H2020 i**P** **O**ver **I**c**N**- the bet**T**er IP (POINT)

Examples

Deploy an ICN-over-SDN Network with the SDN OpenDayLight Controller

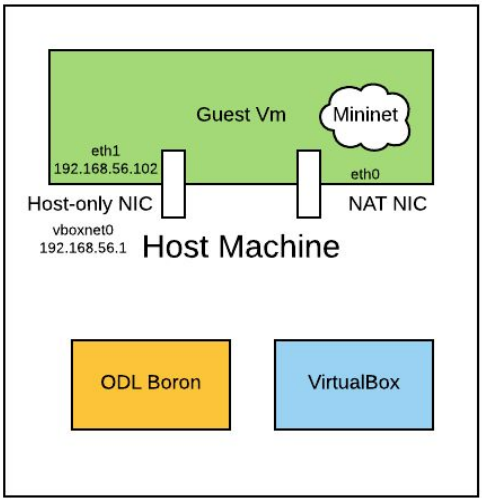
Authors: George Petropoulos, Sebastian Robitzsch, Marievi Xezonaki

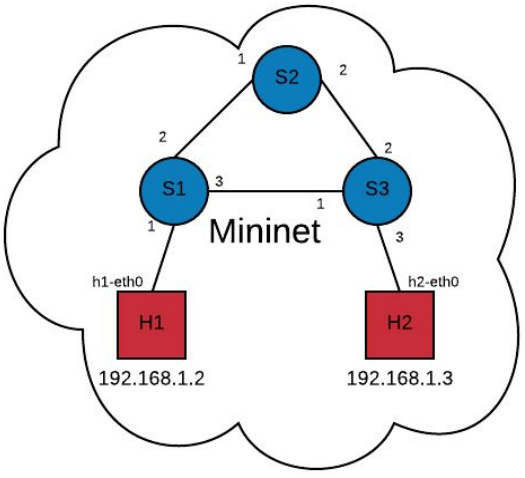
1. Overview

This example is about the ICN-over-SDN deployment capabilities of POINT, using ODL that has the ABM feature published - in the Boron release. The example is based on a Mininet-emulated SDN topology, but could be replicated in any virtualized or physical SDN topology.

2. Prerequisites

The example assumes a basic setup including a host machine with Ubuntu 14.04 or 16.04, with Virtualbox installed, and a Guest Virtual Machine (VM) with Ubuntu 15.04 installed. The Guest VM is configured to include at least one network interface, ideally two network interfaces. One host-only NIC to connect to the host machine, and one NAT NIC to connect to the Internet. For the host-only NIC, it is assumed that prefix 192.168.56.1/24 is used, and Guest VM is given (statically) an IP address 192.168.56.102, while the Host Machine has 192.168.56.1.





In the host machine, it is assumed that the reader has successfully installed the prerequisites to run the Opendaylight Boron SDN controller, as indicated in HowTo-SDN document.
In the Guest VM, it is assumed that the reader has successfully installed Blackadder, following the instructions of the HowTo document.
In addition, in the Guest VM, the example have been tested with Mininet version 2.2.1 and Openvswitch version 2.4.0.

3. Deploy the Network Using Mininet

Assuming that the reader has already cloned in the Mininet VM the POINT repository to a folder `blackadder`, and has already built and installed it, follow the next steps.

In the Mininet VM, transfer the `demo_topo_deployment_tool.py` file from the *mininet-example* folder to the home directory.

```
$ cp ~/blackadder/sdn/mininet-example/demo_topo_deployment_tool.py
~/demo_topo_deployment_tool.py
```

In the host machine, build and execute the POINT Opendaylight application:

```
$ cd blackadder/sdn
$ mvn clean install
$ ./distribution/opendaylight-karaf/target/assembly/bin/karaf
> feature:install tm-sdn
> feature:install point-ui
```

Check that Opendaylight runs properly by checking the Opendaylight console:

```
> log:tail
```

In the Mininet VM, build the resource-manager application:

```
$ cd ~/blackadder/apps/icn-sdn/
$ make all
```

In the Mininet VM, run the custom python Mininet topology:

```
$ cd ~
$ sudo python demo_topo_deployment_tool.py
```

In Mininet console, ping all hosts:

```
> pingall
```

To enable the communication between the SDN controller in the Host Machine, and the 2 Mininet hosts (h1 and h2) in the Guest VM, add a route via the appropriate interface and IP address of your setup:

```
$ sudo ip route add 192.168.1.0/24 via 192.168.56.102 dev vboxnet0
```

Replace 192.168.56.102 and vboxnet0 with the IP address of your VM and the host-only interface of Virtualbox. To test it from your host machine, `ping 192.168.1.2` and `ping 192.168.1.3` (Mininet hosts).

The next step is to configure the SDN controller with the appropriate TM parameters. To realize this, navigate with your web browser to http://localhost:8181/index.html, with username/password, admin/admin, and go to the YANG UI tab at http://localhost:8181/index.html#/yangui/index. In the API tab, expand the `bootstrapping > operations` menu and click the `configureTm` submenu. In the presented form, insert the following parameters:

```
"tmIp": "192.168.1.2",
"tmPort": "12345",
"tmOpenflowId": "host:00:00:00:00:00:01",
```

```
"tmAttachedSwitchId": "openflow:1",
"tmNodeId": "00000001",
"tmLid": "1",
"tmInternalLid": "0",
"tmSourceNodeConnector": "openflow:1:1",
"tablesOptionOn": "false",
"groupsOptionOn": "false",
"deploymentToolOn": "true"
```

The meaning of all the json's fields in the POST message is explained in the Annex.

In your host machine, use the deployment file of https://github.com/point-h2020/point-2.0.0/tree/master/deployment/examples/odl_sample.cfg to deploy an ICN topology over SDN:

```
$ cd deployment/
$ make all
$ ./deploy -c examples/odl_sample.cfg -o --nokill --nostart
```

More information about the deployment file can be found in Chapter 4.

In your Mininet VM, check that ABM Openflow rules are configured:

```
> sudo ovs-ofctl dump-flows s1 -O OpenFlow13
> sudo ovs-ofctl dump-flows s2 -O OpenFlow13
> sudo ovs-ofctl dump-flows s3 -O OpenFlow13
```

You should see entries matching packets of IPv6 type and source or destination addresses.

Then execute the following commands in the Mininet console:

```
> h1 /usr/local/bin/click /tmp/00000001.conf > /tmp/h1_log 2>&1 &
> h2 /usr/local/bin/click /tmp/00000005.conf > /tmp/h2_log 2>&1 &
> h1 ./blackadder/TopologyManager/tm /tmp/topology.graphml >
/tmp/tm_log 2>&1 &
> h1 ./blackadder/examples/samples/publisher > /tmp/pub_log &
> h2 ./blackadder/examples/samples/subscriber
```

You should see correct published data in the Mininet console. In addition, if you dump-flows in switches s1 and s3 with the aforementioned commands, you should check increased packet number on rules with output ports 3 and 1 respectively, while dump-flows in switch s2 indicates no traffic.

To enable monitoring functions, navigate to the ICN-SDN tab, and click the "Enable Monitor" button. Link statistics will be periodically reported to the ICN-SDN application at Host 1. You can check the received monitoring information at the Mininet VM, `/tmp/server.log` file.

In addition, you may trigger a link failure by executing the following command in the Mininet console:

```
> link s1 s3 down
```

And you may revert the link's status back to normal with the following command:

```
> link s1 s3 up
```

Both deletion and addition events will be monitored in the SDN controller console, as well as the indicated `/tmp/server.log` file at the Mininet VM.

4. The traditional deployment tool used to deploy the network

When using ODL to control the SDN switching fabric all rules must be inserted into the switches via ODL and not manually by the deployment tool using bash scripts.

Changes to the Topology Configuration File

Therefore, the following changes marked in red must be made to the .cfg file which is presented to the deployment tool that comes with Blackadder:

```
SUDO = true;
OVERLAY_MODE = "mac";
ODL_ADDRESS = "10.0.0.254"; # IP address of where ODL runs
network = (
      nodes = (
      {#OVS
            testbed_ip = "10.0.0.100";
            running_mode = "user";
            role = [];
            label = "00000100";
            operating_system = "ovs";
            sdn_implementation = "bridges"; #you must select bridges!
            openflow_id = "openflow:8796757777851";# see below
            bridge = "br-icn";
```

The `openflow_id` value can be obtained from the OVS switch(es) directly. Simply issue the following command on each OVS switch in the POINT network:

```
$ DP_ID=`ovs-vsctl list bridge | grep datapath_id | awk '{split($0, tmp, "\""); print tmp[2]}'`
echo $((16#$DP_ID))
```

Also, it is important when creating the OVS bridge on the SDN switch to tell it to not run in standalone mode:

```
$ ovs-vsctl set-controller <BRIDGE> tcp:<IP_ODL>:6633 -- set-fail-mode <BRIDGE> secure
```

With `<BRIDGE>` being the name of the bridge and `<IP_ODL>` the IP address of ODL.

Parameters for ODL configuration

As already mentioned in Chapter 3, ODL must be manually configured. This can be achieved via the ODL REST API and is done by the deployment tool which pushes some node registration information into ODL (use the --nostart argument to not start any POINT software if desired). In order to obtain the tmAttachedSwitchId and the tmSourceNodeConnector, two curl commands must be issued, after the deployment tool has finished . First [1]:

```
$ curl -s -u admin:admin -H "Accept: Application/json" -H "Content-Type: application/json" -X GET http://localhost:8181/restconf/operational/registry:node-registry | jq
```

The response from ODL will look like this:

```
{
  "node-registry": {
      "node-registry-entry": [
      {
      "noneName": "host:08:00:27:15:0e:df",
      "noneId": "00000001"
      },
      {
      "noneName": "host:08:00:27:33:8e:76",
      "noneId": "00000254"
      },
      {
      "noneName": "openflow:8796757777851",
      "noneId": "00000100"
      }
      ]
  }
}
```

Identify the OVS switch using its NID (defined in the topology.cfg file) which has the TM as its neighbour, write down its “noneName” value and issue the second curl command to ODL:

[1] If the system complains that it cannot find jq simply install it via “apt install jq”

```
$ curl -s -u admin:admin -H "Accept: Application/json" -H "Content-Type:
application/json"                          -X                          GET
http://localhost:8181/restconf/operational/registry:node-connector-registr
y | jq
```

Find the same node (using the noneName value of the previous step) in the output (openflow:...:..) and identify the correct port on which the TM can be reached, e.g.:

```
{
  "node-connector-registry": {
      "node-connector-registry-entry": [
      {
      "noneConnectorName": "openflow:8796757777851:10",
      "srcNode": "openflow:8796757777851",
      "dstNode": "host:08:00:27:33:8e:76"
      },
      {
      "noneConnectorName": "openflow:8796757777851:9",
      "srcNode": "openflow:8796757777851",
      "dstNode": "host:08:00:27:15:0e:df"
      }
      ]
  }
}
```

NOTE: it is important to mention that the TM node can only have a single SDN switch as its direct neighbour. Moreover, the SDN switch must be one hop away. The current integration with ODL does not allow to have more than one SDN switch as a direct neighbour of to the TM.

In order to configure ODL and obtain the correct information for the deployment topology file (openflow_id), a HTTP POST must be sent to ODL. This HTTP request has information in its payload given in json. The command is:

```
$ curl -s -u admin:admin -H "Accept: Application/json" -H "Content-Type: application/json" -X POST
"http://localhost:8181/restconf/operations/bootstrapping:configureTm
{
        "input": {
        "tmIp": "10.0.0.254",
        "tmPort": "12345",
        "tmOpenflowId": "host:00:00:00:00:00:01",
        "tmAttachedSwitchId": "openflow:8796757777851",
        "tmNodeId": "00000254",
        "tmLid": "58",
        "tmInternalLid": "153",
        "tmSourceNodeConnector": "openflow:8796757777851:10",
```

```
        "tablesOptionOn": "false",
        "groupsOptionOn": "false",
        "deploymentToolOn": "true"
        }
}
"
```

Some notes on how some of these values can be obtained:

- The *noneConnectorName* equals the tmSourceNodeConnector
- The *srcNode* equals the openflow_id (topology.cfg) and tmAttachedSwitchId
- Also the *noneName* equals the tmAttachedSwitchId
- The *dstNode* field equals tmOpenFlowId.

The meaning of all the json's fields in the POST message and the way to obtain them is explained in detail in the Annex.

5. Monitoring Data from ODL

To support the report of data points from ODL a dedicated application has been written, called ICN-SDN which is located in `~/blackadder/apps/icn-sdn`. To compile the application please refer to this document or simply use the bash script make-all-clean.sh. Once compiled, ODL needs to receive the configuration to monitor its switches. This can be achieved by accessing the ODL GUI and opening the ICN-SDN tab on the left. There you you have to hit "Enable Monitoring" (note, no feedback is given!). Once this is done the ICN-SDN application can be started but requires some additional parameters:

```
sudo ./server <NID_TM> tmOpenflowId  tmAttachedSwitchId
```

Annex

The required input fields in the POST message for ODL configuration are the following:

- **tmIp:** The IP address of the machine hosting the TM and the ICN-SDN application.
- **tmPort:** The port in which the TM-SDN server listens to (the default is *12345*).
- **tmOpenflowId:** The identifier of the machine which hosts the TM as provided by the SDN controller, in the form of host:ab:cd:ef:gh:ij:kl. This value can be obtained with the curl command for the resource `restconf/operational/registry:node-registry`, field `dstNode` of the OVS switch which is the direct neighbour of the TM.
- **tmAttachedSwitchId:** The identifier of the attached SDN switch to the TM as provided by the SDN controller, in the form of *openflow:xyz.* This value can be obtained with the curl command for the resource `restconf/operational/registry:node-registry`, field `srcNode` of the OVS switch which is the direct neighbour of the TM.
- **tmNodeId:** The node ID of the TM.

- **tmLid:** requires the position of '1' in TM's Link ID (LID), starting from 0. *According to George, the default value is 1*. This value can be obtained by invoking the following command on the TM node:
  ```
  cat /tmp/<NID_TM>.conf | grep 0x86dd | \
  awk '{split($0, tmp, ","); s=index(tmp[5],"1"); print s}'
  ```
- **tmInternalLid:** The position of '1' in TM's Internal Link ID (iLID), starting from 0. *According to George, the default value is 0.* This value can be obtained by invoking the following command on the TM node:
  ```
  cat /tmp/<NID_TM>_TMFID.txt | awk '{s=index($0,"1"); print s}'
  ```
- **tmSourceNodeConnector:** The port identifier of the attached SDN switch interface to the TM, in the form of *openflow:xyz:i.* This value can be obtained with the curl command for the resource `restconf/operational/registry:node-registry`, field `noneConnectorName` of the OVS switch which is the direct neighbour of the TM.
- **tablesOptionOn:** Indicates whether the arbitrary bitmask rules will be configured in multiple tables (true) or single table (false). You should select *false*, as tables will not be supported in the trials.
- **groupsOptionOn:** Indicates whether fast failover groups will be configured per interface (true) or not (false). Select *false*.
- **deploymentToolOn:** Indicates whether bootstrapping will occur using the Blackadder deployment tool (true) or through other deployment options (false). Select *true*.