LINCX conformance testing using Ryu

Aleksey Vasilenko, Cloudozer LLP

21/07/2014

1 Summary

- LINCX passes 92% of the OpenFlow 1.3 conformance tests provided by Ryu

Switch	Tests OK	Tests FAILED
Trema Switch	966	25
lagopus	952	27
LINCX	**920**	**71**
ofsoftswitch13	657	330
OVS	673	318
LINC-Switch	535	444
Pica8 P-3290	435	544
ivs	386	605
NEC PF5220	263	728
NoviFlow NoviKit200	251	740
Centec V350	187	804
IBM RackSwitch-G8264	128	863
Hewlett-Packard HP2920	46	945
EdgeCore AS4600-54T	36	937

2 Overview

The Ryu[1] conformance testing suite consists of ∼1000 unit tests[2]. It covers most of the OpenFlow 1.3 specification. LINCX successfully passes 92% of these tests. The certifica-

[1] https://osrg.github.io/ryu

[2] https://github.com/osrg/ryu/tree/master/ryu/tests/switch/of13

tion list published by Ryu[3] shows that most other software switches are less conformant. The detailed results of the LINCX conformance testing are available[4].

3 How to reproduce

See Appendix (chapter 6).

4 Missing features

The following two (required) features are under active development to complete the OpenFlow 1.3 conformance of LINCX:

- Action sets
- Groups

5 Further development

Adding the missing required features and certain optional features, such as meters, should make LINCX compete head-to-head with best current software switches.

Other areas of improvement of LINCX are OpenFlow 1.4 conformance, easier deployment and testing using frameworks other than Ryu (OF-Test, Twister).

6 Appendix

6.1 Working environment

The tests were performed using Arch Linux and Xen 4.4. Other distros and Xen version should work too.

6.2 Setup bridges

Run the following commands as root:

```
$ ip link add ovs1 type veth peer name lincx1
$ ip link add ovs2 type veth peer name lincx2
$ ip link add ovs3 type veth peer name lincx3

$ ip link set up ovs1
$ ip link set up ovs2
$ ip link set up ovs3
$ ip link set up lincx1
$ ip link set up lincx2
$ ip link set up lincx3
```

[3] http://osrg.github.io/ryu/certification.html

[4] https://github.com/FlowForwarding/lincx/blob/master/docs/conformance.md

```
$ brctl addbr br1
$ brctl addbr br2
$ brctl addbr br3
$ ip link set up br1
$ ip link set up br2
$ ip link set up br3
$ brctl addif br1 lincx1
$ brctl addif br2 lincx2
$ brctl addif br3 lincx3

$ brctl addbr xenbr0
$ ip link set up xenbr0
$ ip ad ad 192.168.3.1/24 dev xenbr0
```

7 Setup OVS for traffic management

The Current stable version of OVS can not properly handle MPLS frames. The latest git version is needed. Unfortunately, it is unstable and to survive full Ryu test suite OVS has to be compiled with Clang and launched manually with increased verbosity in a separate shell.

Install OVS from sources:

```
$ git clone git://git.openvswitch.org/openvswitch
$ cd openvswitch
$ ./boot
$ PYTHON=python2 CC=clang ./configure --prefix=/usr --localstatedir=/var --sysconfdir=/etc --w
$ make
$ sudo make install
```

Start OVS (in separate shell):

```
$ systemctl start ovsdb-server
$ ovs-vswitchd -v
```

Configure OVS ports:

```
$ ovs-vsctl add-br ovs -- set bridge ovs fail-mode=secure
$ ovs-vsctl set bridge ovs protocols=OpenFlow13
$ ovs-vsctl add-port ovs ovs1 -- set interface ovs1 ofport_request=1
$ ovs-vsctl add-port ovs ovs2 -- set interface ovs2 ofport_request=2
$ ovs-vsctl add-port ovs ovs3 -- set interface ovs3 ofport_request=3
$ ovs-vsctl set-controller ovs tcp:0.0.0.0:6633
```

8 Setup LINCX

Get LINCX sources:

```
$ git clone https://github.com/FlowForwarding/lincx.git
$ cd lincx
```

Prepare sys.config:

```
$ cp priv/sys.config.sample priv/sys.config
```

Add the following to sys.config:

```
[
  {linc,[
    {of_config,disabled},
    {capable_switch_ports,[
      {port,1,[{interface,"eth1"},{type,vif}]},
      {port,2,[{interface,"eth2"},{type,vif}]},
      {port,3,[{interface,"eth3"},{type,vif}]}
    ]},
    {capable_switch_queues,[]},
    {logical_switches,[
      {switch,0,[
        {backend,linc_max},
        {controllers,[
          {"ryu","192.168.3.1",6633,tcp}
        ]},
        {controllers_listener,{"0.0.0.0",6634,tcp}},
        {queues_status, disabled},
        {ports, [
          {port,1,{queues,[]}},
          {port,2,{queues,[]}},
          {port,3,{queues,[]}}
        ]}
      ]}
    ]}
  ]},

  {enetconf,[
    {capabilities,[
      {base, {1, 0}},
      {base, {1, 1}},
      {startup, {1, 0}},
      {'writable-running', {1, 0}}
    ]},
    {callback_module, linc_ofconfig},
    {sshd_ip, any},
```

```
    {sshd_port, 1830},
    {sshd_user_passwords,[
      {"linc", "linc"}
    ]}
  ]},

  {lager,[
    {handlers,[
      {lager_console_backend, info},
      {lager_file_backend,[
        {"/log/error.log", error, 10485760, "$D0", 5},
        {"/log/console.log", info, 10485760, "$D0", 5}
      ]}
    ]},
    {crash_log,"/log/crash.log"}
  ]},

 {sasl,[
    {sasl_error_logger, {file, "/log/sasl-error.log"}},
    {errlog_type, error},
    {error_logger_mf_dir, "/log"},
    {error_logger_mf_maxbytes, 10485760},
    {error_logger_mf_maxfiles, 5}
  ]}
].
```

Generate munge secret tokens and launch required services (this step can be omitted when one annoying bug is fixed):

```
$ ./scripts/mungeling
-secret dacfa423b8ab3ec21e89009a9d3854ac86b1b3fd 6400e7b41e31c0998eeef9ed356e9fdba13a9c21
$ diod -E
$ systemctl start munged
```

Prepare LINGConfig.mk:

```
$ cp LINGConfig.mk.sample LINGConfig.mk
```

Add the following to LINGConfig.mk (replace secret tokens with actual values):

```
LING_NETSPEC := -ipaddr 192.168.3.2 -netmask 255.255.255.0 -gateway 192.168.3.1
MEMORY := 1024
NUM_PORTS := 3
DEFAULT_BRIDGE := xenbr0
BRIDGE_PREFIX := br
REMOTE_MOUNTS := -secret dacfa423b8ab3ec21e89009a9d3854ac86b1b3fd 6400e7b41e31c0998eeef9ed356
REMOTE_MOUNTS += -9p 192.168.3.1 $(shell pwd)/log /log
```

Build and boot the LINCX image:

```
./rebar get-deps
./rebar compile
make
sudo make boot
```

9 Setup Ryu

Get ryu stable snapshot:

```
$ git clone --branch v3.11 https://github.com/osrg/ryu.git
$ cd ryu
```

Get OVS datapath id:

```
$ TESTER_DPID=`ovs-vsctl -- get bridge ovs datapath_id`
$ TESTER_DPID=`eval echo $TESTER_DPID`
```

Launch Ryu to view the LINCX datapath id:

```
$ PYTHONPATH=. python2 ./bin/ryu-manager --test-switch-tester $TESTER_DPID ryu/tests/switch/te
```

Look for hexadecimal value in the unknown switch message:

```
dpid=000000163e3eeda4 : Connect unknown SW.
```

Stop Ryu and rerun it with correct LINCX datapath id (replace -test-switch-target value with actual one):

```
$ PYTHONPATH=. python2 ./bin/ryu-manager --test-switch-tester $TESTER_DPID --test-switch-targe
```

Wait several minutes for tests to complete and view the results.