own Negotiation
Message that indicates the capabilities that it supports. If the A
bit of the received Negotiation Message from the local device is set
to "1", the remote device SHOULD sent an ACK Message before sending
the new Negotiation Message. After the remote device sends out the
Negotiation Message back to the local device, it waits/or not for the
ACK message, depending on if the A bit of the Negotiation Message
sent to the local device is set to "1". If it’s set to "1", the
remote device waits for the ACK message for a certain time period
before taking further actions. If no ACK Message is received within
this time frame, the remote device SHOULD resend the Negotiation
Message to the local device. The waiting period can be configured
locally at the remote device. This send and wait process CAN be
repeated for at most 3 times before receiving a ACK Message from the
local device. If after 3 times of resending the Negotiation Message,
still no ACK received, then the remote device treats this negotiation
as unsuccessful, and successful otherwise. If the A bit of the
Negotiation Message sent from the remote device is set to "0", then
the remote device treats the negotiation as successful after sending
out the Negotiation Message.

The C bit in the Negotiation Message MUST always be set to "1" before
the negotiation is successful. A successful Capability Negotiation
means that both the local and remote devices have the information of
what capabilities the other side support so that when exchanging any
other messages, only the capabilities that are supported by both ends
SHOULD be carried in the respective capability fields.

Another process of the Capability Negotiation Process is to inform
the remote device of the capability/capabilities that the local
device no longer supports with the indication of C bit set as "0".
To inform the other device of the disabled capability/capabilities, the local device MUST have sent one or more Negotiation Messages that indicate the support of such capability/capabilities previously.

5.2. Data Request and Reply Process

After a successful Capability Negotiation, the local device CAN send the Data Request Message to the remote device per local configuration. An Reply Message SHOULD only be sent after receiving a Request Message. If the A bit of the Request Message is set to "1" (i.e., ACK is required), the local device SHOULD wait for the ACK Message from the remote device for a certain time period before taking further actions, and if no ACK Message is received within this time frame, the local device SHOULD resend the Request Message to the remote device. The waiting period can be configured locally. This send and wait process CAN be repeated for at most 3 times before receiving a ACK Message from the remote device. If the A bit is set to "0", then the local device does not wait for any ACK Message. If no Reply Message is received from the remote device within a time frame, the local device can resend the Request Message. This send and wait process CAN be repeated for at most 3 times before receiving a Reply Message from the remote device. If after 3 times of resending the Request Message, still no Reply Message received, then no further action is taken. The waiting period can be configured locally.

On the other hand, the remote device, after receiving the Request Message from the local device, SHOULD send out the Reply Message to reply the request. If the A bit of the received Request Message from the local device is set to "1", the remote device SHOULD sent an ACK Message before sending the Reply Message.

To request data for multiple protocols/applications, separate Request Messages SHOULD be sent, with each Request Message requesting one specific protocol/application data. Accordingly, the Reply Message is also sent separately per each Request Message.

5.3. Data Notification Process

The Notification Message CAN be sent by the local or the remote device at any time once the Capability Negotiation is successful. Each Notification Message SHOULD only indicate one specific protocol/application. The A bit can be set to "1" or "0" to allow the local/remote device to know if the other device has received the Notification Message through the ACK Message.
6. IANA

TBD

7. Contributors

Jiaqing Zhang, Huawei, zhangjiaqing@huawei.com

8. Acknowledgments

TBD

9. References

[I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., Lapukhov, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.ietf-netconf-yang-push]

[I-D.song-ntf]


Authors' Addresses

Zhenbin Li
Huawei
156 Beiqing Rd
Beijing
China

Email: lizhenbin@huawei.com
Instant Congestion Assessment Network (iCAN) for Data Plane Traffic Engineering
draft-liu-ican-00

Abstract

iCAN (instant Congestion Assessment Network) is a set of mechanisms running directly on network nodes:

- To adjust the flows paths based on real-time measurement of the candidate paths.
- The measurement is to reflect the congestion situation of each path, so that the ingress nodes could decide which flows need to be switched from a path to another.

This is something that current SDN and TE technologies can hardly achieve:

- SDN Controller is slow and far from the data plane, it is neither able to assess the real-time congestion situation of each path, nor able to assure the data plane always go as expected (especially in SRv6 scenarios). However, iCAN can work with SDN perfectly: controller planning multi-path transmission, and iCAN does the flow optimization automatically.
- Traditional TE is not able to adjust the flow paths in real-time.

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."
1. Introduction

Traditional IP routing is shortest path based on static metrics, which can fulfill basic requirement of connectivity. MPLS-TE brings the capability of utilizing non-shortest paths, thus traffic dispatch...
is doable; however, MPLS-TE in only a complementary mechanism because of the scalability issue. Segment routing provides even more flexibility that paths could be easily programmed; and along with the controller, it could be scaled.

However, the above mentioned mechanism all run in the control plane, which implies that they are not able to sense the data plane situation in real-time, thus they are mostly for relative static planning/controlling (minuets, hours or even day-level) of network traffic and not able to adapt to the microscopic traffic change in real-time (e.g. milli-second level). So, in real bearer networks (metro, backbones etc.), it is always underload so that the redundant resources could tolerant the traffic burst, results in a significant waste of network resources.

This draft proposes the iCAN (Instant Congestion Assessment Network) architecture to achieve autonomous adapt to traffic changes in real-time in terms of switching flows between multiple forwarding paths. iCAN includes following things:

- A mechanism between ingress and egress nodes to assess the path congestion situation in RTT level speed, to recognize which paths are underload and which are heavy loaded.
- Recognizing big flows and small flows in the device, in real time
- Ingress node dispatches flows to multiple paths, to make load balance, or to guarantee SLA for specific flows

This draft also discusses use cases and implementation scenarios of iCAN.

2. iCAN Architecture and Key Technical Requirements

2.1. Architecture
As above figure shows, there are 3 entities:

1. Controller
   - Responsible for planning multiple paths for a set of flows that could be aggregated to a pair of Ingress/Egress routers.
   - After delivering the planned paths to the ingress router, the controller would need nothing to do.

2. Ingress router:
   - Serves as a local "controller" for the iCAN system.
   - Responsible for triggering the path congestion assessment, which is coordinated with the egress router through a measurement protocol.
   - After getting the assessment results, the ingress router would calculate which flows need to be switched to a different path, in order to make the paths load balanced or to assure the transport quality of a certain of important flows.
   - In order to do the path switching calculation, the ingress router needs to recognize the TopN flow passing by it, since switching the big flows would make the most effort.
3. Egress router:
   - Only needs to coordinate with the ingress router to do the path assessment.

2.2. Key technical requirements

2.2.1. Path quality assessment

   o Req-1: the assessment MUST reflex the congestion status of the paths. (Note: a candidate congestion metric is proposed as: [I-D.dang-ippm-congestion].)

   o Req-2: the assessment SHOULD be done within a RTT timeslot. Since iCAN is to adapt the traffic change in real-time, the assessment needs to be done very fast.

   o Req-3: the assessment MUST be done for multiple paths between the same ingress/egress routes simultaneously. (Note: a candidate congestion metric is proposed as: [I-D.dang-ippm-multiple-path-measurement].)

2.2.2. Recognition and statistic of flows in devices

   o Req-1: the device SHOULD be able to recognize TopN big flows within a timeslot.

   o Req-2: the device MAY need to statistic all flows’ amount within a timeslot.

2.2.3. Flow switching between paths

   o Req-1: the device SHOULD be able to recognize flow let. The flow switching is done from the next flow let.

   o Req-2: the device MAY need to actively generate gap to artificially create flow let. If the flow needs to be switched immediately, then the device would need to make the gap, to avoid out-of-order packets arriving to the destination through multiple paths.

   o Req-3: the device SHOULD avoid oscillation of frequently switching flows from one to another.

   o Req-4: multiple ingress devices SHOULD be able to coordinate so that they won’t switch flows to the shared path at the same time, to avoid potential congestion in the shared path.
3. Use Cases of iCAN

3.1. Network load balancing

Background problem: traffic is not balanced in current metro network.

While some links are heavily loaded, others might be still lightly loaded: unbalance could lows down the service quality (e.g. SLA could not be guaranteed in the heavily loaded links/paths); unbalance could lows down the network utilization ratio (normally with 30%, e.g. a 100G physical capacity network can only bear at most 30G traffic, a huge waste of network infrastructure).

iCAN could be used for load balance among the multiple paths between a pair of ingress/egress nodes. Once the network is balanced, the real throughput of the network could be elevated significantly.

3.2. SLA assurance

Since iCAN could switch flow in real-time, it can guarantee a set of important flows. Once the path which carries the important flows is to be congested, the other flows could be switched to alternative paths, and the important flows would stably running in the original path.

(More content TBD)

3.3. Fine-Granularity reliability

Traditional reliability protocols such as BFD, can only assess the link on or off. With the path congestion assessment ability, iCAN could also asses the quality degradation.

(More content TBD)

4. Implementation Scenarios

4.1. iCAN with SRv6

- SR Multiple Explicit Paths

For example, there are 3 paths between the ingress and egress nodes, and the multi-path is defined as a SR-List containing LSP1/2/3.

The probe message detects the congestion status of the three SR-list paths. The edge device adjusts the load balancing between the three paths according to the congestion status of the three
SR-lists, and switch the flows from the path with a high congestion to the path with a low congestion.

- SR Multiple Explicit+Loose Paths

In loose path scenario, there needs to be an additional approach to probe the specific paths of a SR tunnel. After that, operations on the probed paths are the same as explicit path scenario.

4.2. iCAN with VxLAN

TBD.

4.3. iCAN with MPLS/MPLS-TE

TBD.

5. Standardization Requirements

1. Multi-path Planning (North Interface between Controller and devices)

2. Path Congestion Assessment (Horizontal Interface between devices), mostly regarding to Req-1&2&3 described in Section 2.2.1.

3. Flow Switching Negotiation (Horizontal Interface between devices), mostly regarding to Req-3&4 described in Section 2.2.3.

(More content TBD.)

6. Security Considerations

TBD.

7. IANA Considerations

TBD.

8. Acknowledgements

Very valuable comments were from Shunsuke Homma, Mikael Abrahamsson and Bruno Decraene.

A commercial router hardware based prototype had been implemented to prove the mechanisms discussed in the document are workable.
9. References

9.1. Normative References


9.2. Informative References

[I-D.dang-ippm-congestion]

[I-D.dang-ippm-multiple-path-measurement]

Author’s Address

Bing Liu
Huawei Technologies
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: leo.liubing@huawei.com
Abstract

The 5G Network slicing is an approach to provide separate independent E2E logical network from user equipment (UE) to applications where each network slice has different SLA requirements. Each E2E network slice consists of multitude of RAN-slice, Core-slice and Transport-slices, each with its own controller. To provide automation, assurance and optimization of the network slices, an E2E network slice controller is needed which interacts with controller in RAN, Core and Transport slices. The interfaces between the E2E network slice controller and RAN and Core controllers are defined in various 3GPP technical specifications. However, 3GPP has not defined the same interface for transport slices.

The aim of this document is to provide the clarification of this interface and to provide the information model of this interface for automation, monitoring and optimization of the transport slices.
1. Introduction ................................. 3
   1.1. Definition of Terms ........................ 3
2. High level architecture of 5G end-to-end network slicing  . 5
3. Logical flow cross layers for automation of an end-to-end network slices ........................ 8
4. Definition of Transport Slice ........................ 11
   4.1. Transport Slices in Distributed RAN .................. 11
   4.2. Transport Slices in Centralized RAN (C-RAN) ........ 12
   4.3. Transport Slices in cloud RAN .................... 13
   4.4. Transport Slice as a set of networks .............. 14
5. Transport Slice Connectivity Interface (TSCI) ............. 16
   5.1. Relationship between TSCI and various IETF data models . 17
   5.2. Realization (aka Implementation) of transport slices .. 18
6. Transport slice connectivity interface (TSCI) information
1. Introduction

Network Slicing is a mechanism which a network operator can use to allocate dedicated infrastructures and services to a customer (aka tenant) from shared operator’s network. In particular a 5G network slice is inherently an E2E concept and is composed of multiple logical independent networks are created in a common operator’s network from an user equipment to applications. In specific, the network slicing receives attention due to factors such as diversity of services and devices in 5G each with its own SLA requirements. Each E2E network slice consists of multitude of RAN-slice, Core-slice and Transport-slices each with its own controller.

To provide automation, assurance and optimization of the network slices, an E2E network slice orchestrator is needed which interacts with controller of RAN, Core and Transport slices. The interfaces between the E2E network slice controller and RAN and Core controllers are defined in various 3GPP technical specifications. However, 3GPP has not defined the same interface for transport slices. The aim of this document is to provide the clarification of this interface and to provide the information model of this interface for automation, monitoring and optimization of the transport slices.

1.1. Definition of Terms

Please refer to [I-D.homma-slice-provision-models] as well.

Customer: Also known as Tenant. Any application, client network, or customer of a network provider, i.e. an NS tenant is a person or group that rents and occupies NSIs from network provider.
E2E Network Slice: A E2E network slice is a virtual network provided by a Slice Provider that has the functionality and performance to support a specific industry sector and/or service. This capability will be the subject of a commercial agreement between the Slice Provider and Slice Buyer, and although it may well support dynamic scale-in/out, the Slice capability will normally be long-lasting i.e. only change on commercial timescales, although this may become more dynamic over time.

In specific, E2E 5G network slices are separate independent logical network E2E from user equipment (UE) to applications in a common infrastructure where each logical network has dedicated SLA. It is an E2E concept which spans multiple network domains (e.g. radio, transport and core), and in some cases more than one administrative domain.

Network Slice Instance (NSI): Also known as Network slice profile (NS profile), NSI is an E2E instance of a network slice blueprint which is instantiated in service provider’s network for specific customer and specific service type. Note that customer and service type can be wildcard.

Transport Slice: It is also called Transport Sub-Slice. A set of connections between various network functions (VNF or PNF) with deterministic SLAs. They can be implemented (aka realized) with various technologies (e.g. IP, Optics, FN, Microwave) and various transport (e.g. RSVP, Segment routing, ODU, OCH etc).

RAN Slice: It is also called RAN Sub-Slice. The context and personality created on RAN network functions for each E2E network slice.

Core Slice: It is also called Core Sub-Slice. The context and personality created on Core network (CN) functions for each e2e network slice.

S-NSSAI: Single Network Slice Selection Assistance Information, defined by 3GPP and is the identification of a Network Slice Sub-Slice: The RAN, Transport or Core Slices are also called Sub-Slice or Subnet; i.e. RAN, Transport and Core Sub-slices or Subnets

Service Level Agreement (SLA): An agreement between a customer (aka tenant) and network provider that describes the quality with which features and functions are to be delivered. It may include information on target KPIs (such as min guaranteed throughput, max tolerable latency, max tolerable packet loss rate); the types of
service (such as the network service functions or billing) to be executed; the location, nature, and quantities of services (such as the amount and location of compute resources and the accelerators require)

gNB: gNB incorporates two major modules; Centralized Unit (CU) and Distribute Unit (DU)

UE: UE: User Equipment such as vehicle Infotainment, Cell phone, IoT sensor etc

2. High level architecture of 5G end-to-end network slicing

To demonstrate the concept of 5G E2E network slicing and the role of various controllers consider a typical 5G network shown in Figure 1 where a mobile network operator Y has two customers (tenants) C1 and Public Safety. The boundaries of administrative domain of the operator includes RAN, Transport, Core and Application domains. Each customer requests to have several logical independent E2E networks from UEs (e.g. vehicle infotainment, mobile phone, IoT meters etc.) to the application, each with distinct SLAs. In 5G, each of these independent networks called an E2E network slice, which consists of RAN, Transport and Core slices (or RAN, Transport and Core Sub-slices).

In Figure 1 there are four E2E network slices, NS1 to NS4, each with its own RAN, Core and Transport slices. To create RAN slice, the RAN network functions such as eNB and gNB should be programmed to have a context for each E2E network slice. This context means that the RAN network functions should allocate certain resources for each E2E network slice such as air interface, various schedulers, policies and profiles to guarantee the SLA requirement for that specific network slice. By the same token, the Core slice means to create the context for each E2E network slice on Core network functions. This means that the RAN and Core network functions are aware of multiple E2E network slices.

When both RAN and Core slices are created, they should be connected together using a set of connections to have an E2E network slice. These set of connections are called Transport Slices, i.e. the transport slices are a group of connections with specific SLAs. The following factors dictate the number of transport slices. The details on transport slices will be discussed in see Section 4:

- The RAN deployment (i.e. distributed RAN, Centralized RAN or Cloud RAN). For example, in Cloud RAN, the RAN network function (i.e. gNB) is decomposed into two network functions (called CU and DU) where one or both will be VNFs and there is a Midhaul network
between them. In this case, there will be a transport slice in RAN to connect DUs to CU.

- The location of the mobile applications. If there is a network between the mobile applications and the 5G Core, another transport slice is needed to connect the 5G Core to Applications.

- Mobile network policy on how to allow creation of the E2E network slices and if the sharing of transport slices between multiple E2E network slices are desirable and allowed.

At the end when RAN, Core and Transport slices are created, they should be bind or associated together to form a working E2E network slice. Since the nature of an E2E network slice is dynamic and the life cycle of each network slice might be a few hours or few months, various controllers are needed to perform the life cycle of various E2E network slices in their respective domains. As shown in Figure 1, each RAN, Core and Transport slice needs a controller. Also an E2E network slice controller is needed to provide the coordination and control of network slices from an E2E perspective.

In summary, an E2E network slice will involve several domains, each with its own controller: 5G RAN domain, transport domain, and 5G core domain. These need to be coordinated in order to deliver an E2E service. Note that in this context a service is not necessarily an IP/MPLS service but rather customer (aka tenant) facing services such as CCTV service, eMBB service and Infotainment service etc.
Figure 1: High level architecture of 5G end-to-end network slicing

Customers (aka Tenants): Public Safety and C1
MNO (Mobile Network Operator): Operator Y

Legend:
----- NS1: E2E network slice 1 for customer C1,
service type Infotainment
====== NS2: E2E network slice 2 for customer C1,
service type Autonomous Driving
+++++ NS3: E2E network slice 3 for customer C1,
service type HD Map (NS3)
--- NS4: E2E network slice 4 for customer Public Safety,
service type Video Surveillance
3. Logical flow cross layers for automation of an end-to-end network slices

Figure 2 provides the logical flow cross layers to achieve the automatic creation of each E2E network slices such as NS1 mentioned in Figure 1. Below are the logical flow among various controllers to achieve this. It is important to note that these steps are logical and in practice some of them can be combined. For example, step 3 can be combined with steps 4 or 5.

1. The customer C1 will request the Operator Y for creation of an E2E network slice for Infotainment service type and SLA of 10 Mbps

2. The E2E network slice controller receives this request and using its pre-defined network slice blueprints (aka network slice templates) creates a network slice profile (aka network slice instance) which contains all the network functions in RAN and core which should be part of this E2E network slice. It then goes through decomposition of this profile and triggers various actions. Steps 3 to 7 shows these actions.

3. A request for creation of the RAN slice will be triggered to RAN Controller.

4. If RAN network functions are virtual, the RAN Slice controller triggers the creation of VNFs in RAN (using for example ETSI interface Os-Ma-nfvo)

5. NFVO manages the life cycle of virtual RAN network functions

6. Since both physical and virtual RAN network functions which are part of the E2E network functions are known to RAN controller, it then triggers the creation of RAN slice by programming RAN network functions

7. Similar to previous steps, a request for creation of the Core slice will be triggered to Core Controller.

8. If Core network functions are virtual, the Core Slice controller triggers the creation of VNFs (using for example ETSI interface Os-Ma-nfvo) and NFVO manages the life cycle of virtual Core network functions

9. Since both physical and virtual Core network functions which are part of the E2E network functions are known to Core controller, it then triggers the creation of Core slice by programming Core network functions
10. In this step, various transport slices (i.e. various connections) need to be created between various network functions, e.g. transport slices between RAN and Core slices, transport slices between RAN network functions or transport slices between core and applications.

11. The transport slices will be implemented (aka realized) in the network.

12. [Optional] If the creation of transport slice involves creation of VNFs (e.g. Firewall, security gateway etc.), triggers the creation of these VNFs (using for example ETSI interface Os-Ma-nfvo).

13. The E2E network slice controller binds (or associates) all these slices together to form a single E2E network slice for specific customer and specific service type.

14. At the end, when the E2E network slice is created, the E2E network slice controller will allocate a unique network slice id (called S-NSSAI) and eventually during the UE network attach, all the UEs will be informed about the existence of this newly created E2E network slice and then they can request it using the 3GPP 5G signaling procedures.

Note that the interfaces 3 and 7 between the E2E network slice Controller and RAN and Core controllers and their information models are defined in various 3GPP technical specifications. However, 3GPP has not defined the same interface for transport slices, i.e. interface 10. The aim of this document is to define this interface. More details will be provided in Section 5.
Figure 2: Logical flow cross layers for automation of an end-to-end network slices
4. Definition of Transport Slice

Since the transport slice is an important concept throughout this document, this section describes this concept in more detail. To this end, consider Figure 3 where a group of physical or virtual network functions (PNF, VNF or both) are connected together. In particular, a single transport slice is defined as:

- A distinct set of connections between multiple VNFs and PNFs each with its own deterministic SLA which is implemented (aka realized) in the network using any technology (e.g. IP, Optics, Microwave and PON), any tunnel types (e.g. IP, MPLS, SR, ODU/OCH) and any L0/L1/L2/L3 service types

```
----- | (---------------------)
| NF11 |
----- | ( Transport Slice ) |
----- | ( ) |
----- | ( ) |
NF21 |
----- | ( ) |
NF12 |
----- | ( A set of distinct ) |
----- | ( connetions connecting ) |
----- | ( physical or virtul network functions (PNF/VNF)) |
----- | ( NF11, NF12, ..., NF1n to ) |
NF2m |
----- | ( ) |
NF1n |
----- | ( ) |
----- | (---------------------)
```

Figure 3: Definition of transport slice

For clarity, in next three sections, a few examples of the transport slices will be provided for following RAN deployments:

- Distributed RAN
- Centralized RAN (C-RAN)
- Cloud RAN

4.1. Transport Slices in Distributed RAN

Distributed RAN is the most common deployment of 4G and 5G RAN networks where as shown in Figure 4, the RAN network is connected to Core network using the transport network. Optionally the mobile applications can be also connected to the Core network using another
overlay network (e.g. Internet where the mobile applications are virtualized in another data center).

In this case, for a single E2E network slice, in addition to RAN and Core slices, there are two sets of transport slices; TS_1 and TS_2. TS_1 is connecting the RAN to Core and TS_2 is connecting Core to Applications.

\[
\begin{array}{c}
\text{--------} \text{End to End Network Slice} \text{--------} \\
\text{--------} \text{RS} \text{---} \text{CS} \text{---} \\
\text{--------} \text{TS}_1 \text{---} \\
\text{--------} \text{TS}_2 \text{---} \\
\end{array}
\]

Legend:

- **TS**: Transport Slice
- **RS**: RAN Slice
- **CS**: Core Slice

**Figure 4: Transport slices in distributed RAN**

### 4.2. Transport Slices in Centralized RAN (C-RAN)

The RAN consists of two functional units, the baseband unit (BBU) and the radio unit (RU). The BBU processes the signal and is connected to the transport network. The RU transmits and receives the carrier signal that is transmitted over the air to the end user equipment (UE). In Centralized RAN (aka C-RAN) as depicted in Figure 5, the RU and BBU are separated by a network called fronthaul network. In this case a single E2E network slice contains three set of transport slices TS_1, TS_2 and TS_3 where TS_1 and TS_2 are identical to distributed RAN case (see Figure 4) and the new TS_3 connects the Radio Units (RU) to the BBUs.
4.3. Transport Slices in cloud RAN

In new cloud-RAN architecture, the baseband unit functionality is further divided into real-time and non-real-time. The former is deployed close to antenna to manages the real-time air interface resources while the non-real-time control functions are hosted centrally in the cloud. In 5G, BBU is split into two parts called CU (Central Unit) and DU (Distributed Unit) as shown in Figure 6 where these entities are connected by new network called Midhaul.

In this deployment for a single E2E network slice, there are four sets of transport slices TS_1, TS_2, TS_3 and TS_4 where TS_1 and TS_2 and TS_3 are identical to distributed and centralized RAN (see Figure 4 and Figure 5). The new transport slice TS_4 will connect DUs to CUs.
To further explore the content of a transport slice, let’s focus on transport slice TS_1 between RAN and Core. Note that the following discussion is also applicable to any other transport slices TS_2, TS_3 or TS_4. As shown in Figure 7 for an individual E2E network slice belongs to a specific customer for a specific service type, the RAN domain is connected to Core domain through the transport network. In this example, the E2E network slice is identified by n-nssai=10 for customer C1 and service type Infotainment. Two RAN network elements BBU1 and BBU2 and two Core network elements AMF1 and UPF1 are all part of the E2E network slice. There are two sets of distinct connections between RAN and Core domains;

- Network-C which connects BBU1 and BBU2 to AMF1 with SLA-C
- Network-U which connects BBU1 and BBU2 to UPF1 with SLA-U
Customer: C1
Service type: Infotainment
S-NSSAI: 10
Network-C (from BBU-1.tp1, BBU-2.tp1 to AMF1.tp1 with SLA-C)
Network-U (from BBU-1.tp2, BBU-2.tp2 to UPF1.tp2 with SLA-U)

Transport slice TS_1: { Network-C and Network-U }

Note that the SLA-C and SLA-U can be different. The combination of these two networks is called a single transport slice TS_1. Note that the definition of the transport slice does not specify how these connections should be realized (or implemented). It also does not specify which technology (e.g. IP, MPLS, Optics etc.) or tunnel type (e.g. MPLS, Segment Routing etc.) should be used during the realization. Those are part of realization of the transport slice done by transport slice controller. With this approach the definition is logically separated from implementation of transport slices. This gives a tremendous programmability and flexibility during the realization of transport slices using any type of technologies and tunnel types.
In summary, a transport slice is a distinct set of technology-agnostic connections each with its own deterministic SLA which can be implemented (aka realized) using any technology, tunnel type and service type.

5. Transport Slice Connectivity Interface (TSCi)

As shown in Figure 2, the interfaces 3 and 7 between the E2E network slice Controller and RAN and Core controllers, respectively and their information models are defined in various 3GPP technical specifications [TS.28.530-3GPP], [TS.28.531-3GPP], [TS.28.540-3GPP] and [TS.28.541-3GPP]. However, 3GPP has not defined the same interface for transport slices, i.e. interface 10. The aim of this document is to provide the clarification of this interface and to provide the information model of this interface. The interface is called the Transport Slice Connectivity interface (TSCi) which provides some means for automation, monitoring and optimization of the transport slices.

This new interface is needed in order to simplify the creation of the Transport slices and hides all the complexity of implementation (aka realization) from higher E2E network slice controller similar to their RAN and Core counterparts defined by 3GPP.

The aim of this document is to define a new interface and its information model between the E2E network slice controller and the transport slice controller. The characteristics on this new interface are:

a. The interface allows a request and response about resources. It should not allow negotiation, as this would be complex and not have a clear benefit

b. This interface is needed by the same layer that configures 3GPP RAN and Core slices to support the E2E path management in 5G networks. The provider of this interface is the higher layer controller which needs to create a connectivity between two network functions. The provider of this interface is the lower layer controller which provide the implementation (aka realization) of this connectivity (i.e. transport slice).

c. This interface is needed in industry to achieve true standard based automation of 5G E2E network slices. It provides a technology-agnostic intent-based interface to the E2E network slice controller similar to interfaces defined by 3GPP for RAN and Core slices
d. This interface is independent of type of network functions which needs to be connected, i.e. it is independent of any specific repository, software usage, protocol, or platform which realizes the physical or virtual network functions.

e. The interface standardizes a way to learn about what resources are available in the network which impact the creation of the transport slices.

f. This technology independent interface simplifies the creation of the transports slices by describing it in a standard way along with all the network functions (such as eNB, gNB, CU, DU, Core, Mobile application server, etc.) that compose a transport slice, their properties, attributes and their SLA requirements, i.e. the connectivity resources requested from an E2E network slice controller to transport slice controller with their corresponding SLA requirements.

g. This interface provides capabilities for transport slice monitoring and analytics. It means that the TSCi interface allows enabling/disabling the collection of the transport slices telemetry, assurance and Threshold Crossing Alert (TCA) data and providing them to the E2E network slice controller.

h. This interface provides capabilities for optimization of the transport slices. Since the nature of an E2E network slice is dynamic, it is important to make sure the transport slice SLA are valid and in case any SLA violation happens, the transport slice controller performs the closed-loop action and informs the upper layer E2E network slice controller for the violation and closed-loop action. This interface allows this functionality.

i. This interface allows binding and association between RAN to Transport slices and between Core to Transport slices.

j. This interface complements various IETF service, tunnel, path models by providing an abstract layer on top of these models.

5.1. Relationship between TSCi and various IETF data models

The transport slice connectivity interface and its relationship to various IETF data models are not addressed in any IETF WGs as it has very unique characteristics of the 5G E2E network slicing. The new interface complements various IETF service, tunnel, path data models by providing an abstract layer on top of these models.
Figure 8: Relationship between transport slice interface and IETF Service/Tunnels/Path data models

Figure 8 shows more details on how the new transport slice connectivity interface (TSCi) relates to various IETF service/tunnels/path models. The transport slice controller receives a request for creation of a transport slice. This request is an abstract intent-based request which contains the required connections between various network functions in RAN and Core. The transport slice controller will then realize or implement those connections using various IETF models.

In a sense, the new transport slice connectivity interface provides an abstract layer on top of IETF models. The IETF service, path and tunnel data models can be any existing IETF service models such as L3SM or L2SM ([RFC8049] and [RFC8466]). It also can be any future data models.

5.2. Realization (aka Implementation) of transport slices

Since the transport slice connectivity interface describes the connections not the services, it is essential to make a distinction between connections and implemented services. Referring to Figure 9, assume using the new transport slice interface, the E2E network slice controller requests the creation of a transport slice which has multiple connections between RAN and Core. One of these connections is shown below between RAN1 and UPF1. The E2E network slice controller can request a connection between RAN1 to UPF1 for a
specific tenant, specific service type and specific SLA (e.g. customer Public Safety for service CCTV with latency of 5 [ms] or better).

---

<------------------------>
IP/MPLS service, path and tunnel between BR1 & BR2

<------------------------>
Connectivity between RAN1 and UPF1 (which is part of a Transport Slice)

Legend:

<--- Connection part of the transport slice
<===> Implementation (aka realization) of the transport slice

Figure 9: Distinction between Connections and Services

To realize (aka implement) this connection, the transport slice controller, will find the endpoint for the L0/L1/L2/L3 services, find the best path and create a service between these endpoints. In this example, in order to implement the connection between RAN1 and UPF1, the transport slice controller will first find the best boarder routers BR1 and BR2, find the best path between them and finally creates a Service/path from BR1 to BR2. It is important to note that:

- The endpoints of the Connection are different from the endpoints of the Services, paths and tunnels. This is the unique characteristic of transport slices where the endpoints of the connections are different from endpoints of the Services (i.e. endpoint of the transport slices are RAN1 and UPF1 whereas the endpoint of services are BR1 and BR2.

o The Service/path API can be any existing IETF service models such as L3SM or L2SM ([RFC8049] and [RFC8466]). It also can be any future service model.

o In order to have the connectivity between RAN1 and UPF1, the RAN and Core slices should be associated to Transport slice. This is also a by-product of the Transport slice connectivity interface when all allocated resources for access points (such as allocated VLAN IDs, IP addresses etc) are conveyed to RAN and Core Slices. This will be done by coordination between transport slice controller and E2E network slice controller.

6. Transport slice connectivity interface (TSCi) information model

Based on the definition of a transport slice (see Section 4), the high-level information model of a transport slice connectivity interface should conform with Figure 10:
module: transport-slice-connectivity

---rw transport-slice
   ---rw transport-slice-info
      ---rw ts-id
      ---rw ts-name
      ---...

   ---rw network-slice-info [ns-id]
      ---rw list of s-nnsai (i.e. E2E network slice id)
      ---rw customer (aka tenant)
      ---rw service type (e.g. CCTV, infotainment etc)
      ---rw NS IDs (i.e. list of S-NSSAI)
      ---...

   ---rw transport-slice-networks* [network-id]
      ---rw network-id
      ---...

      ---rw node* [node-id]
         ---rw node-id
         ---...

      ---rw connection-link* [link-id]
         ---rw link-id
         ---rw endpoint-A
         ---rw endpoint-B
         ---...

      ---rw transport-slice-policy* [policy-id]
         ---rw policy-id
         ---rw policy-type (e.g. sla-policy, selection-policy, assurance-policy)
         ---...

Figure 10: High-level information model of transport slice connectivity interface

The proposed transport slice information model should include the following building blocks:

- transport-slice-info: Information related to transport slice
- network-slice-info: A list of all E2E network slices mapped to transport slice
- transport-slice-networks: Each transport slice is a set of networks. Each network contains:
  * list of nodes (i.e. list of RAN and Core network functions)
  * list of connection-links (i.e. list of connections between nodes)
* list of transport-slice-policies (i.e. various SLA, Selection and Monitoring policies)

6.1. transport-slice-info

It contains information such as transport slice name, transport slice ID etc. The details will be provided in next version of this draft.

6.2. network-slice-info

As discussed in Section 3, a request for creation of a transport slice starts with the fact that a customer (aka tenant) will request an E2E network slice from an operator for a certain service type (e.g. CCTV, Infotainment, URLLC, etc.). So, there is a mapping between transport slice and the E2E network slice. Depending on the deployment, it is possible to map multiple E2E network slice to a transport slice, i.e. the mapping between transport slice to E2E network slice is one to many.

The network-slice-info contains the list of E2E network slices which are mapped to the transport slice with all relevant information such as S-NSSAI, name of customer, service type etc. The details will be provided in next version of this draft.

6.3. transport-slice-networks

As per Section 4, a transport slice will contain one or more transport-slice-networks.

6.4. node

As discussed in Section 4, the transport slice comprises one or more connectivity networks between RAN and Core network elements. It is also possible to have the connectivity networks between RAN and RAN network elements for some RAN deployments (example for midhaul network). As discussed in section 2.2, when the transport slice is realized (implemented), the network elements which are the endpoints of realization of the transport slice might be different. As a result, the nodes in this model represent RAN or Core network elements. However, the model is flexible where nodes might represent the routers or switches which are the endpoints of the transport slice realization.

The attributes of the node are IP address, node name, domain ID and termination points. The details will be provided in next version of this draft.
6.5. connection-link

Each transport-slice-networks contain one or more connections between various nodes described in Section 6.4. The connection-link is a list of links which are represented by endpoint-A, endpoint-B etc. The details will be provided in next version of this draft.

6.6. transport-slice-policy

To establish a transport slice, one or more transport-slice-networks will be created each with certain SLA requirement which is represented by transport-slice-policy. This draft proposes three types of transport slice policies to be supported:

- transport-slice-sla-policy
- transport-slice-selection-policy
- transport-slice-assurance-policy

The summary of these policies will be provided here. The details will be provided in next version of this draft.

6.6.1. transport-slice-sla-policy

This is a mandatory policy. The transport-slice-policy represents in a generic and technology-agnostic way the SLA requirement needed to realize a group of connections which are part of a transport slice. It is defined per transport-slice-network. It contains the bounded latency, bandwidth, reliability, security etc.

6.6.2. transport-slice-selection-policy

This is an optional policy. In some deployment, the E2E network slice controller might want to assist the transport slice controller on how to realize a transport slice by providing some information regarding the type of technologies and tunnels. This information will be provided in transport-slice-selection-policy.

6.6.3. transport-slice-assurance-policy

This is a mandatory policy. The E2E network slice controller shall influence the transport slice controller for transport slice assurance on how to perform monitoring, analytics and optimization. To this end, the transport-slice-assurance-policy will be used. It contains, the type of assurance needed, time interval, how often inform the E2E network slice controller etc.
6.7. IETF models

Currently none of the IETF data models address the modeling of transport slice connectivity as shown in Figure 6. However, the various IETF data models might be augmented to address the information model of the transport slice connectivity interface. Following is the list of candidates IETF YANG models. This is not an extensive list and the next version of the draft will provide a more comprehensive list.

6.7.1. ACTN model

As defined in [RFC8453], [I-D.king-teas-applicability-actn-slicing] and [I-D.ietf-teas-actn-vn-yang] a VN (Virtual Network) is the abstract customer view of the TE network. The VN can be seen as a set of edge-to-edge abstract links (a Type 1 VN). Each abstract link is referred to as a VN member and is formed as an E2E tunnel across the underlying networks.

This definition is very similar to definition of the transport slice which means that the VN YANG model can be augmented to address the modeling of the transport slice shown in Figure 6. This is work in progress for next version of this document.

6.7.2. i2rs model

Similar to [I-D.qiang-coms-netslicing-information-model], the data model for network topologies developed in [[I-D.ietf-i2rs-yang-network-topo] can be augmented to address the modeling of the transport slice. This is work in progress for next version of this document.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

TBD

9. Informative References

[I-D.boucadair-connectivity-provisioning-protocol]


Authors’ Addresses

Reza Rokui
Nokia
Canada

Email: reza.rokui@nokia.com
Shunsuke Homma
NTT
3-9-11, Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Email: homma.shunsuke@lab.ntt.co.jp

Diego R. Lopez
Telefonica I+D
Spain

Email: diego.r.lopez@telefonica.com

Xavier de Foy
InterDigital Inc.
Canada

Email: Xavier.Defoy@InterDigital.com

Luis M. Contreras-Murillo
Telefonica I+D
Spain

Email: luismiguel.contrerasmurillo@telefonica.com

Jose J. Ordonez-Lucena
Telefonica I+D
Spain

Email: joseantonio.ordonezluccena@telefonica.com

Pedro Martinez-Julia
NICT
Japan

Email: pedro@nict.go.jp