open
MUL

MuL SDN 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

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. Get necessary packages

a) For Ubuntu 12.04.3 LTS, the following packages should be installed:

```
$ sudo apt-get install flex bison libwxgtk2.6-dev build-essential expect g++-multilib
tofrodos zlib1g-dev gawk libffi-dev gettext python python-all-dev swig libcurl4-gnutls-dev
libglib2.0-dev libevent-dev libssl-dev autoconf libtool
$ pip install --upgrade pip
$ pip install -r python_req.txt
$ sudo apt-get install python-daemon
```

b) Set CFLAGS and environment required by compilation process:

```
$ source pythonpath.sh

For i386, the following can be used :
  $ export CFLAGS="-I /usr/include/glib-2.0/ -I /usr/lib/i386-linux-gnu/glib-2.0/include/"

For x86-64, we can pass the following CFLAGS :
  $ export CFLAGS="-I /usr/include/glib-2.0/ -I /usr/lib/x86_64-linux-gnu/glib-2.0/include/"
```

2. Configure and make MuL

```
$ ./autogen.sh
$  ./configure  --with-vty=yes
$ make
```

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
   c. Location : $(mul-code)/services/loadable/topo_routing/

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

      ii. 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
      iii. CLI app
         • This provides a common cli based provisioning tool for all MuL components.
         • Location : $(mul-code/application/cli
      iv. 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 webserver
./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.
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:

```
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64# ./mul
2013/12/27 19:11:40 MUL: [THREAD] INIT |4| switch-threads |2| app-threads
2013/12/27 19:11:40 MUL: [HA] Init
2013/12/27 19:11:40 MUL: [APP] mul-vty registered
2013/12/27 19:11:41 MUL: [SWITCH] Pinned to thread |1|
2013/12/27 19:11:41 MUL: [HA] New role confirmed |HA-role-equal|
2013/12/27 19:11:41 MUL: [APP] mul-cli registered
2013/12/27 19:11:44 MUL: [APP] auxiliary conn
```

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:

```
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64# ./multr
2013/12/27 19:16:34 multr: tr_module_init
2013/12/27 19:16:34 multr: mul_route_init
2013/12/27 19:16:34 multr: mul_lldp_init
2013/12/27 19:16:34 multr: Service Create mul-tr:12345
2013/12/27 19:16:34 multr: lldp_port_add: adding 2 to switch 0x782bcb684d8d
2013/12/27 19:16:34 multr: lldp_port_add: adding 1 to switch 0x782bcb684d8d
2013/12/27 19:16:38 multr: mul_route_apsp_init_state:
```

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

open
MUL

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:

```
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64# ./mulcli -V 10000
2013/12/27 19:22:44 mulcli: cli_module_init
2013/12/27 19:22:44 mulcli: [mul-core] service instance created
2013/12/27 19:22:44 mulcli: [mul-tr] service instance created
2013/12/27 19:22:44 mulcli: [mul-fab-cli] service instance created
2013/12/27 19:22:44 mulcli: No such service [mul-makdi]
2013/12/27 19:22:44 mulcli: [MAKDI] service not found
```

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:

```
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64# ./mull2sw
2013/12/27 19:26:34 mull2sw: l2sw_module_init
2013/12/27 19:26:34 mull2sw: L2 Switch 0x782bcb684d8d added
```

open
MUL

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:

```
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64#
root@devstack2-PowerEdge-R710:/usr/src/mul-release/bins/x86-64# ./mulfab
2013/12/27 19:20:00 mulfab: [mul-core] service instance created
2013/12/27 19:20:00 mulfab: fabric_module_init
2013/12/27 19:20:00 mulfab: Service Create mul-fab-cli:12346
2013/12/27 19:20:00 mulfab: mul_route_service_get:
shm_open: Success
2013/12/27 19:20:00 mulfab: fab_switch_add:switch (0x782bcb684d8d) added
2013/12/27 19:20:00 mulfab: fab_port_add:switch (0x782bcb684d8d) port(2) added
2013/12/27 19:20:00 mulfab: fab_port_add:switch (0x782bcb684d8d) port(1) added
```

open
MUL

1.4.2.6 Running MUL controller in HA mode

[TBD]