YANG Data Models MPLS Base and Static LSPs

draft-ietf-mpls-base-yang-04

draft-ietf-mpls-static-yang-04

code @ https://github.com/ietf-mpls-yang/te

Tarek Saad, Kamran Raza, and Rakesh Gandhi, Cisco Systems
Vishnu Pavan Beeram, Juniper Networks
Xufeng Liu, Jabil
Igor Bryskin, Huawei
Himanshu Shah, Ciena
R. Jones, Brocade
B. Wen, Comcast

Agenda

- Updates to I-Ds (since IETF98)
- Open issues
- Next steps

Summary of Changes

- Migrated MPLS label stack from "`leaf-list`" to "`list`" with "`key index`"
- Met with authors of "draft-acee-rtgwg-yang-rib-extend":
  - Discussed moving per route attribute extensions from MPLS YANG model to RIB-EXT model
  - Discussed existing RIB's model per-AF next-hop address limitation

Update # 1 < draft-ietf-mpls-base-04>
MPLS label stack modelling (list ordering)

```
+--rw mpls-static:outgoing-labels
  +--rw mpls-static:outgoing-labels* [index]
     +--rw mpls-static:index     -> ../config/index
     +--rw mpls-static:config
     |  +--rw mpls-static:index?   uint8
     |  +--rw mpls-static:label?   rt-types:mpls-label
     +--ro mpls-static:state
        +--ro mpls-static:index?   uint8
        +--ro mpls-static:label?   rt-types:mpls-label
```

Label Index:

- distinguisher to allow repetition of same label in the stack
- can be used to derive the “offset” per entry in label stack – e.g. index == offset?

List ordering modes:

- Ordered-by-user – User defined order:
  - order of appearance in configuration should be same as that in label stack (top to bottom)
  - can use insert/delete before/after, top, etc.
- Ordered-by-system (default):
  - order of appearance in the configuration list is not important
  - backend sorts/generates label offset in stack from label entries
  - backend may use list index as label offset “as-is”
    - if so, need to validate presence of n-1 before accepting n
    - if not, recomputes label offset from sorted label list entries
    - if not, backend needs to resort/recompute offset everytime a new entry is added/removed from the list

Update # 2
MPLS label stack modelling (list ordering)

```
+--rw mpls-static:outgoing-labels
  +--rw mpls-static:outgoing-labels* [index]
    +--rw mpls-static:index     -> ../config/index
    +--rw mpls-static:config
    |  +--rw mpls-static:index?   uint8
    |  +--rw mpls-static:label?   rt-types:mpls-label
    +--ro mpls-static:state
      +--ro mpls-static:index?   uint8
      +--ro mpls-static:label?   rt-types:mpls-label

+--ro route*
  +--ro next-hop
  |    +--:(next-hop-list)
  |      +--ro next-hop-list
  |        +--ro next-hop*
  |          +--ro outgoing-interface?   if:interface
  |          +--ro mpls:index?          string
  |          +--ro mpls:backup-index?    string
  |          +--ro mpls:loadshare?       uint16
  |          +--ro mpls:role?            nhlfe-role
  |          +--ro mpls:remote-labels* [index]
  |            +--ro mpls:index    uint8
  |            +--ro mpls:label?   rt-types:mpls-label
```

Label Index:

- distinguisher to allow repetition of same label in the stack
- can be used to derive the “offset” per entry in label stack – e.g. index == offset?

List ordering modes:

- Ordered-by-user – User defined order:
  - order of appearance in configuration should be same as that in label stack (top to bottom)
  - can use insert/delete before/after, top, etc.
- Ordered-by-system (default):
  - order of appearance in the configuration list is not important
  - backend sorts/generates label offset in stack from label entries
  - backend may use list index as label offset “as-is”
    - if so, need to validate presence of n-1 before accepting n
    - if not, recomputes label offset from sorted label list entries
    - if not, backend needs to resort/recompute offset everytime a new entry is added/removed from the list

Open Issue # 1
MPLS label encoding

```
grouping mpls-label-stack {
  description
    "A grouping that specifies an MPLS label stack.";
  container mpls-label-stack {
    description
      "Container for a list of MPLS label stack entries.";
    list entry {
      key "id";
      description
        "List of MPLS label stack entries.";
      leaf id {
        type uint8;
        description
          "Identifies the sequence of an MPLS label stack entries.
           An entry with smaller ID value is precedes an entry in
           the label stack with a smaller ID.";
      }
      leaf label {
        type rt-types:mpls-label;
        description
          "Label value.";
      }
      leaf ttl {
        type uint8;
        description
          "Time to Live (TTL).";
        reference
          "RFC3032: MPLS Label Stack Encoding.";
      }
      leaf traffic-class {
        type uint8 {
          range "0..7";
        }
        description
          "Traffic Class (TC).";}
    }
  }
} // mpls-label-stack
```

Currently MPLS-Static model uses

- list of rt-types:mpls-label values only

ietf-routing-type defines label stack as:

  - label, ttl, traffic-class

- allows setting TC per entry in stack at ingress
  - TC -> PHB on LSRs
  - rules defined in RFC3270 (short/Uniform) Pipe model
- allows setting TTL per entry in stack on ingress
  - rules defined in RFC3443 for short/uniform Pipe models (section 3.4)
  - should this show in state (if so, is it iTTL, outTTL?)
- Should same grouping be used in MPLS/RIB state too?
  - Usually routes appearing in RIB (installed by signaling protocols e.g. LDP, BGP, RSVP) will not set TC, TTL, etc.

Open Issue # 2
Use extended RIB route attributes

```
+--ro route*
  +--ro next-hop
  |    +--:(next-hop-list)
  |       +--ro next-hop-list
  |          +--ro next-hop*
  |             +--ro outgoing-interface?   if:interface
  |             +--ro mpls:index?           string
  |             +--ro mpls:backup-index?    string
  |             +--ro mpls:loadshare?       uint16
  |             +--ro mpls:role?            nhlfe-role
  |             +--ro mpls:remote-labels* [index]
  |                +--ro mpls:index    uint8
  |                +--ro mpls:label?   rt-types:mpls-label
```

- Migrate to using the newly introduced RIB route extended model < draft-acee-rtgwg-yang-rib-extend>
  - for repair-path attributes
    - originally defined in MPLS model but applicable to routes for other Afs
  - Follow-up on missing per-path attributes (e.g. loadshare)

Open Issue #3:

Migration to NMDA style

- Migrate from OC-style to the recommended NMDA style

Next Steps

- Address the open issues
- Request further review and comments on the models

Thank You