OpenFlow start up

Let’s take a look! SDN
Contents

1 SDN OpenSource Projects
2 Openflow tutorial
3 Mininet (emulate Openflow N/W)
4 Nox controller guide
5 Routeflow guide
## Contents

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SDN OpenSource Projects</td>
</tr>
<tr>
<td>2</td>
<td>Openflow tutorial</td>
</tr>
<tr>
<td>3</td>
<td>Mininet (emulate Openflow N/W)</td>
</tr>
<tr>
<td>4</td>
<td>Nox controller guide</td>
</tr>
<tr>
<td>5</td>
<td>Routeflow guide</td>
</tr>
<tr>
<td>Name</td>
<td>Language</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>OpenFlow Reference</td>
<td>C</td>
</tr>
<tr>
<td>NOX</td>
<td>Python, C++</td>
</tr>
<tr>
<td>POX</td>
<td>Python</td>
</tr>
<tr>
<td>Beacon</td>
<td>Java</td>
</tr>
<tr>
<td>Floodlight</td>
<td>Java</td>
</tr>
<tr>
<td>RouteFlow</td>
<td>C++, Python</td>
</tr>
<tr>
<td>Maestro</td>
<td>Java</td>
</tr>
<tr>
<td>FlowVisor</td>
<td></td>
</tr>
<tr>
<td>Trema</td>
<td>Ruby, C</td>
</tr>
</tbody>
</table>
OpenSource SDN Controllers

- The first SDN controller
- Linux
- C++ and Python
- OpenFlow Interface
- Event-based programming model
- Packet construction/dissection libraries
- Applications:
  - Forwarding (reactive), topology discovery, host tracking,...
- Download from http://www.noxrepo.org

- Python only
- Build a new platform in pure Python
- Target Linux, Mac OS, and Windows
- Good for research & education
- Download from http://www.noxrepo.org

Floodlight

- A Java-based OpenFlow Controller (forked from Beacon)
- Apache Licensed
- REST API
- Pure java (no OSGI know how required)
- Dead simple to build and run
- Tested and hardened in real environments
- Physical OpenFlow switches and real networks
- Code included in commercial product from Big Switch Networks
- Download from http://floodlight.openflowhub.org
Contents

1. SDN OpenSource Projects
2. Openflow tutorial
3. Mininet (emulate Openflow N/W)
4. Nox controller guide
5. Routeflow guide
2-1 We are going to ...
2-2 Let’s start SDN (step 1)

1. Download & Install Virtual Box
   VirtualBox 4.2.6 for Windows hosts
   https://www.virtualbox.org/wiki/Downloads

2. Download a compressed VM image
   OpenFlowTutorial-101311.zip

3. Unzip the image in Windows Explorer
   Linux Users: unzip OpenFlowTutorial-101311.zip
Let's start SDN (step2)

1. **Name and Operating System**
   - Name: OpenflowTutorial
   - OS: Linux
   - BIOS: Ubuntu

2. **Memory Size**
   - Memory available: 512 MB
   - Total: 3072 MB

3. **Hard Drive**
   - Available hard drive: 8.00 GB
   - Selected hard drive: OpenFlowTutorial-disk1.vhd (6.60 GB)

4. **Final Screen**
   - Username: openflow
   - Password: openflow
Let’s start SDN (step3)

Virtualbox settings

Note: if you are running VirtualBox, you should make sure your VM has **two network interfaces**. One should be a **NAT interface** that it can use to access the Internet, and the other should be a **host-only interface** to enable it to communicate with the host machine.
Mininet is a tool that emulates an arbitrary openflow network on your machine.

Contents

1. SDN OpenSource Projects
2. Openflow tutorial
3. Mininet (emulate Openflow N/W)
4. Nox controller guide
5. Routeflow guide
MININET Download and Get Started

Mininet is a tool that emulates an arbitrary openflow network on your machine.

• **How download MININET**
  Openflow tutorial을 설치하면 기본으로 구성되어있음. [http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet](http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet)
  1. git clone git://github.com/mininet/mininet
  2. mininet/util/install.sh –a

• **How to run mininet:**
  1. Open a new terminal (try ssh’ing to your VM).
  2. Run: sudo mn
     : Note: The default topology is a line with two hosts (h2 and h3) and a switch
  3. You should see the mininet terminal: mininet>
  4. Now, ping h3 from h2: mininet> h2 ping h3
MININET edit by code

mytopo.py

```python
from mininet.topo import Topo, Node

class MyTopo( Topo ):
    "Simple topology example."

def __init__( self, enable_all = True ):
    "Create custom topo."

        # Add default members to class.
        super( MyTopo, self ).__init__()

        # Set Node IDs for hosts and switches
        leftHost = 1
        leftSwitch = 2
        rightSwitch = 3
        rightHost = 4

        # Add nodes
        self.add_node( leftSwitch, Node( is_switch=True ) )
        self.add_node( rightSwitch, Node( is_switch=True ) )
        self.add_node( leftHost, Node( is_switch=False ) )
        self.add_node( rightHost, Node( is_switch=False ) )

        # Add edges
        self.add_edge( leftHost, leftSwitch )
        self.add_edge( leftSwitch, rightSwitch )
        self.add_edge( rightSwitch, rightHost )

        # Consider all switches and hosts 'on'
        self.enable_all()

Topos = { 'mytopo': ( lambda: MyTopo() ) }
```
3.3 MININET edit by tools

```
~/mininet/examples$ sudo miniedit.py
```
3-4 MININET edit by tools

Easy!
3-5 MININET test

- Simulate network & ping
  sudo mn --topo single,3 --mac --switch ovsk --controller remote
  mininet> h2 ping -c3 h3

- Flow 확인
  dpctl dump-flows tcp:127.0.0.1:6634

- ADD Flow
  dpctl add-flow tcp:127.0.0.1:6634 in_port=1, actions=output:2
  dpctl add-flow tcp:127.0.0.1:6634 in_port=2, actions=output:1

- DELETE Flow
  dpctl del-flows tcp:127.0.0.1:6634 in_port=1

  add : dpctl add-flow tcp:127.0.0.1:6634 in_port=1, actions=output:2
Contents

There are a handful of controllers available: Beacon, Floodlight, NOX, NOX Classic, POX, Ryu, Threma, ...
In this tutorial, we use NOX Classic.

1  SDN OpenSource Projects
2  Openflow tutorial
3  Mininet (emulate Openflow N/W)
4  Nox controller guide
5  Routeflow guide
4-1 About NOX history

The official version of NOX-Classic supports OpenFlow 1.0

<table>
<thead>
<tr>
<th>NOX</th>
<th>NOX-Classic (Deprecated)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Active</strong></td>
<td><strong>Next</strong></td>
</tr>
<tr>
<td><strong>Release</strong></td>
<td><strong>Active</strong></td>
</tr>
<tr>
<td><strong>Branch</strong></td>
<td><strong>Release</strong></td>
</tr>
<tr>
<td>verity</td>
<td>impulse</td>
</tr>
<tr>
<td>None yet!</td>
<td>destiny</td>
</tr>
<tr>
<td>2012-05-11</td>
<td>2012-07-20</td>
</tr>
<tr>
<td>N/A</td>
<td>2010-09-15</td>
</tr>
<tr>
<td>toz zip</td>
<td>toz zip</td>
</tr>
<tr>
<td></td>
<td>toz zip</td>
</tr>
</tbody>
</table>

- To get the latest version of NOX, you’d do the following:
  - git clone http://noxrepo.org/git/nox
- If you don’t want to use the default branch
  - git checkout branch-name

NOX-Classic has had a Qt based GUI for quite a while now, and in slightly modified form, it made it into POX as well.
4-2 About NOX version

• **Bootstraping NOX 0.9.0**
  
  You do not need to install NOX - it comes ready to run on the VM.  
  
  cd ~/nox/build/src
  
  ./nox_core -v -i ptcp: pytutorial

• **Check NOX Version**

  openflow@openflowtutorial:~/nox/build/src$ ./nox_core -V
  
  NOX 0.9.0(zaku)~full~beta (nox_core), compiled Oct 13 2011 17:49:51
  
  Compiled with OpenFlow 0x01
Download and bootstrapping nox

$ git clone https://github.com/noxrepo/nox-classic.git
$ ls
    mininet nox nox-classic oflops oftest openflow openvswitch pox
$ cd nox-classic/
$ ./boot.sh
$ mkdir build/
$ cd build/
$ ../configure
$ make
$ cd src
$ ./nox_core -V

NOX 0.9.1~full~beta (nox_core), compiled Feb 6 2013 23:36:29
Compiled with OpenFlow 0x01


4-4  NOX, Let's Take a look

-  `src / nox / coreapps / examples / pyswitch.py`

```python
# --
# Given a packet, Learn the source and peg to a switch/inport
# --
def do_learning(dpdl, inport, packet):
    global inst

    # Learn MAC on incoming port
    srcaddr = packet.src.toString()
    if ord(srcaddr[0]) & 1:
        return
    if inst.has_key(srcaddr):
        dst = inst[srcaddr]
        if dst[0] != inport:
            log.msg('MAC has moved from ' + str(dst) + 'to' + str(inport), system='pyswitch')
        else:
            return
    else:
        log.msg('learned MAC '+'mac_to_str(packet.src)+' on Rd Rd' + str(dpdl)[1], system='pyswitch')
    inst[srcaddr] = (inport, time(), packet)

    # Replace any old entry for (switch,mac).
    mac = mac_to_int(packet.src)
```

```python
# --
# If we've learned the destination MAC set up a flow and
# send only out of its inport. Else, flood.
# --
def forward(dpdl, inport, packet, buf, bufid):
    dstaddr = packet.dst.toString()
    if not ord(dstaddr[0]) & 1 and inst.has_key(dstaddr):
        port = inst[dpdl][dstaddr]
        if port != inport:
            log.err('**warning** learned port = inport', system='pyswitch')
            inst.send_openflow(dpdl, buf, bufid, openflow.OFPP_FLOOD, inport)
        else:
            # We know the outport, set up a flow
            log.msg('installing flow for ' + str(packet), system='pyswitch')
            flow = extract_flow(packet)
            flow[core.IN_PORT] = inport
            actions = [openflow.OFPAT_OUTPUT, [0, port]]
            inst.install_datapath_flow(dpdl, flow, CACHE_TIMEOUT,
                                         openflow.OFP_FLOW_PERMANENT, actions,
                                         bufid, openflow.OFP_DEFAULT_PRIORITY,
                                         inport, buf)
    else:
        # Haven't learned destination MAC. Flood
        inst.send_openflow(dpdl, bufid, buf, openflow.OFPP_FLOOD, inport)
```
4-5 Bootstrapping NOX GUI

- Nox GUI를 실행하기 위해서 Monitoring 실행
  ~/nox-classic/build/src$ ./nox_core -v -v -i ptcp:6633 monitoring switch routing trackhost_pktin

- nox-gui.py scripts 위치 확인.
  ~/nox-classic/src$ ls
  builtin etc gui include lib Makefile.am Makefile.in Make.vars nox nox-gui.py nox_main.cc scripts tests utilities

- NOX GUI 실행, error 확인.
  openflow@openflowtutorial:~/nox-classic/src$ ./nox-gui.py
  Traceback (most recent call last):
    File "./nox-gui.py", line 18, in <module>
      from PyQt4 import QtGui, QtCore
  ImportError: No module named PyQt4

- fix errors.
  sudo apt-get update
  sudo apt-get install pyqt4-dev-tools
    python-qt4, python-simplejson python-qt4-sql libqt4-sql-sqlite

- X server 실행 (X ming..)

- NOX controller 실행. NOX GUI 실행 => GUI pop-up 확인
  ./nox-gui.py
**NOX GUI**

- **gui start**
  1) `~/nox-classic/build/src$ ./nox_core -v -v -i ptpc:6633 monitoring switch routing trackhost_pktin flowtracer`
  2) `~/nox-classic/src$ ./nox-gui.py`
  3) `~/mininet$ sudo mn --topo tree,depth=2,fanout=3`
4-7 NOX GUI, a little work

- Flow manage
4-8  NOX GUI, a little work

- Flow entry
## Contents

1. SDN OpenSource Projects
2. Openflow tutorial
3. Mininet (emulate Openflow N/W)
4. Nox controller guide
5. Routeflow guide
What is RouteFlow?

RouteFlow is an open source solution to provide legacy IP routing services (e.g., OSPF, RIP) over OpenFlow networks. As a consequence of running the routing protocols flow entries are installed in the OpenFlow devices that correspond to the FIB generated by the routing engine (Quagga).
start using RouteFlow
1. Download the pre-configured VM  
2. Start VM
3. Get Open vSwitch
4. Get Mongo DB & install

The diagram below illustrates what we're running in this test: one switch (switch1), connecting two hosts (b1 and b2). This switch is being controlled by RouteFlow, and a virtual machine (rfvm1) is running Linux and doing the forwarding intelligence tasks for switch1.

https://github.com/CPqD/RouteFlow/blob/master/README.md#test-cases
appendix
  - pox
  - floodlight
6-1  **POX controller Tutorial**

- **Install the POX openflow controller**
  1) Download: [http://www.noxrepo.org/pox/versionsdownloads/](http://www.noxrepo.org/pox/versionsdownloads/)
  2) or **Clone from GitHub**, even easier!

  ```bash
  $ git clone http://www.github.com/noxrepo/pox
  $ cd pox
  $ ./pox.py openflow.of_01 --address=<your ip> --port=6633
  (ex. $ ./pox.py openflow.of_01 --address=10.1.1.1 --port=6634)
  Ready.
  POX> Yay!
  
  Selecting Branch/version
  $ git clone [http://github.com/noxrepo/pox](http://github.com/noxrepo/pox)
  $ cd pox
  /pox$ git checkout betta
  
  [https://openflow.stanford.edu/display/ONL/POX+Wiki#POXWiki-SelectingaBranchVersion](https://openflow.stanford.edu/display/ONL/POX+Wiki#POXWiki-SelectingaBranchVersion)
POX GUI Tutorial

- POX gui guide
  - git clone https://github.com/noxrepo/pox
  - cd pox
  - git checkout betta
  - cd ext
  - git clone https://github.com/MurphyMc/poxdesk
  - cd poxdesk
  - wget http://downloads.sourceforge.net/qooxdoo/qooxdoo-2.0.2-sdk.zip
  - unzip qooxdoo-2.0.2-sdk.zip
  - mv qooxdoo-2.0.2-sdk qx
  - cd poxdesk
  - ./generate.py
  - cd ..../..
  - ./pox.py samples.pretty_log web messenger messenger.log_service messenger.ajax_transport openflow.of_service poxdesk
Floodlight controller Tutorial

- Install guide

  Prerequisites
  
  ```
  sudo apt-get install build-essential default-jdk ant python-dev eclipse
  ```

  1. Download and build
  
  ```
  git clone git://github.com/floodlight/floodlight.git
  cd floodlight
  git checkout stable
  ant;
  ```

  2. Running floodlight
  
  ```
  java -jar target/floodlight.jar
  ```

  Tutorial site:
  
  http://floodlight.openflowhub.org/getting-started/
Thank you | kt ds