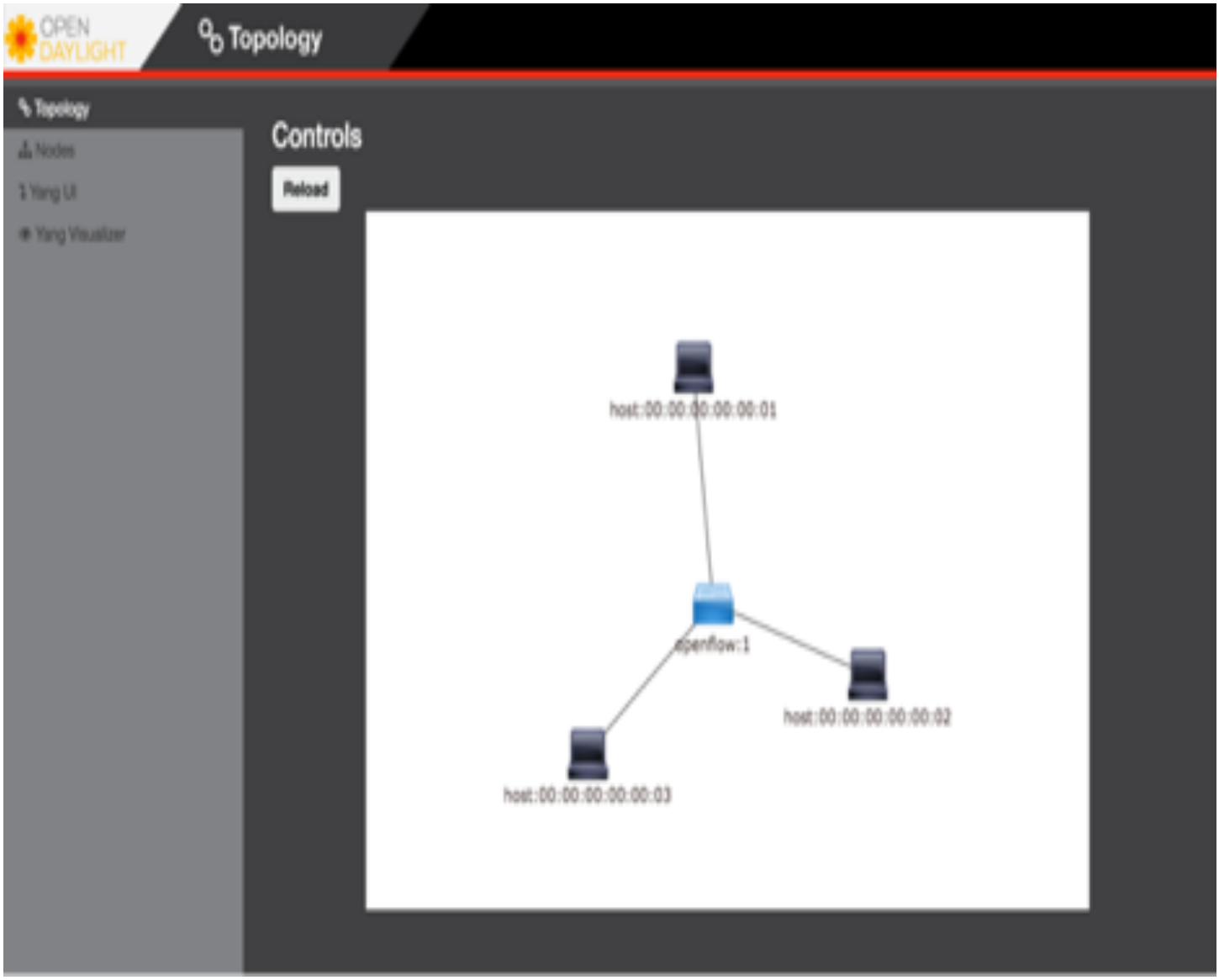
SCREEN SHOTS OF PUSH & PULL CREATION

SCREEN SHOT 1: SIMPLE NETWORK TOPOLOGY CREATED WITH MININET IN AWS INSTANCE

http://ec2-52-90-221-43.compute-1.amazonaws.com:8181/index.html#/topology



SCREEN SHOT 2: REVERSE ENGINEERING RESTCONF INFO TO FETCH DEVICE DETAILS

http://ec2-52-90-221-43.compute-1.amazonaws.com:8181/restconf/operational/opendaylight-inventory:nodes

Modeling the schema to parse the device information from opendaylight


```
<table xmlns="urn:opendaylight:flow:inventory">
  <id>0</id>
  <aggregate-flow-statistics xmlns="urn:opendaylight:flow:statistics">
    <flow-count>11</flow-count>
    <packet-count>64</packet-count>
    <byte-count>4884</byte-count>
  </aggregate-flow-statistics>
  <flow>
    <id>#UF$TABLE*0-2</id>
    <flags/>
    <table_id>0</table_id>
    <hard-timeout>0</hard-timeout>
    <priority>2</priority>
    <idle-timeout>0</idle-timeout>
    <instructions>
      <instruction>
        <order>0</order>
        <apply-actions>
          <action>
            <order>2</order>
            <output-action>
              <output-node-connector>CONTROLLER</output-node-connector>
              <max-length>65535</max-length>
            </output-action>
          </action>
          <action>
            <order>1</order>
            <output-action>
              <output-node-connector>2</output-node-connector>
              <max-length>65535</max-length>
            </output-action>
          </action>
          <action>
            <order>0</order>
            <output-action>
              <output-node-connector>3</output-node-connector>
              <max-length>65535</max-length>
            </output-action>
          </action>
        </apply-actions>
      </instruction>
    </instructions>
    <match>
      <in-port>openflow:1:1</in-port>
    </match>
    <cookie>309847654363090125O</cookie>
```

```
<flow-statistics xmlns="urn:opendaylight:flow:statistics">
  <duration>
    <second>534</second>
    <nanosecond>16000000</nanosecond>
  </duration>
  <packet-count>13</packet-count>
  <byte-count>1022</byte-count>
</flow-statistics>
</flow>
<flow>
  <id>#UF$TABLE*0-1</id>
  <flags/>
  <table_id>0</table_id>
  <hard-timeout>0</hard-timeout>
  <priority>2</priority>
  <idle-timeout>0</idle-timeout>
  <instructions>
    <instruction>
      <order>0</order>
      <apply-actions>
        <action>
          <order>2</order>
          <output-action>
            <output-node-connector>CONTROLLER</output-node-connector>
            <max-length>65535</max-length>
          </output-action>
        </action>
        <action>
          <order>1</order>
          <output-action>
            <output-node-connector>1</output-node-connector>
            <max-length>65535</max-length>
          </output-action>
        </action>
        <action>
          <order>0</order>
          <output-action>
            <output-node-connector>2</output-node-connector>
            <max-length>65535</max-length>
          </output-action>
        </action>
      </apply-actions>
    </instruction>
  </instructions>
  <match>
    <in-port>openflow:1:3</in-port>
  </match>
```

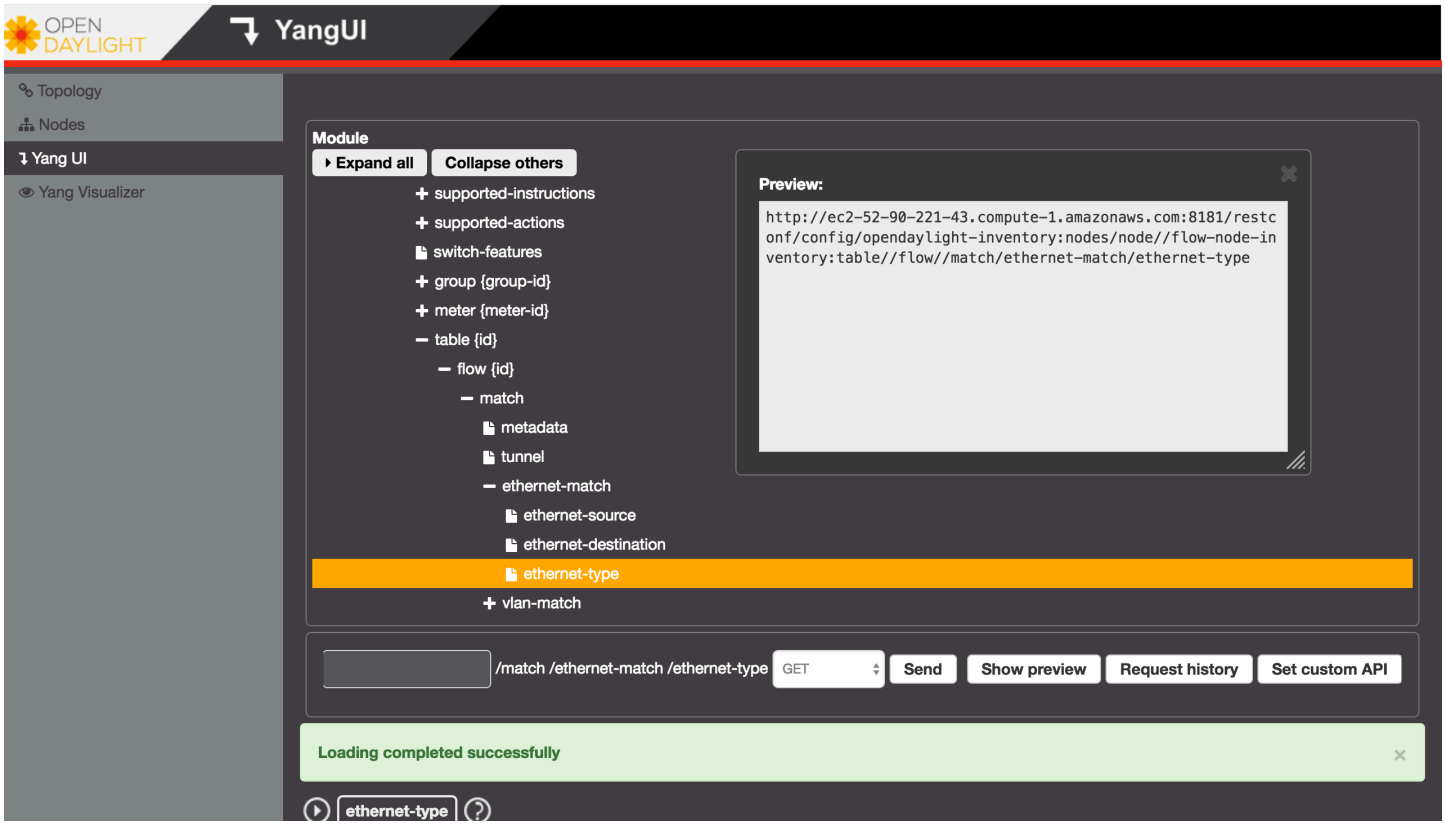
SCREEN SHOT 3: JSON CODE SNIPPET of device details extraction

https://github.com/ratchagan/REST_API/blob/master/RESTPULL.java

```
24          JSONParser jsonParser = new JSONParser();
25          JSONObject jsonObject = (JSONObject) jsonParser.parse(responseStr);
26          int sizeNodes = jsonObject.size();
27          JSONObject nodesObj = (JSONObject) jsonObject.get("nodes");
28          //Gets the entire topology
29          JSONArray deviceArray = (JSONArray) nodesObj.get("node");
30          //gets the device 1,2,3..
31          int sizeNodes2 = deviceArray.size();
32          //JSONObject jsonObject1 = (JSONObject) jsonParser.parse(responseStr);
33          //System.out.println(deviceArray.toString());
34
35          String device[][] = new String[sizeNodes2][5];
36
37          for (int i=0;i<sizeNodes2;i++)
38          {
39              JSONObject deviceObj = (JSONObject) deviceArray.get(i);
40              device[i][0] = (String) deviceObj.get("id");
41              device[i][1] = (String) deviceObj.get("flow-node-inventory:description");
42              device[i][2] = (String) deviceObj.get("flow-node-inventory:hardware");
43              device[i][3] = (String) deviceObj.get("flow-node-inventory:manufacturer");
44              device[i][4] = (String) deviceObj.get("flow-node-inventory:ip-address");
45
46
47              JSONArray portArray = (JSONArray) deviceObj.get("node-connector");
48              int portArraySize = portArray.size();
49              String portSize[][] = new String[1][1] ;
50              portSize [0][0]= String.valueOf(portArray.size());
51              paraMap.put("portSize+"+device[i][0],portSize);
52              String port[][]= new String[portArraySize][4];
53              for(int o=0;o<portArraySize;o++)
54              {
55              JSONObject portObj = (JSONObject) portArray.get(o);
56              port[o][0] = (String) portObj.get("id");
57              port[o][1] = (String) portObj.get("flow-node-inventory:hardware-address");
58              port[o][2] = (String) portObj.get("flow-node-inventory:port-number");
```

SCREENSHOT 4: CONSTRUCTING FLOW STRUCTURE THROUGH YANG MODEL

Modeling the flow:



http://ec2-52-90-221-43.compute-1.amazonaws.com:8181/restconf/config/opendaylight-inventory:nodes/node//flow-node-inventory:table//flow//match/ethernet-match/ethernet-type

SCREENSHOT 5: MAPPING FLOW INTO JSON

```
166             ///Construct the json here
167         if (flowdata[4]==null)
168             flowdata[4] = "\"10.0.0.1/24\" ";
169         if (flowdata[5] ==null)
170             flowdata[5] = "10";
171
172         String[] json_string=new String[3];
173         json_string[2]=flow.toString().split("table-0:flow-")[1]+"\n";
174         json_string[0]=flow.toString().split(topologyURI+":")[1].split(":openflow-service")[0]+"\n";
175
176         json_string[1]= "{\"flow\": [{\"id\":\"0\",\"match\": {\"ethernet-match\": {\"ethernet-type\":"
177                 + " {\"type\": "+flowdata[1]+"}},\"ipv4-source\": "+flowdata[4]+"},\"instructions\": "
178                 + "{\"instruction\": [{\"apply-actions\": {\"action\": [{\"order\": \"1\",\"flood-action\": {}}]},"
179                 + "\"order\": \"1\" }]},\"cookie_mask\": \"10\",\"out_port\": "+flowdata[5] +",\"out_group\": \"2\",\"flow-name\": \"FooXf2
180                 + "\"installHw\": \"false\",\"barrier\": \"false\",\"strict\": \"true\",\"priority\": \"2\",\"idle-timeout\": \"0\",\"hard-
181                 + "\"cookie\": \"10\",\"table_id\": \"0\"}]}";
```

http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/0

EXAMPLE FLOW IN JSON FORMAT

```
{
  "flow": [
    {
      "id": "0",
      "match": {
        "ethernet-match": {
          "ethernet-type": {
            "type": "2048"
          }
        },
        "ipv4-source": "10.0.0.1/24"
      },
      "instructions": {
        "instruction": [
          {
            "apply-actions": {
              "action": [
                {
                  "order": "1",
                  "flood-action": {}

}
                ]   },
              "order": "1"
            }
          ]
        },
        "cookie_mask": "10",
        "out_port": "10",
        "out_group": "2",
        "flow-name": "FooXf22",
        "installHw": "false",
        "barrier": "false",
        "strict": "true",
        "priority": "2",
        "idle-timeout": "0",
        "hard-timeout": "0",
        "cookie": "10",
        "table_id": "0"
      }
    ]
}
```

EXAMPLE FLOW IN XML FORMAT:

```
<flow xmlns="urn:opendaylight:flow:inventory">
<id>0</id>
<strict>true</strict>
<out_port>10</out_port>
<idle-timeout>0</idle-timeout>
<barrier>false</barrier>
<hard-timeout>0</hard-timeout>
<table_id>0</table_id>
<priority>2</priority>
<cookie>10</cookie>
<instructions>
<instruction>
<order>1</order>
<apply-actions>
<action>
<order>1</order>
</action>
</apply-actions>
</instruction>
</instructions>
<out_group>2</out_group>
<installHw>false</installHw>
<match>
<ethernet-match>
<ethernet-type>
<type>2048</type>
</ethernet-type>
</ethernet-match>
<ipv4-destination>10.0.0.1/24</ipv4-destination>
</match>
<cookie_mask>10</cookie_mask>
<flow-name>FooXf22</flow-name>
</flow>
```

SCREENSHOT 6: DEBUGGING

https://ask.opendaylight.org/question/11946/flow-pushed-using-config-rest-api-is-not-reflected-via-operational-rest-api/

PROBLEM STATEMENT:

I pushed a flow using the following REST call,

http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:2/flow-node-inventory:table/0/flow/18/match/ethernet-match/ethernet-type

But the flow is not getting updated in the operational REST call

http://localhost:8181/restconf/operational/opendaylight-inventory:nodes/

Issue Resolved:

When checking the odl controller log,the following error was thrown

**2016-04-28 02:41:19,839 | WARN | ult-dispatcher-3 | FlowForwarder | 203 - org.opendaylight.openflowplugin.applications.forwardingrules-manager - 0.1.0.Lithium | * * TableID in URI tableId=null and in palyload tableId=0 is not same. * *

In the opendaylight controller code

https://github.com/opendaylight/openflowplugin/blob/master/applications/forwardingrules-manager/src/main/java/org/opendaylight/openflowplugin/applications/frm/impl/FlowForwarder.java

private static boolean tableIdValidationPrecondition (final TableKey tableKey, final Flow flow) {
Preconditions.checkNotNull(tableKey, "TableKey can not be null or empty!");
Preconditions.checkNotNull(flow, "Flow can not be null or empty!");
*** * if (! tableKey.getId().equals(flow.getTableId())) * * {**
LOG.warn("TableID in URI tableId={} and in palyload tableId={} is not same.",

```
flow.getTableId(), tableKey.getId());
return false;
}
return true;
}
```

Found out, the flow id and the table id had to be same for the flow to be pushed to the switch.

The correct REST call for pushing a flow, is as below

http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/0

with the required match fields
https://wiki.opendaylight.org/view/OpenDaylight_OpenFlow_Plugin:End_to_End_Flows