Core Routing Module

`draft-lhotka-netmod-routing-cfg-00`

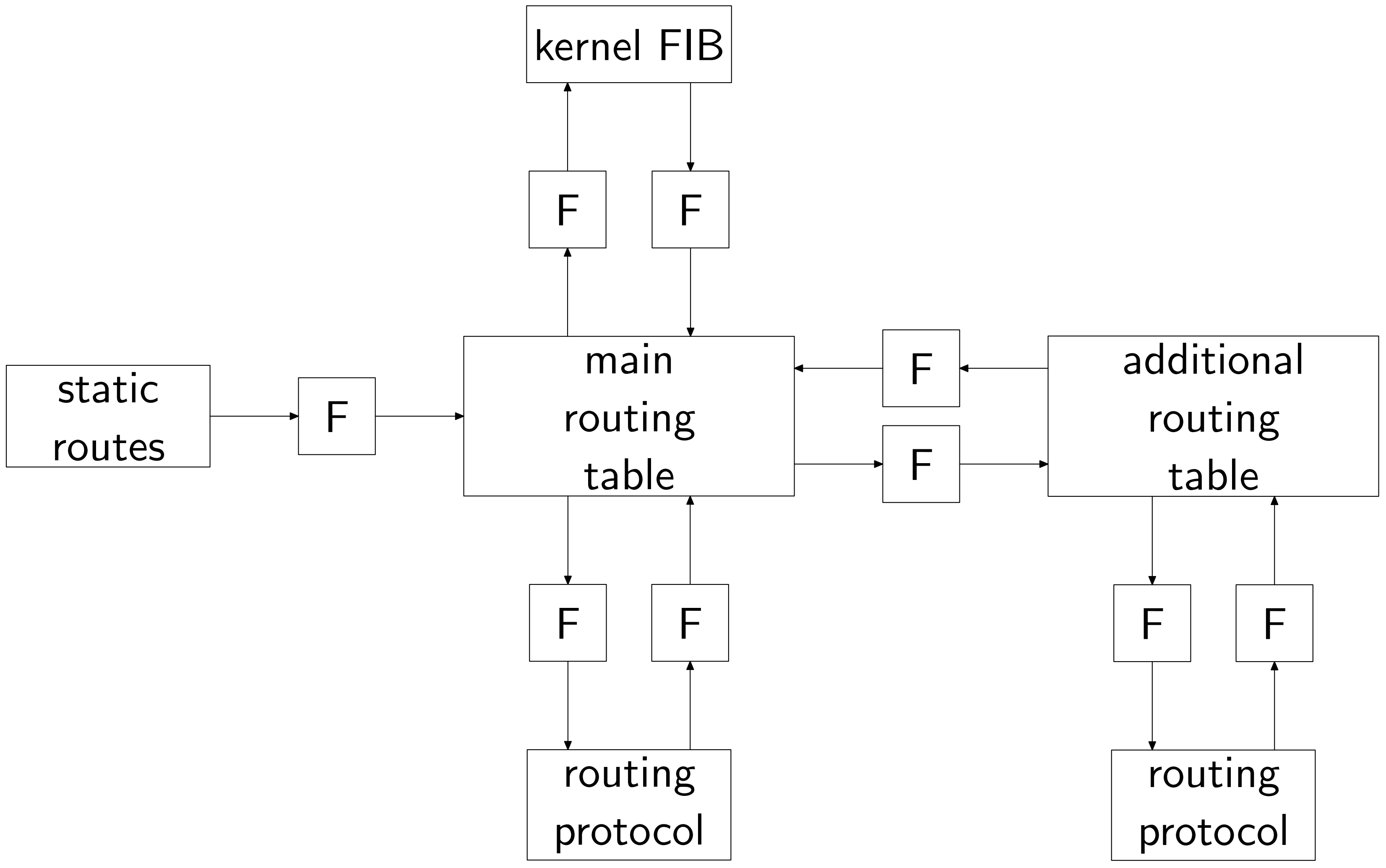
Ladislav Lhotka
⟨*lhotka@cesnet.cz*⟩

29 March 2011

Objectives

- The data model should be suitable for the common address families, in particular IPv4 and IPv6, unicast and multicast.
- Simple routing setups, such as static routing, should be configurable in a simple way, ideally without the need to write additional YANG modules.
- The framework must allow for complicated setups including multiple routing tables and multiple routing protocols, and controlled redistribution of routing information.
- Vendors should be able to map data models using this framework to their proprietary data models and configuration interfaces.

Block Diagram



Framework Components

- route
- routing table
- routing protocol instance
- route pipe
- route filter

Route

Routes are used in both configuration data (static routes) and operational state data (routing tables). Minimal content model, open to enhancements.

- `route-type` – `"unicast"` (default), `"multicast"` and `"unicast"`
- `destination-prefix`
- `next-hop`
- `outgoing-interface`

Routing Table

Operational state data, a list of routes bundled with administrative data:

- `source-protocol`,
- `last-modified` – date and time of last modification or creation.

New RPC operation, `delete-route`, is used for deleting a route from a specified routing table.

Every compliant implementation MUST automatically configure the main routing table. Additional routing tables may be configured, each identified by a unique name.

Questions:

1. Should the name of the main routing table be fixed?
2. Should the (main) kernel routing table also be mandatory?

Routing Protocol Instance

Any number of routing protocol instance may be configured, each is identified by a unique name.

Each routing protocol instance MUST be connected to a routing table, by default it is the main routing table.

Two route filters control the exchange of routes – `import-filter` (protocol $\rightarrow$ table) and `export-filter` (table $\rightarrow$ protocol).

The module defines two pseudo-protocols: `"direct"` (exactly one instance) and `"static"`.

Defining New Routing Protocols

New identity MUST be defined for each protocol type, e.g.

```
identity rip {
  base rt:routing-protocol;
  description "Identity for the RIP routing protocol.";
}
```

New route attributes MAY be defined.

The `routing-protocol-instance` container MAY be augmented with protocol-specific configuration data, e.g.

```
augment "/rt:routing/rt:routing-protocol-instances/" +
        "rt:routing-protocol-instance" {
  container rip-configuration {
    when "../rt:type='rip'";
    ...
  }
}
```

Route Pipe

A unidirectional link connecting two routing tables. Each route pipe is identified by a unique name.

The transport of routes is controlled by means of a route filter.

Route pipes are defined “out of band”:

```
<route-pipe>
  <name>...</name>
  <origin>...</origin>
  <recipient>...</recipient>
  <route-filter>...</route-filter>
</route-pipe>
```

Route Filter

Route filter may be defined to allow or deny selected routes, but also to manipulate them. Each route filter is identified by a unique name.

The module defines no route filtering language, this is expected to be done in other modules (and by another WG?). Therefore, the only two configurable policies are “allow all” and “deny all”.

Potential YANG Deficiencies

1. Lack of support for YANG data types in XPath expressions – `leafref`, `instance-identifier`, `identityref`, perhaps a few important derived types such as IP addresses.

   ```
   leaf location {
    type string;
    when "derived_from(../type, 'physical-interface')";
   }
   ```

2. Requirement on uniqueness of `leaf-list` entries – how to model AS PATH?

3. In order to create `leafref` links from module A to data nodes appearing in module B, module A has to import module B. But imports cannot be circular, so what if a link from B to A is also needed? Solution: A and B are one module (see the IPFIX data model).

   A more flexible mechanism for addressing nodes in other modules is needed, circularity has to be avoided only when importing definitions.