Core Routing Data Model

`draft-ietf-netmod-routing-cfg-16`

Ladislav Lhotka
⟨lhotka@nic.cz⟩

12 November 2014

Core Routing Data Model

Framework enabling coordinated configuration and management of routing protocols.

Three YANG modules:

- *ietf-routing* – AF-independent parts,
- *ietf-ipv4-unicast-routing*, *ietf-ipv6-unicast-routing* – data nodes specific for IPv4 and IPv6.

The YANG modules and I-D are being developed at

`https://github.com/yang-routing/yang-routing`

Essential Components

- routing instance
- route and RIB
- routing protocol
- route filter

```
+--rw routing
   +--rw routing-instance* [name]
   |  +--rw name
   |  +--rw type?
   |  +-- ...
   |  +--rw routing-protocols
   |     +--rw routing-protocol* [type name]
   |        +--rw name
   |        +--rw type
   |        +-- ...
   +--rw ribs
   |  +--rw rib* [name]
   |     +--rw name
   |     +--rw address-family
   |     +-- ...
   +--rw route-filters
      +--rw route-filter* [name]
         +--rw name
         +--rw type
         +-- ...
```

Scope

- vertical: everything from hosts to high-end routers,
- horizontal: common data model for all implementations.

The data model needn't mirror any CLI/implementation but implementors should be able to map their CLI to the data model, and the configuration data to their router implementation.

YANG has flexible modularity mechanisms (modules/submodules, augments, features, deviations) but this may still be too ambitious(?).

The CLI legacy is rather strong – most of the routing know-how is expressed in terms of a CLI configuration and commands.

Data Model Dilemmas

- interface configuration/state data: everything under `if:interface`, or scattered interface lists
- route redistribution: directly between routing protocols, or via RIBs;
- protocol-centric v. instance-centric design.

Alternative Scenarios

❶ standard data models,

❷ multiple vendors-specific models,

❸ combination of both.