Traffic Engineering Database in FRR

Design & Architecture

Olivier Dugeon

Why FRR need TED ?

Segment Routing

- To compute Traffic Engineered path (SR-TE)
- To compute next hop for Flexible Algorithm
- Both IS-IS and OSPF are concern
  - Could remain internal to each routing daemon, but code could benefit of common code factorization

BGP Link State

- Link State are extracted from Traffic Engineering Link State convey by OSPF and IS-IS
  - BGP LS needs to collect information from IS-IS and / or OSPF in a common way

RSVP-TE & PCE

- This is mandatory to setup and compute path
  - Both protocols assume the existence of an up to date TED that represent the network topology

A common representation of TED

First approach could consist to use directly LSA (OSPF) and LSP (IS-IS)

- TLVs and Sub-TLVs are not the same even if they convey similar values e.g. TE metric
- Impose to have 2 decoders for BGP Link State, RSVP-TE & co
- Impose to have 2 set of messages to convey information between daemons

A common TED approach overcome these limitations

- OSPF, respectively IS-IS are in charge to convert LSA, respectively LSP, to TED structure
- BGP Link State, RSVP-TE & co use the common decoder
- Same set of messages are used to convey TED information

Possibility to model TED structure in yang

- But it is impossible to model a Connected Graph mandatory for path computation (see https://git.opendaylight.org/gerrit/c/bgpcep/+/86954/12/docs/graph/graph-user-guide-graph-model.rst)

Graph model

```
module: graph
+--rw graph-topology
+--rw graph* [name]
       +--rw name          string
       +--rw graph-type?   enumeration
       +--rw asn?          uint32
       +--rw vertex* [vertex-id]
       |  +--rw vertex-id      uint64
       |  +--rw name?          string
       |  +--rw router-id?     inet:ip-address
       |  +--rw vertex-type?   enumeration
       |  +--rw srgb
       |  |  +--rw lower-bound?   uint32
       |  |  +--rw range-size?    uint32
       |  +--rw asn?           uint32
       +--rw edge* [edge-id]
       |  +--rw edge-id              uint64
       |  +--rw local-vertex-id?    uint64
       |  +--rw remote-vertex-id?   uint64
       |  +--rw name?               string
       |  +--rw edge-attributes
       |     +--rw metric?                     uint32
       |     +--rw te-metric?                  uint32
       |     +--rw color?                      uint32
       |     +--rw local-address?              inet:ip-address
       |     +--rw remote-address?             inet:ip-address
       |     +--rw local-identifier?           uint32
       |     +--rw remote-identifier?          uint32
       |     +--rw max-link-bandwidth?         decimal64
       |     +--rw max-resv-link-bandwidth?    decimal64
       |     +--rw unreserved-bandwidth*       decimal64
       |     +--rw delay?                      delay
       |     +--rw min-max-delay
       |     |  +--rw min-delay?   delay
       |     |  +--rw max-delay?   delay
       |     +--rw jitter?                     delay
       |     +--rw loss?                       loss
       |     +--rw residual-bandwidth?         decimal64
       |     +--rw available-bandwidth?        decimal64
       |     +--rw utilized-bandwidth?         decimal64
       |     +--rw adj-sid?                    uint32
       |     +--rw backup-adj-sid?             uint32
       |     +--rw srlgs*                      uint32
       +--rw prefix* [prefix]
       +--rw prefix        inet:ip-prefix
       +--rw prefix-sid?   uint32
       +--rw vertex-id?    uint64
```

Exchange TED between daemons

OSPF API

- Already available, but too OSPF centric (convey LSA only)
- Need to write similar API for IS-IS
- Need to change the API to convey TED structure

Dedicated daemon

- Cons. Setup an extra daemon which need to be monitored to convey some messages
- Pro. Not add more task to ZEBRA layer

Add new ZAPI message

- Pro. Easiest (Base on actual route redistribution) and Fastest solution (no need to code a new bus infrastructure)
- Conf. Add one more message to ZEBRA, but just to pass information between daemon (relay mode)

New paradigm with a dedicated FRR BUS

- Split ZEBRA in 2 parts:
  - Kernel interaction (actual ZEBRA)
  - Communication between daemons (new FRR BUS)

Proposed code

First target: BGP Link State

New set of functions inside Library

- TED (Graph) structure
- TED management (CRUD) including Vertex, Edge & Prefix

New ZAPI messages to redistribute TED information

- Send & receive complete TED
- Register daemon to receive TED update
- Send TED update

Path Computation Algorithms inside Library

- In a second step if needed

Thanks