OPEN NETWORKING FOUNDATION

OpenFlow™ Controller/Switch NDM Synchronization

v1.0

15 August 2014

NOTE: ONF specification
TS_OpenFlow_Negotiable_Datapath_Models_v.1.0_062014
is closely related to this specification.

ONF Document Type: OpenFlow Spec
ONF Document Name: OpenFlow Controller-Switch NDM Synchronization v1.0

Disclaimer

1 Introduction

This document describes how OFCPs can use OF-Config 1.2 to negotiate an agreed NDM (negotiable datapath model) with an OpenFlow Capable switch. Negotiation can include discovery of available switch-supported NDMs, request for parameter constraints, and ultimately activation of specific NDMs with specific parameters.

2 OF-Config-based NDM Negotiation

2.1 NDM Negotiation Overview

For system implementations that support the OF-Config1.2 protocol, a method is defined for the OFCP to determine which NDMs are supported by a capable switch, and subsequently to associate an NDM with a logical switch. In cases where the NDM implementation has adjustable parameters, the OFCP can attempt to impose value constraints on one or more of the NDM parameters, and switch will either reject the constraints, or will respond with a collection of parameter values that satisfy the requested constraints.

The following section describes the negotiation process.

2.2 Basic Flow of Operations

1. OFCP connects to an OpenFlow capable switch and learns the following through NETCONF capabilities exchange:
    a. Does the capable switch implement NDMs?
        i. Detectable through the optional presence of the NDM YANG module identifiers (URIs)
    b. A list of NDMs that the capable switch implements through their respective unique identifier (URIs)

2. For some very simple “OFCP clients” (e.g. shell scripts), it may be difficult to process the information provided in the capabilities exchange. An alternative approach is to query the subtree that would contain the basic NDM parameters by performing a 'get' on the following node:

   ```
   /capable-switch/resources/ndm
   ```

   If that subtree does not exist (i.e. the response is empty), the client can safely assume that the capable switch does not implement NDMs.

3. If the capable switch does indeed implement NDM supports, the client can proceed by querying the capable switch for available NDMs performing a 'get' operation on the following node:

   ```
   /capable-switch/resources/ndm/available-ndms
   ```

   The capable switch responds with a list of available NDMs as a list with the following leafs:

   ```
   ndm:authority  string
   ndm:name       string
   ndm:type       enumeration (ttp, fpmod)
   ndm:version    string
   ```

   Any empty response means that while it does implement NDM functionality there are currently no NDMs available for use.

4. If the client finds NDMs for which it doesn't currently have access to the associated YANG module, it can use the (optional) *get-schema* operation as described in RFC 6022, Section 3.1 to fetch the YANG modules for specific NDMs.

The client now knows:

a. Which NDMs the capable switch supports through the capabilities exchange in (1. above) or the query in (2. above).
b. The formal set of configuration parameters (including default values) and negotiation RPCs for all NDMs as defined in the corresponding YANG modules.

5. The client may now proceed by instantiating (and parameterizing) an NDM on any of the available logical switches either according to the data definitions in the YANG module (5.a below); or by first collecting suggested parameters from the capable switch using the suggest-ndm-parameters RPC if present (5.b below)

    a. The client uses basic NETCONF *edit-config* operations to create an instantiated (parameterized) NDM on any of the logical switches

    b. The client uses the *suggest-ndm-parameters* RPC with NDM-specific input values to query the switch for suggested parameters. The switch responds with a complete and valid set of data for the client to use in creating an NDM. The OFCP proceeds as described in (5.1)

       The *suggest-ndm-parameters* RPC may be called for several NDMs to collect suggested parameters before deciding on which NDM to create as described in (6. above)

2.3 Example NDM Negotiation

This section contains an informal set of terminal and protocol dumps corresponding to the general steps above to provide an example. We make use of the *netconf-console* tool to query the NETCONF server.

The command below initiates a capabilities-exchange with the NETCONF server (within a capable switch) to find out which protocol capabilities the switch supports and which YANG modules are implemented. We can see below that the server supports of-config, ndm, and two specific ndm-modules l2l3 and l2l3acl.

```
calle@macbook:openflow $netconf-console --hello
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
[...]
    <capability>urn:onf:of111:config:yang?module=of-config&amp;revision=2011-12-07</capability>
<capability>urn:opennetworking.org:yang:ndm?module=ndm&amp;revision=2013-04-30</capability>
    <capability>urn:opennetworking.org:yang:ndm:l2l3?module=l2l3</capability>
    <capability>urn:opennetworking.org:yang:ndm:l2l3acl?module=l2l3acl</capability>
  </capabilities>
  <session-id>13</session-id>
</hello>
```

The command below queries the server for configuration and operational data under the `/capable-switch/resources/ndm` node. The server returns data (in this case the name, type and version of the two supported NDMs). The client can now safely assume that the server supports NDMs.

```
calle@macbook:openflow $netconf-console --get -x /capable-switch/resources/ndm
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <capable-switch xmlns="urn:onf:of111:config:yang">
      <resources>
        <ndm xmlns="urn:opennetworking.org:yang:ndm">
          <available-ndms>
            <name>l2l3acl</name>
            <type>ttp</type>
            <version>draft</version>
          </available-ndms>
          <available-ndms>
```



```
            <name>l2l3</name>
            <type>ttp</type>
            <version>draft</version>
          </available-ndms>
        </ndm>
      </resources>
    </capable-switch>
  </data>
</rpc-reply>
calle@macbook:openflow $
```

This is an example where the server has no support for NDMs:

```
calle@macbook:openflow $netconf-console --get -x /capable-switch/resources/xlndm
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data/>
</rpc-reply>
```

The below shows how to fetch a specific YANG module in-band using the `get-schema` and the name of the module we want to fetch (l2l3acl). Module content edited for space.

```
calle@macbook:openflow $netconf-console --get-schema=l2l3acl
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring"><![CDATA[module l2l3acl {

  namespace "urn:opennetworking.org:yang:ndm:l2l3acl";
  prefix "l2l3acl";

  import of-config {
    prefix ofc;
  }

  import ndm {
    prefix ndm;
  }

  grouping l2l3acl-params {
    leaf ingress-vlan-table-size {
      type uint32;
    }
[...]
}]]></data>
</rpc-reply>
calle@macbook:openflow $
```

The YANG module includes parameter information for the NDM, including the default values and which parameters may be adjusted. If there are no adjustable parameters, or if the OFCP client finds the default parameters acceptable as-is, then it knows which parameters to use with the NDM. (Otherwise, see “Example Parameter Adjustment” below.) At this point, the OFCP client has the information it needs to perform an edit-config operation to associate a parameterized NDM with the logical switch. First we’ll take a look at the content of the NDM we’re planning to add:

```
calle@macbook:openflow $more add-l2l3acl-implementation.xml
<capable-switch xmlns="urn:onf:of111:config:yang"
                xmlns:ndm="urn:opennetworking.org:yang:ndm"
                xmlns:l2l3acl="urn:opennetworking.org:yang:ndm:l2l3acl">
  <logical-switches>
    <switch>
      <id>LogicalSwitch6</id>
      <resources>
        <ndm:parameterized-ndm>
          <l2l3acl:l2l3acl>
            <l2l3acl:ingress-vlan-table-size>128</l2l3acl:ingress-vlan-table-size>
            <l2l3acl:acl-table-size>128</l2l3acl:acl-table-size>
            <l2l3acl:router-mac-table-size>128</l2l3acl:router-mac-table-size>
            <l2l3acl:l3-table-size>128</l2l3acl:l3-table-size>
```



```
            <l2l3acl:l2-table-size>128</l2l3acl:l2-table-size>
            <l2l3acl:egress-vlan-table-size>128</l2l3acl:egress-vlan-table-size>
          </l2l3acl:l2l3acl>
        </ndm:parameterized-ndm>
      </resources>
    </switch>
  </logical-switches>
</capable-switch>
```

The next step is to add the NDM with all parameters set to default values. We use the edit-config operation as follows:

```
calle@macbook:openflow $netconf-console --edit-config=add-l2l3acl-implementation.xml
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <ok/>
</rpc-reply>
```

Once the edit-config has completed, we can take a look at the newly parameterized NDM in the configuration using the get-config operation:

```
calle@macbook:openflow $netconf-console --get -x "/capable-switch/logical-
switches/switch[id='LogicalSwitch6']/resources/parameterized-ndm"
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <capable-switch xmlns="urn:onf:of111:config:yang">
      <logical-switches>
        <switch>
          <id>LogicalSwitch6</id>
          <resources>
            <parameterized-ndm xmlns="urn:opennetworking.org:yang:ndm">
              <l2l3acl xmlns="urn:opennetworking.org:yang:ndm:l2l3acl">
                <ingress-vlan-table-size>128</ingress-vlan-table-size>
                <acl-table-size>128</acl-table-size>
                <router-mac-table-size>128</router-mac-table-size>
                <l3-table-size>128</l3-table-size>
                <l2-table-size>128</l2-table-size>
                <egress-vlan-table-size>128</egress-vlan-table-size>
              </l2l3acl>
            </parameterized-ndm>
          </resources>
        </switch>
      </logical-switches>
    </capable-switch>
  </data>
</rpc-reply>
calle@macbook:openflow $
```

2.4 Example Parameter Adjustment

If an NDM implementation supports parameter adjustment, the OFCP client may choose to request parameters other than the defaults. The suggest-ndm-parameters RPC, shown below, can be used to ask the switch to suggest a full set of parameters that comply with certain constraints. The constraints are represented as requested parameters, which may be larger table sizes, or may represent optional functionality (see the discussions of the support meta members and the built-in OptFunc parameter, both in section 3 of this document.). First we take a look at the requested parameters suggested by the client:

```
calle@macbook:openflow $more action.xml
<suggest-ndm-parameters xmlns="urn:opennetworking.org:yang:ndm">
  <l2l3acl xmlns="urn:opennetworking.org:yang:ndm:l2l3acl">
    <ingress-vlan-table-size>128</ingress-vlan-table-size>
    <router-mac-table-size>128</router-mac-table-size>
  </l2l3acl>
</suggest-ndm-parameters>
```


Then we execute the RPC and capture the output. If the output is empty, then the NDM implementation was not able to generate a set of parameters that satisfy the provided parameter constraints. If that occurs, the OFCP client can try again using different constraints, or decide not to use the NDM.

If the output is not empty, the returned parameters (in the rpc-rep ly below) can be used in an edit-config operation (similar to the earlier example, except that now we specify all the parameters) to add a parameterized NDM.

```
calle@macbook:openflow $netconf-console --rpc=action.xml
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <l2l3acl xmlns="urn:opennetworking.org:yang:ndm:l2l3acl">
    <ingress-vlan-table-size>128</ingress-vlan-table-size>
    <acl-table-size>128</acl-table-size>
    <router-mac-table-size>128</router-mac-table-size>
    <l3-table-size>128</l3-table-size>
    <l2-table-size>128</l2-table-size>
    <egress-vlan-table-size>128</egress-vlan-table-size>
  </l2l3acl>
</rpc-reply>
calle@macbook:openflow $
```