Openmul PRISM SDN Routing - HOWTO

Document Revision History

Rev. No.	Date	Revised By	Comments
1.0	30.10.2013	openmul development team	Doc to describe how-to use mul controller
1.1	07.07.2014	openmul development team	Updates

Contents

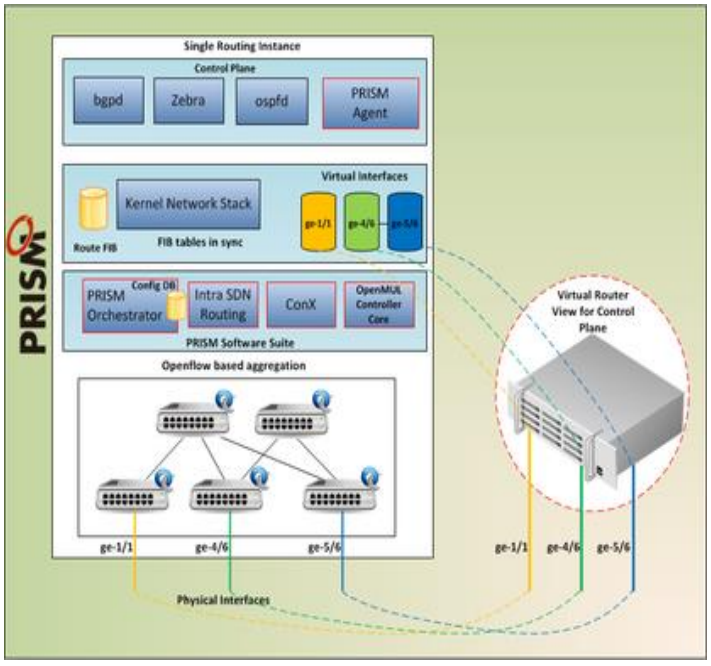
1 Building/Installing MuL controller

Please refer to OpenMUL-HOWTO guide for instructions on installing MuL controller

2 What is PRISM routing?

PRISM stands for ***Perfect Routing protocol integration with SDN using MUL controller***. As the name suggests PRISM's main objective is to run any legacy control plane using SDN/Openflow interface on any set of switches. The main feature which sets PRISM apart is the fact it runs one control plane instance irrespective of number of switches. Also, switches need not be directly connected and can have intermediate non-SDN L2 or L3 switches.

The end-goal is to provide game-changing flexibility of creating on-demand routing and forwarding entities with open interfaces for use across data-centers, enterprise or campus networks. Users will have the flexibility to operate normal Linux networking commands without the need to dig into the number of devices, software or hardware make of devices as well as proprietary CLIs of legacy devices



- White-box SDN based dynamic routing out of box

- Programmable network interface cluster with high visibility

- Transforms white-box or Openflow switch to powerful routers

- Supports aggregation of N-devices (stacking) under a single management entity

- Seamless integration with any legacy devices and protocols

- Support for Quagga, BIRD routing suites etc

- Openstack plugin support (As a data-center gateway)

2.1 Running PRISM

2.1.1 Mapping PRISM virtual interfaces to Switch Physical Ports

PRISM framework creates the virtual interfaces for each physical port which would be connected as routing interfaces to the legacy network. This mapping is stored in a file located at /etc/mul/prism/vif.conf. Contents of the file are as <DPID>|<port>|<interface-name> . After creating this file and starting the prism components these interfaces will appear as normal interfaces in Linux and be configured with normal linux networking tools.

2.1.2 Running PRISM using inbuilt script

A utility script is provided to run PRISM components which are located at:

$(mul-code-dir)/mul.sh

```
$ cd (mul-code-dir)/

# Options available
./mul.sh start prism
./mul.sh stop
```

The various options taken by this script are described below:

Option Name	Description
start prism	Runs mul-core, topology, prism framework, path connector service and webserver component.
stop	Stop all running MuL components.

Before running MuL for the very first time, one needs to run the following commands:

```
$ cd (mul-code-dir)/
$ ./mul.sh init
```

2.1.3 Running PRISM component wise

The following sections will explain the procedure to follow if one is interested to run each of MuL's components separately or in verbose mode.

2.1.3.1 Run director/core

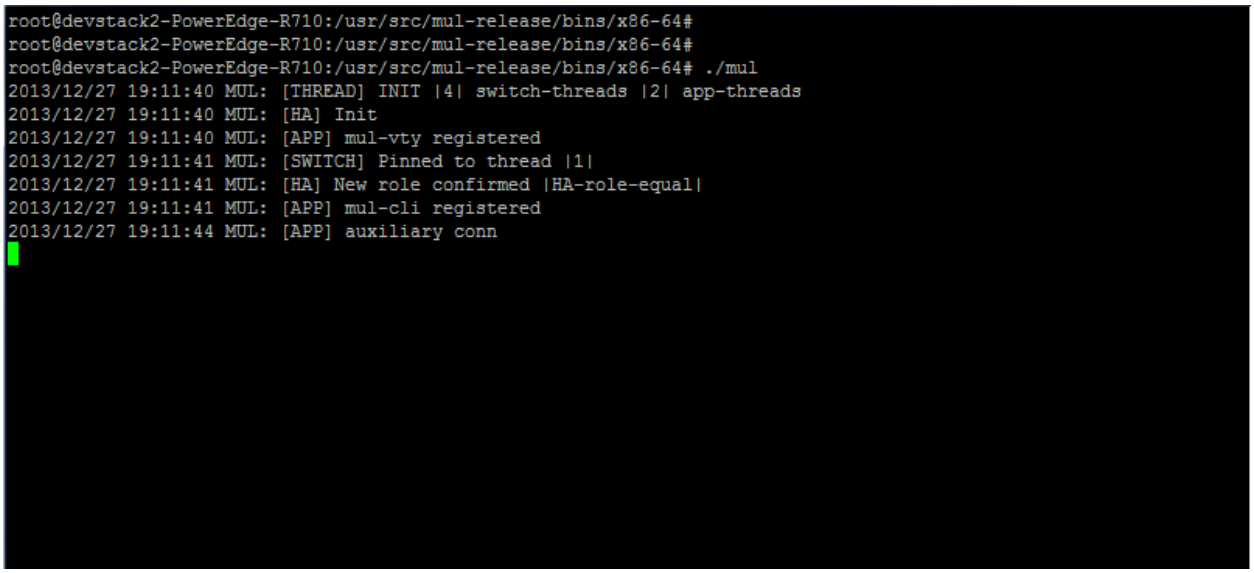
```
$ cd (mul-code-dir)/
$ sudo ./mul –d

OR (for verbose mode)

$ sudo ./mul
```

Possible options to use -

```
mul options:
-d              : Daemon Mode
-S <num>        : Number of switch handler threads
-A <num>        : Number of app handler threads
-P <port>       : Port Number for incoming switch connection
-H <peer>       : Peer IP address for HA
-n              : Don't install default flows in switch
-p              : Enable OF packet dump for all switches
-s              : Enable ssl for all switch connections
-l <level>      : Set syslog levels 0:debug, 1:err(default) 2:warning
-x              : Verify switch-ca cert. Only applicable along with -s option
-h              : Help
```

For running mul with ssl, kindly refer to MUL-HOWTO-SSL in the docs/.

Example:



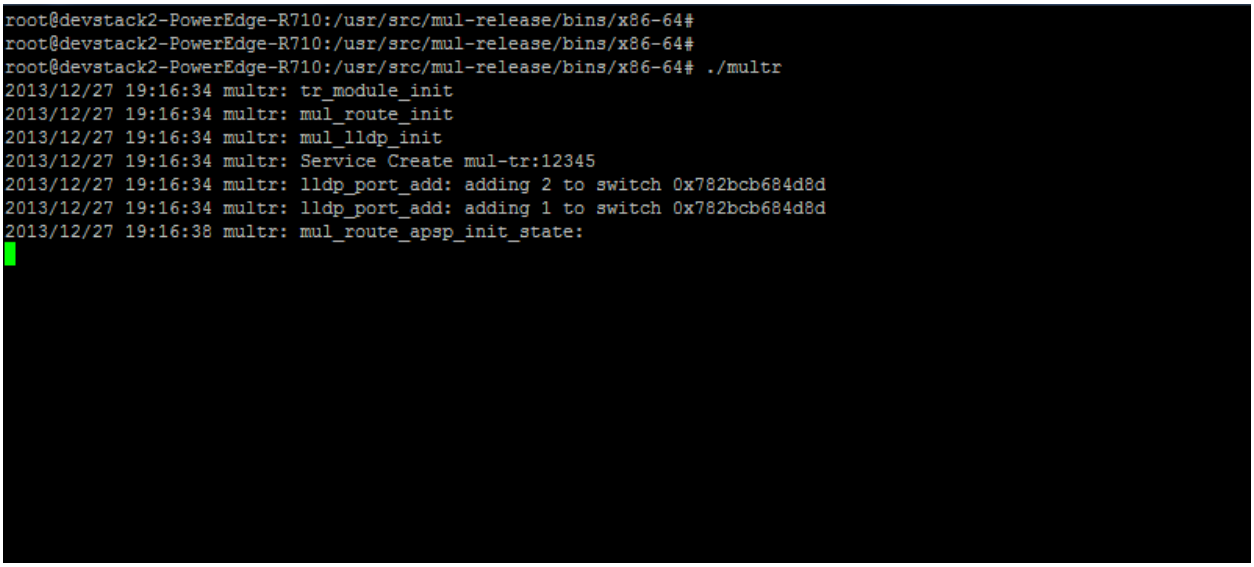
2.1.3.2 Running Mul topology/routing service

```
$ cd (mul-code-dir)/services/loadable/topo_routing
$ sudo ./multr -d

OR (for verbose)

$ sudo ./multr
```

Possible options to use -

```
    -d                  : Daemon mode (Run in background)
    -s <server-ip>      : Controller server IP address to connect (localhost by default)
```

Example:



2.1.3.3 Running Applications: cli

This application provides a unified cli which hooks in with each of the MuL controller components thereby providing a single point of management of all MuL controller components:

```
$ cd (mul-code-dir)/application/cli
$ sudo ./mulcli –V 10000  -d          ## Access to fabric cli on telnet port 10000

OR (for verbose mode)

$ sudo ./mulcli –V 10000
```

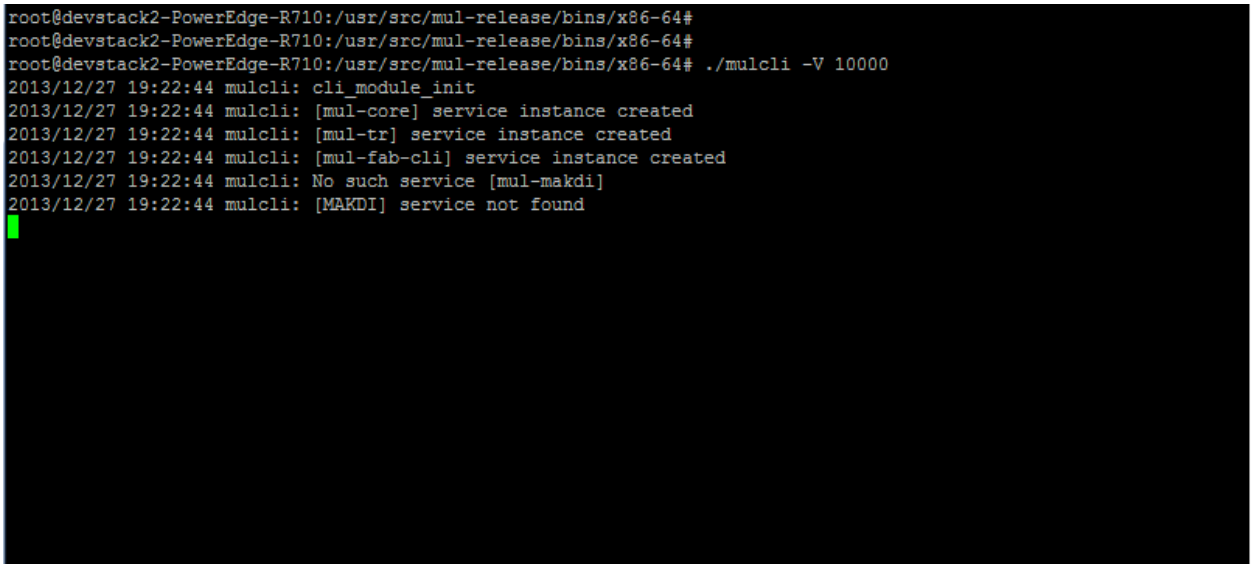
Possible options to use -

```
    -d                  : Daemon mode (Run in background)
    -s <server-ip>      : Controller server ip address to connect (localhost by default)
    -V <telnet-port>    : cli telnet port number
    -H <peer>           : Peer IP address for HA
```

NOTE – This application will auto-detect which of the mul's application are present.

More comprehensive cli command guide can be found in docs/ folder.

Example:



2.1.3.4 Running Path Connector Service: ConX

ConX service takes care of managing the internal fabric path. It can maintain various active and backup paths for the internal path thereby achieving highly reliable and lossless fast path recovery.

```
$ cd (mul-code-dir)/services/loadable/conx
$ sudo ./mulconx -d

OR (for verbose mode)

$ sudo ./mulconx
```

Example:

2.1.3.5 Running Applications: PRISM

PRISM (Perfect Routing Integration of SDN using MuL) takes a modular approach to solve the problem of the lack of simple and centralized management plane all the while making the network highly agile. In a nutshell, it lets a single control plane routing instance run as is across a set of Openflow devices bringing them under a centralized management plane.

```
$ cd (mul-code-dir)/application/prism/app
$ sudo ./prismapp –d

OR (for verbose mode)

$ sudo ./prismapp
```

```
$ cd (mul-code-dir)/application/prism/agent
$ sudo ./prismagent –d

OR (for verbose mode)

$ sudo ./prismagent
```

Example:

```
root@KulServer-2:/home/jyyang/nikhil/concave/prism/mul-kc-code/application/prism/agent# ./prismagent -n
```

```
root@KulServer-2:/home/jyyang/nikhil/concave/prism/mul-kc-code/application/prism/app# ./prismapp
```

2.1.4 Configuring (mapped) virtual Interfaces

2.1.4.1 Manual Configuration

```
root@KulServer-2:/home/jyyang/nikhil/concave/prism/mul-kc-code# ifconfig pr-s1-eth0 1.1.1.1/24 up
root@KulServer-2:/home/jyyang/nikhil/concave/prism/mul-kc-code# ifconfig pr-s1-eth0
pr-s1-eth0 Link encap:Ethernet  HWaddr fa:9b:5a:2d:34:ba
          inet addr:1.1.1.1  Bcast:1.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::f45f:e2ff:fe63:a748/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:82 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:13539 (13.5 KB)

root@KulServer-2:/home/jyyang/nikhil/concave/prism/mul-kc-code#
```

2.1.4.2 Configuration using Control plane stack (eg Quagga)

```
root@KulServer-2:/home/jyyang/nikhil/concave/prism/mul-kc-code# telnet localhost zebra
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.20.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
Zebra> en
Password:
Zebra# con t
Zebra(config)# interface pr-s1-eth0
Zebra(config-if)# ip address 1.1.1.1/24
Zebra(config-if)# no shut
Zebra(config-if)# no shutdown
Zebra(config-if)# exit
Zebra(config)# exit
```

```
Zebra# show interface pr-s1-eth0
Interface pr-s1-eth0 is up, line protocol detection is disabled
  index 7658 metric 1 mtu 1500
  flags: <UP,BROADCAST,RUNNING,MULTICAST>
  HWaddr: fa:9b:5a:2d:34:ba
  inet 1.1.1.1/24 broadcast 1.1.1.255
  inet6 fe80::f45f:e2ff:fe63:a748/64
Zebra#
```

2.1.5 Dynamic routing using quagga protocol suite

PRISM supports all quagga configurations for OSPF, BGP, RIP, IS-IS on PRISM interfaces. For more information on quagga suite, visit the quagga project - http://www.nongnu.org/quagga/