A use case for Schema Mount

February 22, 2016

The term *schema mount* is used to be solution neutral

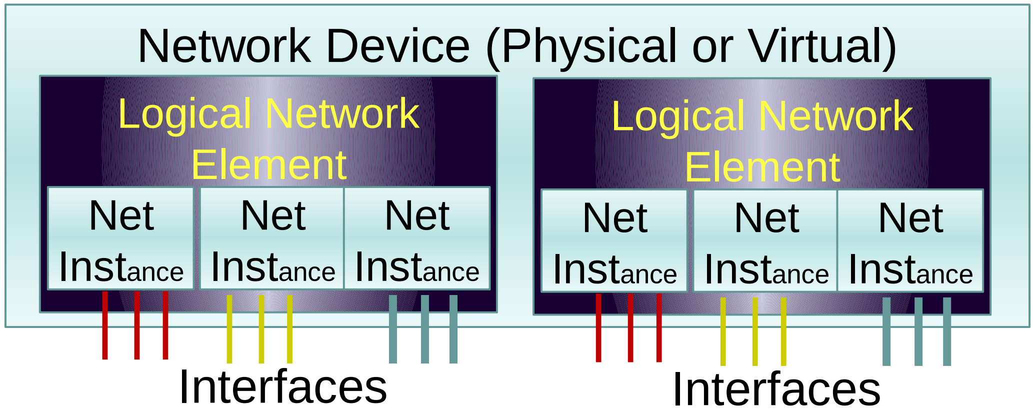
Schema Mount Use Case

- We cover a single use case
  - Not all possible use cases
- Use case driven by **draft-rtgyangdt-rtgwg-device-model-02**
  - Repo: https://github.com/ietf-rtg-area-yang-arch-dt/meta-model/
  - Authors: Acee Lindem, Christian Hopps, Dean Bogdanovic, **Lou Berger** (Ed.)
  - Contributors: Anees Shaikh, Kevin D'Souza, Luyuan Fang, Qin Wu, Rob Shakir, Stephane Litkowski, Yan Gang

Topics

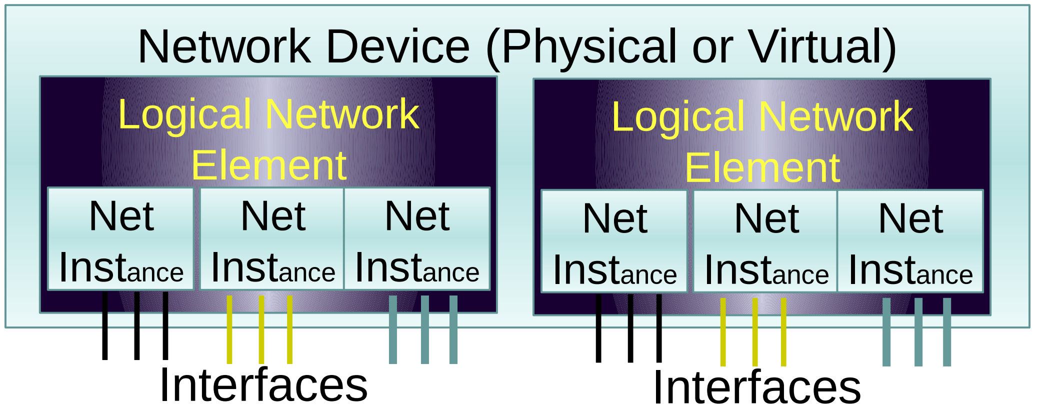
- Challenges
- Original solution
- Schema mount based solution
- Concluding observations

Challenge 1: LNEs – Logical Network Elements



- Separate management sub-domains
  - Sub-domains can be managed independently and, optionally, by a top level manager
- Conceptually
  - LNE ~= "Guest“
  - Network-device ~= "Host"

Challenge 2: NIs – Network Instances



- Separate routing / switching domains
- Can represent of an RFC 4364 VRF or a Layer 2 Virtual Switch Instance (VSI) or a bridge/router (i.e., both)
  - General virtualized instance implying a separate L2, L3, or L2/L3 context.
  - For L3, this implies a unique IPv4/IPv6 address space.

Challenge Context

Top-level is sometimes referred to as *root-level*

- There are many "top-level" modules out there
  - Some RFCs
  - Many drafts
  - Many private/proprietary/consortia
  - Some from other SDOs (e.g., from IEEE)
- None are LNE aware
- One is almost NI aware
  - draft-ietf-netmod-routing-cfg has *routing instances*
- One example: RFC7223 – A "top-level" module

```
Namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces";
+--rw interfaces
|  +--rw interface* [name]
|     +--rw name                        string
|     +--rw description?                string
|     +--rw type                        identityref
|     +--rw enabled?                    boolean
|     +--rw link-up-down-trap-enable?   enumeration
```

Original (draft -01) Approach

- An explicit structure with LNEs and NIs

```
+--rw device                           (Real or virtual)
     +--rw info
     +--rw hardware
     +--rw interfaces                   (RFC7223, RFC7277, drafts)
     +--rw qos
     +--rw logical-network-elements (logical partition)
         +--rw networking-instances (rtg-cfg draft, e.g., VRF/VSI)
```

- Pro:
  - Can support any type of device
  - No YANG modification required
- Cons:
  - Every model and device would see at least 1 LNE and NI
  - Would impact every module
    - Each module would need to pick path based on model type
      - Physical at the top
      - Per management domain, under LNE
      - Per VRF/VSI, under NI

Current (draft -02) Approach

The term *schema mount* is used to be solution neutral

- Rely on “schema” mount
  - Works for any module – ***without modification***
- Adds two tables
  - LNE: logical-network-inventory
  - NI: networking-instance
- Each table defines a per {LNE, NI} instance root
  - Under which any top-level model may be *instantiated*
    - Note this is defined in the schema
  - Choice of available model is up to the implementation
    - Some type of device profile definition is expected
  - ietf-yang-library is used to enumerate available models

Example: A Top-Level Device

```
Namespace "urn:ietf:params:xml:ns:yang:...";
   +--rw ietf-yang-library
   |
   +--rw interfaces
   +--rw hardware
   +--rw qos
   |
   +--rw system-management
   +--rw networking-services
   +--rw oam-protocols
   |
   +--rw routing
   +--rw mpls
   +--rw ieee-dot1Q
   |
   +--rw ietf-acl
   +--rw ietf-key-chain
   |
   +--rw logical-network-element
   +--rw networking-instance
```

```
module: network-device
   +--rw system-management
      +--rw system-management-global
      |  +--rw statistics-collection
      |  ...
      +--rw system-management-protocol* [type]
      |  +--rw type=syslog
      |  +--rw type=dns
      |  +--rw type=ntp
      |  +--rw type=ssh
      |  +--rw type=tacacs
      |  +--rw type=snmp
      |  +--rw type=netconf
```

```
module: network-device
   +--rw networking-services
      +--rw networking-service* [type]
         +--rw type=ntp-server
         +--rw type=dns-server
         +--rw type=dhcp-server
```

```
module: network-device
   +--rw oam-protocols
      +--rw oam-protocol* [type]
         +--rw type=bfd
         +--rw type=cfm
         +--rw type=twamp
```

```
module: network-device
   +--rw routing
      +--rw control-plane-protocols
      |  +--rw control-plane-protocol* [type]
      |     +--rw type        identityref
      |     +--rw policy
      +--rw ribs
         +--rw rib* [name]
            +--rw name            string
            +--rw description?    string
            +--rw policy
```

```
module: network-device
  +--rw mpls
     +--rw global
     +--rw lsps* [type]
       +--rw type=static
       +--rw type=constrained-paths
       +--rw type=igp-congruent
```

Example: LNE Model

```
//network-device state
    module: logical-network-element
+--rw logical-network-inventory
   +--rw logical-network-element* [name]
      +--rw name="one"                string
      +--rw manged=true               boolean
      +--rw root                      schema-mount
         //Example LNE state when exposed to network-device
         +--rw ietf-yang-library
         +--rw interfaces
         +--rw hardware
         +--rw qos
         +--rw system-management
         +--rw networking-services
         +--rw oam-protocols
         +--rw routing
         +--rw mpls
         +--rw ieee-dot1Q
         +--rw networking-instance
```

Example: LNE Model

```
module: networking-instance
   +--rw networking-instances
      +--rw networking-instance* [name]
         +--rw name                                   string
         +--rw type?                                  identityref
         +--rw enabled?                               boolean
         +--rw description?                           string
         +--rw networking-instance-policy
         |  ...
         +--rw root?                        schema-mount
         |  ...
augment /if:interfaces/if:interface:
   +--rw bind-networking-instance-name?   string
augment /if:interfaces/if:interface/ip:ipv4:
   +--rw bind-networking-instance-name?   string
augment /if:interfaces/if:interface/ip:ipv6:
   +--rw bind-networking-instance-name?   string
```

Key Requirements of This Use Case

1. That any data model can be instantiated within another module
   – Instantiated means that information is maintained only within the ‘mounted’ context
   – This use case only requires mounting of top-level models
2. That no additional model is needed to support 1
   – The schema defines what other modules can be mounted
3. That a server can control which models are mounted
4. That all capabilities that exist with the mounted module are available e.g. RPC operations, notifications, and augmentations

Observations

- We are happy with any solution that enables our use case
- Both solutions drafts address some, but not all of the use case
  - Both require additional modules
  - Both solutions look like reasonable starting points
  - Perhaps can merge them and add additional needed capability
- We need a solution direction ASAP
  - Without some form of schema mount we will need to revert to the draft -01 ridge structure