Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 18, 2016

L. Berger
LabN Consulting, L.L.C.
C. Hopps
Deutsche Telekom
A. Lindem
Cisco Systems
D. Bogdanovic
May 17, 2016

Logical Network Element Model
draft-rtgyangdt-rtgwg-lne-model-00

Abstract

This document defines a logical network element module. This module along with the network instance module can be used to manage the logical and virtual resource representations that may be present on a network device. Examples of common industry terms for logical resource representations are Logical Systems or Logical Routers. Examples of of common industry terms for virtual resource representations are Virtual Routing and Forwarding (VRF) instances and Virtual Switch Instances (VSIs).

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] module to support the creation of logical network elements on a network device. A logical network element (LNE) is an independently managed virtual device made up of resources allocated to it from the host, or parent, network device. (An LNE running on a host network device conceptually parallels a virtual machine running on a host system.) This document also defines the necessary augmentations for allocating host resources to a given LNE. As the interface management model [RFC7223] is the only a module that currently defines host resources, this document currently defines only a single augmentation to cover the assignment of interfaces to an LNE.

As each LNE is an independently managed device, each will have its own set of YANG modeled data that is independent of the host device and other LNEs. For example, multiple LNEs may all have their own "Tunnel0" interface defined which will not conflict with each other and will not exist in the host's interface model. An LNE will have it's own management interfaces possibly including independent instances of netconf/restconf/etc servers to support configuration of

their YANG models. As an example of this independence, an implementation may choose to completely rename assigned interfaces, so on the host the assigned interface might be called "Ethernet0/1" while within the LNE it might be called "eth1".

In addition to standard management interfaces, a host device implementation may support accessing LNE configuration and operational YANG models directly from the host system. When supported, such access is accomplished through a schema-mount mount point [I-D.ietf-netmod-schema-mount] under which the root level LNE YANG models may be accessed.

Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric.

This document was motivated by, and derived from, [RTG-DEVICE-MODEL].

1.1. Status of Work and Open Issues

The top open issues are:

1. This document will need to match the evolution and standardization of [I-D.openconfig-netmod-opstate] or [I-D.ietf-netmod-opstate-reqs] by the Netmod WG.

It will also make use of emerging YANG functionality supported by YANG Schema Mount This document is expected to use whatever Schema Mount solution is agreed upon by the Netmod Working Group.

2. Overview

In this document, we consider network devices that support protocols and functions defined within the IETF Routing Area, e.g, routers, firewalls and hosts. Such devices may be physical or virtual, e.g., a classic router with custom hardware or one residing within a server-based virtual machine implementing a virtual network function (VNF). Each device may sub-divide their resources into logical network elements (LNEs) each of which provides a managed logical device. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric. Each LNE may also support virtual routing and forwarding (VRF) and virtual switching instance (VSI) functions, which are referred to below as a network instances (NIs). This breakdown is represented in Figure 1.

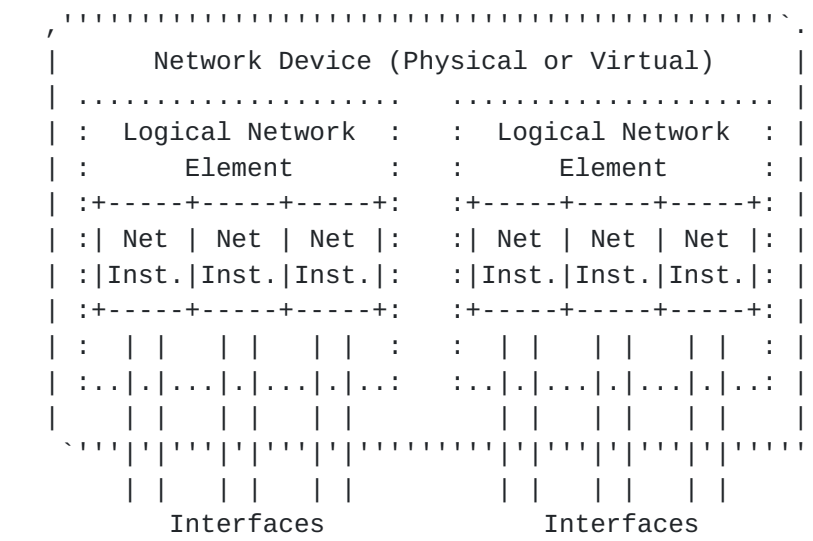

Figure 1: Module Element Relationships

A model for LNEs is described in Section 3 and the model for network instances is covered in [NI-MODEL]. For more information on how these models may be used within an overall device model structure, see [RTG-DEVICE-MODEL].

The interface management model [RFC7223] is and existing model that is impacted by the definition of LNEs and network instances. This document and [NI-MODEL] define augmentations to the interface module to support LNEs and NIs. Similar elements, although perhaps only for LNEs, may also need to be included as part of the definition of the future hardware and QoS modules.

Interfaces are a crucial part of any network device's configuration and operational state. They generally include a combination of raw physical interfaces, link-layer interfaces, addressing configuration, and logical interfaces that may not be tied to any physical interface. Several system services, and layer 2 and layer 3 protocols may also associate configuration or operational state data with different types of interfaces (these relationships are not shown for simplicity). The interface management model is defined by [RFC7223].

The logical-network-element and network-instance modules augment the existing interface management model in two ways: The first, by the logical-network-element module, adds an identifier which is used on physical interface types to identify an associated LNE. The second, by the network-instance module, adds a name which is used on interface or sub-interface types to identify an associated network instance. Similarly, this name is also added for IPv4 and IPv6 types, as defined in [RFC7277].

The interface related augmentations are as follows:

```
    module: ietf-logical-network-element
    augment /if:interfaces/if:interface:
       +--rw bind-lne-name?   string

    augment /if:interfaces/if:interface:
       +--rw bind-network-instance-name?   string
    augment /if:interfaces/if:interface/ip:ipv4:
       +--rw bind-network-instance-name?   string
    augment /if:interfaces/if:interface/ip:ipv6:
       +--rw bind-network-instance-name?   string
```

The following is an example of envisioned combined usage. The interfaces container includes a number of commonly used components as examples:

```
          +--rw interfaces
          |  +--rw interface* [name]
          |     +--rw name                          string
          |     +--rw lne:bind-lne-name?            string
          |     +--rw ethernet
          |     |  +--rw ni:bind-network-instance-name? string
          |     |  +--rw aggregates
          |     |  +--rw rstp
          |     |  +--rw lldp
          |     |  +--rw ptp
          |     +--rw vlans
          |     +--rw tunnels
          |     +--rw ipv4
          |     |  +--rw ni:bind-network-instance-name? string
          |     |  +--rw arp
          |     |  +--rw icmp
          |     |  +--rw vrrp
          |     |  +--rw dhcp-client
          |     +--rw ipv6
          |        +--rw ni:bind-network-instance-name? string
          |        +--rw vrrp
          |        +--rw icmpv6
          |        +--rw nd
          |        +--rw dhcpv6-client
```

The [RFC7223] defined interface model is structured to include all interfaces in a flat list, without regard to logical or virtual instances (e.g., VRFs) supported on the device. The bind-lne-name and bind-network-instance-name leaves provide the association between an interface and its associated LNE and NI (e.g., VRF or VSI).

3. Logical Network Elements

A logical network element is a network-device which is contained within another network-device. Using host-virtualization terminology one could refer to an LNE as a "Guest", and the containing network-device as the "Host". While LNEs may be implemented via host-virtualization technologies this is not a requirement.

Logical network elements represent the capability of some devices to partition resources into independent logical routers and/or switches. Device support for multiple logical network elements is implementation specific. Systems without such capabilities need not include support for the logical-network-element module. In physical devices, some hardware features are shared across partitions, but control plane (e.g., routing) protocol instances, tables, and configuration are managed separately. For example, in virtual routers or VNFs, this may correspond to establishing multiple logical instances using a single software installation. The model supports configuration of multiple instances on a single device by creating a list of logical network elements, each with their own configuration and operational state related to routing and switching protocols, as shown below:

```
module: ietf-logical-network-element
   +--rw logical-network-inventory
      +--rw logical-network-element* [name]
         +--rw name?   string
         +--rw description? string
         +--rw managed?     boolean
         +--rw root?        schema-mount
augment /if:interfaces/if:interface:
   +--rw bind-lne-name?     string
```

`name` identifies the logical network element. `managed` indicates if the host network device is able to manage the LNE via the `root` structure.

3.1. LNE Management - Host Network Device View

There are multiple implementation approaches possible to enable a network device to support the logical-network-element module and multiple LNEs. Some approaches will allow the management functions operating at network device level to access LNE configuration and operation information, while others will not. Similarly, even when LNE management from the network device is supported by the implementation, it may be prohibited by user policy.

The `managed` boolean mentioned above is used to indicate when LNE management from the network device context is possible. When the `managed` boolean is `false`, the LNE cannot be managed by the host system and can only be managed from within the context of the LNE as described in the next section, Section 3.2.

When the `managed` boolean is `true`, the LNE can be managed from both the context of the LNE and the host network device. In this case, the same information that is available from within the LNE context is made available via the `root` element, with paths modified as described in [I-D.ietf-netmod-schema-mount].

As an example, consider the case where an LNE with a `name` of "one" is defined on a network device. In this case the following structure might be made available:

```
.....................................................................
                                                (network-device state)

+--rw yanglib:modules-state        [I-D.ietf-netconf-yang-library]
+--rw lne:logical-network-elements [I-D.rtgyangdt-rtgwg-lne-model]
    +--rw logical-network-element* [name]
        +--rw name="one"           string
        +--rw manged=true          boolean
        +--rw root                 schema-mount
           |
.....................................................................
           |                        (exposed LNE state if managed=true)
           |
           +--rw yanglib:modules-state  [I-D.ietf-netconf-yang-library]
           +--rw if:intefaces           [RFC7223]
           +--rw hardware
           +--rw qos
           +--rw system-management
           +--rw network-services
           +--rw oam-protocols
           +--rw rt:routing             [I-D.ietf-netmod-routing-cfg]
           +--rw mpls
           +--rw ieee-dot1Q
           +--rw ni:network-instances   [I-D.rtgyangdt-rtgwg-ni-model]
```

As an LNE is a network device itself, all modules that may be present at the top level network device may also be present for the LNE, be made available under `root`, and be accessible via paths modified per [I-D.ietf-netmod-schema-mount]. The list of available modules is expected to be implementation dependent. As is the method used by an implementation to support LNEs.

Resources assigned to the LNE will be represented in that LNE's resource modules. e.g., an LNE's interfaces module will contain the interfaces assigned to that LNE from the containing network-device.

3.2. LNE Management - LNE View

Management functions operating with the context of an LNE are accessed through standard LNE's management interfaces, e.g., NETCONF and SNMP. When accessing an LNE via an LNE's management interface, a network-device representation will be presented, but its scope will be limited to the specific LNE. Normal YANG/NETCONF mechanisms, together with yang library [I-D.ietf-netconf-yang-library], can be used to identify the available modules. Each supported module will be presented as a top level module. Only LNE associated resources will be reflected in resource related modules, e.g., interfaces, hardware and perhaps QoS. From the management perspective, there will be no difference between the available LNE view (information) and an a physical network device.

Multiple implementation approaches are possible to provide LNE views, and these are outside the scope of this document.

3.3. LNE Instantiation

TBD -- need to resolve if instantiation is based on new list entry creation per the pending Schema Mount solution definition.

4. Security Considerations

LNE portion is TBD

NI portion is TBD

5. IANA Considerations

This YANG model currently uses a temporary ad-hoc namespace. If it is placed or redirected for the standards track, an appropriate namespace URI will be registered in the "IETF XML Registry" [RFC3688]. The YANG structure modules will be registered in the "YANG Module Names" registry [RFC6020].

6. Logical Network Element Model

The structure of the model defined in this document is described by the YANG module below.

```
<CODE BEGINS> file "ietf-logical-network-element@2016-05-01.yang"
module ietf-logical-network-element {
```

```
   yang-version "1";

   // namespace
   namespace "urn:ietf:params:xml:ns:yang:ietf-logical-network-element";

   prefix "lne";

   // import some basic types
   import ietf-interfaces {
     prefix if;
   }

   // meta
   organization "IETF RTG YANG Design Team Collaboration
                 with OpenConfig";

   contact
       "Routing Area YANG Architecture Design Team -
        <rtg-dt-yang-arch@ietf.org>";

   description
     "This module is used to support multiple logical network
      elements on a single physical or virtual system.";

   revision "2016-05-01" {
     description
       "IETF Routing YANG Design Team Meta-Model";
     reference "TBD";
   }

   // feature statements
   feature bind-lne-name {
     description
       "Logical network element to which an interface is bound";
   }

   // top level device definition statements
   container logical-network-elements {
     description "Allows a network device to support multiple logical
                  network element (device) instances";
     list logical-network-element {
       key name;
       description "List of logical network elements";
       leaf name {
         type string;
         description "Device-wide unique identifier for the
                      logical network element";
       }
```

```
      leaf managed {
        type boolean;
        description
          "True if the host can manage the LNE using the root mount
           point";
      }
      leaf description {
        type string;
        description
          "Description of the logical network element";
      }
      leaf root {
        type schema-mount;
        description "Root for models supported per logical
                     network element";
      }
    }
  }

  // augment statements
  augment "/if:interfaces/if:interface" {
    description
        "Add a node for the identification of the logical network
        element associated with an interface. Applies to interfaces
        that can be assigned on a per logical network element basis.
        A <TBD> error is returned when the interface type cannot be
        assigned.";

    leaf bind-lne-name {
      type string;
      description
        "Logical network element ID to which interface is bound";
    }
  }

  // rpc statements

  // notification statements

}
<CODE ENDS>
```

7. References

7.1. Normative References

[I-D.ietf-netmod-schema-mount]
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-01 (work in progress), April 2016.

[NI-MODEL]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "Network Instance Model", draft-rtgyangdt-rtgwg-ni-model-00.txt (work in progress), May 2016.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

[RTG-DEVICE-MODEL]
Lindem, A., Ed., Berger, L., Ed., Bogdanovic, D., and C. Hopps, "Network Device YANG Organizational Models", draft-rtgyangdt-rtgwg-device-model-04.txt (work in progress), May 2016.

7.2. Informative References

[I-D.ietf-netconf-yang-library]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", draft-ietf-netconf-yang-library-05 (work in progress), April 2016.

[I-D.ietf-netmod-opstate-reqs]
Watsen, K. and T. Nadeau, "Terminology and Requirements for Enhanced Handling of Operational State", draft-ietf-netmod-opstate-reqs-03 (work in progress), January 2016.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

Appendix A. Acknowledgments

The Routing Area Yang Architecture design team members included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Lou Berger, Qin Wu, Rob Shakir, Stephane Litkowski, and Yan Gang.

The RFC text was produced using Marshall Rose's xml2rfc tool.

Appendix B. Contributors

Contributors' Addresses

TBD

Authors' Addresses

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Dean Bogdanovic

Email: ivandean@gmail.com