# OpenDaylight Project PCMM Southbound Plugin

## History

|  |  |  |
| --- | --- | --- |
| Thomas Kee | 2013/10/17 | Add delete SF from CLI command and add PICA8 inband scripts. Add perl script documentation. |
| Thomas Kee | 2013/10/15 | Demo details |
| Thomas Kee | 2013/10/10 | Initial living information for integration |

# Goal

* Demonstration at SCTE Show on 21 October 2013 to preview the possibility of using an SDN controller to manage CMTS’s as network elements.
* Seed development of an southbound plugin to enable CTMS support and garner industry support with CableLabs sponsorship.
* Possibly incubate the demo source as an open source OpenDayLight project (but not for Hydrogen release)

## Documentation

[PacketCable™ Specification Multimedia Specification PKT-SP-MM-I05-091029](http://www.cablelabs.com/specifications/archives/PKT-SP-MM-I05-091029.pdf)

[OpenDaylight Project Developer Wiki](https://wiki.opendaylight.org/view/Main_Page)

## Framework

OpenDaylight is an open source project under the Linux Foundation with the mutual goal of furthering the adoption and innovation of Software Defined Networking (SDN) through the creation of a common industry supported framework.

The following diagram highlights the OpenDaylight Framework and how the PacketCable components fit in to the solution.



## Overview

OpenDaylight supports the OSGi framework and bidirectional REST for the northbound API between the Controller Platform and Network Applications.  The OSGi framework is used where applications will be running in the same address space as the controller while REST APIs will be used when applications will be running in different address spaces, or on different machines, as the controller.

### PacketCable SDN Application Logic

This component is introduced at the Network Applications Orchestration and Services layer. The component is proposed as an application responsible for the workflow logic that represents the use cases of interest: L2VPN, Buffer Management, Service Flow QoS Management, and general configuration automation.

### Flow Programmer

This standard component exists at the Controller Platform layer and is responsible for programming flows at network elements that register support for flow management.  It adheres to this [FlowProgrammer Restful API](https://jenkins.opendaylight.org/controller/job/controller-merge/ws/opendaylight/northbound/flowprogrammer/target/site/wsdocs/resource_FlowProgrammerNorthbound.html).

### PacketCable Manager

This component is introduced at the Controller Platform layer.  This component manages the PCMM/DOCSIS specific attributes that are applied to an existing Service Flow, configuration, and addition and removal of CMTS elements.  It should export a PacketCable Manager RESTful API as part of ODL architecture.

### PCMM/COPS

This component is introduced at the Southbound Interfaces & Protocol Plugins layer.  This component is responsible for the PCMM/COPS/PDP functionality required to service requests from PacketCable Manager and FlowManager.  Requests are transposed into PCMM Gate Control messages and transmitted via COPS to the CMTS. This plugin adheres to the PCMM/COPS/PDP functionality defined in the [CableLabs specification.](http://www.cablelabs.com/specifications/archives/PKT-SP-MM-I05-091029.pdf)

## Demonstration

Enables a minimal functionality southbound plugin to create and delete best effort service flows that show cases a high throughput video flow and a low throughput video flow control via the OpenDayLight controller by an SDN application utilizing the northbound restful API.



## Locating Demo Assets on Controller

cd /home/user/SDN-OpenFlow-Lab/

drwxr-xr-x  5 user user     4096 Sep 30 12:19 packetcable

drwxr-xr-x  4 user user     4096 Sep 30 10:28 protocol\_plugins

-rwxr-xr-x  1 user user       53 Oct 15 18:39 run\_odl\_sdnapp\_menu.sh

-rwxrwxrwx  1 user user      148 Oct  1 11:51 run\_odl.sh

drwxr-xr-x  5 user user     4096 Sep 30 12:19 sdn

## Starting and Running OpenDayLight

The following sequence of commands on either of the ofc-laptop Linux computers will start the controller environment. After the system boots, open a shell and perform the following commands:

1. Turn off the Network Manager so as to not have it interfere with the statically configured interfaces.

/etc/init.d/network-manager stop

1. Start the interfaces

ifup –a

1. Change direct to the SDN work directory

cd SDN\_Openflow\_Lab

1. Start the OpenDayLight

./run\_odl.sh

Wait for the osgi console to startup and then point a browser at

http://localhost:8080/

From the osgi console, verify the plugin is active

osgi> ss | grep packetcable

110        ACTIVE org.opendaylight.controller.protocol\_plugins.packetcable\_0.4.0.SNAPSHOT

true

osgi> dm 110

[110] org.opendaylight.controller.protocol\_plugins.packetcable

  org.opendaylight.controller.sal.flowprogrammer.IPluginInFlowProgrammerService(protocolPluginType=PC) registered

  org.opendaylight.controller.sal.utils.INodeFactory(protocolName=PC,protocolPluginType=PC) registered

  org.opendaylight.controller.sal.utils.INodeConnectorFactory(protocolName=PC,protocolPluginType=PC) registered

  org.opendaylight.controller.sal.inventory.IPluginInInventoryService(scope=Global,protocolPluginType=PC) registered

    org.opendaylight.controller.sal.inventory.IPluginOutInventoryService (scope=Global) service required available

  org.opendaylight.controller.sal.reader.IPluginInReadService(protocolPluginType=PC,containerName=default) registered

  org.opendaylight.controller.sal.inventory.IPluginInInventoryService(protocolPluginType=PC,containerName=default) registered

osgi> s | grep packetcable

110        file:/home/mininet/controller/opendaylight/distribution/opendaylight/target/distribution.opendaylight-0.1.1-SNAPSHOT-osgipackage/opendaylight/plugins/protocol\_plugins.packetcable-0.4.0-SNAPSHOT.jar

  ACTIVE org.opendaylight.controller.protocol\_plugins.packetcable\_0.4.0.SNAPSHOT [110]

true

osgi>

## Demo SDN Application

### ./run\_odl\_sdnapp\_menu.sh

### restapi.py

This python script is the demo SDN application and drives and exercises various interfaces offered by the ODL northbound API. Most of the functions where written for mininet’s default topology and may need some changes if used with another topology. To start the application

~/SDN-OpenFlow-Lab/run\_odl\_sdnapp\_menu.sh

or directly

~/SDN-OpenFlow-Lab/sdn/python $ python restapi.py

There are some specific notes about the currently implemented functionality with regard to defining flows.

1. The flow actions are meaningless and no actions are implemented. They can exist as long as they are compliant flow programmer constructs.
2. The name can be changed, but do not use symbols such as dashes or underscores. This in not a PCMM restriction, but a current flow programmer issue.
3. The priority MUST be a value in the range of 64 to 128.
4. A priority of 100 will instruct the PCMM plugin to apply the high bandwidth QoS parameters.
5. Flow Programmer SAL dispatches flows to plugins designated by node type.



The script has the following two flows used for the PCMM demonstration.

flow\_pcmm\_1 = {

"actions": [

"DROP"

],

"etherType": "0x800",

"installInHw": "true",

"name": "flowpcmmHighBW",

"node": {

"id": "51966",

"type": "PC"

},

"nwDst": "10.32.4.208",

"nwSrc": "10.32.154.2",

"priority": "100",

"tpDst": "8081"

}

flow\_pcmm\_2 = {

"actions": [

"OUTPUT=2"

],

"etherType": "0x800",

"installInHw": "true",

"name": "flowpcmmlowBW",

"node": {

"id": "51966",

"type": "PC"

},

"nwDst": "10.32.4.208",

"nwSrc": "10.32.154.2",

"priority": "64",

"tpDst": "8081"

}

[ Insert ODL screenshot with flow details ]

To use the application add a PCMM flow 1 (12) to start with the High bandwidth and toggle (16) from the high and low bandwidth service flow for bad video stream.

Alternatively, start with the Low bandwidth add a PCMM flow 2 (14) and remove a PCMM flow (15) and change to the High bandwidth by adding a PCMM flow 1 (12) service flow for good video stream.

------------------------------

MENU

------------------------------

1. Add Flow 1

2. Add Flow 2

3. Add Several Flows

4. Remove Flow 1

5. Remove Flow 2

6. Remove All Flows

7. Toggle Flow

8. List Flow Stats

9. List Topology

10. List Flows

11. List Ports

12. Add PCMM Flow 1

13. Remove PCMM Flow 1

14. Add PCMM Flow 2

15. Remove PCMM Flow 2

16. Toggle PCCM Flows

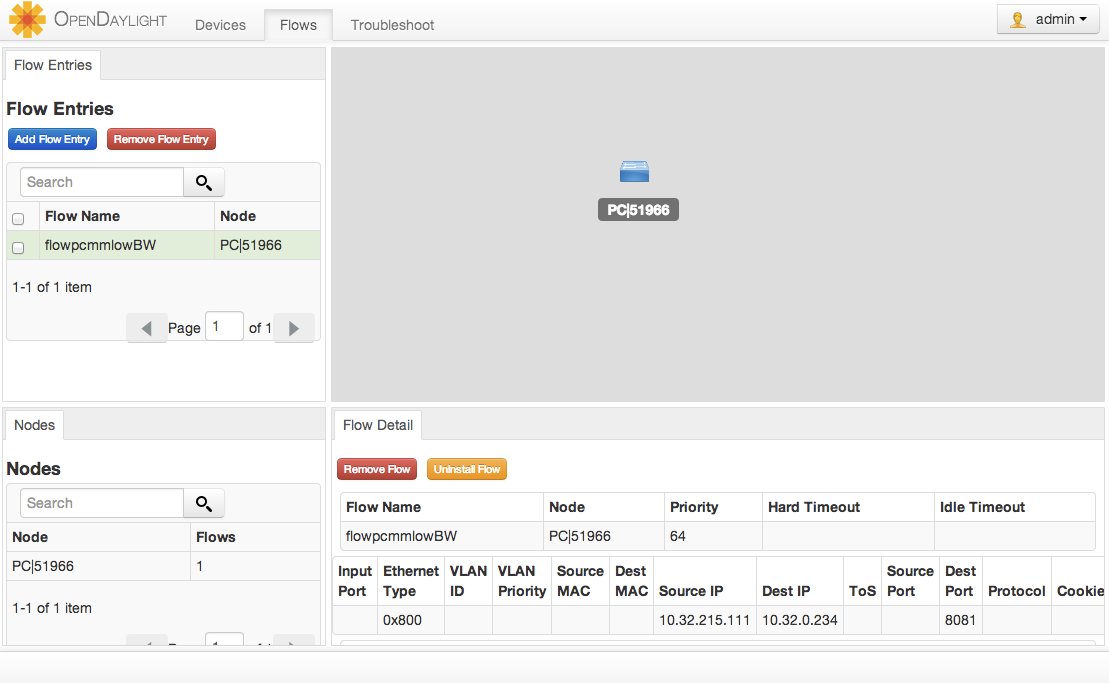
q. Quit

Enter selection:

## OpenDayLight Web UI

Enter the URL of OpenDayLight in your we browser <http://127.0.0.1:8080/> and use the credentials of admin/admin to enter the application.

The ‘fake” CMTS show up in the UI as a “PC” network element type with the id of 51966 in the topology area of the screen. When you select the Flows view either the flowpcmmlowBW or flowpcmmhighBW flow name should appear as a Flow entry the details of which are displayed in the Flow Details frame. If something goes wrong the flow can be removed by selecting the Remove Flow button.

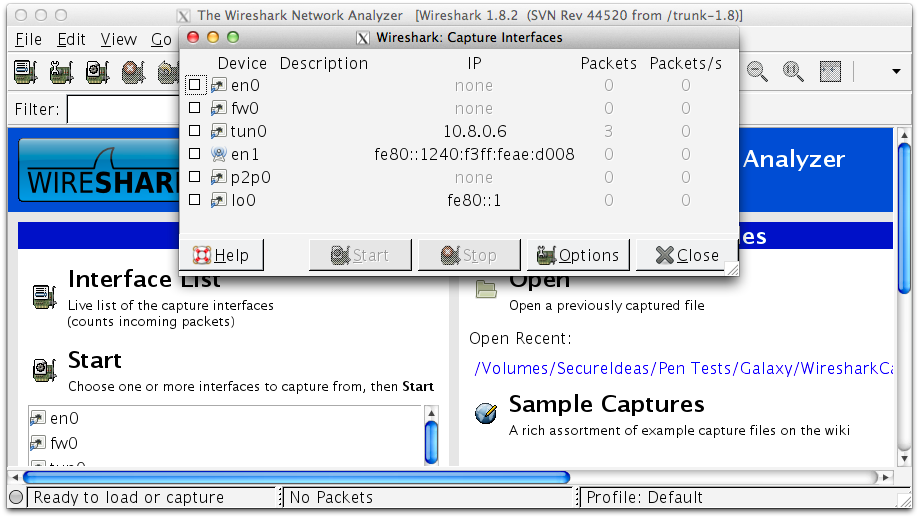


## Debugging with Wireshark

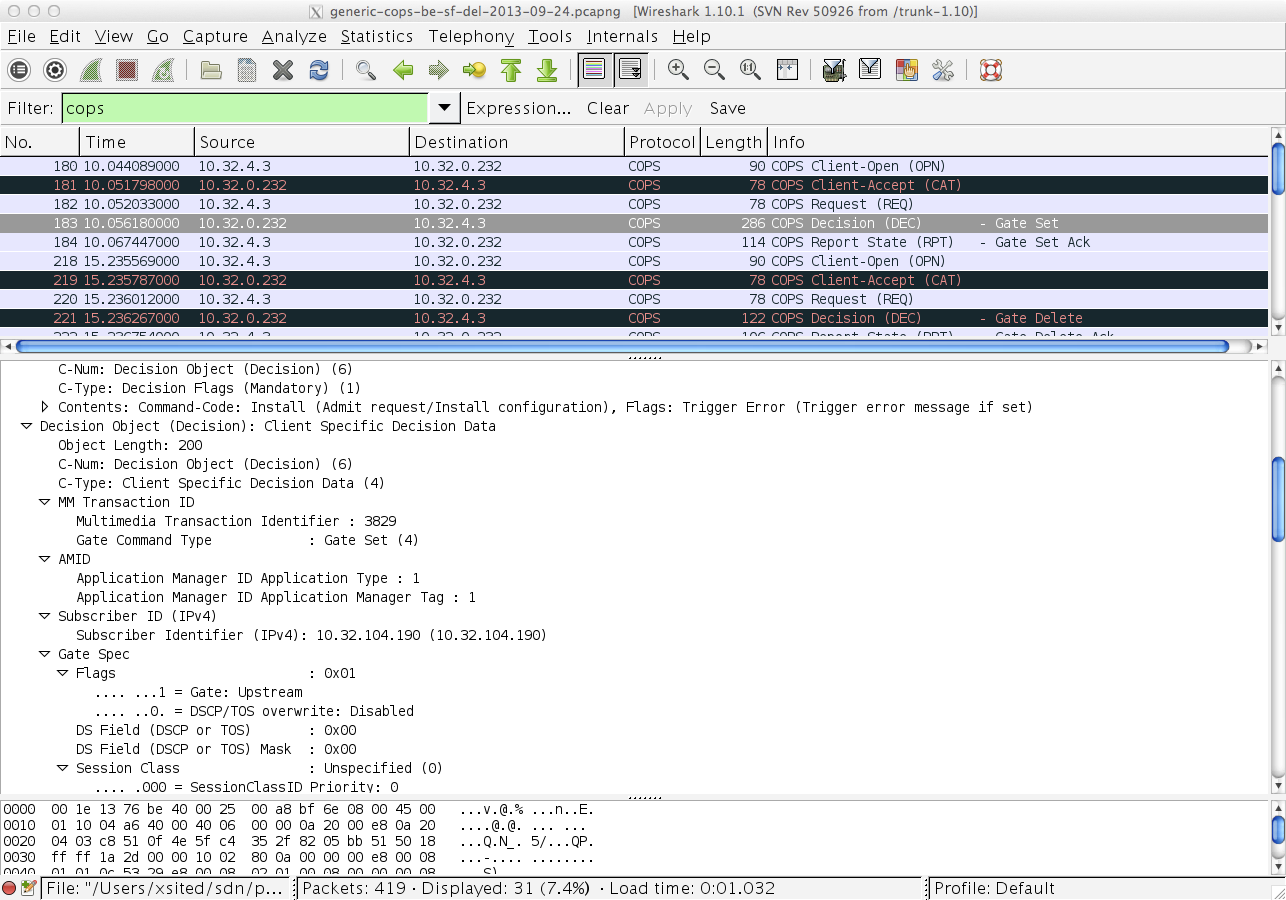
To start wireshark with privileges issue the following command:

sudo wireshark &

1. First select the interface to monitor.



1. Use the Filter to only display COPS messages by applying “cops” in the filter field.



## Working with the CMTS

### Log Into CMTS

telnet 10.32.4.3

Username: cisco

Password: Cisco

### Find the Cable Modem

10k2-DSG#show cable modem

                                                                                  D

MAC Address    IP Address      I/F           MAC           Prim RxPwr  Timing Num I

                                             State         Sid  (dBmv) Offset CPE P

0010.188a.faf6 0.0.0.0         C8/0/0/U0     offline       1    0.00   1482   0   N

74ae.7600.01f3 10.32.115.150   C8/0/10/U0    online        1    -0.50  1431   0   Y

0010.188a.fad8 10.32.115.142   C8/0/10/UB    w-online      2    -0.50  1507   1   Y

000e.0900.00dd 10.32.115.143   C8/0/10/UB    w-online      3    1.00   1677   0   Y

e86d.5271.304f 10.32.115.168   C8/0/10/UB    w-online      6    -0.50  1419   1   Y

### Show PCMM Plugin Connection

10k2-DSG#show packetcabl ?

  cms     Gate Controllers connected to this PacketCable client

  event   Event message server information

  gate    PacketCable gate information

  global  PacketCable global information

10k2-DSG#show packetcable cms

GC-Addr        GC-Port  Client-Addr    COPS-handle  Version PSID Key PDD-Cfg

10k2-DSG#show packetcable cms

GC-Addr        GC-Port  Client-Addr    COPS-handle  Version PSID Key PDD-Cfg

10.32.0.240    54238    10.32.15.3     0x4B9C8150/1    4.0   0    0   0

### Show COPS Messages

debug cops details

### Use CM Mac Address to List Service Flows

10k2-DSG#show cable modem

                                                                                  D

MAC Address    IP Address      I/F           MAC           Prim RxPwr  Timing Num I

                                             State         Sid  (dBmv) Offset CPE P

0010.188a.faf6 ---             C8/0/0/UB     w-online      1    0.50   1480   1   N

74ae.7600.01f3 10.32.115.150   C8/0/10/U0    online        1    -0.50  1431   0   Y

0010.188a.fad8 10.32.115.142   C8/0/10/UB    w-online      2    -0.50  1507   1   Y

000e.0900.00dd 10.32.115.143   C8/0/10/UB    w-online      3    0.00   1677   0   Y

e86d.5271.304f 10.32.115.168   C8/0/10/UB    w-online      6    -0.50  1419   1   Y

10k2-DSG#show cable modem 000e.0900.00dd service-flow

SUMMARY:

MAC Address    IP Address      Host          MAC           Prim  Num Primary    DS

                               Interface     State         Sid   CPE Downstream RfId

000e.0900.00dd 10.32.115.143   C8/0/10/UB    w-online      3     0   Mo8/0/2:1  2353

Sfid  Dir Curr  Sid   Sched  Prio MaxSusRate  MaxBrst     MinRsvRate  Throughput

          State       Type

23    US  act   3     BE     0    0           3044        0           39

30    US  act   16    BE     0    500000      3044        0           0

24    DS  act   N/A   N/A    0    0           3044        0           17

UPSTREAM SERVICE FLOW DETAIL:

SFID  SID   Requests   Polls      Grants     Delayed    Dropped    Packets

                                             Grants     Grants

23    3     784        0          784        0          0          784

30    16    0          0          0          0          0          0

DOWNSTREAM SERVICE FLOW DETAIL:

SFID  RP\_SFID QID    Flg Policer               Scheduler             FrwdIF

                         Xmits      Drops      Xmits      Drops

24    33019   131550     0          0          777        0          Wi8/0/2:2

Flags Legend:

$: Low Latency Queue (aggregated)

~: CIR Queue

### Deleting a PCMM Gate Message from the CMTS

10k2-DSG#test cable dsd 000e.0900.00dd 30

## Using Test.java (did not do this)

This is a Java test harness to exercise the PCMM/COPS functionality for the demo without OpenDayLight.

java -classpath .:src/main/java/pcmm.jar:src/main/java/jcops.jar:/usr/share/junit/junit.jar Test

Test - starting Client

MENU

=============

1. Add Flow 1

2. Add Flow 2

3. Toggle Flow

4. Remove Flow 1

5. Remove Flow 2

0. Quit

Enter Choice:

## Using pcmm.pl

This is a standalone tool to test PCMM without OpenDayLight.

/home/user/SDN-OpenFlow-Lab/sdn/perl/

./pcmm.pl

Quick update and documentation for this baseline PCMM tool with three techniques for the desired affect for the demonstration.

Main Menu:

1. Add Service Flow 1

2. Add Service Flow 2

3. Remove Service Flow 1

4. Remove Service Flow 2

5. Toggle Classifier (flip class)

6. Toggle Classifier (sf del/add)

7. Toggle Classifier (raise/lower be2 priority)

8. Print Stuff

9. Quit

Each scenario moves the video traffic from an open to constrained BE service flow.  Video stream is on 8081.

1. Flip classifier IP protocol port between 1369 and 8081.

1 -> 5 -> 5 -> 5 …. 3 -> 4 (or 9)

2. Delete the service flow and create the alternative desired flow (only one exists at anyone time.)

1 -> 6 -> 6 ->6  …. 3 -> 4 (or 9)

3. Raise and lower the priority of the classifier on BE2.  BE1 is 69 and BE2 is toggled between 64 and 128.

1 -> 7 -> 7 ->7 …. 3 -> 4 (or 9)

## Setting up the Development Environment

### Check Out the Controller and PCMM Plugin

sudo apt-get install maven

git clone https://git.opendaylight.org/gerrit/p/controller.git

# XXX - CHANGE THIS

git clone https://github.com/xsited/protocol\_plugins.git

In the parent is pom.xml for the entire packetcable projects and this will build a working controller distribution based on the controller + packetcable modules and leave a bundle in packetcable/target/protocol\_plugins.packetcable-0.4.0-SNAPSHOT.jar

### Building OpenDayLight

JDK 1.7+ and Maven 3+ are required:

From the toplevel issue the following instructions to build the controller:

cd controller/opendaylight/distribution/opendaylight

mvn clean install

or if you want to avoid SNAPSHOT checking use:

mvn clean install -nsu

// mvn clean install -DskipTests

From the toplevel issue the following instructions to build the packetcable SB plugin:

cd protocol\_plugins/packetcable

## mvn clean install

Upon successful completion of a build install and run from the toplevel:

cp protocol\_plugins/packetcable/target/protocol\_plugins.packetcable-0.4.0-SNAPSHOT.jar controller/opendaylight/distribution/opendaylight/target/distribution.opendaylight-0.1.1-SNAPSHOT-osgipackage/opendaylight/plugins/protocol\_plugins.packetcable-0.4.0-SNAPSHOT.jar

cd controller/opendaylight/distribution/opendaylight/target/distribution.opendaylight-0.1.1-SNAPSHOT-osgipackage/opendaylight/

export JAVA\_HOME=/usr

./run.sh

## Adding Pica8 to Demo Topology

### Add a New Bank of Port to a New Bridge

#!/bin/sh

SW\_IP=172.16.1.3

ovs-vsctl --db=tcp:$SW\_IP:6633 add-br br2 -- set bridge br2 datapath\_type=pica8

for i in 25 26 27 28 29 30 31 32 33 34 35 36 ; do

   ovs-vsctl --db=tcp:$SW\_IP:6633 add-port br2 ge-1/1/$i -- set interface ge-1/1/$i type=pica8

done

### Configuring Pica8 (Open vSwitch) for Inband Communication to the Controller

Configure an IP address for br2 and configure following CLI which will enable the inband communication between controller (connected with ge-1/1/1 or others) and switch.

Find the MAC address of the switch.

cat /pica/bin/system/pica\_switch\_mac

60:EB:69:D2:9C:D7

Use the MAC address and ovs-ofctl to direct controller traffic over br2.

ifconfig br2 10.32.4.201/24

ovs-ofctl del-flows br2

ovs-ofctl add-flow br2 priority=65300,dl\_src=60:EB:69:D2:9C:D7,actions=all

ovs-ofctl add-flow br2 priority=65300,dl\_dst=60:EB:69:D2:9C:D7,actions=local

ovs-ofctl add-flow br2 priority=65300,dl\_dst=FF:FF:FF:FF:FF:FF,actions=all,local

ovs-ofctl dump-flows br2

OFPST\_FLOW reply (OF1.2) (xid=0x2):

 cookie=0x0, duration=37.289s, table=0, n\_packets=0, n\_bytes=0, priority=65300,dl\_src=60:eb:69:d2:9c:d7 actions=AL

 cookie=0x0, duration=23.792s, table=0, n\_packets=0, n\_bytes=0, priority=65300,dl\_dst=60:eb:69:d2:9c:d7 actions=LL

 cookie=0x0, duration=8.944s, table=0, n\_packets=0, n\_bytes=0, priority=65300,dl\_dst=ff:ff:ff:ff:ff:ff actions=ALL

ping 10.32.4.240

PING 10.32.4.240 (10.32.4.240) 56(84) bytes of data.

From 10.32.4.201 icmp\_seq=1 Destination Host Unreachable

From 10.32.4.201 icmp\_seq=2 Destination Host Unreachable

ifconfig

eth0      Link encap:Ethernet  HWaddr 08:9E:01:93:5D:73

          inet addr:172.16.1.3  Bcast:172.16.1.255  Mask:255.255.255.0

          inet6 addr: fe80::a9e:1ff:fe93:5d73/64 Scope:Link

          UP BROADCAST MULTICAST  MTU:1500  Metric:1

          RX packets:0 errors:0 dropped:0 overruns:0 frame:0

          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:1000

          RX bytes:0 (0.0 B)  TX bytes:238 (238.0 B)

          Base address:0x4000

lo        Link encap:Local Loopback

          inet addr:127.0.0.1  Mask:255.0.0.0

          inet6 addr: ::1/128 Scope:Host

          UP LOOPBACK RUNNING  MTU:16436  Metric:1

          RX packets:149 errors:0 dropped:0 overruns:0 frame:0

          TX packets:149 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:0

          RX bytes:28061 (27.4 KiB)  TX bytes:28061 (27.4 KiB)

ifconfig -a

eth0      Link encap:Ethernet  HWaddr 08:9E:01:93:5D:73

          inet addr:172.16.1.3  Bcast:172.16.1.255  Mask:255.255.255.0

          inet6 addr: fe80::a9e:1ff:fe93:5d73/64 Scope:Link

          UP BROADCAST MULTICAST  MTU:1500  Metric:1

          RX packets:0 errors:0 dropped:0 overruns:0 frame:0

          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:1000

          RX bytes:0 (0.0 B)  TX bytes:238 (238.0 B)

          Base address:0x4000

eth1      Link encap:Ethernet  HWaddr 08:9E:01:93:5D:74

          BROADCAST MULTICAST  MTU:1500  Metric:1

          RX packets:0 errors:0 dropped:0 overruns:0 frame:0

          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:1000

          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

          Base address:0x6000

lo        Link encap:Local Loopback

          inet addr:127.0.0.1  Mask:255.0.0.0

          inet6 addr: ::1/128 Scope:Host

          UP LOOPBACK RUNNING  MTU:16436  Metric:1

          RX packets:155 errors:0 dropped:0 overruns:0 frame:0

          TX packets:155 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:0

          RX bytes:28531 (27.8 KiB)  TX bytes:28531 (27.8 KiB)

sit0      Link encap:IPv6-in-IPv4

          NOARP  MTU:1480  Metric:1

          RX packets:0 errors:0 dropped:0 overruns:0 frame:0

          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:0

          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

**Add Scripts to Pica8 to connect to Inband Controller**

This creates new bridge br2 with the ports in bank 3 of the switch for the NSI side and sets the controller to 10.32.4.240:6633

root@sw1#cat make\_br.sh

#!/bin/sh

SW\_IP=172.16.1.3

ovs-vsctl --db=tcp:$SW\_IP:6633 show

ovs-vsctl --db=tcp:$SW\_IP:6633 del-br br2

ovs-vsctl --db=tcp:$SW\_IP:6633 add-br br2 -- set bridge br2 datapath\_type=pica8

for i in 25 26 27 28 29 30 31 32 33 34 35 36 ; do

    ovs-vsctl --db=tcp:$SW\_IP:6633 add-port br2 ge-1/1/$i -- set interface ge-1/1/$i type=pica8

done

ovs-vsctl --db=tcp:$SW\_IP:6633 set-controller tcp:10.32.4.240:6633

This controller routing out the br2 interface on the NSI side. Use ifconfig eth0 to obtain the MAC address to use in the following script.

root@sw1#cat inband\_controller.sh

#MAC=`cat /pica/bin/system/pica\_switch\_mac`

MAC="08:9E:01:93:5E:AE"

echo $MAC

ifconfig br2 10.32.4.201/24

ovs-ofctl del-flows br2

ovs-ofctl add-flow br2 priority=65300,dl\_src=$MAC,actions=all

ovs-ofctl add-flow br2 priority=65300,dl\_dst=$MAC,actions=local

ovs-ofctl add-flow br2 priority=65300,dl\_dst=FF:FF:FF:FF:FF:FF,actions=all,local

ovs-ofctl dump-flows br2

**Useful Links**

[OpenDaylight Main Website](http://www.opendaylight.org/)

[OpenDaylight Wiki](https://wiki.opendaylight.org/view/Main_Page)

[YANG Models Wiki](https://wiki.opendaylight.org/view/YANG_Tools:Available_Models)

[YANG Modeling in the Southbound Plugin Development (Model-Driven SAL)](https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:Southbound_Plugin_Development_Guide)

**Meetings**

[Technical Steering Committee](https://wiki.opendaylight.org/view/TSC:Main)

[Technical Work Stream](https://wiki.opendaylight.org/view/Tech_Work_Stream:Main)