Laboratory 9: Software Defined Networking II

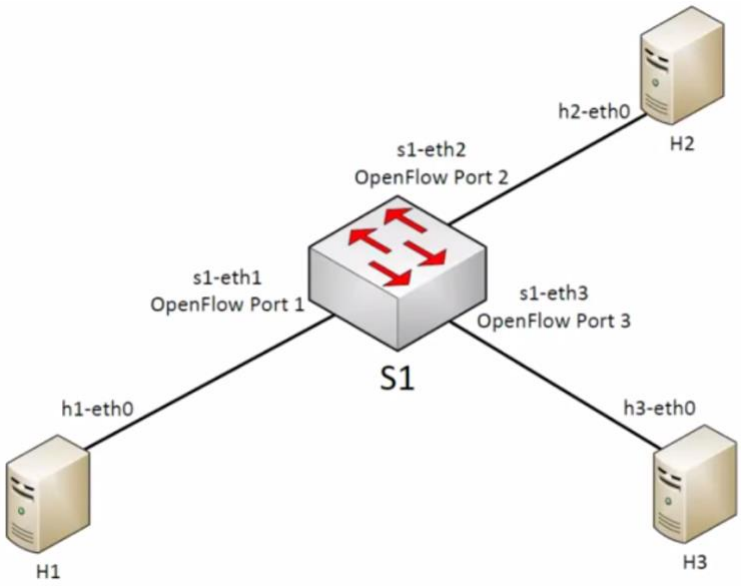
Objective:

- To learn the layer 2 and 3 SDN switch configuration
- To learn how to use OpenDaylight controller
- To learn the basic of North Bound Interface (NBI) API

Create topology:

After SSH to mininet VM, let's create a simple topology with 3 hosts and 1 switch as shown in the figure below,

```
$ sudo mn --topo=single,3 --mac --controller=none
```




```
mininet> dump                    // Use this command to dump hosts, switch & ports' IDs

mininet> net        // Use this command to view links (not connectivity!)
```

**pingall should not work at this stage since no flows can be found in the switch!

```
mininet> sh ovs-ofctl add-flow s1 action=normal     // This will insert normal L2 switch forwarding, do pingall to check its working!

mininet> sh ovs-ofctl dump-flows s1          // To check all flows of the switch

mininet> sh ovs-ofctl del-flows s1          // Use this command to delete all flows of a switch
```

SUMMARY OF BASIC COMMANDS

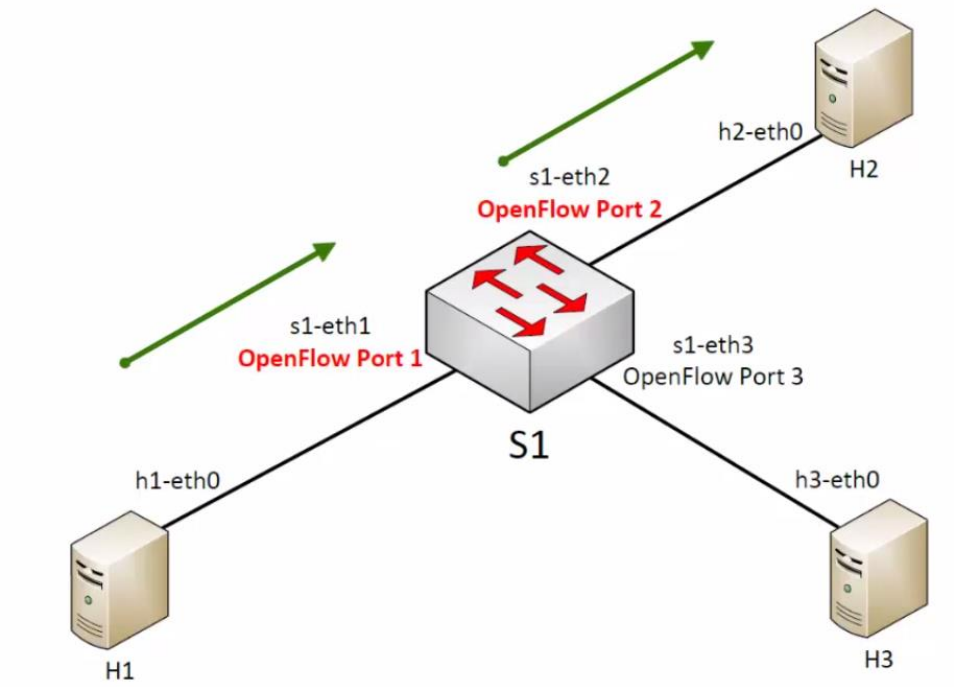
Command	Action
`sh ovs-ofctl dump-flows s1`	view all flows of the switch
`sh ovs-ofctl del-flows s1`	to delete all flows of a switch
`dump`	to dump hosts, switch & ports' IDs

Add flow based on Layer 1 matching:

Consider, traffic from,

Input port 1 -> Output to port 2 &

Input port 2 -> Output to port 1




```
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=1,actions=output:2
```

```
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=2,actions=output:1

mininet> h1 ping -c2 h2

mininet> h3 ping -c2 h1   // This doesn't work because no flow is installed when inport = 3

mininet> sh ovs-ofctl add-flow s1 actions=drop
```

IMPORTANT: When you do not specify the priority, it assigns priority=32768. Priority number ranges from 0 to 65,535.

```
mininet> sh ovs-ofctl dump-flows s1        // use this command to check the flows installed
```

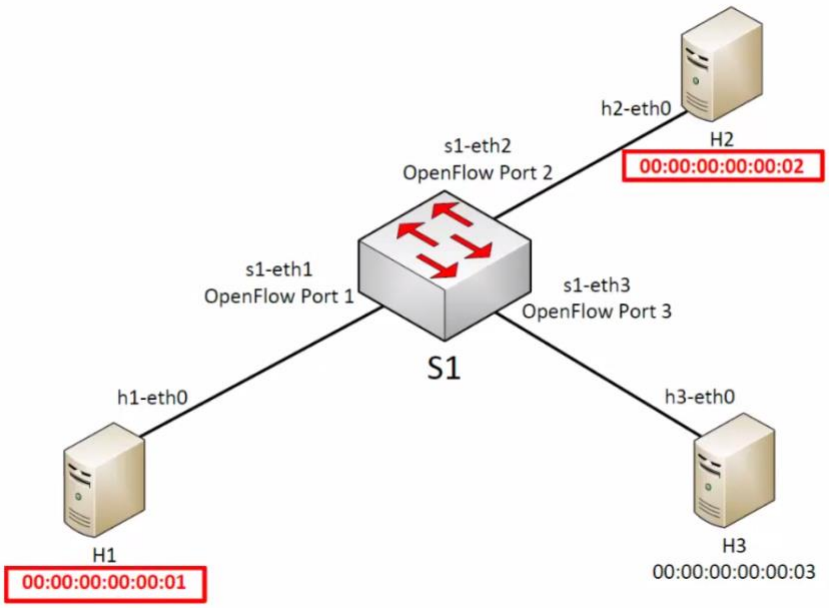
How to remove the wildcard flow entry only?

```
mininet> sh ovs-ofctl del-flows s1 --strict        // This deletes only the wildcard flow entry
```

Now let us delete all flows and

Add flow based on Layer 2 (MAC address) matching:

```
 mininet> sh ovs-ofctl del-flows s1
```



**Note: The following commands are all single line commands!

```
mininet> sh ovs-ofctl add-flow s1
dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,actions=output:2

mininet> sh ovs-ofctl add-flow s1
dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,actions=output:1
```

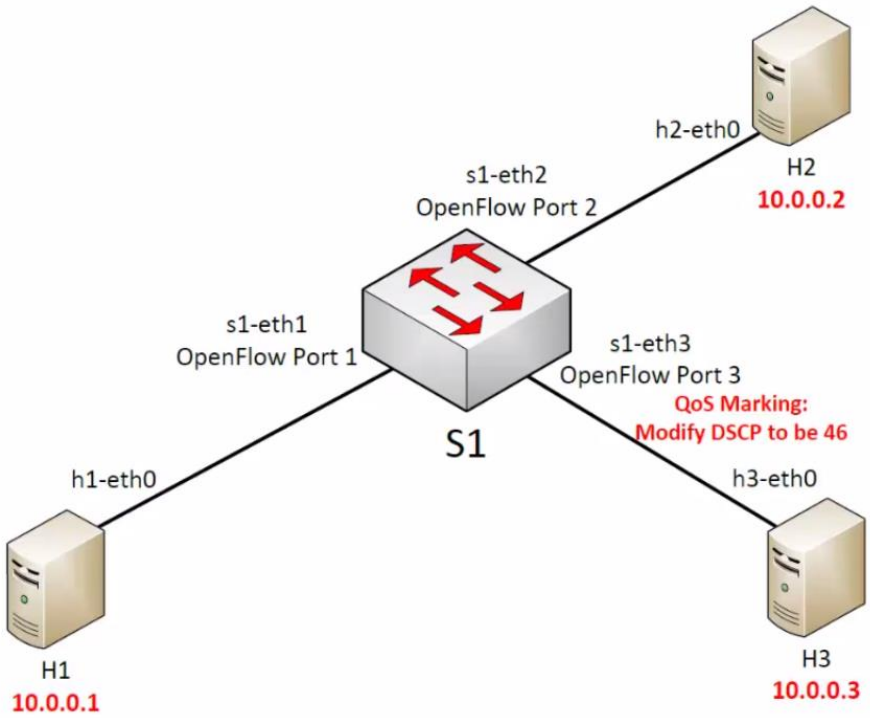
```
mininet> sh ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,actions=flood

mininet> h1 ping h2

mininet> h2 ping h3
```

These should work whereas h1 ping h3 should not!

Now let us delete all flows and

Add flow based on Layer 3 (IP address) matching:

```
mininet> sh ovs-ofctl del-flows s1
```




```
mininet> sh ovs-ofctl add-flow s1
priority=500,dl_type=0x800,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=normal

mininet> sh ovs-ofctl add-flow s1
priority=800,ip,nw_src=10.0.0.3,actions=mod_nw_tos:184,normal

mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.1,actions=output:1

mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.2,actions=output:2

mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.3,actions=output:3
```

The switch now working as a layer 3 switch with a static routing table. Note that in the above task we did not use a SDN controller and we wrote flow entries manually.

Let's close our topology and clean mininet to be ready for next task.

```
mininet> exit

$ sudo mn -c
```

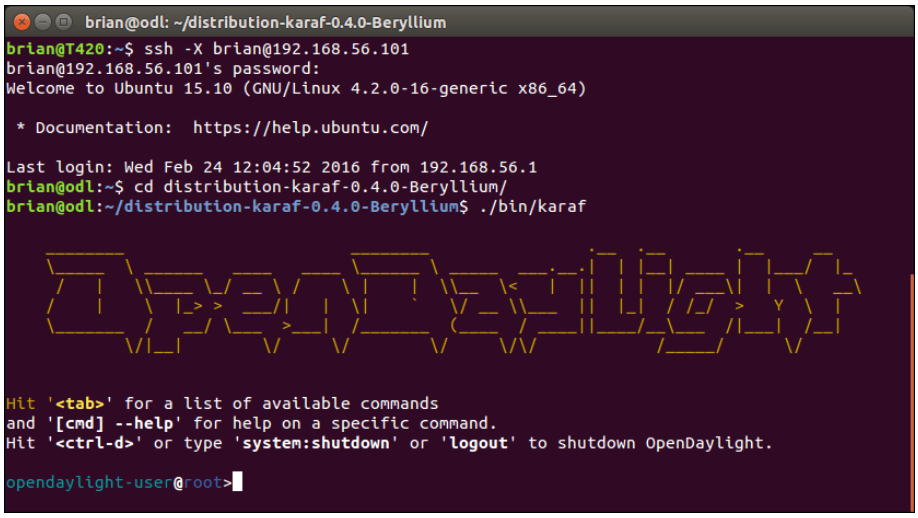
OpenDaylight Controller:

One of Mininet's key features is that it makes it very easy to create a complete virtual network including hosts, switches, links and OpenFlow controllers. By default, Mininet runs Open vSwitch in OpenFlow mode, which requires an OpenFlow controller.

As you probably know, Mininet comes with built-in Controller() classes to support several controllers, including the OpenFlow reference controller (`controller`), Open vSwitch's `ovs-controller`, and the now-deprecated NOX Classic.

You can easily choose which one you want when you invoke the `mn` command:

```
$ sudo mn --controller ref

$ sudo mn --controller ovsc

$ sudo mn --controller NOX,pyswitch
```



We are going to add another controller: OpenDaylight. We have already installed it on our mininet VM. Open a new terminal and use SSH to connect mininet. Channge directory to the OpenDaylight directory and run the `karaf` command inside the package distribution folder.

```
$ cd /opt/opendaylight/

$ ./bin/karaf
```

Now the OpenDaylight controller is running similar to the screenshot you see. Leave it as it is and return to the other SSH terminal. Later you can turn off OpenDaylight controller using `system:shutdown`.

In other SSH terminal enter:

```
$ sudo mn --topo linear,3 --mac --controller=remote,ip=192.168.0.140,port=6633 --switch ovs,protocols=OpenFlow13
```

The OpenDaylight Graphical User Interface

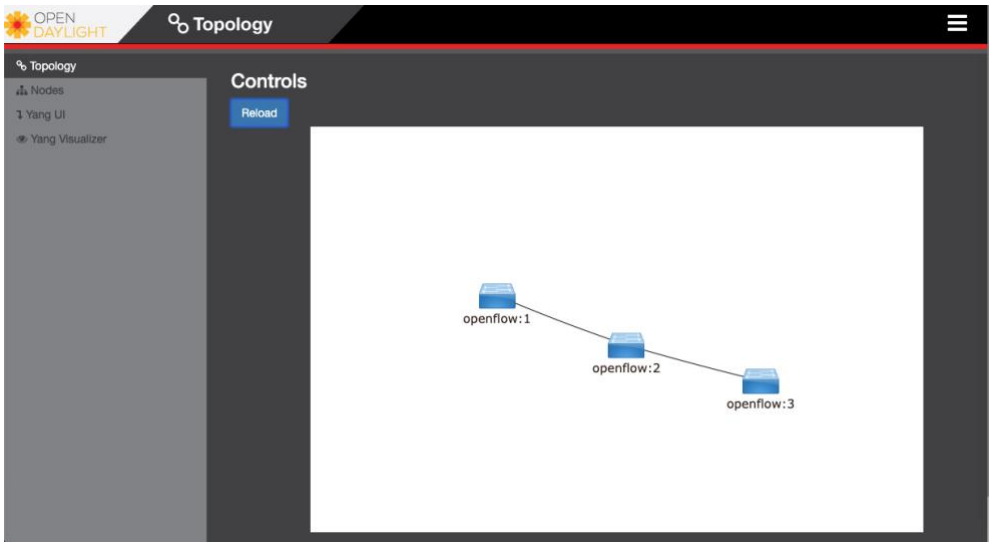
Open a browser on your host system and enter the URL of the OpenDaylight User Interface (DLUX UI). It is running on the mininet/OpenDaylight VM so the IP address is 192.168.0.140 and the port, defined by the application, is 8181:

So the URL is: http://192.168.0.140:8181/index.html .

The default username and password are both *admin*.

Topology

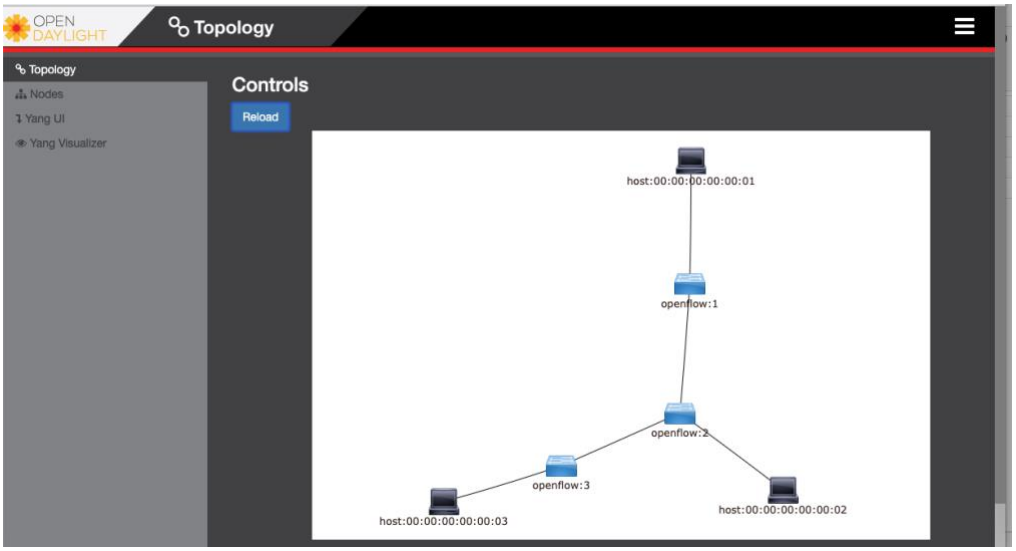
Now we see the network topology in the OpenDaylight controller's *topology* tab. You should see a similar topology to the following screenshot. At this moment, we can only see switches since our controller has not leaned the network yet.



Now use *pingall* to help controller to learn topology.

```
Mininet> pingall
```

Now in the browser click on the reload button to refresh the topology. You can see the connected hosts as well.

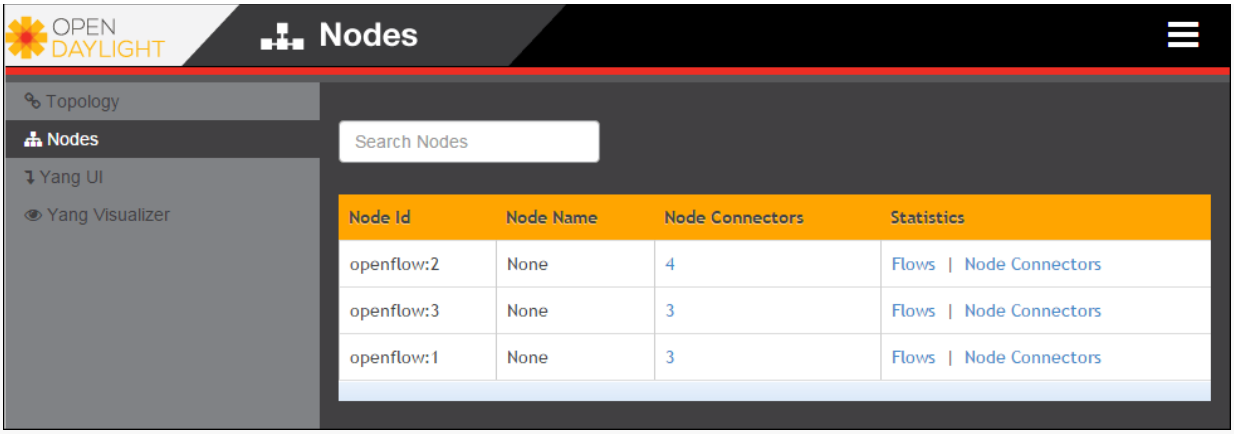


You can see the network that is emulated by the Mininet network emulator. You may test OpenDaylight functionality by building different network topologies in Mininet with different attributes, and by using OpenDaylight to run experiments on the emulated

network. For example, you may break links between switches in Mininet to test how the network responds to faults.

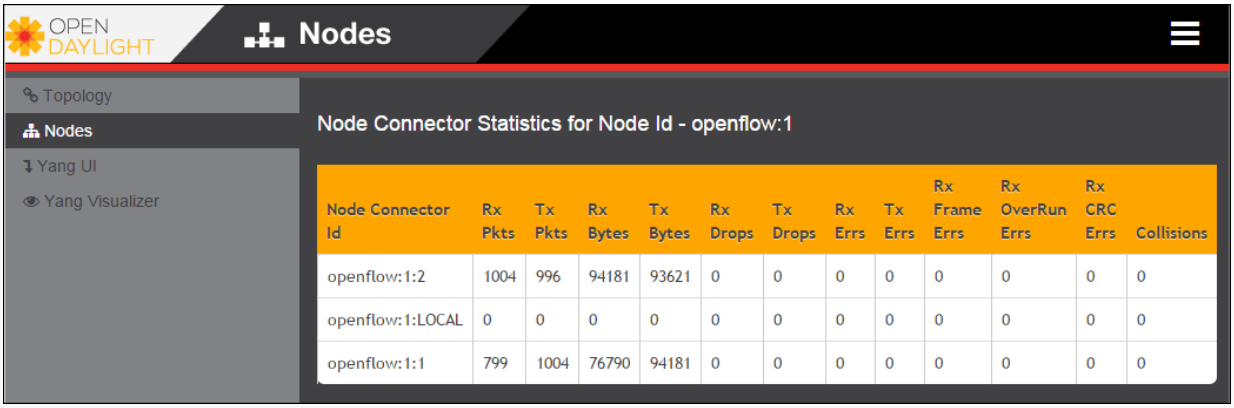
Nodes

Click on the *Nodes* tab to see information about each switch in the network:



List of nodes

Click on the *Node Connectors* link in each row to see information about each port on the switch:



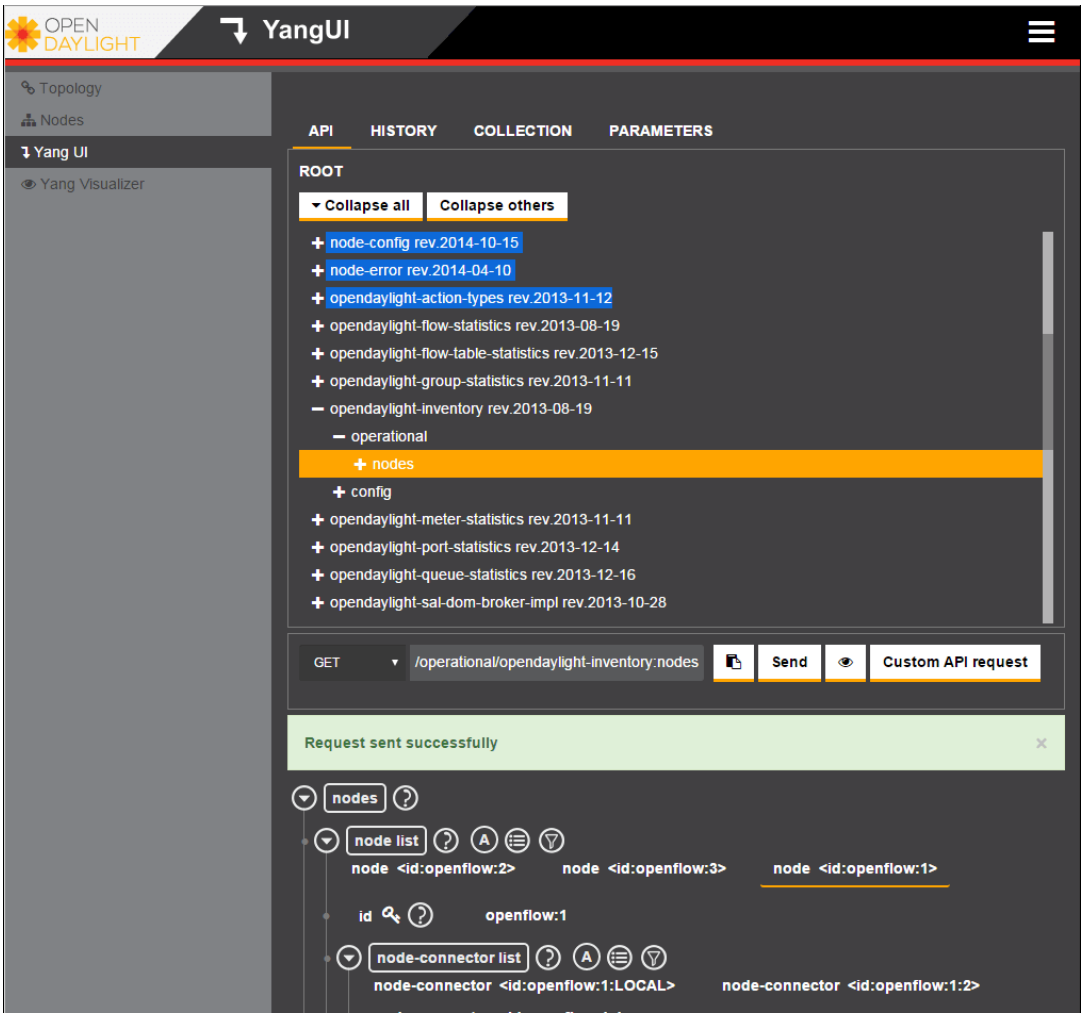
Interfaces

Yang UI

The OpenDaylight Yang UI is a graphical REST client for building and sending REST requests to the OpenDaylight data store. We can use the Yang UI to get information from the data store, or to build REST commands to modify information in the data store — changing network configurations.

Click on the *Yang UI* tab. Then click on the *Expand all* button to see all available APIs. Not all of them will work because we did not install all features. One API that will work is

the *Inventory* API. Click on it, then navigate down to the *nodes* attribute and click on the *Send* button to send the *GET* API method to the controller.



Scroll down to see all the inventory information about the network: nodes, ports, statistics, etc. Click on the switches and interfaces to see the details of each.

You can copy the GET API and past it in your browser address bar to see the response's content. You also can get an individual node details:

http://192.168.0.140:8181/restconf/operational/opendaylight-inventory:nodes/node/openflow:1

where openflow:1 is the node ID for first switch. We also can change configuration, set flows and so on.

Refrences:

https://docs.opendaylight.org/en/stable-neon/user-guide/openflow-plugin-project-user-guide.html

http://www.brianlinkletter.com/using-the-opendaylight-sdn-controller-with-the-mininet-network-emulator/