NETMOD
Internet-Draft
Intended status: Standards Track
Expires: August 6, 2015

T. Zhou
S. Liu
Y. Xia
S. Jiang
Huawei Technologies Co., Ltd

February 2, 2015

YANG Data Models for Intent-based NEtwork MOdel
draft-zhou-netmod-intent-nemo-00

Abstract

This document describes a basic YANG data model for network intent. The basic model can be augmented by additional YANG modules defining data models for intent related protocols and functions to support various network scenarios and applications. The basic network intent data model provides common building blocks for extensions, such as specific node and policy information.

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 August 6, 2015.

Table of Contents

1. Introduction

Cloud computing and Software Defined Networking (SDN) are moving the IT world from a network-centric view to an application-centric view. Intent based North Bound Interface (NBI) provides applications the ability signal the "intent" (E.g. create a network between 3 applications) to the network layer rather than specify the details of the network. The network intent is prescriptive ("go to the store") rather than descriptive ("follow this route to the store"), leaving the details to the network implementation.

The NEMO specifications ([I-D.xia-sdnrg-nemo-language] and [I-D.xia-sdnrg-service-description-language]) describe a set of intent-based primitives to manipulate and manage virtual networks. Behind the NEMO language, there is a set of basic network models abstracting the network intent from the top down according to the service requirement. The NEMO intent model provides consistent

abstraction for network services while concealing various implementation techniques and multi-vendor devices.

This document introduces YANG [RFC6020] [RFC6021] data models for network intent based on NEMO abstraction. This set of models facilitates the standardization for the interface of intent networking. The basic model can be augmented by additional YANG modules defining data models for intent related protocols and functions to support various different network scenarios and applications. The basic network intent data model provides common building blocks for extensions, such as specific node and policy information.

2. Requirements Language and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

Intent-based NEtwork MOdel provides a high level description of requirements to network with the abstraction from top down. It conceals complexity to the network implementation but eases the application invocation.

3. NEMO YANG Model Overview

The 80/20 rule of thumb in network application is that 80% of network applications only use 20% of the network capability. Most of the network operation is the combination of 4 basic operations: node, link, flow and policy, of which the network intent comprises as shown in Figure 1.

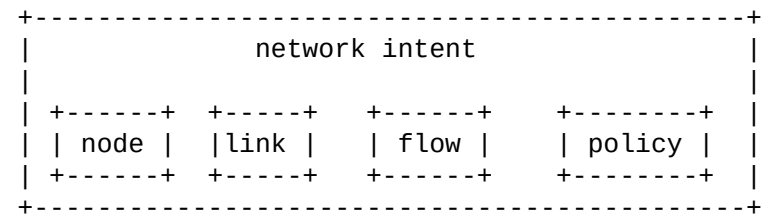

Figure 1

- Node operation includes creation, modification and deletion of a network element (NE).

- Link operation is used for the connectivity among NEs.

- Flow operation identities the flow to be operated with flow characters matching.

- Policy operation controls the behavior of entities (including node, link and flow) following the same pattern "with <condition>, to execute <action>".

3.1. Node Intent Module

3.1.1. Design of node intent module

The node intent model describes the network elements with the packet processing capability. According to the functionality, various specific nodes fall into three classes:

- The forwarding node only deals with L2/3 forwarding.

- The processing node provides L4-7 network services, and will modify the body of packets.

- The logical node describes a set of network elements and their links exposing properties as one entity.

A basic node model is shown in Figure 2.

```
module: nemo-node
   +--rw node-def
      +--rw tenant-id     yang:uuid
      +--rw node-id       yang:uuid
      +--rw name?         string
      +--rw type          node-type
      +--rw owner         yang:uuid
      +--rw properties
```

Figure 2

- tenant-id: uuid type to globally identify the tenant who owns the intent.

- node-id: uuid type to globally identify a node instance.

- name: defines the name of the node instance.

- type: describes the specific type of the node, e.g. firewall.

- owner: specifies the parent logical node where this node instance located.

- properties: can be augmented to describe various specific nodes derived from the basic node model abstraction.

3.1.2. Node intent YANG module

```
module nemo-node{
   namespace "urn:TBD:params:xml:ns:yang:intent:nemo";
   prefix nemonode;

   import ietf-yang-types {
      prefix yang;
   }

   description "A node is used to specify one resource entity which
                is responsable for one of the following functions:
                forwarding, processing, logical abstraction.";

   revision 2015-02-02{
      description "Initial revision";
   }

   typedef node-type {
      type string {
         pattern "[a-zA-Z][_0-9a-zA-Z]*";
      }
   }

   container node-def{
      leaf tenant-id{
         type yang:uuid;
         mandatory true;
         description "Specify the tenant id who owns this node object.";
      }

      leaf node-id{
         type yang:uuid;
         mandatory true;
         description "Uniquely identifies node object.";
      }

      leaf name{
         type string;
         description "name of the node object";
      }
```

```
      leaf type{
         type node-type;
         mandatory true;
         description "Specifies the concrete type of the node entity
                      object, there are different properties defination
                      for different type node.";
      }

      leaf owner{
         type yang:uuid;
         mandatory true;
         description "Used to indicate a parent node which contains
                      this node logically.";
      }

      container properties{
         description "property name and value inforamtion for a kind of
                      node.";
      }
   }
}
```

3.2. Link Intent Module

3.2.1. Design of link intent module

The link intent model describes the connectivity between node entities. The following figure shows the model abstraction.

```
module: nemo-link
   +--rw link-def
      +--rw tenant-id     yang:uuid
      +--rw link-id       yang:uuid
      +--rw name?         string
      +--rw type          link-type
      +--rw endnodes
      |  +--rw one-node-id        yang:uuid
      |  +--rw another-node-id    yang:uuid
      +--rw properties
```

Figure 3

- tenant-id: uuid type to globally identify the tenant who owns the intent.
- link-id: uuid type to globally identify a link instance.
- liname: defines the name of the link instance.

- type: describes the specific type of the link which has different dedicated properties.
- endnodes: describes the two ends of the link.
- properties: can be augmented to describe various specific links derived from the basic link model abstraction.

3.2.2. Link intent YANG module

```
module nemo-link{
   namespace "urn:TBD:params:xml:ns:yang:intent:nemo";
   prefix nemolink;

   import ietf-yang-types {
      prefix yang;
   }

   description "A link is used to specify a connection relationship
                between two node object.";

   revision 2015-02-02{
      description "Initial revision";
   }

   typedef link-type {
      type string {
         pattern "[a-zA-Z][_0-9a-zA-Z]*";
      }
   }

   container link-def{
      leaf tenant-id{
         type yang:uuid;
         mandatory true;
         description "Specify the tenant id who owns the link object.";
      }

      leaf link-id{
         type yang:uuid;
         mandatory true;
         description "Uniquely identifies link object.";
      }

      leaf name{
         type string;
         description "name of the link object.";
      }
```

```
       leaf type{
          type link-type;
          mandatory true;
          description "Specifies the concrete type of the link entity
                       object, there are different properties defination
                       for different type link";
       }

       container endnodes{
          leaf one-node-id{
             type yang:uuid;
             mandatory true;
             description "Uniquely identifies node entity object which
                          is as the endpoint of the link.";
          }
          leaf another-node-id{
             type yang:uuid;
             mandatory true;
             description "Uniquely identifies node entity object which
                          is as the endpoint of the link.";
          }
       }

       container properties{

       }
    }
 }
```

3.3. Flow Intent Module

3.3.1. Design of flow intent module

The flow intent model describes a sequence of packets with certain common characters, such as source/destination IP address, port, and protocol. From the intent perspective, flow is the special traffic with user concern, which may be per device or across many devices. So the flow characters also include ingress/egress node, and so on.

The following figure shows the model abstraction.

```
module: nemo-flow
   +--rw flow-def
      +--rw tenant-id    yang:uuid
      +--rw flow-id     yang:uuid
      +--rw name?      string
      +--rw match
```

Figure 4

- tenant-id: uuid type to globally identify the tenant who owns the intent.
- flow-id: uuid type to globally identify a flow instance.
- name: defines the name of the flow instance.
- match: describes common characters of a flow, such as source/ destination IP address, port, and protocol. It’s not limited to IP based packet headers, but can be augmented to describe various high level match items according to the user intent.

3.3.2. Flow intent YANG module

```
 module nemo-flow{
    namespace "urn:TBD:params:xml:ns:yang:intent:nemo";
    prefix nemoflow;

    import ietf-yang-types {
       prefix yang;
    }

    description "A flow object represents a kind of service data flow.";

    revision 2015-02-02{
       description "Initial revision";
    }

    container flow-def{
       leaf tenant-id{
          type yang:uuid;
          mandatory true;
          description "Specify the tenant id who owns the flow object.";
       }

       leaf flow-id{
          type yang:uuid;
          mandatory true;
          description "Uniquely identifies flow object.";
       }

       leaf name{
          type string;
          description "name of the flow object";
       }

       container match{
          description "Flow match items.";
       }
    }
 }
```

3.4. Policy Intent Module

3.4.1. Design of policy intent module

The policy intent controls the behavior of specific entities by APP, such as flow policy, node policy. All the policies follow the same pattern "with <condition>, to execute <action>", and can be applied to any entity.

The following figure shows the model abstraction.

```
module: nemo-policy
   +--rw policy-def
      +--rw tenant-id    yang:uuid
      +--rw policy-id    yang:uuid
      +--rw applyto      yang:uuid
      +--rw name?        string
      +--rw priority?    uint32
      +--rw condition
      +--rw actions
```

Figure 5

- tenant-id: uuid type to globally identify the tenant who owns the intent.
- policy-id: uuid type to globally identify a policy instance.
- applyto: indicates the entity to which the policy will be applied.
- name: defines the name of the policy instance.
- condition: describes the trigger condition to execute the policy.
- actions: describes the actions to be executed when the condition meets. This item can be augmented to various specific actions to describe the intent.

3.4.2. Policy intent YANG module

```
module nemo-policy{
   namespace "urn:TBD:params:xml:ns:yang:intent:nemo";
   prefix nemopolicy;

   import ietf-yang-types {
      prefix yang;
   }

   description "Describes network control policy.";

   revision 2015-02-02{
      description "Initial revision";
   }

   container policy-def{

      leaf tenant-id{
         type yang:uuid;
         mandatory true;
```

```
         description "Specify the tenant id who owns the policy
                      object.";
      }

      leaf policy-id{
         type yang:uuid;
         mandatory true;
         description "Uniquely identifies policy capability entity
                      object.";
      }

      leaf applyto{
         type yang:uuid;
         mandatory true;
         description "Used to indicate the entity object which will be
                      applied policy.";
      }

      leaf name{
         type string;
         description "name of the policy object";
      }

      leaf priority{
         type uint32;
         mandatory false;
         default "0";
         description "A priority is used to specify executive order for
                      policy.";
      }

      container condition{
         description "Used to indicate the policy condition, policy will
                      be carred out just when the expression is true.";
      }

      container actions{
         description "network control action name";
      }
   }
}
```

3.5. Intent Batch Module

3.5.1. Design of intent batch module

In addition to submit intent one by one, intents can be submitted as a batch with the combination of 4 basic intents.

The below figure shows how this batch is described.

```
module: nemo-intent-batch
   +--rw tenant-id     yang:uuid
   +--rw nodes*        yang:uuid
   +--rw links*        yang:uuid
   +--rw flows*        yang:uuid
   +--rw policies*     yang:uuid
   +--rw properties
```

Figure 6

3.5.2. Intent batch YANG module

```
module nemo-intent-batch{
   namespace "urn:TBD:params:xml:ns:yang:intent:nemo";
      prefix nemointentbatch;

      import ietf-yang-types {
         prefix yang;
      }

      description "A intent batch consists of some nemo model object.";

      revision 2015-02-02{
         description "Initial revision";
      }

      leaf tenant-id{
         type yang:uuid;
         mandatory true;
         description "Specify the tenant id who owns the policy
                      object.";
      }

      leaf-list nodes {
         type yang:uuid;
         description "node entity instance list";
      }

      leaf-list links {
         type yang:uuid;
         description "link entity instance list";
      }

      leaf-list flows {
         type yang:uuid;
         description "flow entity instance list";
      }

      leaf-list policies {
         type yang:uuid;
         description "policy entity instance list";
      }

      container properties{
         description "property name and value information for the whole
                      intent service.";
      }
}
```

4. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in the YANG module which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., <edit-config>) to these data nodes without proper protection can have a negative effect on network operations.

5. IANA Considerations

This memo includes no request to IANA.

6. Acknowledgements

The authors would like to thanks the valuable comments made by Wei Cao, Xiaofei Xu, Fuyou Miao and Wenyang Lei.

This document was produced using the xml2rfc tool [RFC2629].

7. Informative References

[I-D.xia-sdnrg-nemo-language]
Xia, Y., Jiang, S., Zhou, T., and S. Hares, "NEMO (NEtwork MOdeling) Language", draft-xia-sdnrg-nemo-language-01 (work in progress), October 2014.

[I-D.xia-sdnrg-service-description-language]
Xia, Y., Jiang, S., and S. Hares, "Requirements for a Service Description Language and Design Considerations", draft-xia-sdnrg-service-description-language-01 (work in progress), October 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Authors' Addresses

Tianran Zhou
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: zhoutianran@huawei.com

Shixin Liu
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: liushixin@huawei.com

Yinben Xia
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: xiayinben@huawei.com

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: jiangsheng@huawei.com