

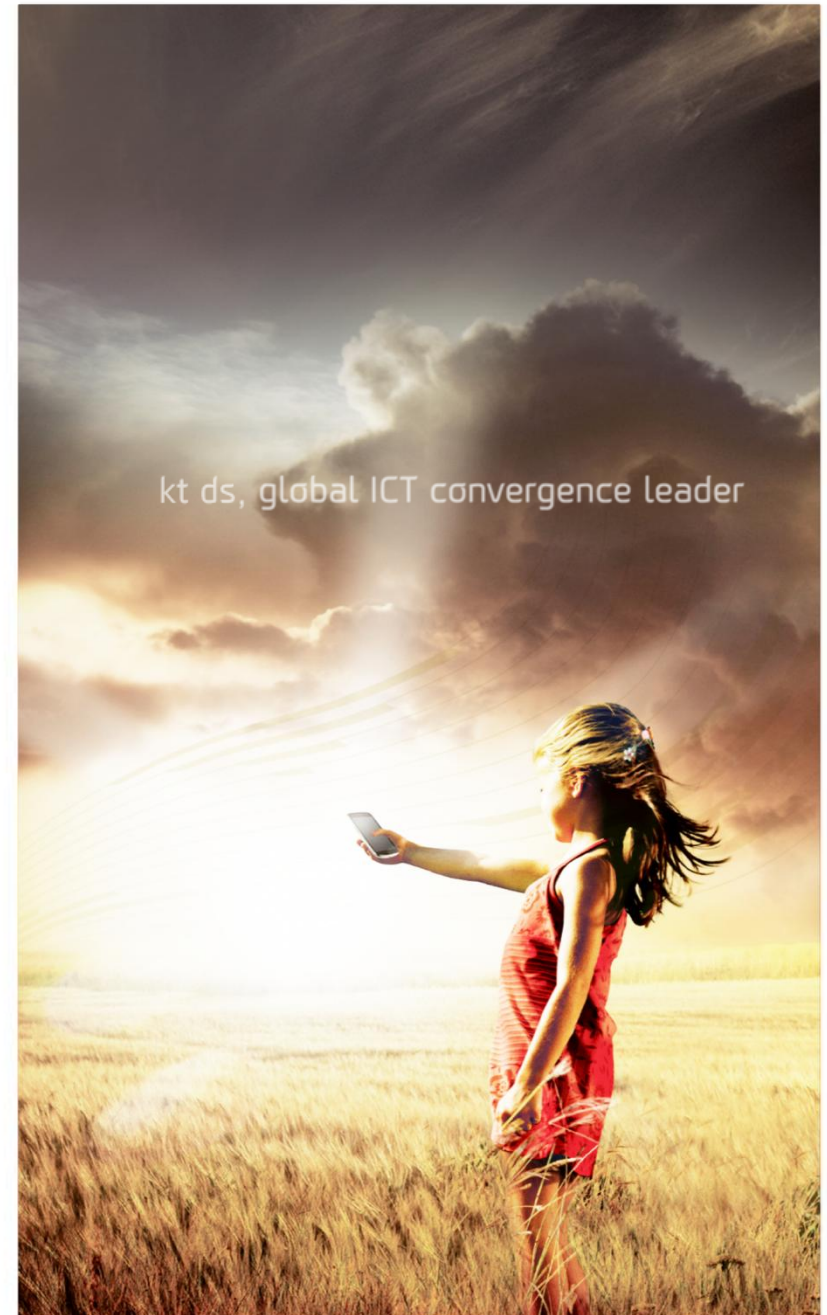
OpenFlow start up

Let's take a look! SDN

이미주, 이승재 | 2013.02.14

ruby@kt.com, sjlee7373@kt.com

kt ds



Contents

1 SDN OpenSource Projects

2 Openflow tutorial

3 Mininet (emulate Openflow N/W)

4 Nox controller guide

5 Routeflow guide

Contents

1 SDN OpenSource Projects

2 Openflow tutorial

3 Mininet (emulate Openflow N/W)

4 Nox controller guide

5 Routeflow guide

1-1 OpenSource SDN Projects

Name	Language	Platform(s)	License	Original Author	Notes
OpenFlow Reference	C	Linux	OpenFlow License	Stanford/Nicira	not designed for extensibility
NOX	Python, C++	Linux	GPL	Nicira	actively developed
POX	Python	Any			
Beacon	Java	Win, Mac, Linux, Android	GPL (core), FOSS Licenses for your code	David Erickson (Stanford)	web UI framework, regression test framework
Floodlight	Java	Any	Apache	BigSwitch, based on Beacon	
RouteFlow	C++, Python	Linux	Apache	CPqD (Brazil)	virtual IP routing as a service
Maestro	Java	Win, Mac, Linux	LGPL	Zheng Cai (Rice)	
FlowVisor				On.lab	
Trema	Ruby, C	Linux	GPL	NEC	regression test Framework

1-2 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>

Nox classic

- ✓ C++ and Python
- ✓ branch
 - Destiny (active) 0.9.1
 - Zaku (release) 0.9.0

"New" nox

- ✓ Remove Python from nox
- ✓ C++ Only
- ✓ branch
 - 0..9.2



- 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 environmets
- 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

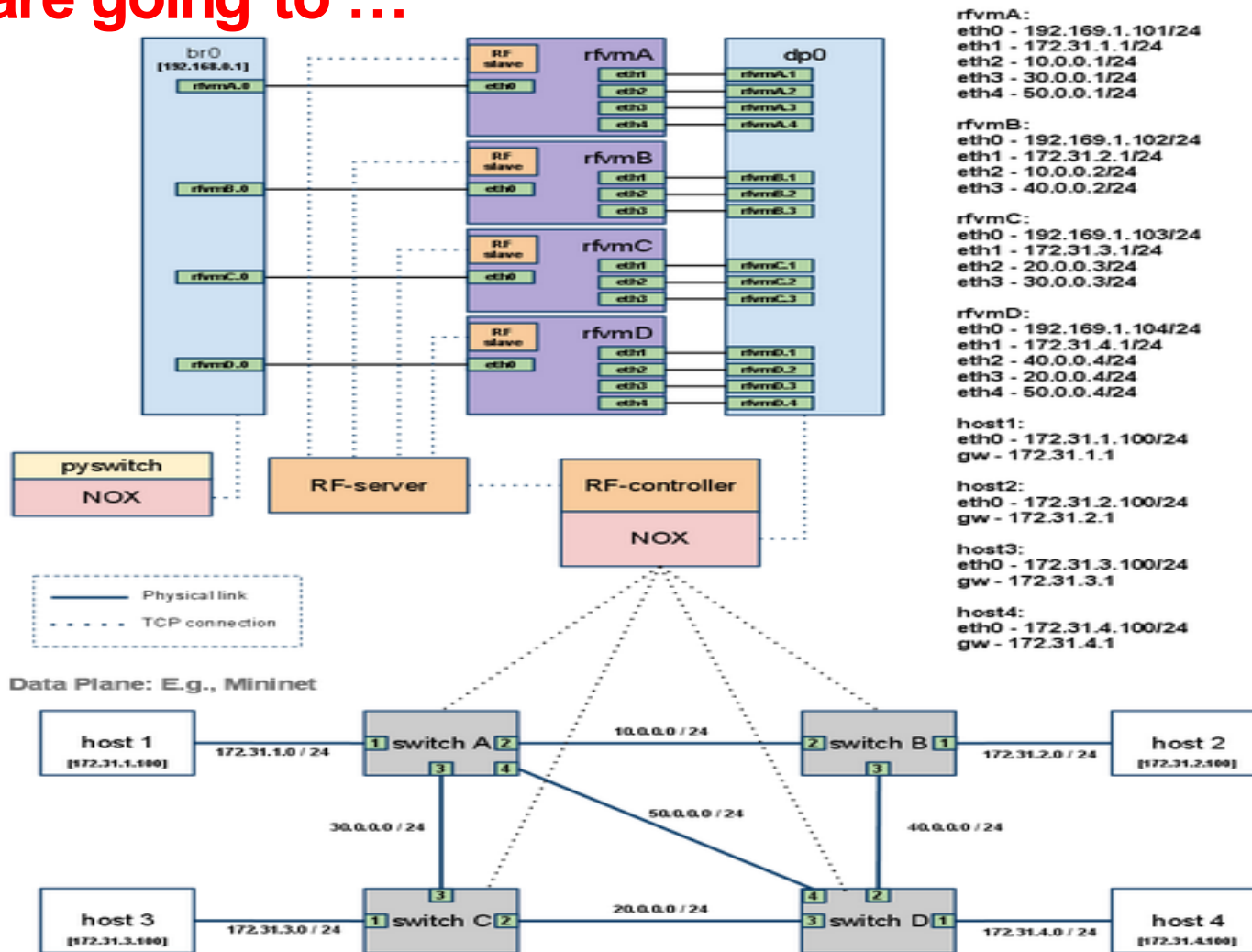
2 Openflow tutorial

3 Mininet (emulate Openflow N/W)

4 Nox controller guide

5 Routeflow guide

2-1 We are going to ...



환경구성

Openflow
tutorial

Emulate
Network
(MININET)

Controller
(NOX)

routeflow

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

http://www.openflow.org/wk/index.php/OpenFlow_Tutorial#Start_Network

3. Unzip the image in Windows Explorer

Linux Users : unzip OpenFlowTutorial-101311.zip

Windows Users

2-3 Let's start SDN (step2)

1

가상 머신 만들기

이름 및 운영 체제

새 가상 머신을 나타내는 이름을 입력하고 설치할 운영 체제를 선택하십시오. 입력한 이름은 VirtualBox에서 가상 머신을 식별하는 데 사용됩니다.

이름(N):

종류(T):

버전(V):

2

가상 머신 만들기

메모리 크기

가상 머신에 할당할 메모리(RAM) 크기를 메가바이트 단위로 입력하십시오.

추천 메모리 크기는 512 MB입니다.

MB

4 MB 3072 MB

3

가상 머신 만들기

하드 드라이브

필요하다면 가상 머신에 하드 드라이브를 추가할 수 있습니다. 새 하드 드라이브를 만들거나, 목록에서 선택하거나, 폴더 아이콘을 통하여 다른 위치에 있는 가상 하드 드라이브를 선택할 수 있습니다.

더 자세한 구성이 필요하다면 이 단계를 건너뛰고 가상 머신을 만든 다음 설정을 진행하십시오.

추천하는 하드 드라이브 크기는 8.00 GB입니다.

☐ 가상 하드 드라이브를 추가하지 않음(D)

☐ 지금 가상 하드 드라이브 만들기(C)

☒ 기존 가상 하드 드라이브 파일 사용(U)

4

123 [실행 중] - Oracle VM VirtualBox

가상 머신 실행 화면

```
[ 10.2317981 piix4_smbus 0000:00:07.0: SMBus base address uninitialized - upgrade BIOS or use force_addr=0xaddr
Ubuntu 11.04 openflowtutorial tty1
openflowtutorial login: openflow
Password:
Last login: Sun Feb 3 21:27:52 PST 2013 on tty1
Welcome to Ubuntu 11.04 (GNU/Linux 2.6.38-8-generic i686)

* Documentation:  https://help.ubuntu.com/

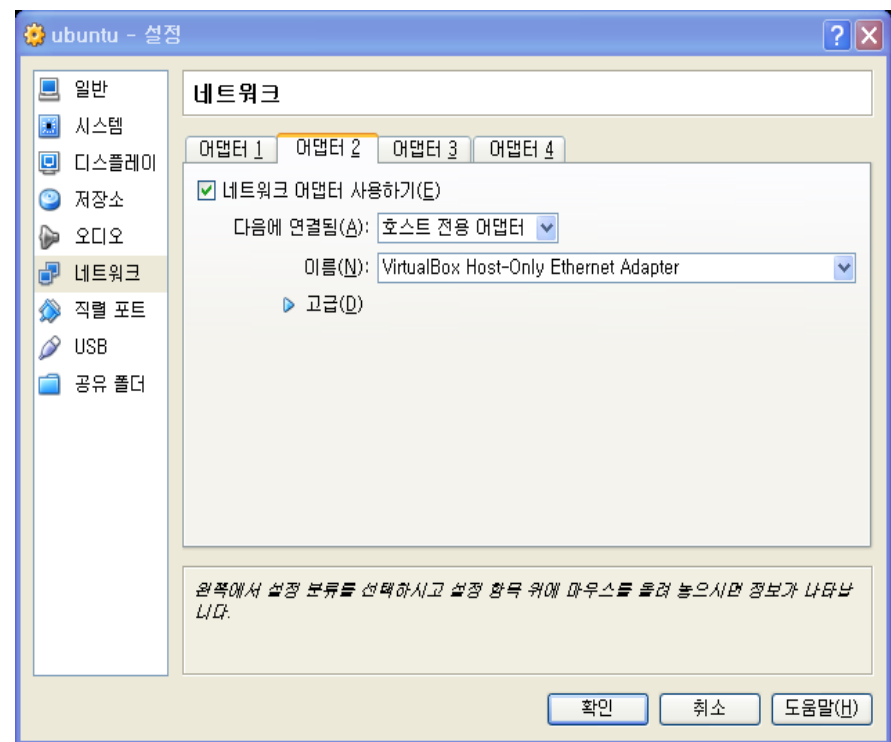
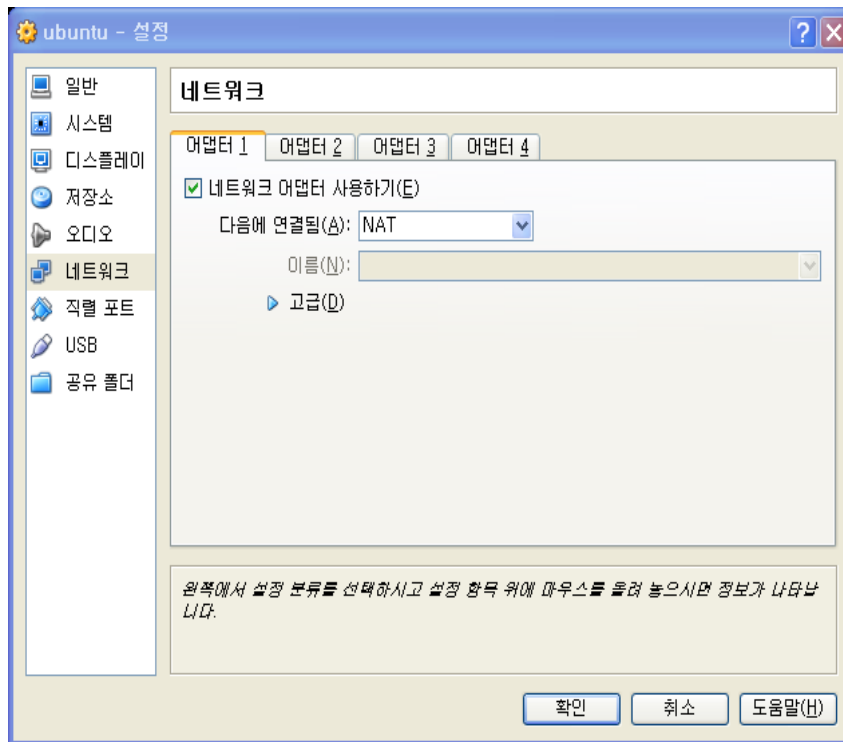
System information disabled due to load higher than 1.0
New release 'oneiric' available.
Run 'do-release-upgrade' to upgrade to it.

openflow@openflowtutorial:~$ ls
mininet nox oflops oftest openflow openuswitch
openflow@openflowtutorial:~$ _
```

Username: openflow
Password : openflow

2-4 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.

Contents

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

1 SDN OpenSource Projects

2 Openflow tutorial

3 Mininet (emulate Openflow N/W)

4 Nox controller guide

5 Routeflow guide

3-1 MININET Download and Get Started

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

- **How download MININET**

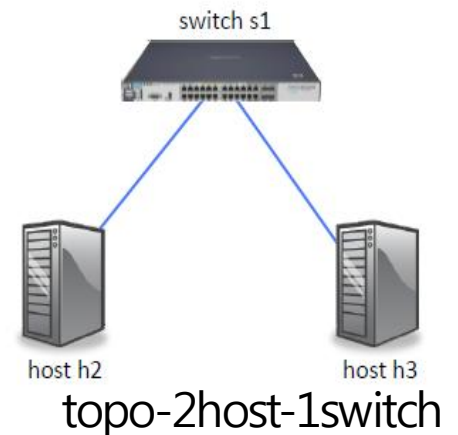
Openflow tutorial을 설치하면 기본으로 구성되어있음.

<http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet>

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`



3-2 MININET edit by code

mytopo.py

```
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__()

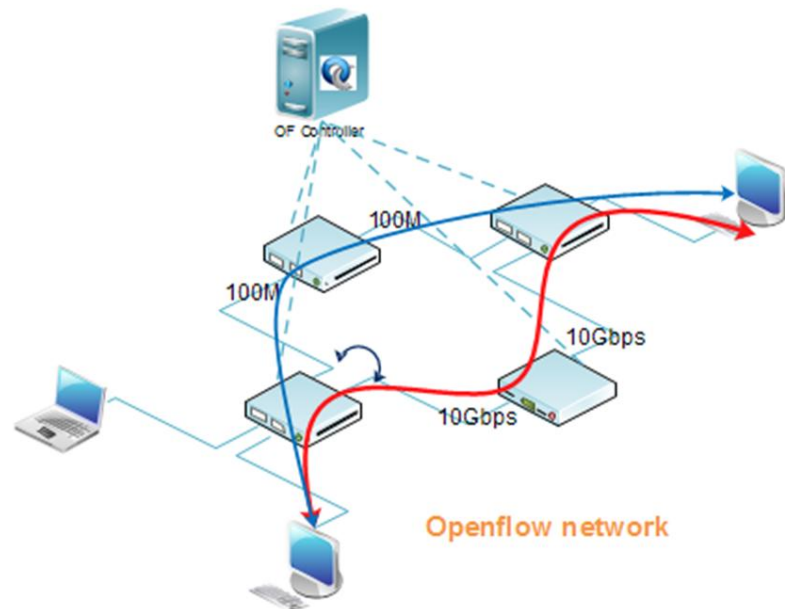
        # Set Node IDs for hosts and switches
        leftHost = 1
        leftSwitch = 2
        rightSwitch = 3
        rightHost = 4

        # 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 )

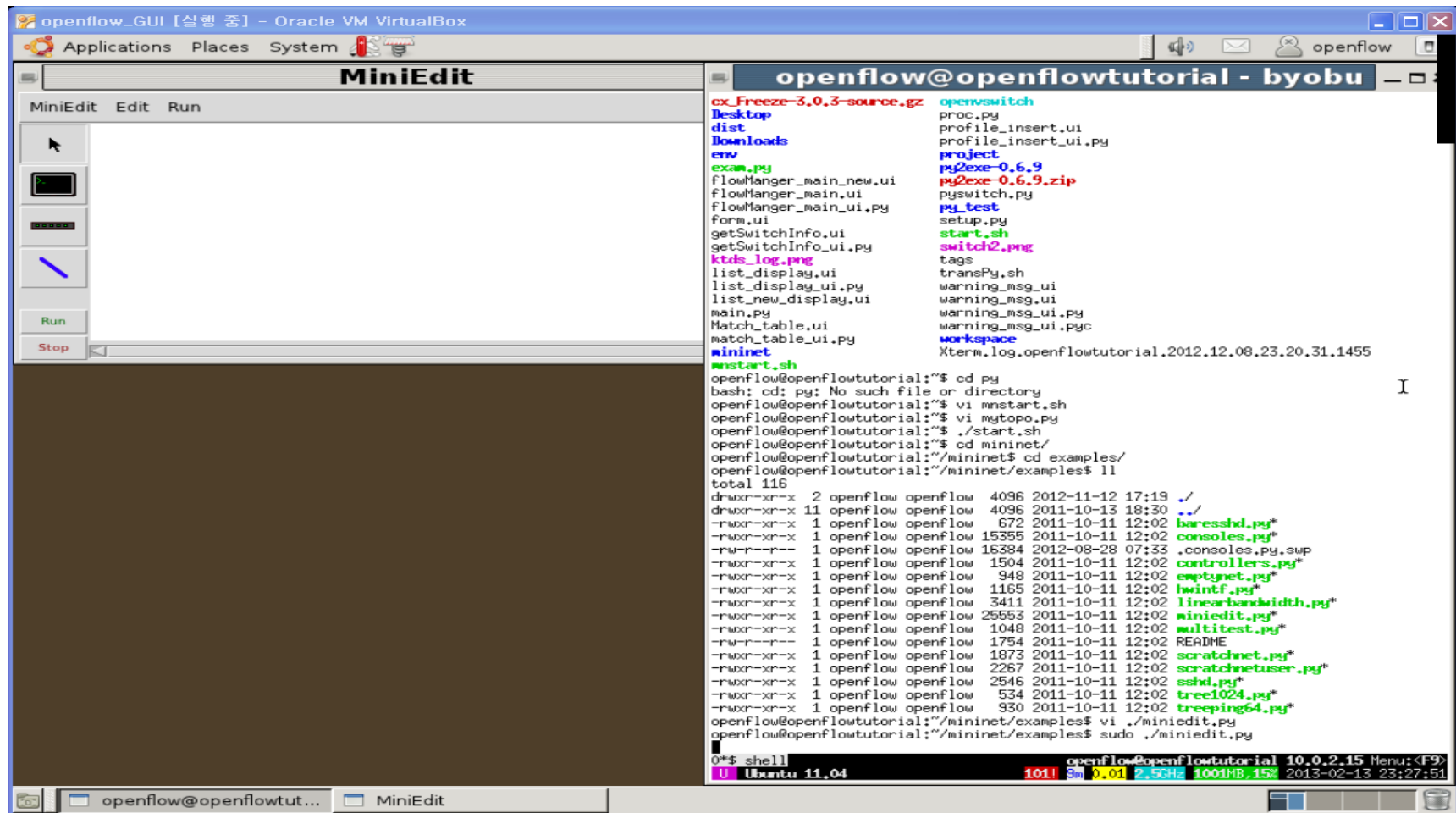
        # Consider all switches and hosts 'on'
        self.enable_all()

topos = { 'mytopo': ( lambda: MyTopo() ) }
```



3-3 MININET edit by tools

~/mininet/examples\$ sudo miniedit.py



The screenshot shows a VirtualBox window titled "openflow_GUI [실행 중] - Oracle VM VirtualBox". Inside, there is a "MiniEdit" application window and a terminal window titled "openflow@openflowtutorial - byobu".

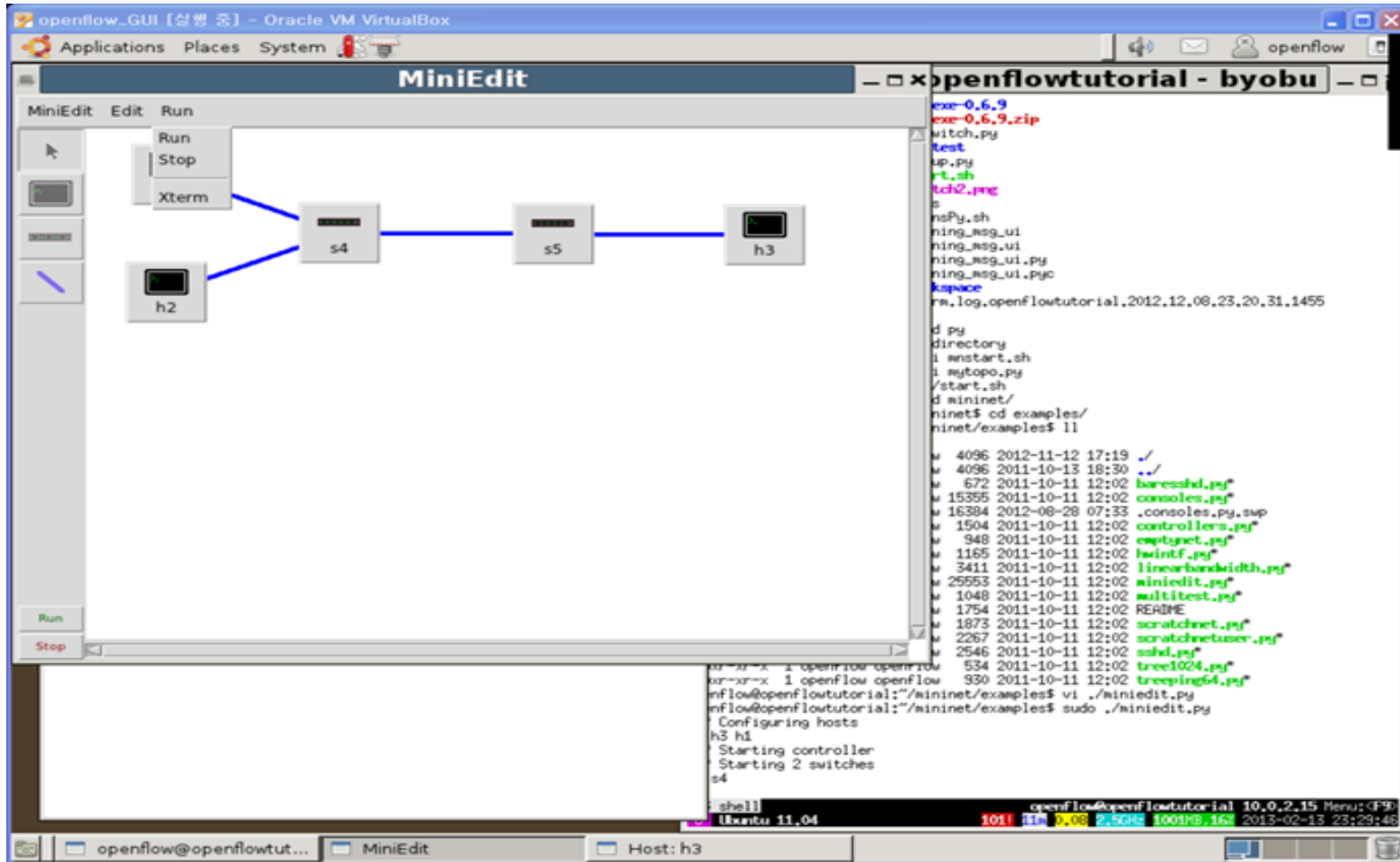
The MiniEdit window has a menu bar with "MiniEdit", "Edit", and "Run". It has a toolbar with icons for file operations and a "Run" button. The main editing area is empty.

The terminal window shows the following commands and output:

```
openflow@openflowtutorial:~$ cd py
bash: cd: py: No such file or directory
openflow@openflowtutorial:~$ vi mstart.sh
openflow@openflowtutorial:~$ vi mytopo.py
openflow@openflowtutorial:~$ ./start.sh
openflow@openflowtutorial:~$ cd mininet/
openflow@openflowtutorial:~/mininet$ cd examples/
openflow@openflowtutorial:~/mininet/examples$ ll
total 116
drwxr-xr-x 2 openflow openflow 4096 2012-11-12 17:19 ./
drwxr-xr-x 11 openflow openflow 4096 2011-10-13 18:30 ../
-rwxr-xr-x 1 openflow openflow 672 2011-10-11 12:02 baressh.py*
-rwxr-xr-x 1 openflow openflow 15355 2011-10-11 12:02 consoles.py*
-rwxr-xr-x 1 openflow openflow 16384 2012-08-28 07:33 consoles.py.swp
-rwxr-xr-x 1 openflow openflow 1504 2011-10-11 12:02 controllers.py*
-rwxr-xr-x 1 openflow openflow 948 2011-10-11 12:02 emptynet.py*
-rwxr-xr-x 1 openflow openflow 1165 2011-10-11 12:02 hwintf.py*
-rwxr-xr-x 1 openflow openflow 3411 2011-10-11 12:02 linearbandwidth.py*
-rwxr-xr-x 1 openflow openflow 25553 2011-10-11 12:02 miniedit.py*
-rwxr-xr-x 1 openflow openflow 1048 2011-10-11 12:02 multitest.py*
-rwxr-xr-x 1 openflow openflow 1754 2011-10-11 12:02 README
-rwxr-xr-x 1 openflow openflow 1873 2011-10-11 12:02 scratchnet.py*
-rwxr-xr-x 1 openflow openflow 2267 2011-10-11 12:02 scratchnetuser.py*
-rwxr-xr-x 1 openflow openflow 2546 2011-10-11 12:02 sshd.py*
-rwxr-xr-x 1 openflow openflow 534 2011-10-11 12:02 tree1024.py*
-rwxr-xr-x 1 openflow openflow 930 2011-10-11 12:02 treeping64.py*
openflow@openflowtutorial:~/mininet/examples$ vi ./miniedit.py
openflow@openflowtutorial:~/mininet/examples$ sudo ./miniedit.py
0*$ shell
U Ubuntu 11.04 101% 8m 0.01 2.5GHz 1001MB 15% 2013-02-13 23:27:51
```

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
```

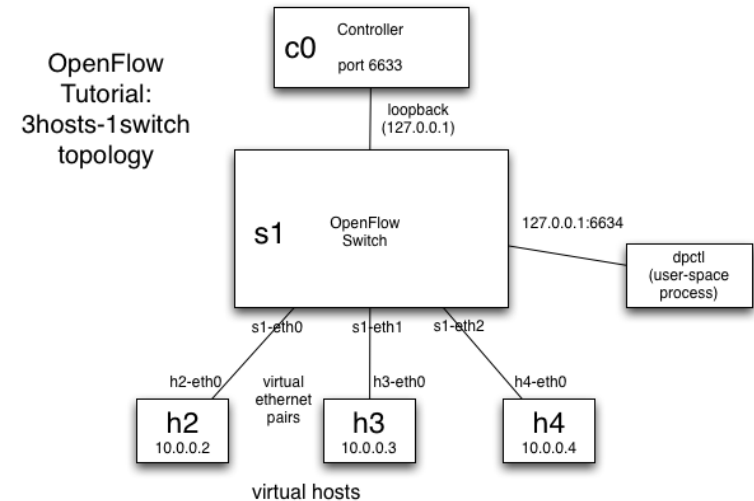
- ADD Flow

```
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

NOX

	<i>Active</i>	<i>Release</i>
<i>Branch</i>	verity	None yet!
<i>Release Date</i>	2012-05-11	N/A
<i>Snapshots</i>	tgz zip	

NOX-Classic (Deprecated)

	<i>Next</i>	<i>Active</i>	<i>Release</i>
<i>Branch</i>	impulse	destiny	zaku
<i>Release Date</i>	2012-??-??	2012-07-20	2010-09-15
<i>Snapshots</i>		tgz zip	tgz zip

- 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

POX

	<i>Active</i>	<i>Release</i>
<i>Branch</i>	master	None yet!
<i>Release Date</i>	2012?	N/A
<i>Snapshots</i>	tgz zip	N/A

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

- **Bootstrapping NOX 0.9.0**



by *Murphy*
McCauley

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
```

4-3 Bootstrapping NOX 0.9.1

- 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 ith OpenFlow 0x01
```

4-4 NOX, Let's Take a look

● src / nox / coreapps / examples / pyswitch.py

```
# --
# Given a packet, Learn the source and peg to a switch/inport
# --
def do_l2_learning(dpid, inport, packet):
    global inst

    # Learn MAC on incoming port
    srcaddr = packet.src.tostring()
    if ord(srcaddr[0]) & 1:
        return
    if inst.st[dpid].has_key(srcaddr):
        dst = inst.st[dpid][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 %d %d%' (dpid,inport), system='pyswitch')

    # Learn or update timestamp of entry
    inst.st[dpid][srcaddr] = (inport, time(), packet)

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

```
# --
# If we've learned the destination MAC set up a flow and
# send only out of its inport. Else, flood.
# --
def forward_l2_packet(dpid, inport, packet, buf, bufid):
    dstaddr = packet.dst.tostring()
    if not ord(dstaddr[0]) & 1 and inst.st[dpid].has_key(dstaddr):
        prt = inst.st[dpid][dstaddr]
        if prt[0] == inport:
            log.err('**warning** learned port = inport', system="pyswitch")
            inst.send_openflow(dpid, bufid, buf, openflow.OFPP_FLOOD, inport)
        else:
            # We know the output, 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, prt[0]]]]
            inst.install_datapath_flow(dpid, 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(dpid, bufid, buf, openflow.OFPP_FLOOD, inport)
```

4-5 Bootstrapping NOX GUI

- **Nox GUI**를 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

4-6 NOX GUI

● gui start

- 1) `~/nox-classic/build/src$ ./nox_core -v -v -i ptcp:6633` monitoring switch routing trackhost_pktin flowtracer
- 2) `~/nox-classic/src$ ./nox-gui.py`
- 3) `~/mininet$ sudo mn --topo tree,depth=2,fanout=3`

The screenshot shows the NOX GUI with a log of messages on the left and a network topology diagram in the center. The log includes timestamps, component names, and message details. The topology diagram shows a network of nodes and links.

Timestamp	Component	Verbosity	Message
00147	sdnmessenger	DBG	SDN message of length 85
00148	messenger_core	DBG	Copy 85 bytes to message
00151	sdnmessenger	DBG	SDN message of length 70
00156	messenger_core	DBG	Copy 70 bytes to message
00148	sdnmessenger	DBG	Message posted as JSOINMsg_event
00149	messenger_core	DBG	Copy 81 bytes to message
00143	sdnmessenger	DBG	SDN message of length 84
00158	messenger_core	DBG	Received packet of length 84
00153	sdnmessenger	DBG	SDN: (command:subscribe,aid:4,type:tcp)
00152	sdnmessenger	DBG	Message posted as JSOINMsg_event
00142	messenger_core	DBG	Received packet of length 84
00151	sdnmessenger	DBG	SDN message of length 81
00159	sdnmessenger	DBG	SDN: (command:subscribe,aid:8,type:flowtracer,msg...
00156	sdnmessenger	DBG	SDN: (command:subscribe,aid:7,type:sample_routing,m...
00160	python	ERR	Python handler returned invalid Disposition.
00157	sdnmessenger	DBG	SDN: (command:subscribe,aid:6,type:spanning_tree,m...
00158	sdnmessenger	DBG	SDN: (command:subscribe,aid:5,type:monitoring,msg...
00153	sdn_link	DBG	Adding 0a99c268 to interested host2sw list
00154	sdn_link	DBG	Adding 0a99c268 to interested sw2sw list

Topology

Default STP Monitoring Routing FlowTracer

Toggle: (N)odes (L)inkIDs (P)orts (R)efresh Topology

Send JSOIN command to NOX component

({"type":"lavi","command":"request","node_type":"all"})

The screenshot shows the query results for a packet. The results are displayed in a text area with a blue background. The results include packet details, counters, and actions.

```
Query results came back (dpid=0xb)
4 flows:

Match :
in_port: 4 dl_src: 0xe3cf8a5e7d3L dl_dst: 0x42e31c5a9423L nw_src: 10.0.0.9 nw_dst: 10.0.0.1 tp_src: 0
tp_dst: 0 nw_dst_n_wild: 0 nw_proto: ICMP dl_type: IP dl_vlan: 65535 dl_vlan_pcp: 0 nw_tos: 0
Counters :
Packet count: 4 Hard timeout: 0 Byte count: 392 Idle timeout: 5 Duration nsec: 838000000 Duration sec: 3
Priority: 65535 Cookie: 17960886152958125591 Table id: 0
Actions :
Port: 1

Match :
in_port: 4 dl_src: 0xe3cf8a5e7d3L dl_dst: 0x42e31c5a9423L nw_src: 10.0.0.9 nw_dst: 10.0.0.1 tp_src: 0
tp_dst: 0 nw_dst_n_wild: 0 nw_proto: (overwritten)ARP-Reply dl_type: ARP dl_vlan: 65535 dl_vlan_pcp: 0
nw_tos: 0
Counters :
Packet count: 1 Hard timeout: 0 Byte count: 42 Idle timeout: 5 Duration nsec: 879000000 Duration sec: 3
Priority: 65535 Cookie: 2560826493513136816 Table id: 0
Actions :
Port: 1

Match :
in_port: 1 dl_src: 0x42e31c5a9423L dl_dst: 0xe3cf8a5e7d3L nw_src: 10.0.0.1 nw_dst: 10.0.0.9 tp_src: 0
```

4-7 NOX GUI, a little work

- Flow manage

The screenshot displays the KTDS NOX Graphical User Interface, which is divided into two main windows: the **FlowManager Main Form** and the **List Display** window.

FlowManager Main Form: This window contains a menu bar (File, View, Components, Colors, Help) and a toolbar. It features two input fields: "Get_Switch INFO" and "Get_Flow_table". Below these is a list of hexadecimal values (000000000003, 000000000004, 000000000005, 000000000006) and a "PROFILE_LIST" section. At the bottom, there are buttons for "timestamp", "component", and "verbosity", along with "Clear filter", "Clear Log DB", "Filter !", "Autoscroll", and "FlowManager".

List Display: This window is titled "KTDS NOX Graphical User Interface" and "List Display". It contains three tables: "Flow_Table", "Match Table", and "Action Table".

Parameter	Value
NAME	test
Actions	Ref.Action Tab
Match	Ref.Match Tab
Priority	65535
Cookie	
Idle Time out	20
Hard Time out	
Out port	4

Parameter	Value
DL_DST	
DL_SRC	
DL_TYPE	2048
DL_VLAN	
DL_PCP	
INPUT PORT	1
NW_DST	
NW_Protocol	1
NW_SRC	10.0.0.1
Type Of Service	
TP_DST	
TP_SRC	
Wildcards	4178158

Parameter	ACTION value
OUTPUT	4
SET_VLAN_VID	
SET_VLAN_PCP	
STRIP_VLAN	
SET_ETH_SRC	
SET_ETH_DST	
SET_IP_SRC	
SET_IP_DST	
SET_IP_TOS	
SET_SRC_PORT	
SET_DST_PORT	
EXQUEUE	

Below the tables are buttons for "Save", "Change", and "Confirm". A "command to controller" section includes "Add Flow", "Chg_Flow", and "Del Flow" buttons. At the bottom right, it says "Design by KTDS. S.J.MJ" and "contact us: sjlee7373@kt.com".

Toggle: (N)odes Node(I)Ds Lin(K)s (L)inkIDs (P)orts (R)efresh Topology

Log Output:

```
64 bytes from 10.0.0.7: icmp_req=601 ttl=64 time=0.040 ms
64 bytes from 10.0.0.7: icmp_req=602 ttl=64 time=0.107 ms
64 bytes from 10.0.0.7: icmp_req=603 ttl=64 time=0.135 ms
64 bytes from 10.0.0.7: icmp_req=604 ttl=64 time=0.046 ms
64 bytes from 10.0.0.7: icmp_req=605 ttl=64 time=0.043 ms
64 bytes from 10.0.0.7: icmp_req=606 ttl=64 time=0.037 ms
64 bytes from 10.0.0.7: icmp_req=593 ttl=64 time=0.036 ms
64 bytes from 10.0.0.7: icmp_req=594 ttl=64 time=0.075 ms
64 bytes from 10.0.0.7: icmp_req=595 ttl=64 time=0.137 ms
64 bytes from 10.0.0.7: icmp_req=596 ttl=64 time=0.109 ms
64 bytes from 10.0.0.7: icmp_req=597 ttl=64 time=0.158 ms
64 bytes from 10.0.0.7: icmp_req=598 ttl=64 time=0.034 ms
```

4-8 NOX GUI, a little work

- Flow entry

The screenshot displays the KTDS_NOX Graphical User Interface. The window title is "KTDS_NOX Graphical User Interface". The menu bar includes "File", "View", "Components", "Colors", and "Help".

On the left, a text area shows query results: "Query results came back (dpid=0x3) 6 flows:". Below this, a blue box contains details for a specific flow: "Match : in_port: 1 nw_src: 10.0.0.1 nw_proto: ICMP dl_type: IP", "Counters : Packet count: 10 Hard timeout: 0 Byte count: 980 Idle timeout: 20", "Duration nsec: 358000000 Duration sec: 5 Priority: 65535 Cookie: 0", "Table id: 0", "Actions : OutPut Port: 4".

Below the text area is a table with headers: "timestamp", "component", "verbosity", and "message". The table is currently empty.

At the bottom left, there are buttons for "Timestamp", "Component", "Verbosity", "Clear filter", "Clear Log DB", "Filter !", "Autoscroll", and "FlowManager".

On the right, a network topology diagram shows a central green node connected to three yellow nodes. The central node is labeled "00 00 00 00 00 03". The yellow nodes are labeled "00 00 00 00 00 01", "00 00 00 00 00 02", and "00 00 00 00 00 03". The connections are labeled with numbers 1, 2, and 3.

Below the diagram are buttons for "Default", "STP", "Monitoring", and "Routing". A toggle bar at the bottom right shows "Toggle: (N)odes Node(I)Ds Lin(K)s (L)inkIDs (P)orts (R)efresh Topology".

At the bottom, a log window displays network traffic details:

```
64 bytes from 10.0.0.7: icmp_req=628 ttl=64 time=0.084 ms
64 bytes from 10.0.0.7: icmp_req=629 ttl=64 time=0.162 ms
64 bytes from 10.0.0.7: icmp_req=630 ttl=64 time=0.035 ms
64 bytes from 10.0.0.7: icmp_req=631 ttl=64 time=0.139 ms
64 bytes from 10.0.0.7: icmp_req=666 ttl=64 time=0.140 ms
64 bytes from 10.0.0.7: icmp_req=667 ttl=64 time=0.054 ms
64 bytes from 10.0.0.7: icmp_req=668 ttl=64 time=0.169 ms
64 bytes from 10.0.0.7: icmp_req=669 ttl=64 time=0.042 ms
64 bytes from 10.0.0.7: icmp_req=670 ttl=64 time=0.044 ms
64 bytes from 10.0.0.7: icmp_req=671 ttl=64 time=0.069 ms
```

Contents

1 SDN OpenSource Projects

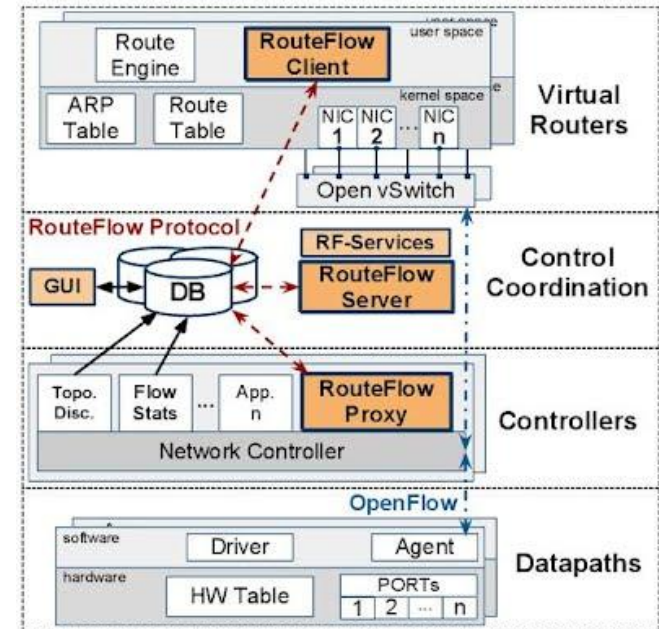
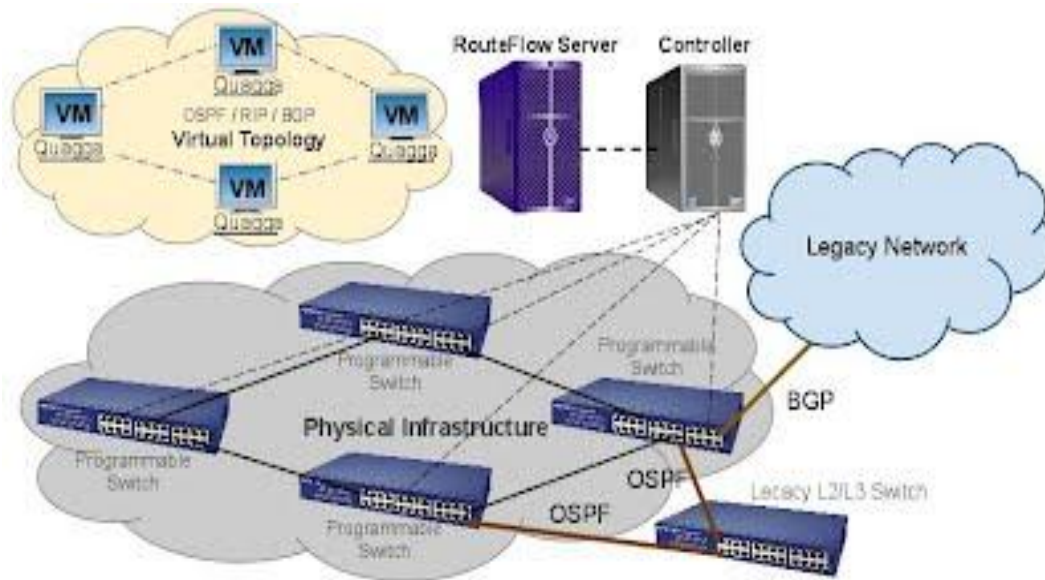
2 Openflow tutorial

3 Mininet (emulate Openflow N/W)

4 Nox controller guide

5 Routeflow guide

5-1 What is RouteFlow?

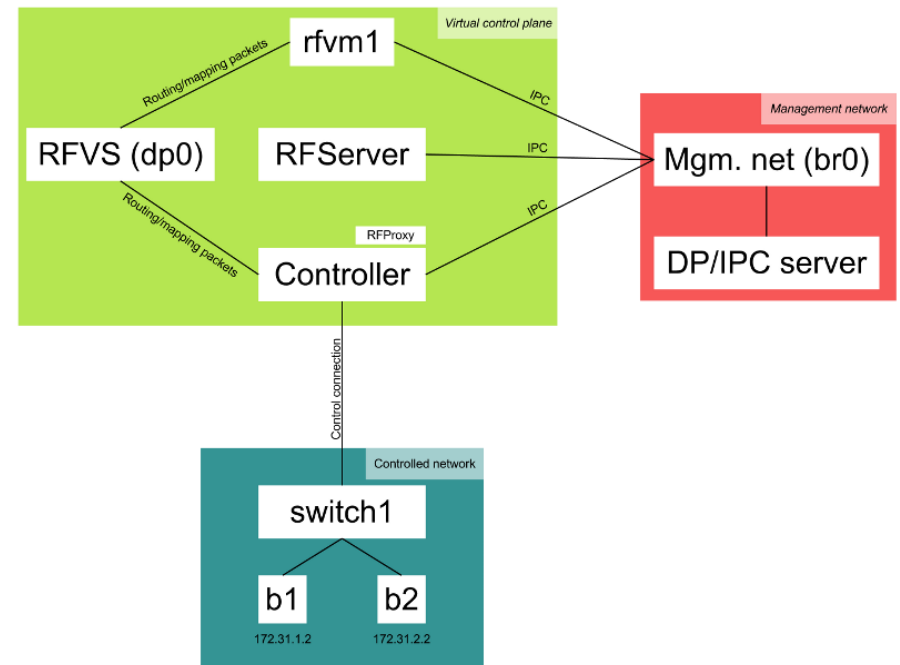


RouteFlow is an open source solution to provide legacy IP routing services (e.g., OSPF, RIP) over OpenFlow networks. As a consequence of running the routing protocols flow entries are installed in the OpenFlow devices that correspond to the FIB generated by the routing engine (Quagga).

5-2 RouteFlow Guide

- start using RouteFlow
 1. Download the pre-configured VM
<ftp://ftp.cpqd.com.br/pub/routeflow/RouteFlow.zip>
 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.



appendix

- pox
- floodlight

6-1 POX controller Tutorial

● Install the POX openflow controller

- 1) Download : <http://www.noxrepo.org/pox/versionsdownloads/>
- 2) or **Clone from GitHub**, even easier!
\$ 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  
$ cd pox  
/pox$ git checkout betta
```

<https://openflow.stanford.edu/display/ONL/POX+Wiki#POXWiki-SelectingaBranchVersion>

6-2 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
```

6-3 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

