Core Routing Modules

`draft-ietf-netmod-routing-cfg-02`

Ladislav Lhotka
⟨*lhotka@nic.cz*⟩

26 March 2012

Main changes

- Each router contains a list of logical interfaces assigned to it.
- New module *ietf-ipv6-unicast-routing*.
- Additional semantic constraints via **must** statements.
- Routes to directly connected networks end up in the FIB table by default.
- Simplifications and cleanup, use of additional groupings.

Main Data Tree

```
+--rw routing
   +--rw router [name]
      +--rw name                    string
      +--rw description?            string
      +--rw enabled?                boolean
      +--rw interfaces
      |  +--rw interface [name]
            ...
      +--rw routing-protocols
      |  +--rw routing-protocol [name]
            ...
      +--rw route-filters
      |  +--rw route-filter [name]
            ...
      +--rw routing-tables
         +--rw routing-table [name]
            ...
```

Router Interfaces

A router instance can use logical interfaces that are assigned to it by including the corresponding entry in the list of router interfaces. Each interface may be assigned to no more than one router.

```
list interface {
  key "name";
  leaf name {
    type if:interface-ref;
  }
}
```

Other modules may augment this list with additional routing-specific configuration parameters and/or operational state data, see IPv6 RA parameters in *ietf-ipv6-unicast-routing*.

Module *ietf-ipv6-unicast-routing*

❶ provides IPv6-specific content of routes,

```
grouping route-content {
  uses rt:route-content;
  leaf dest-prefix {
    type inet:ipv6-prefix;
  }
  leaf next-hop {
    type inet:ipv6-address;
  }
}
```

from *ietf-routing*

❷ defines interface parameters required by RFC 4861 (mainly IPv6 router advertisements), except *IsRouter* – *ietf-ip* module seems to be a better place for it.

IPv6 RA Parameters

```
+--rw rt:interface [name]
  +--rw name                                  if:interface-ref
  +--rw v6ur:ipv6-router-advertisements
  |  +--rw v6ur:send-advertisements?    boolean
  |  +--rw v6ur:max-rtr-adv-interval?   uint16
  |  +--rw v6ur:min-rtr-adv-interval?   uint16
  |  +--rw v6ur:managed-flag?           boolean
  |  +--rw v6ur:other-config-flag?      boolean
  |  +--rw v6ur:link-mtu?               uint32
  |  +--rw v6ur:reachable-time?         uint32
  |  +--rw v6ur:retrans-timer?          uint32
  |  +--rw v6ur:cur-hop-limit?          uint8
  |  +--rw v6ur:default-lifetime?       uint16
  |  +--rw v6ur:prefix-list
  |     +--rw v6ur:prefix [seqno]
  |        +--rw v6ur:seqno                uint8
  |        +--rw v6ur:prefix-spec?         inet:ipv6-prefix
  |        +--rw (control-adv-prefixes)?
  |           +--:(no-advertise)
  |           |  +--rw v6ur:no-advertise?         empty
  |           +--:(advertise)
  |              +--rw v6ur:valid-lifetime?       uint32
  |              +--rw v6ur:on-link-flag?         boolean
  |              +--rw v6ur:preferred-lifetime?   uint32
  |              +--rw v6ur:autonomous-flag?      boolean
```

Open Issues

1. Interactions with other core modules, namely *ietf-interfaces* and *ietf-ip*.
2. Extra module for IPv6 neighbour discovery parameters?
3. Relax the isolation of router instances?
4. Generalize the *get-route* RPC method?
5. Semantics of **unique** statement.

Interactions with *ietf-interfaces* and *ietf-ip*

- The *is-router* flag should be defined somewhere, probably in *ietf-ip*, separately for IPv4 and IPv6.
- If *is-router* is false for an interface and address family, the subtree `/rt:routing/rt:router/rt:interfaces` may still contain routing-specific configuration for that interface and address family but it won't be used.
- Analogically for the *enabled* flags under *ip:ipv4*, *ip:ipv6* and *if:interface*.

IPv6 ND parameters

Thomas Morin suggested in his review that IPv6 ND parameters be moved to a separate module.

Options:

1. Keep the IPv6 ND parameters in the *ietf-ipv6-unicast-routing*.
2. Create a new module *ietf-ipv6-nd*.
3. Remove the parameters and leave it to IPv6 experts.

Isolation of Router Instances

"Although it it not enforced by the data model, different router instances normally do not internally share any data."

Thomas Morin: this might limit the usefulness of multiple router instances. For example, in RFC 4364 (MPLS/BGP VPNs) the master router exchanges routes in both directions with VRF instances.

Proposal: Remove all constraints regarding the isolation of router instances. Implementations will have to specify the rules for information exchange among router instances.

get-route

"Which route is used for forwarding datagrams to the destination with address X?"

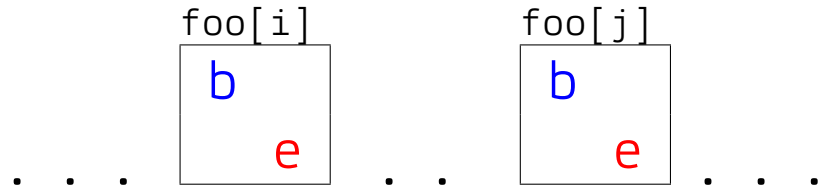
- input: destination address,
- output: the (unique) active route from FIB.

Other methods for querying the routing system will be needed, but do we have enough information to define them now and include them in the first revision of the core routing module?

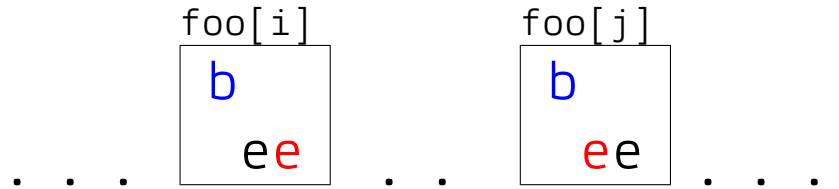
YANG **unique** statement

RFC 6020: *The "unique" constraint specifies that the combined values of all the leaf instances specified in the argument string, including leafs with default values, MUST be unique within all list entry instances in which all referenced leafs exist.*

```
list foo {
  unique "a/b c/d/e";
  ...
  leaf b { ... }
  ...
  leaf e { ... }
}
```



but what about this:



Unique assignment of router interfaces

```
list router {
  unique "interfaces/interface/name";
  ...
}
```

This represents a common case of partitioning a set of objects into disjoint subsets, where the subsets are organized in multiple hierarchy levels.

The (ugly) alternative is:

```
list router {
  must "not(preceding-sibling::router[
        interfaces/interface/name =
        current()/interfaces/interface/name])";
  ...
}
```