OpenMUL Controller - 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
1.2	17.01.2015	openmul development team	How to use build.sh script

Contents

1 Installing MuL controller

1.1 Getting MuL Code

The latest MuL code can be downloaded using git as follows:

```
$ git clone git@github.com:openmul/openmul.git
```

For getting source code based on permissive license (eg Apache or BSD), please contact support@openmul.org

1.2 Building MuL Code from source

1. Run all-in-one build script for Ubuntu/CentOS/Fedora for any architecture :

```
$ cd openmul
$./build.sh
```

Note- It will just build the necessary binaries not install it.

1.3 Components of openmul

Before running Mul, one to understand that MuL has many components, each running as a separate application. MuL is basically a suite of co-operative applications comprising of:

1) Mul director/core
    a. Major component of mul
    b. Handles all low level connections and does openflow processing.
    c. Location : $(mul-code)/mul

2) Mul services
    a. These provide basic infra services built on top of mul director
    b. Currently available :
        i. Topology discovery service
        ii. Path finding service
        iii. Path Connector Service
    c. Location : $(mul-code)/services/loadable/topo_routing/ (i & ii)
        $(mul-code)/services/loadable/conx/ (iii)
3) MuL system apps
    a. System apps are built using a common api provided by mul-director and mul-services

b. These are hardly aware of Openflow and hence designed to work across different openflow versions provided switches support common requirement of these apps
c. Currently available :
    i. PRISM
        - PRISM - Perfect Routing Integration of SDN using MuL
        - Enables IP peering between legacy island and SDN island
        - Integrated with Quagga/ZebOS
        - Less dependent on Switch Hardware
        - Provides single point of management
        - Possible to support multiple islands connectivity
        - Supports multiple controller clustering

    ii. L2 learning app
        - A simple learning application
        - Location : $(mul-code/application/l2switch

    iii. Fabric app
        - Enables multi-tenant aware end to end P2P connections between end-hosts across a mesh of Openflow switches
        - Data center oriented application
        - Based on P+V model
        - Location : $(mul-code/application/fabric

    iv. Makdi app – Openflow Service chaining application
        - Enables service chaining across an Openflow fabric
        - Enables integration with external L4-L7 device by recognizing metadata eg vlan for a particular service
        - Follows ETSI MANO standard where possible

    v. CLI app
        - This provides a common cli based provisioning tool for all MuL components.
        - Location : $(mul-code/application/cli
    vi. NBAPI webserver
        - This provides RESTful Api's for mul controller
        - Location: application/nbapi

1.4 Running MUL

1.4.1 Running Mul using inbuilt script

A utility script is provided to run various combinations of MuL components in:

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

Options available:

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

# Options available
./mul.sh start standalone
./mul.sh start l2switch
./mul.sh start fabric
./mul.sh start prism
./mul.sh stop
```

The various options taken by this script are described below:

Option Name	Description
start standalone	Runs mul-core, cli and webserver component. Should suffice if users want to configure everything from cli/rest-api.
start l2switch	Runs mul-core, l2switch, cli and webserver component.
start fabric	Runs mul-core, topology, fabric and webserver component.
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
```

1.4.2 Running Mul 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.

1.4.2.1 Run director/core

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

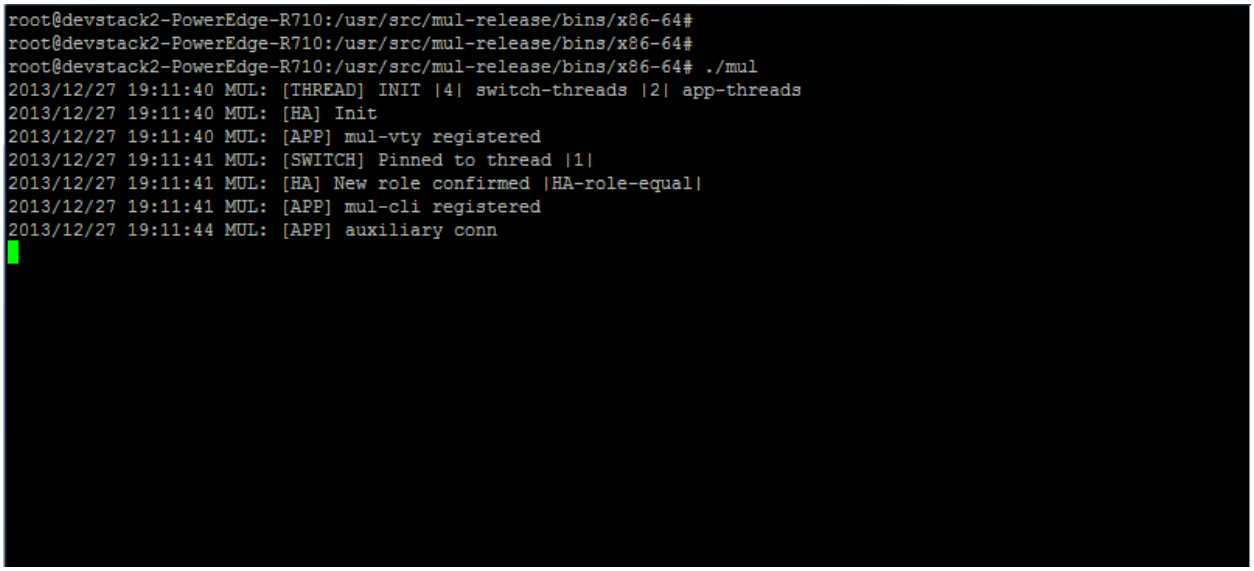
```
$ sudo ./mul –d

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:



1.4.2.2 Running Mul topology/routing service

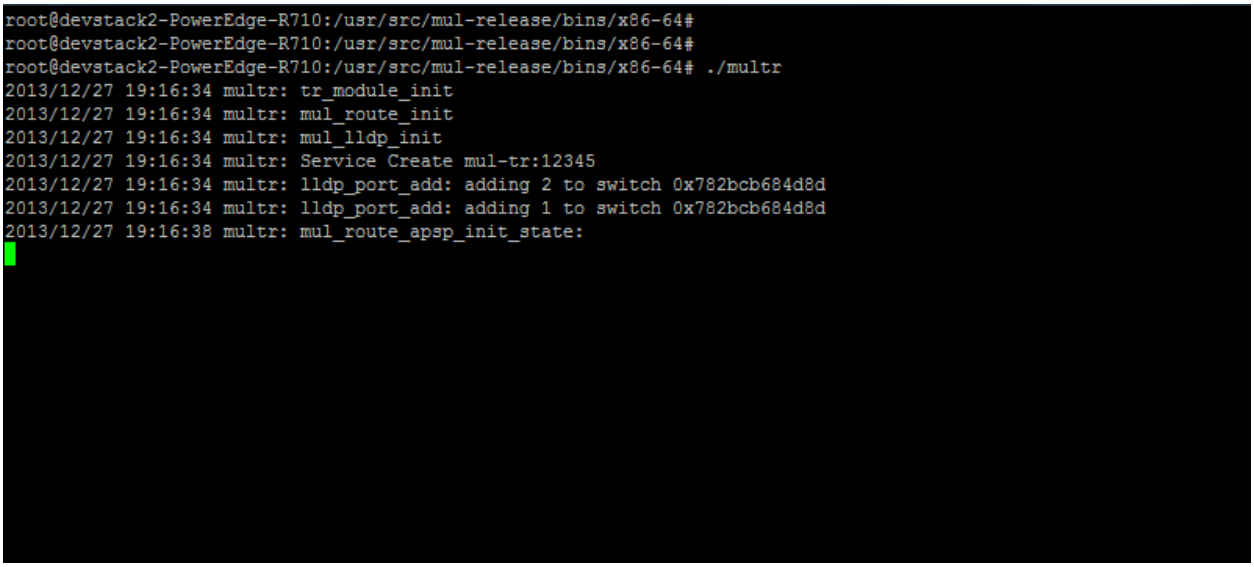
```
$ cd (mul-code-dir)/services/loadable/topo_routing
$ sudo ./multr -d
```

```
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:



1.4.2.3 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
```

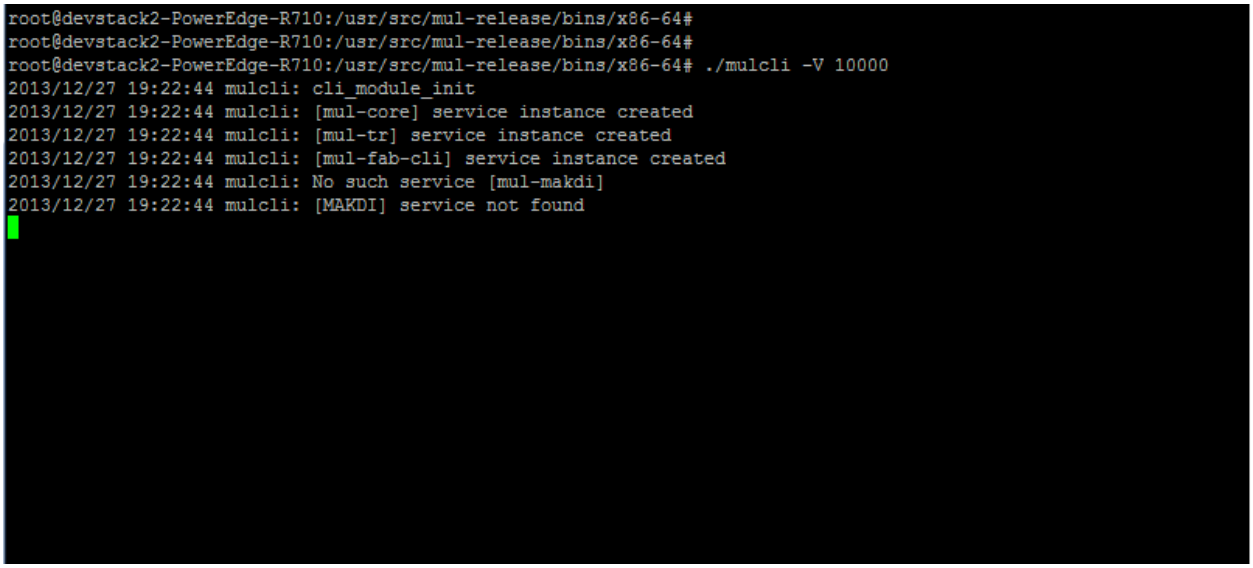
Possible options to use -

```
    -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:



1.4.2.4 Running Applications: l2switch

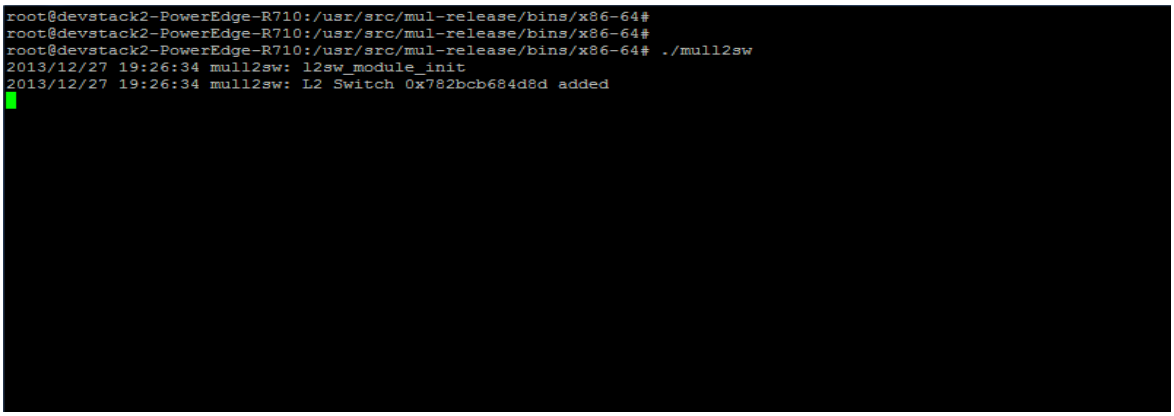
L2switch application provides bare-bones forwarding scheme based on l2 learning.

```
$ cd (mul-code-dir)/application/l2switch
$ sudo ./mull2sw
```

Possible options to use –

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

Example:



Once we run l2switch app, network wide l2-switching takes place so if you have any hosts connected to an Openflow network they will start “pinging” each other.

Note – This does not support spanning tree or similar algorithm yet. So, this application should not be used in a loopy network.

1.4.2.5 Running Applications: fabric

The fabric app provides a slightly complex forwarding scheme wherein it provides a loop and learning free point to point virtual network connectivity between two or more hosts. Furthermore, it is multi-tenant aware.

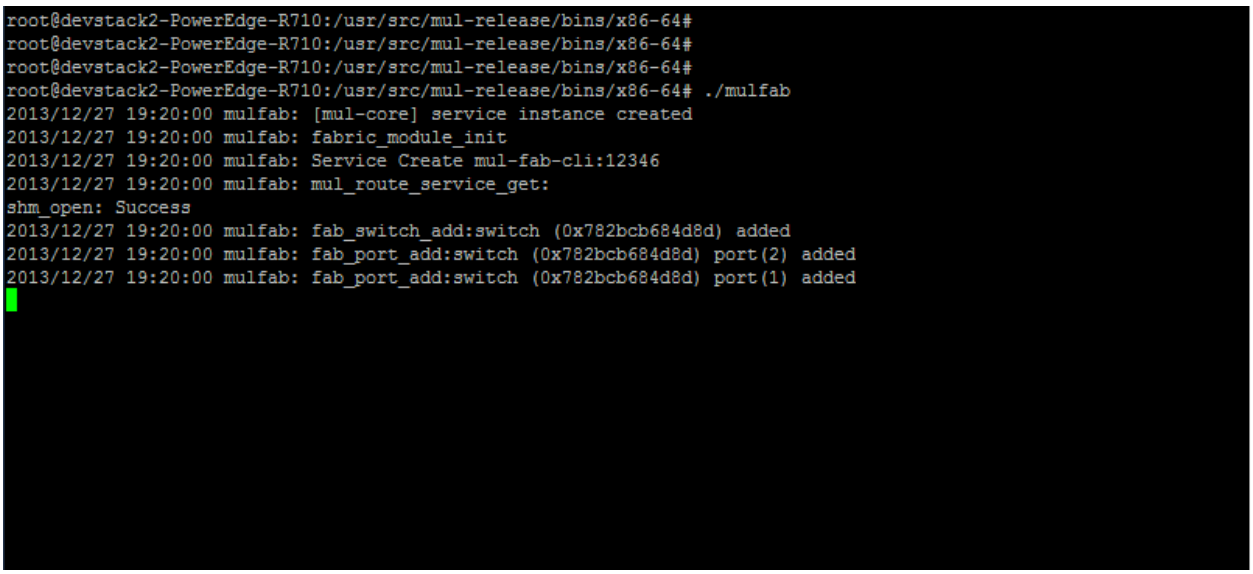
```
$ cd (mul-code-dir)/application/fabric
$ sudo ./mulfab –d

OR (for verbose mode)

$ sudo ./mulfab
```

Possible options to use -

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

NOTE – This application needs topo-routing service to be also running.

Sample:



1.4.2.6 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
```

For more on how to run PRISM, please refer PRISM-HOWTO document.

1.4.2.7 Running MUL controller in HA mode

MUL core component supports multiple controller functionality as defined by Openflow spec v1.3 and above. To use it one has to first enable the necessary functionality in the Openflow switches.

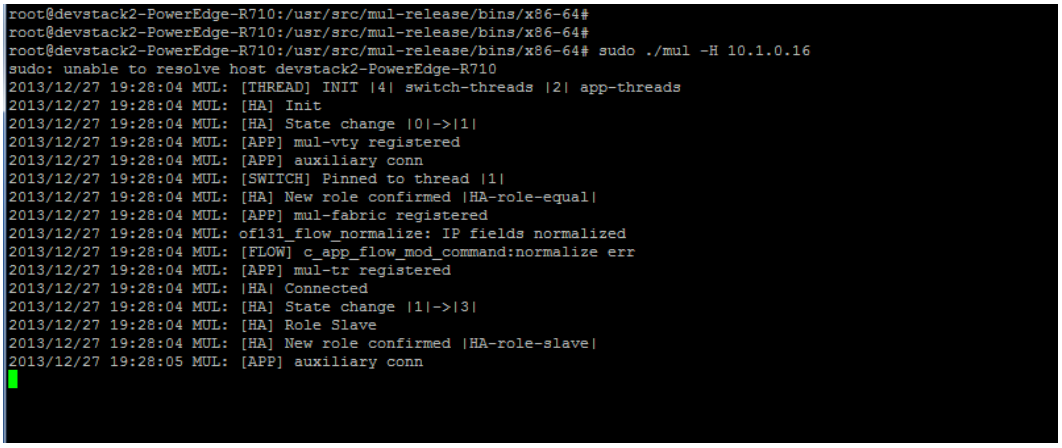
In this release, only mul core is HA aware and other applications are not yet HA aware

Mul core can be run in multiple controllers HA mode using the following:

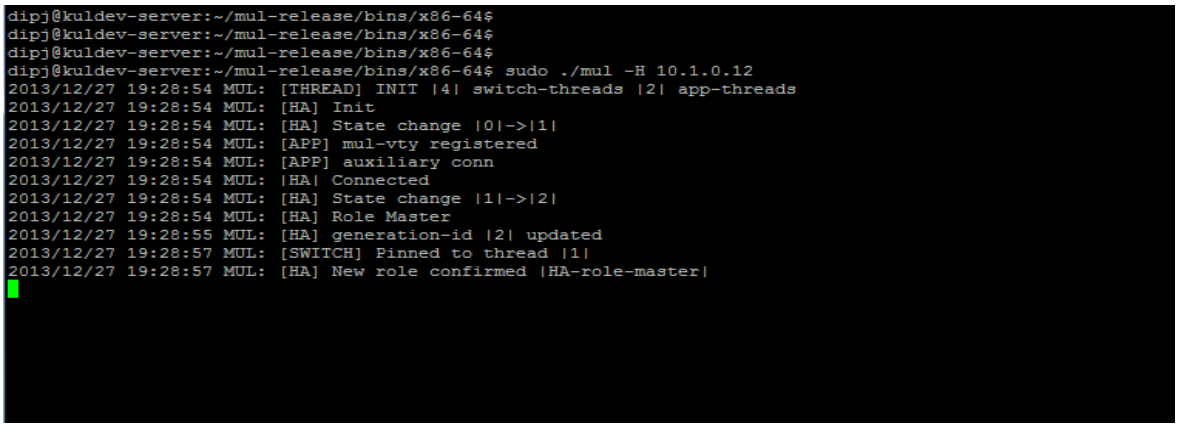
```
$ cd (mul-release-dir)/bins/<arch>/
$ sudo ./mul –H <IP address of Peer>
```

Example:

Controller 1 -



Controller 2 -



NOTE – *It is recommended that before connecting the switches to controller(s), the controllers be connected in HA-mode.*