i2rs
Internet-Draft
Intended status: Informational
Expires: September 12, 2017

Y. Zhuang, Ed.
D. Shi
Huawei
R. Gu
China Mobile

March 11, 2017

YANG Data Model for Fabric Service delivery in Data Center Network
draft-zhuang-i2rs-dc-fabric-service-model-02

Abstract

This document defines a YANG data model that can be used to deliver fabric service for users within a data center network. This model is intended to be instantiated by management system. It provides an abstraction of services for a fabric network to be used by users. However it is not a configuration model used directly onto network infrastructures. It should be used combining with such as fabric topology data model defined in [I-D.zhuang-i2rs-yang-dc-fabric-network-topology] with specific fabric topology information to generate required configuration onto the related network elements to deliver the service.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2017.

Table of Contents

1. Introduction

Network service provisioning is currently coupled with specific network topology and technology applied, which is technology and device oriented.

In the area of data center, this approach makes the management complex due to massive network devices involved and various applications deployed by multiple users (also known as tenants).

In the traditional way, the administrator has to be aware of the entire data center network before delivering services for users. When service request comes up, administrator has to divide the request into appropriate configurations and operations for all involved devices manually. Finally, these configurations are deployed onto network infrastructure, which requires personnel skills.

Actually different users share the same network infrastructure. A more dynamical way to deploy and manage the network is eager to be found out. Here we decompose the network management system into several layers in order to have network service provision more flexible and automatic. Each network layer is dedicated to be managed. What is more, all the layers can be combined to fulfill the delivery of the user's service.

We can use three layers in data center network. The bottom one is physical infrastructure with massive devices. The middle one is fabric topology defined in [I-D.zhuang-i2rs-yang-dc-fabric-network-topology]. Unlike the physical layer, the fabric layer is used to display a fabric network including features and boundaries. In the fabric layer, a set of fabrics can exist with each managed independently. Furthermore, a bottom-up abstraction of fabric service is proposed to provide application centric interfaces facing to users which define network services regardless of beneath fabric topology and physical connections in the up layer.

This document defines a YANG [RFC6020] [RFC7950] data model focusing on the fabric service interfaces to define user fabric network services regardless of specific beneath network topology and devices. This model defines the generic configuration for fabric services within DC networks.

For example, this model can be used by the network orchestrator in which the fabric service interfaces are exposed. When a service from user application is requested, orchestrator adopts this model including service information and processes it into the topology layer through a DC controller. Thus a service is automatically and dynamically provided.

The service data model includes three main modules:

(a)Module "ietf-fabric-service" defines a module for user network service over fabric networks from the application centric view. To do so, it augments general network topology model defined in [I-D.ietf-i2rs-yang-network-topo] with logical components such as logical switches, logical routers as well as logical ports to carry network services requested by user applications.

(b)Module "ietf-fabric-endpoint" defines a module for hosts that run applications and generate traffics. The major usage of this module is to indicate the attachment points of a host in a user service network as well as in a physical network when it is initialed, so as to build bindings between physical layer and topology layer dynamically.

(c) Module "ietf-fabric-capable-device" defines a module for configuration and operation onto network devices generated based on information in fabric topology and fabric service modules.

Besides, the model "ietf-fabric-topology" defined in [I-D.zhuang-i2rs-yang-dc-fabric-network-topology] with topology and resource as well as technology information is used to work together to implement configurations and operations of the fabric service onto the specific fabric infrastructure.

2. Concept and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.1. Terminology

Fabric topology: a data center network can be decomposed to a set of fabric networks, while each of these fabrics composes a set of physical nodes/links of the physical infrastructure to form a fabric network. The fabric topology includes attributes of fabrics, such as gateway mode, involved nodes, roles of involved nodes etc al.

Fabric Service: it is used as a service interface of fabric networks to users, which uses logical elements to represent network connections between hosts for applications, regardless of a specific fabric topology deployment. Each service instance is based on a fabric topology, while a fabric can provide multiple service instances for different users, each of which is isolated to others.

Endpoint: an endpoint represents a host, which can be a virtual machine on a server or a bare-metal server.

Fabric capable device: a physical device (e.g. a switch) that supports fabric service and fabric topology models.

3. Fabric service framework overview

This draft provides a network service interface on top of fabrics network layer. Users can use these network service interfaces to deploy their applications over a data center network automatically and dynamically.

From the application centric point of view, user hosts can be considered to connect with other hosts through a switch if they are L2 reachable, alternatively, connect through a router if they are L3 reachable simply. So a user network can be abstracted into a logical

network where L2 reachable represents logical switches connecting hosts and L3 reachable represents logical routers connecting switches.

With this concept, a user can use appropriate logical elements to define their networks and configure attributes of these elements such as vlan id, gateway etc al. All of these form a network service. For example, a fabric service diagram for a user is shown as below.

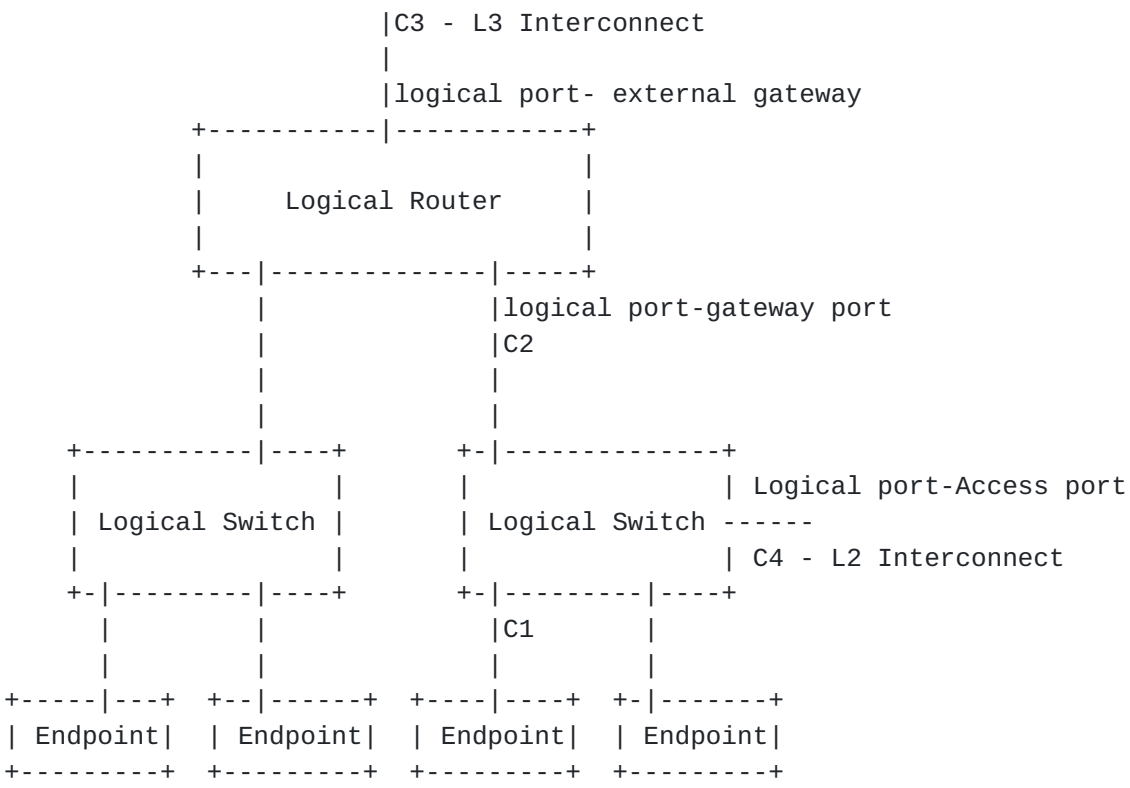


Figure 1: Diagram of a fabric service

In the diagram, abstraction of network connections is focused as a very initial effort to abstract services for fabric-based DC networks. Based on the connection, we can add other network appliance for which the fabric service should be extended.

3.1. Service element

There are four major components regarding as service elements within a fabric service as depicted in Figure 1.

Logical Switch:

Works as a switch within a logical fabric network to provide L2 connections between hosts or to a logical router or to external networks. It can be bounded to one or several physical switches.

Logical Router:

Works as a router to provide L3 connections between logical switches or to external networks.

Logical Port:

Provides port function on logical switches and logical routers which claims their connections to others.

Endpoint:

Represents user hosts which can be a VM or a bare-metal server.

3.2. Functionality of connections

There are 4 connections between elements within the fabric service framework listed as follows:

C1: Endpoint attachment. It is used by an endpoint to connect to a logical switch.

C2: L2 to L3 attachment. Interface between a logical switch and a logical router within the same fabric.

C3: L3 interconnection which connects to a logical router.

C4: L2 interconnection which connects to a logical switch in another fabric.

Thinking of the functionality of different connections, a logical port can act as an access port (which provides C1/C4/C3 connection to a network element), or a service port (which provide C2 gateway connection) as shown in Figure 2.

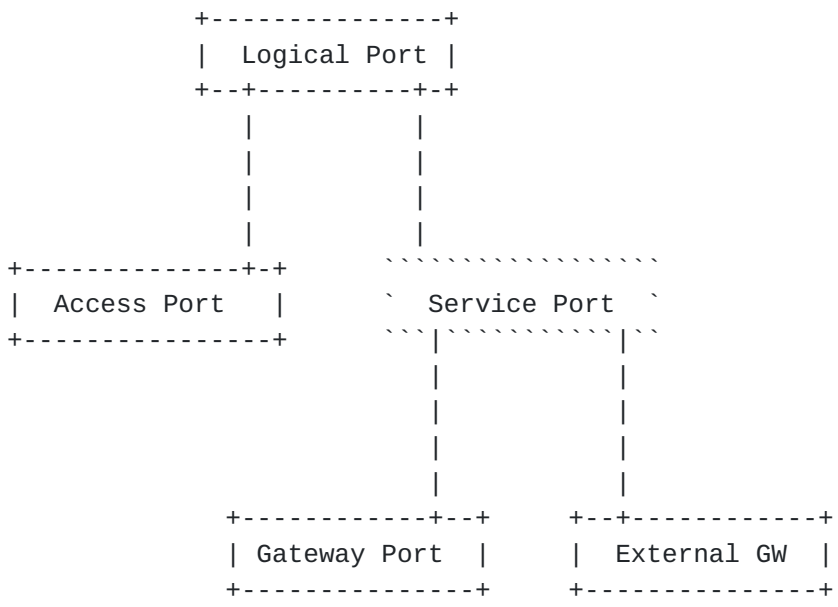


Figure 2: Types of Logical port

When a logical port is noticed as an access port, there will be a corresponding physical port. In this situation, the required access configuration can be deployed on this physical port directly. However, there will be a gateway service if a logical port is noticed as a service port. In this situation, the management system should combine the gateway function and fabric territory at fabric topology layer together with the gateway configuration on the service port. By the combination, it is easy to figure out the appropriate devices in the physical infrastructure and their configurations for these devices respectively.

4. Fabric service model usage

4.1. Usage architecture

In section 3, a fabric service interface is provided for users to define their networks in a more concentrated and intuitive way. To be detailed, when a fabric service comes, the topology manager will parse services into configuration/operations of specific network devices automatically. In this process, service interface information and topology information from fabric topology defined in [I-D.zhuang-i2rs-yang-dc-fabric-network-topology] is needed.

The whole process is shown in Fig.3. Fabric service module is used define network services for applications maybe by an orchestration for example, according to the topology architecture stated in [I-D.draft-ietf-i2rs-usecase-reqs-summary]. The topology information maintenance should be done by a topology manager. By combining

information from different layers, a topology manager automatically generates configurations and operations of related devices and deploys them respectively over the physical fabric infrastructures. Here, the fabric-capable-device module can be used to configure involved devices.

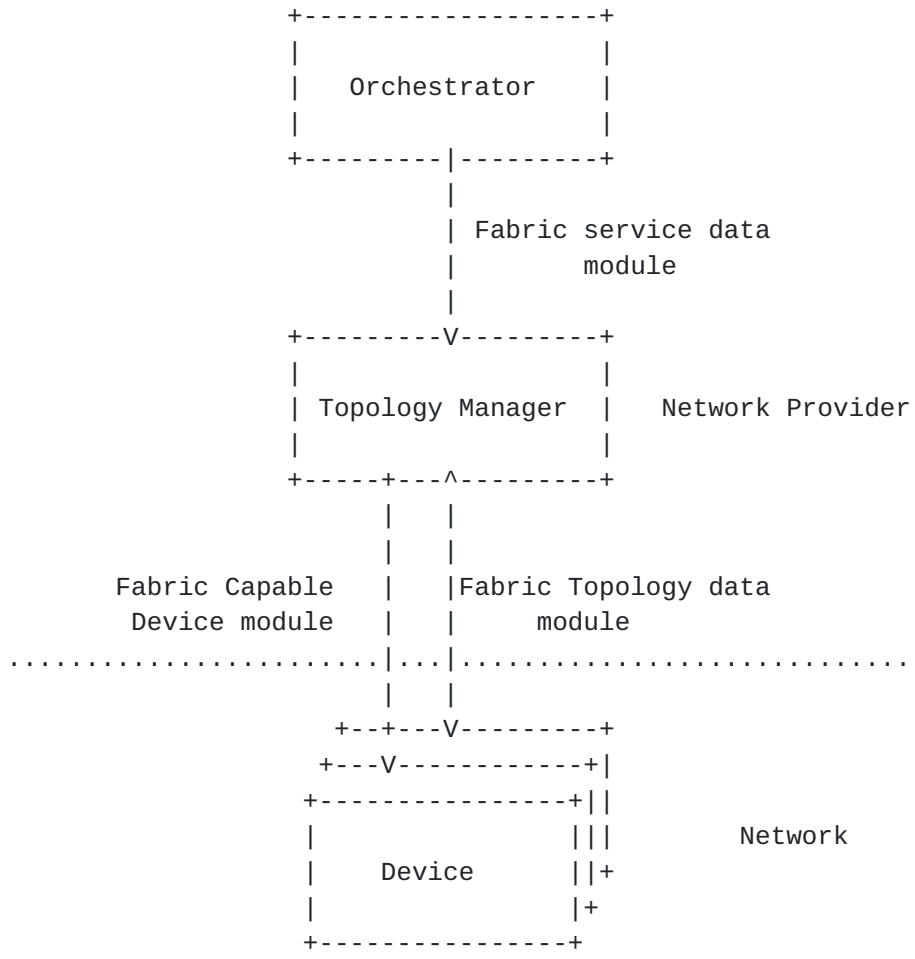


Figure 3: Fabric service Usage architecture

4.2. Multi-Layer interconnection

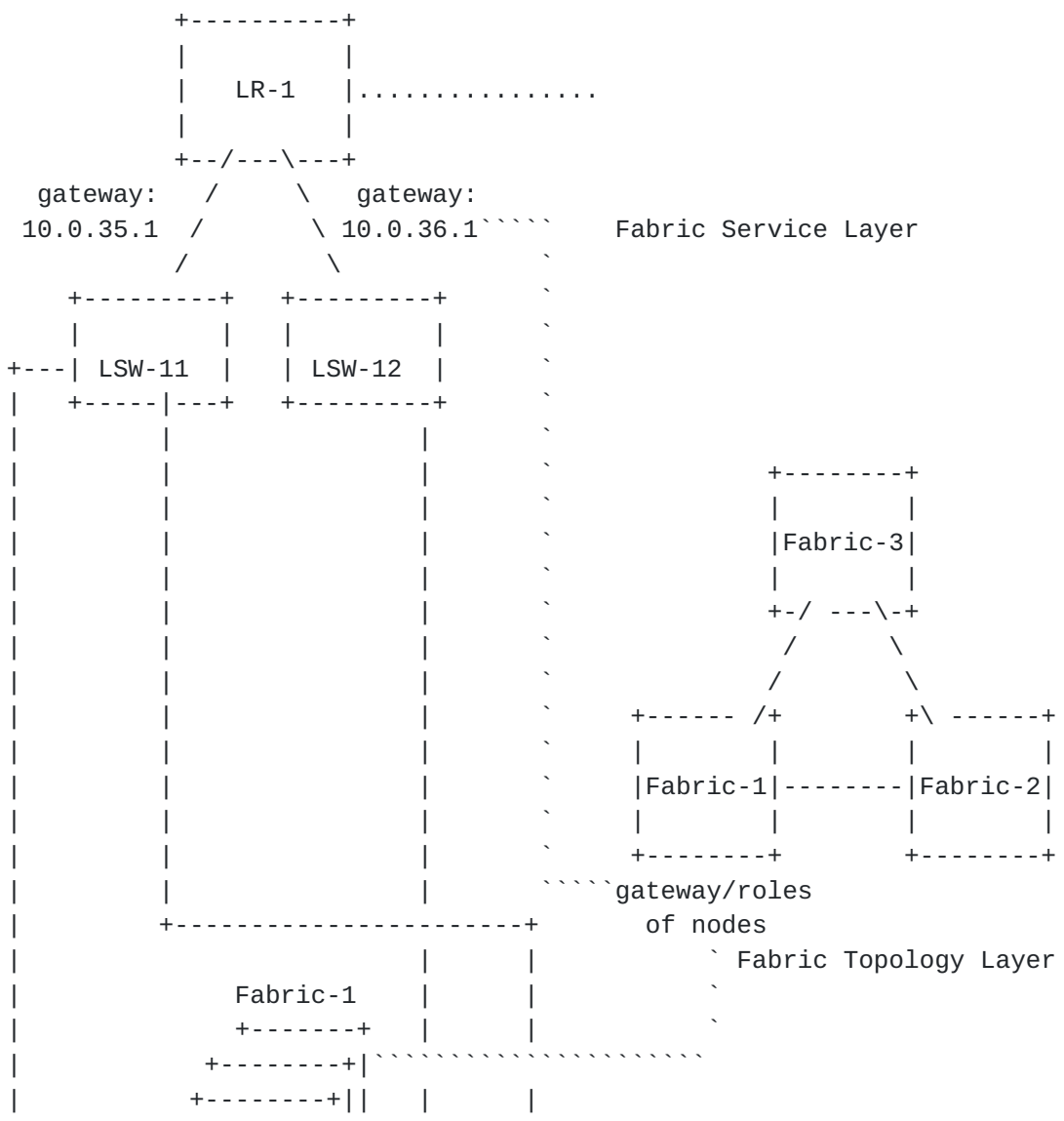
There are three layers in this usage.

At the service layer, a fabric service model is abstracted from fabric network used as an application-centric interface to define user networks. It focuses on the connection services from users' perspective. Using the fabric service interface, an administrator can define a logical network for each user over a single fabric network while each logical networks can be managed separately.

For the fabric topology layer, it collects and maintains the fabric topology information (including territory of the physical fabric, connections, gateway functions, roles of devices within the fabric and specific technologies for each fabric) upon the physical network layer.

With information provided by both fabric service as well as fabric topology, a fabric topology manager will calculates and generates configuration and operation for involved network devices in the physical layer so as to distribute and deploy them onto network infrastructure.

The diagram of the management architecture and its relationship is depicted as below.



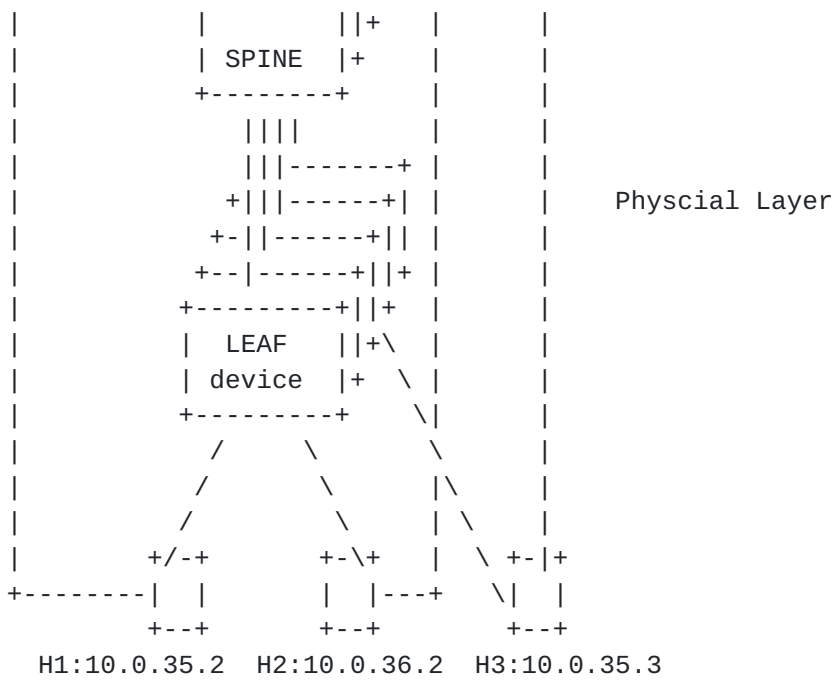


Figure 4: Multi-layer interconnection

The mapping of nodes with access logical ports is realized by endpoints e.g. H1,H2 and H3 in Fig.4. An endpoint is instantiated by the orchestrator to indicate the locations of a host both in the logical layer as well as in the physical layer, so as to deliver services requested from the logical port onto the physical port in a dynamic manner. For H1 and H3, they are considered to connect to the same switch for user in the logical layer, even they attach to the different devices.

Besides, gateway configuration is defined at service layer while the gateway mode and gateway devices (for distributed gateway, the gateway should be deployed on LEAF devices, while for centralized gateway, the configuration should be on SPINE) are defined in fabric topology layer. By combing the gateway information from both layers, the system can automatically figure out the involved devices and generate appropriate configurations onto them.

5. Design of the data model

5.1. Fabric service module

As explained previously, network service for user network can be abstracted to sets of logical switches, logical routers and logical ports. Upon these logical elements, acl policies and gateway functions can be attached.

The fabric service module is defined by YANG module "ietf-fabric-service". The module is depicted in the following diagram.

```
 module: ietf-fabric-service
  augment /nw:networks/nw:network/nw:node:
    +--ro lsw-attribute
       +--ro lsw-uuid?     yang:uuid
       +--ro name?         string
       +--ro segment-id?   uint32
       +--ro network?      inet:ip-prefix
       +--ro external?     boolean
       +--ro fabric-acl* [fabric-acl-name]
          +--ro fabric-acl-name    string
  augment /nw:networks/nw:network/nw:node:
    +--ro lr-attribute
       +--ro lr-uuid?      yang:uuid
       +--ro name?         string
       +--ro vrf-ctx?      uint32
       +--ro fabric-acl* [fabric-acl-name]
       |  +--ro fabric-acl-name    string
       +--ro routes
          +--ro route* [destination-prefix]
             +--ro description?          string
             +--ro destination-prefix    inet:ipv4-prefix
             +--ro (next-hop-options)?
                +--:(simple-next-hop)
                   +--ro next-hop?             inet:ipv4-address
                   +--ro outgoing-interface?   nt:tp-id
  augment /nw:networks/nw:network/nw:node/nt:termination-point:
    +--ro lport-attribute
       +--ro lport-uuid?         yang:uuid
       +--ro name?               string
       +--ro port-layer
       |  +--ro layer-1-info
       |  |  +--ro location?   nt:tp-id
       |  +--ro layer-2-info
       |  |  +--ro access-type?      access-type
       |  |  +--ro access-segment?   uint32
       |  +--ro layer-3-info
       |     +--ro ip?               inet:ip-address
       |     +--ro network?          inet:ip-prefix
       |     +--ro mac?              yang:mac-address
       |     +--ro forward-enable?   boolean
       |     +--ro logical-switch?   nw:node-id
       +--ro fabric-acl* [fabric-acl-name]
       |  +--ro fabric-acl-name    string
       +--ro port-function
       |  +--ro (function-type)?
```

```
|     +--:(ip-mapping)
|        +--ro ip-mapping-entry* [external-ip]
|           +--ro external-ip    inet:ipv4-address
|           +--ro internal-ip?   inet:ipv4-address
+--ro underlayer-ports* [port-ref]
   +--ro port-ref    instance-identifier
```

Figure 5: Fabric Service Module

To provide a logical network topology for DC fabric network, the module augments the original ietf-network and ietf-network-topology modules:

- New nodes for logical switch and logical router with additional data objects are introduced by augmenting the "node" list of the network module.

- Termination points for logical ports are augmented with logical port information and its reference to termination ports in the underlay topologies. As stated in section 3, the logical port may act as an access port which will be bounded to some physical port, or else it may be as a service point which connects to internal gateway or external gateway. Besides, it can also be attached with ACL rules.

5.2. Endpoint module

To represent user attachments points and map logical fabric configurations and operations of applications onto the physical fabric infrastructure, an endpoint is instantiated to represent a host of a user that runs applications.

The fabric endpoint module is defined by YANG module "ietf-fabric-endpoint". The module is depicted as follows:

```
module: ietf-fabric-endpoint
   +--ro endpoints
      +--ro endpoint* [endpoint-uuid]
         +--ro endpoint-uuid       yang:uuid
         +--ro own-fabric?         fabric:fabric-id
         +--ro mac-address?        yang:mac-address
         +--ro ip-address?         inet:ip-address
         +--ro gateway?            inet:ip-address
         +--ro public-ip?          inet:ip-address
         +--ro location
         |  +--ro node-ref?         fabrictype:node-ref
         |  +--ro tp-ref?           fabrictype:tp-ref
         |  +--ro access-type?      fabrictype:access-type
         |  +--ro access-segment?   uint32
         +--ro logical-location
            +--ro node-id?   nw:node-id
            +--ro tp-id?     nt:tp-id
```

Figure 6: Fabric endpoint module

By indicating locations of an endpoint in "location" container, the logical network elements such as logical nodes and logical termination points are bounded to the network elements in a specific fabric. Then the network configurations and operations from the logical network together with its belonged fabric topology information will further be distributed onto the bounding/related physical elements by the network topology manager.

Besides, the module defines three rpc commands to register, unregister and locate the endpoint onto both logical network and physical network shown as follows.

```
 rpcs:
    +---x register-endpoint
    |  +---w input
    |  |  +---w fabric-id?          fabric:fabric-id
    |  |  +---w endpoint-uuid?      yang:uuid
    |  |  +---w own-fabric?         fabric:fabric-id
    |  |  +---w mac-address?        yang:mac-address
    |  |  +---w ip-address?         inet:ip-address
    |  |  +---w gateway?            inet:ip-address
    |  |  +---w public-ip?          inet:ip-address
    |  |  +---w location
    |  |  |  +---w node-ref?         fabrictype:node-ref
    |  |  |  +---w tp-ref?           fabrictype:tp-ref
    |  |  |  +---w access-type?      fabrictype:access-type
    |  |  |  +---w access-segment?   uint32
    |  |  +---w logical-location
    |  |     +---w node-id?   nw:node-id
    |  |     +---w tp-id?     nt:tp-id
    |  +--ro output
    |     +--ro endpoint-id?   yang:uuid
    +---x unregister-endpoint
    |  +---w input
    |     +---w fabric-id?   fabric:fabric-id
    |     +---w ids*         yang:uuid
    +---x locate-endpoint
       +---w input
          +---w fabric-id?     fabric:fabric-id
          +---w endpoint-id?   yang:uuid
          +---w location
             +---w node-ref?         fabrictype:node-ref
             +---w tp-ref?           fabrictype:tp-ref
             +---w access-type?      fabrictype:access-type
             +---w access-segment?   uint32
```

Figure 7: Fabric endpoint module RPC

5.3. Fabric capable device module

The fabric-capable-device module is to configure individual network elements i.e. devices of physical network based on belonged fabric topology and fabric services expressed by users. This module can be used by a topology manager.

The structure of "ietf-fabric-capable-device" data model is depicted in the following diagram.

```
    module: ietf-fabric-capable-device
      augment /nw:networks/nw:network/nw:node:
         +--rw supported-fabric*       identityref
         +--rw capability-supported*   fabrictype:service-capabilities
         +--ro attributes
         |  +--ro fabric-id?    nw:node-id
         |  +--ro fabric-ref?   fabrictype:node-ref
         +--rw config
            +--rw bridge-domain* [id]
            |  +--rw id         string
            |  +--rw flood?     boolean
            |  +--rw arp?       boolean
            |  +--rw segment?   uint32
            +--rw bd-port* [bd-port-id]
            |  +--rw bd-port-id     string
            |  +--rw bdid?          string
            |  +--rw ref-tp-id?     nw:node-id
            |  +--rw access-type?   fabrictype:access-type
            |  +--rw access-tag?    uint32
            |  +--rw fabric-acl* [fabric-acl-name]
            |     +--rw fabric-acl-name    string
            +--rw vrf* [id]
            |  +--rw id         string
            |  +--rw name?      string
            |  +--rw vrf-ctx?   int32
            +--rw bdif* [id]
               +--rw id               string
               +--rw bdid?            string
               +--rw vrf?             int32
               +--rw ip-address?      inet:ip-address
               +--rw mask?            int32
               +--rw mac-address?     yang:mac-address
               +--rw fabric-acl* [fabric-acl-name]
               |  +--rw fabric-acl-name    string
               +--rw port-function
                  +--rw (function-type)?
                     +--:(ip-mapping)
                        +--rw ip-mapping-entry* [external-ip]
                           +--rw external-ip    inet:ipv4-address
                           +--rw internal-ip?   inet:ipv4-address
      augment /nw:networks/nw:network/nw:node/nt:termination-point:
         +--rw port-role?   fabrictype:fabric-port-role
         +--rw port-ref?    fabrictype:tp-ref
      augment /nw:networks/nw:network/nw:node/nt:termination-point:
         +--rw bd-ids*   instance-identifier
```

Figure 8: Fabric capable device module

The ietf-fabric-capable-device augments the generic ietf-network and ietf-network-topology modules with specific attributes for fabric services as follows:

- Additional data objects for nodes are introduced by augmenting the "node" list of the network module to carry specific fabric features for a fabric capable device. New objects include type of supported fabric such as VXLAN fabric or VLAN fabric, supported capability such as whether the device supports Network Address Transformation or Service Function Chain functions. Also, the "host" fabric is indicated to show which fabric the device belonged to.

- Besides, new objects for configuring the device are contained in the "config" container, which includes the L2 attributes and L3 attributes of the bridge.

- Termination points are also augmented with attributes to indicate the role of the device port in a fabric and its role in the bridge network.

6. Fabric Service YANG Modules

```
<CODE BEGINS> file "ietf-fabric-types@2016-10-13.yang"
module ietf-fabric-types {

    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-fabric-types";
    prefix fabrictypes;

    import ietf-inet-types { prefix "inet"; revision-date "2013-07-15"; }

    import ietf-network-topology { prefix nt;  }
        import ietf-network { prefix nw;  }

        import ietf-yang-types { prefix "yang"; revision-date "2013-07-15";}

        organization
    "IETF I2RS (Interface to the Routing System) Working Group";

        contact
    "WG Web:    <http://tools.ietf.org/wg/i2rs/ >
     WG List:   <mailto:i2rs@ietf.org>

     WG Chair:  Susan Hares
                <mailto:shares@ndzh.com>
```

```
     WG Chair:  Russ White
                <mailto:russ@riw.us>

     Editor:    Yan Zhuang
                <mailto:zhuangyan.zhuang@huawei.com>

     Editor:    Danian Shi
                <mailto:shidanian@huawei.com>";

    description
        "This module contains a collection of YANG definitions for Fabric.";


    revision "2016-10-13" {
        description
            "Initial revision of faas.";
        reference       "draft-zhuang-i2rs-yang-dc-fabric-network-topology-02
and draft-zhuang-i2rs-dc-fabric-service-model-00";
    }

    identity fabric-type {
                description
                        "base type for fabric networks";
    }

    identity vxlan-fabric {
        base fabric-type;
                description
                        "vxlan fabric";
    }

    identity vlan-fabric {
        base fabric-type;
                description
                        "vlan fabric";
    }

    typedef service-capabilities {
        type enumeration {
            enum ip-mapping {
                                description "NAT";
                        }
            enum acl-redirect{
                                description "acl redirect, which can provide
SFC function";
                        }
            enum dynamic-route-exchange{
                                description "dynamic route exchange";
                        }
```

```
        }
```

```
                description
                        "capability of the device";
    }
    /*
     * Typedefs
     */
    typedef node-ref {
        type instance-identifier;
        description "A reference to a node in topology";
    }

    typedef tp-ref {
        type instance-identifier;
        description "A reference to a termination point in topology";
    }

    typedef link-ref {
                type instance-identifier;
        description "A reference to a link in topology";
    }

    typedef device-role {
        type enumeration {
            enum SPINE {
                description "a spine node";
            }
            enum LEAF {
                description "a leaf node";
            }
            enum BORDER {
                description "a border node";
            }
        }
        default "LEAF";
        description "device role type";
    }

    typedef fabric-port-role {
        type enumeration {
            enum internal {
                description "the port used for devices to access each other.";
            }
            enum external {
                description "the port used for fabric to access outside
network.";
            }
            enum access {
                description "the port used for Endpoint to access fabric.";
```

```
            }
```

```
            enum reserved {
                description " not decided yet. ";
            }
        }
                description "the role of the physical port ";
    }

    typedef fabric-port-type {
        type enumeration {
            enum layer2interface {
                                description "l2 if";
            }
            enum layer3interface {
                                description "l3 if";
            }
            enum layer2Tunnel {
                                description "l2 tunnel";
            }
            enum layer3Tunnel {
                                description "l3 tunnel";
            }
        }
                description
                        "fabric port type";
    }

    typedef underlayer-network-type {
        type enumeration {
            enum VXLAN {
                description "vxlan";
            }
            enum TRILL {
                description "trill";
            }
            enum VLAN {
                description "vlan";
            }
        }
                description "";
    }

    typedef layer2-protocol-type-enum {
        type enumeration {
            enum VLAN{
                                description "vlan";
                    }
            enum VXLAN{
                                description "vxlan";
```

```
                        }
            enum TRILL{
                                description "trill";
                        }
            enum NvGRE{
                                description "nvgre";
                        }
        }
                description "";
    }

    typedef access-type {
        type enumeration {
            enum exclusive{
                                description "exclusive";
                        }
            enum vlan{
                                description "vlan";
                        }
        }
        description "";
    }

    grouping fabric-port {
                description
                        "attributes of a fabric port";
        leaf name {
            type string;
                        description "name of the port";
        }
        leaf role {
            type fabric-port-role;
                        description "role of the port in a fabric";
        }
        leaf type {
            type fabric-port-type;
                        description "type of the port";
        }
        leaf device-port {
            type tp-ref;
                        description "the device port it mapped to";
        }
        choice tunnel-option {
                        description "tunnel options";

            case gre {
                 leaf src-ip {
                    type inet:ip-prefix;
```

```
                                    description "source address";
                }
                leaf dest-ip {
                    type inet:ip-address;
                                    description "destination address";
                }
            }
        }
    }

    grouping route-group {
                description
                        "route attributes";
        list route {
            key "destination-prefix";
                        description "route list";

            leaf description {
                type string;
                description "Textual description of the route.";
            }
            leaf destination-prefix {
                type inet:ipv4-prefix;
                mandatory true;
                description "IPv4 destination prefix.";
            }
            choice next-hop-options {
                                description "choice of next hop options";
                case simple-next-hop {
                    leaf next-hop {
                        type inet:ipv4-address;
                        description "IPv4 address of the next hop.";
                    }
                    leaf outgoing-interface {
                        type nt:tp-id;
                        description "Name of the outgoing interface.";
                    }
                }
            }
        }
    }

    grouping port-functions {
                description
                        "port functions";

        container port-function {
                        description "port functions";
```

```
            choice function-type {
                                description "type of functions";
                case ip-mapping {
                    list ip-mapping-entry {
                                                key "external-ip";
                                                description "list of NAT
entry";
                        leaf external-ip {
                            type inet:ipv4-address;
                                                        description "external
address";
                        }
                        leaf internal-ip {
                            type inet:ipv4-address;
                                                        description "internal
address";
                        }
                    }
                }
            }
        }
    }
    grouping acl-list {
                description "acl list";
        list fabric-acl {
                        key fabric-acl-name;
                        description "fabric acl list";
            leaf fabric-acl-name {
                type string;
                                description "acl name";
            }
        }
    }
        ///groupings for logical element
        grouping logical-switch {
                description "grouping attributes for a logical switch.";

        leaf lsw-uuid {
            type yang:uuid;
                        description "logical switch id";
        }
        leaf name {
            type string;
                        description "logical switch name";
        }
        leaf segment-id {
            type uint32;
                        description "segement id";
```

```
        }
        leaf network {
            type inet:ip-prefix;
```

```
                    description "subnet";
        }
        leaf external {
            type boolean;
                    description "whether its a lsw to external network";
        }
        uses acl-list;
    }

    grouping logical-router {
                description "grouping atttributes for a logical router";
        leaf lr-uuid {
            type yang:uuid;
                    description "logical router id";
        }
        leaf name {
            type string;
                    description "logical router name";
        }
        leaf vrf-ctx {
            type uint32;
                    description "logical router vrf id";
        }

        uses acl-list;

                container routes {
                        description "routes";
                uses route-group;
        }
    }

    grouping logical-port {
                description "grouping attributes for logical ports";
        leaf lport-uuid {
            type yang:uuid;
                    description "logical port id";
        }
        leaf name {
            type string;
                    description "logical port name";
        }
        container port-layer {
                    description "layer information of the lport";

            container layer-1-info {
                            description "layer 1 information of the lport";
                leaf location {
```

```
                type nt:tp-id;
                                    description "L1 tp id";
            }
        }
        container layer-2-info {
                            description "layer 2 information of the lport";
            leaf access-type {
                type access-type;
                                    description "l2 access type";
            }
            leaf access-segment {
                type uint32;
                                    description "access segement";
            }
        }
        container layer-3-info {
                            description "layer 3 information of the lport";
            leaf ip {
                type inet:ip-address;
                                    description "ip address";
            }
            leaf network {
                type inet:ip-prefix;
                                    description "ip prefix";
            }
            leaf mac {
                type yang:mac-address;
                                    description "mac address";
            }
            leaf forward-enable {
                type boolean;
                                    description "whether enable forward";
            }
            leaf logical-switch {
                type nw:node-id;
                                    description "lsw id";
            }
        }
    }

    uses acl-list;
    uses port-functions;

    list underlayer-ports {
        key port-ref;
                    description "list of the corresponding underlay ports";
        leaf port-ref {
            type instance-identifier;
```

```
                                description "port reference";
            }
        }
    }
}
<CODE ENDS>

<CODE BEGINS> file "ietf-fabric-service@2017-03-03.yang"
module ietf-fabric-service {

    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-fabric-service";
    prefix fabric-services;

    import ietf-network { prefix nw;  }
    import ietf-network-topology { prefix nt;  }
        import ietf-fabric-types { prefix fabrictype; revision-date
"2016-10-13"; }

    organization
    "IETF I2RS (Interface to the Routing System) Working Group";

        contact
   "WG Web:    <http://tools.ietf.org/wg/i2rs/ >
     WG List:   < mailto:i2rs@ietf.org>

     WG Chair:  Susan Hares
                <mailto:shares@ndzh.com>

     WG Chair:  Russ White
                <mailto:russ@riw.us>

     Editor:    Yan Zhuang
                <mailto:zhuangyan.zhuang@huawei.com>

     Editor:    Danian Shi
                <mailto:shidanian@huawei.com >";

    description
        "This module contains a collection of YANG definitions for Fabric
services.
                Copyright (c) 2016 IETF Trust and the persons identified as
                authors of the code.  All rights reserved.

                Redistribution and use in source and binary forms, with or
                without modification, is permitted pursuant to, and subject
                to the license terms contained in, the Simplified BSD License
                set forth in Section 4.c of the IETF Trust's Legal Provisions
                Relating to IETF Documents
```

(http://trustee.ietf.org/license-info).

```
                This version of this YANG module is part of
                draft-zhuang-i2rs-yang-fabric-services;
                see the RFC itself for full legal notices.";

        revision "2017-03-03" {
            description
                        "remove rpc commands";
                reference
                        "draft-zhuang-i2rs-yang-fabric-service-01";
        }
    revision "2016-10-12" {
        description
            "Initial revision of fabric service.";
                reference
                        "draft-zhuang-i2rs-yang-fabric-service-00";
    }

    augment "/nw:networks/nw:network/nw:node" {
        description "Augmentation for logic switch nodes provided by
fabrices.";

        container lsw-attribute {

                        description "attributes for logical switches";
            uses fabrictype:logical-switch;
        }
    }

    augment "/nw:networks/nw:network/nw:node" {
        description "Augmentation for logical router nodes provided by fabric
services.";

        container lr-attribute {

                        description "attributes for logical routers";
            uses fabrictype:logical-router;
        }
    }

    augment "/nw:networks/nw:network/nw:node/nt:termination-point" {
        description "Augmentation for logical port provided by fabric
services.";

        container lport-attribute {

                        description "attributes for logical ports";
            uses fabrictype:logical-port;
        }
    }
```

```
}
```

<CODE ENDS>

<CODE BEGINS> file "ietf-fabric-endpoint@2016-10-12.yang"
module ietf-fabric-endpoint {

    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-fabric-endpoint";
    prefix fabric-endpoints;

    import ietf-inet-types { prefix "inet"; revision-date "2013-07-15"; }
    import ietf-yang-types { prefix "yang"; revision-date "2013-07-15"; }
    import ietf-network { prefix nw; }
    import ietf-network-topology { prefix nt; }
        import ietf-fabric-types { prefix fabrictype; revision-date
"2016-10-13"; }
        import ietf-fabric-topology { prefix fabric; }

        organization
    "IETF I2RS (Interface to the Routing System) Working Group";

        contact
    "WG Web: <http://tools.ietf.org/wg/i2rs/ >
     WG List: <mailto:i2rs@ietf.org>

     WG Chair: Susan Hares
                <mailto:shares@ndzh.com>

     WG Chair: Russ White
                <mailto:russ@riw.us>

     Editor: Yan Zhuang
                <mailto:zhuangyan.zhuang@huawei.com>

     Editor: Danian Shi
                <mailto:shidanian@huawei.com>";

    description
        "This module contains a collection of YANG definitions for endpoints in
Fabric service.

                Copyright (c) 2016 IETF Trust and the persons identified as
                authors of the code. All rights reserved.

                Redistribution and use in source and binary forms, with or
                without modification, is permitted pursuant to, and subject
                to the license terms contained in, the Simplified BSD License
                set forth in Section 4.c of the IETF Trust's Legal Provisions
                Relating to IETF Documents

(http://trustee.ietf.org/license-info).

```
                This version of this YANG module is part of
                draft-zhuang-i2rs-yang-dc-fabric-network-topology;
                see the RFC itself for full legal notices.";

    revision "2016-10-12" {
        description
            "Initial revision of faas.";
                reference
                        "draft-zhuang-i2rs-yang-fabric-service-00";
    }

    grouping device-location {
        description "the location for this endponits in the physical network.";

        leaf node-ref {
            type fabrictype:node-ref;
                        description "node reference";
        }

        leaf tp-ref {
            type fabrictype:tp-ref;
                        description "port reference";
        }

        leaf access-type {
            type fabrictype:access-type;
            default "exclusive";
                        description "access type";
        }

        leaf access-segment {
            type uint32;
            default 0;
                        description "access segement";
        }
    }

    grouping endpoint-attributes {
                description "endpoint attributes";

        leaf endpoint-uuid {
            type yang:uuid;
                        description "endpoint id";
        }

        leaf own-fabric {
            type fabric:fabric-id;
                        description "fabric id";
```

```
        }

        leaf mac-address {
            type yang:mac-address;
                        description "mac addr";
        }

        leaf ip-address {
            type inet:ip-address;
                        description "ip addr";
        }

        leaf gateway {
            type inet:ip-address;
                        description "gateway ip";
        }

        leaf public-ip {
            type inet:ip-address;
                        description "public ip addr";
        }

        container location {
                        description "physical location of the endpoint";
            uses device-location;
        }

        container logical-location {
            description "The location for this endpoint in the logical
network.";

            leaf node-id {
                type nw:node-id;
                                description "node id";
            }

            leaf tp-id {
                type nt:tp-id;
                                description "port id";
            }
        }
    }

    container endpoints {
                config false;
        description "endpoints registry for faas.";

        list endpoint {
            key "endpoint-uuid";
```

```
                        description "endpoint list";

                        uses endpoint-attributes;
        }
    }

    /********************RPC*************************************/
    rpc register-endpoint {
        description
            "Register a new endpoing into the registry.";

        input {
           leaf fabric-id {
                type fabric:fabric-id;
                                description "fabric id";
            }

            uses endpoint-attributes;
        }
        output {
            leaf endpoint-id {
                type yang:uuid;
                                description "endpoint id";
            }
        }
    }

    rpc unregister-endpoint {
        description "Unregister an endpoint or endpoints from the registry.";
        input {
           leaf fabric-id {
                type fabric:fabric-id;
                                description "fabric id";
            }

            leaf-list ids {
                type yang:uuid;
                                description "a list of ids";
            }
        }
    }

    rpc locate-endpoint {
        description "Set the physical location of the endpoing.";
        input {
           leaf fabric-id {
                type fabric:fabric-id;
                                description "fabric id";
```

```
            }

            leaf endpoint-id {
                type yang:uuid;
                                description "endpoint id";
            }
            container location {
                        description "locations";
                uses device-location;
            }
        }
    }
}
<CODE ENDS>

<CODE BEGINS> file "ietf-fabric-capable-device@2016-09-29.yang"
module ietf-fabric-capable-device {
yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-fabric-capable-device";
    prefix fabric-device;

    import ietf-inet-types { prefix "inet"; revision-date "2013-07-15"; }
    import ietf-network { prefix "nw"; }
        import ietf-network-topology { prefix nt; }
    import ietf-yang-types { prefix "yang"; revision-date "2013-07-15"; }
    import ietf-fabric-types { prefix fabrictype; revision-date "2016-09-29"; }

        organization
    "IETF I2RS (Interface to the Routing System) Working Group";

        contact
    "WG Web:    <http://tools.ietf.org/wg/i2rs/ >
     WG List:   <mailto:i2rs@ietf.org>

     WG Chair:  Susan Hares
                <mailto:shares@ndzh.com>

     WG Chair:  Russ White
                <mailto:russ@riw.us>

     Editor:    Yan Zhuang
                <mailto:zhuangyan.zhuang@huawei.com>

     Editor:    Danian Shi
                <mailto:shidanian@huawei.com>";

    description
        "This module contains a collection of YANG definitions for Fabric.";
```

```
    revision "2016-09-29" {
        description
            "Initial revision of faas.";
                reference
                        "draft-zhuang-i2rs-yang-dc-fabric-network-topology-02";
    }

    identity fabric-capable-device-context {
                description
                        "identity of fabric capable device";
    }

    grouping fabric-capable-device-attribute {
                description
                        "attributes of fabric capable device";

        leaf fabric-id {
            type nw:node-id;
                        description "fabric id";
        }
        leaf fabric-ref {
            type fabrictype:node-ref;
                        description "fabric reference";
        }
    }

    grouping fabric-capable-device-config {
                description
                        "fabric service configuration of fabric capable
device";

        list bridge-domain {
            key id;
                        description "list of bridge domains";
            leaf id {
                type string;
                                description "id of the device";
            }
            leaf flood {
                type boolean;
                default false;
                                description "flood";
            }
            leaf arp {
                type boolean;
                default false;
                                description "arp";
            }
```

```
            leaf segment {
```

```
                type uint32;
                        description "segment";
            }
        }

        list bd-port {
                key bd-port-id;
            description "bridge port list";
                leaf bd-port-id {
                type string;
                        description "bridge port id";
            }
            leaf bdid {
                type string;
                        description "bridge id";
            }
            leaf ref-tp-id {
                type nt:tp-id;
                        description "reference of termination port id";
            }
            leaf access-type {
                type fabrictype:access-type;
                        description "access type";
            }
            leaf access-tag {
                type uint32;
                        description "access tag";
            }
            uses fabrictype:acl-list;
        }

        list vrf {
                key id;
            description "vrf list";
                leaf id {
                type string;
                        description "vrf id";
            }
            leaf name {
                type string;
                        description "name";
            }
            leaf vrf-ctx {
                type int32;
                        description "context";
            }
        }
```

```
        list bdif {
                        key id;
            description "bridge interface list";
                        leaf id {
                type string;
                                description "bridge interface id";
            }
            leaf bdid {
                type string;
                                description "bridge id";
            }
            leaf vrf {
                type int32;
                                description "vrf";
            }
            leaf ip-address {
                type inet:ip-address;
                                description "ip address";
            }
            leaf mask {
                type int32;
                                description "mask";
            }
            leaf mac-address {
                type yang:mac-address;
                                description "mac address";
            }
            uses fabrictype:acl-list;
            uses fabrictype:port-functions;
        }
    }

    augment "/nw:networks/nw:network/nw:node" {
                description "augmentation for fabric capable device nodes";
        leaf-list supported-fabric {
            type identityref {
                base fabrictype:fabric-type;
            }
                        description "supported fabric types";
        }

        leaf-list capability-supported {
                        type fabrictype:service-capabilities;
                        description "service capability list";
        }

        container attributes {
            config false;
```

```
                        description "attributes of the device";
            uses fabric-capable-device-attribute;
        }

        container config {
                        description "configuration of the device";
            uses fabric-capable-device-config;
        }
    }

    augment "/nw:networks/nw:network/nw:node/nt:termination-point" {
                description
                        "References a termination point to a tp in a fabric";

        leaf port-role {
            type fabrictype:fabric-port-role;
                        description "role in a fabric";
        }
        leaf port-ref {
            type fabrictype:tp-ref;
                        description "port reference in fabric";
        }
    }

    augment "/nw:networks/nw:network/nw:node/nt:termination-point" {
                description
                        "References a termination point to a tp in bridges";
        leaf-list bd-ids {
            type instance-identifier;
                        description "bridge id list";
        }
    }
}
```

<CODE ENDS>

7. Security Considerations

None.

8. IANA Considerations

None.

9. References

9.1. Normative References

[I-D.ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo-12 (work in progress), March 2017.

[I-D.zhuang-i2rs-yang-dc-fabric-network-topology]
Zhuangyan, Z., Shi, D., Gu, R., and H. Ananthakrishnan, "A YANG Data Model for Fabric Topology in Data Center Network", draft-zhuang-i2rs-yang-dc-fabric-network-topology-03 (work in progress), March 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, DOI 10.17487/RFC2234, November 1997, <http://www.rfc-editor.org/info/rfc2234>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

9.2. Informative References

[I-D.ietf-i2rs-usecase-reqs-summary]
Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", draft-ietf-i2rs-usecase-reqs-summary-03 (work in progress), November 2016.

Authors' Addresses

Yan Zhuang (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: zhuangyan.zhuang@huawei.com

Danian Shi
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: shidanian@huawei.com

Rong Gu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: gurong_cmcc@outlook.com