

# A short walk-through of Mininet and POX

This tutorial has three parts. The first part covers the basics of the Mininet network emulation environment under which your programming assignment will be carried out. The second part covers the basics of the POX controller that you need program. The third part provides some guidelines for you to do your programming assignment.

## Part 1: The Mininet network emulation environment

### 1. Setup

The easiest way to get started is to **use a pre-packaged Mininet/Ubuntu Virtual Machine (VM)**. This VM contains the Mininet itself, all OpenFlow binaries and controllers, and tweaks to the kernel configuration to support larger Mininet networks.

#### Download Virtualization System

You can use any virtualization system of your choice, but we recommend installing VirtualBox. It's free and runs on Windows, Linux and OS X.

VirtualBox (Recommended): <https://www.virtualbox.org/wiki/Downloads>

For other virtualizations systems, visit [here](#).

#### Download Mininet VM

We recommend installing the following Mininet VM image:

<https://www.dropbox.com/sh/344tr8xay5q9x3a/AAAlhnsUqgF99ZIF3soVdd4a?dl=0> (32 bit)

<https://www.dropbox.com/sh/uudwuxngh5tb9ye/AADOe3IGqgO-CCAixTKHC3Lha?dl=0> (64 bit)

It comes with the latest version of Mininet a.k.a. HiFi and two OpenFlow Controllers (POX and Pyretic). The download will take some time. It's ~ 800MB, compressed.

#### Setup Virtual Machine

- **Start VirtualBox**
- **Select File>Import Appliance and select the .ova file**
- **Press the "Import" button.**
  - This will unpack and import the VM in your local machine. It will take a while, as the unpacked image is about 3 GB.

## **Boot VM**

Now, you are ready to start your VM. Press the "Start" arrow icon or double-click your VM within the VirtualBox window.

In the VM console window, log in with the user name and password for your VM. The username and password for this VM are:

- User name - **mininet**
- Password - **mininet**

Note that this user is a sudoer, so you can execute commands with root permissions by typing *sudo command*, where *command* is the command you wish to execute with root permission.

## **2. Mininet**

Mininet is a powerful network emulation tool. It creates a realistic virtual network, running real kernel, switch and application code on a single machine. You can easily interact with your network using the Mininet Command Line Interface (CLI).

The network topology used in this tutorial consists of 3 hosts, 1 switch and 1 POX controller (see below).

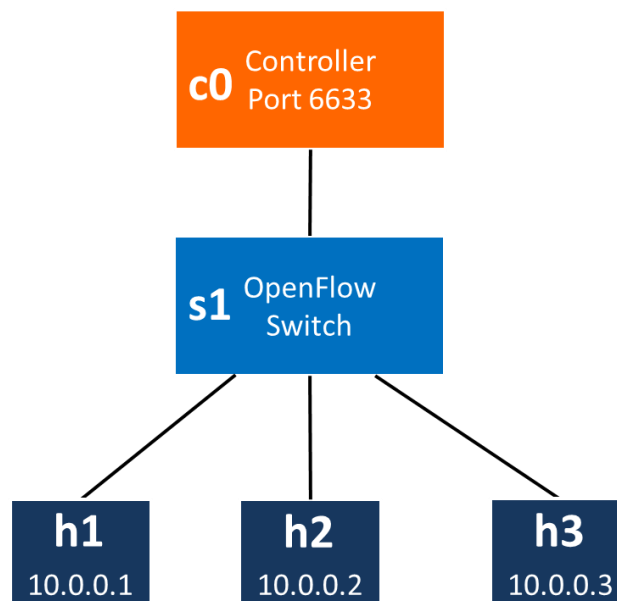


Figure 1: The basic network with 1 switch, 1 controller and 3 hosts.

To create this network in the VM, in a SSH terminal, enter:

```
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
```

Options:

--topo single,3: linear topology which includes 3 virtual hosts

--mac: set the host MAC and IP addresses to small, unique, easy-to-read IDs

--switch ovsk: the switch used here is an Open vSwitch

--controller remote: the controller is at a remote location

## 2.1. Mininet Basics

Display Mininet CLI commands:

```
mininet> help
```

Display nodes:

```
mininet> nodes
```

If the first string of the CLI command is a host, switch or controller name, the command is executed on that node. For instance, to show the interface of host h1:

```
mininet> h1 ifconfig
```

Test connectivity between hosts. For example, test the connectivity between h1 and h2:

```
mininet> h1 ping h2
```

Alternatively, you can test the connectivity between all hosts by typing:

```
mininet> pingall
```

Another way to run interactive commands and watch debug output is to run xterm for one or more hosts:

```
mininet> xterm h1 h2 h3
```

Note: Make sure you have your X11 forwarding (Xming for Windows system and X11 for Mac OS) is properly set up and run in background. See the next session for installing SSH and enabling X server.

Exit Mininet:

```
mininet> exit
```

Clean up:

```
$ sudo mn -c
```

### 3. SSH and X server

Once your VirtualBox VM is up, you can (and should) remotely login and start your network emulation environment, e.g., Mininet and the POX controller, from your remote terminals.

#### Download X Server and SSH capable terminal

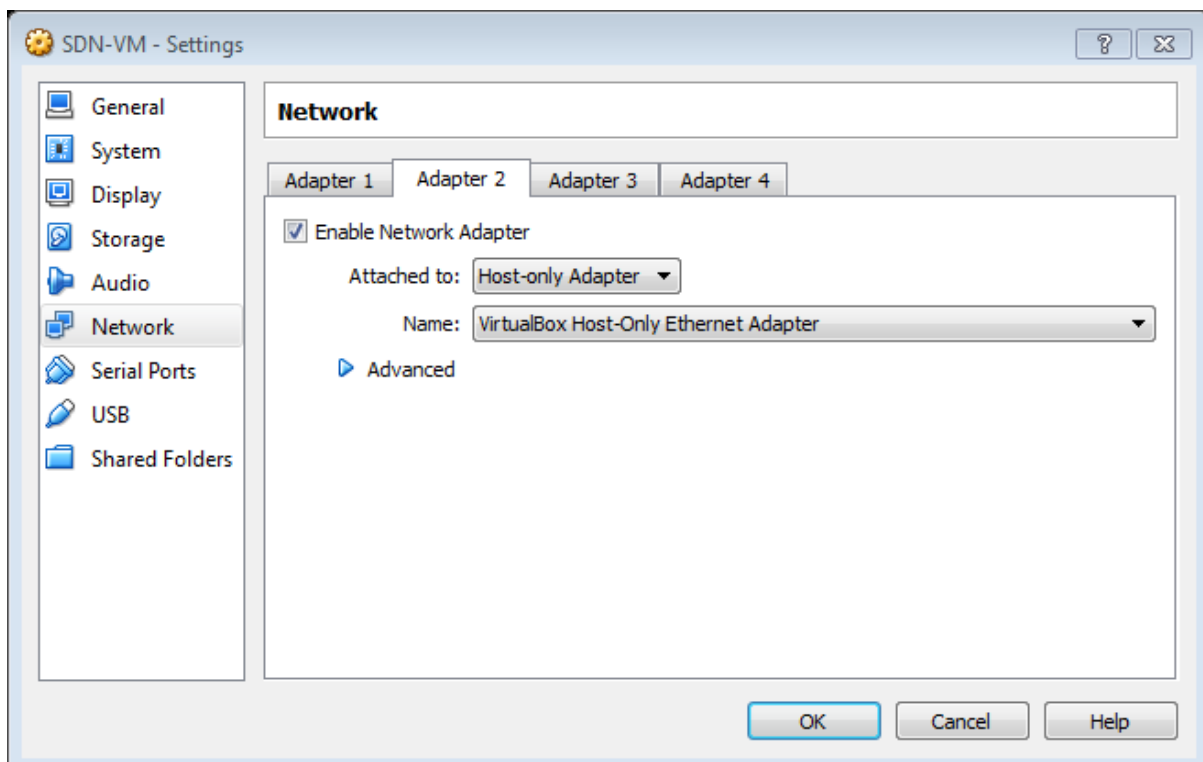
- For Windows install [Xming](#) and [Putty](#).
- For MAC OS install [XQuartz](#) and Terminal.app (builtin)
- Linux comes with X server and Gnome terminal + SSH (builtin)

SSH is short for Secure SHell. It provides a way to securely access another computer remotely.

#### 3.1. Enable SSH

Before you can connect to the VM via a remote SSH terminal, you need to setup your VM to enable SSH session.

Select your VM and go to the Settings Tab. Go to Network -> Adapter 2. Select the “Enable adapter” box, and attach it to “host-only network”. Make sure that your VM is turned off. Otherwise VirtualBox will not allow you to make these changes. This will allow you to easily access your VM through your host machine.



## 3.2. Set Up Network Access

To reduce the course VM's size, it does not provide a desktop environment. This tutorial and programming assignment will be done through forwarding, where programs display graphics through an X server.

To start the X forwarding, you will need to find the guest IP address.

### IP Address in VirtualBox

If you are using VirtualBox, you need to make sure that your VM has two network interfaces. One is a NAT interface which is used to access the Internet and the other is a host-only interface which is used to communicate with host machine. You can check the IP address by typing:

```
$ ifconfig -a
```

in Mininet environment. The NAT interface could be eth0 and its IP address should be 10.x.x.x and the host-only interface could be eth1 and its IP address should be 192.168.x.x. Later you will SSH to the host-only interface.

```
SDN-VM [Running] - Oracle VM VirtualBox
Machine View Devices Help
eth0    Link encap:Ethernet  HWaddr 08:00:27:52:6d:10
        inet addr:10.0.2.15 Bcast:10.0.2.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:50 errors:0 dropped:0 overruns:0 frame:0
        TX packets:50 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:5891 (5.8 KB) TX bytes:4870 (4.8 KB)

eth1    Link encap:Ethernet  HWaddr 08:00:27:a8:2b:e5
        inet addr:192.168.56.101 Bcast:192.168.56.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:50 errors:0 dropped:0 overruns:0 frame:0
        TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:5716 (5.7 KB) TX bytes:684 (684.0 B)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:65536 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

mininet@mininet-vm:~$ ifconfig -a_
```

### Access VM via SSH

#### 1. Mac OS X and Linux

Open a terminal. In that terminal, run:

```
$ ssh -X [user]@[Guest IP Here]
```

Replace [user] with the correct user name for your VM image.

Replace [Guest IP Here] with the IP you just found out. If ssh does not connect, make sure that you can ping the IP address you are connecting to.

Enter the password for your VM image. Next, try starting up an X terminal using

```
$ xterm
```

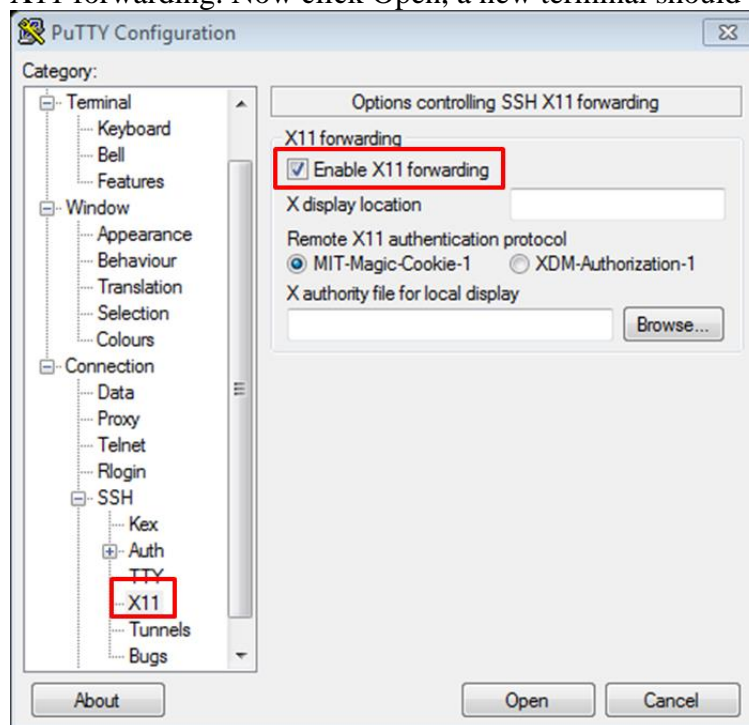
and a new terminal window should appear.

## 2. Windows

To use X11 applications such as xterm, the Xming server must be running, and you must make an ssh connection with X11 forwarding enabled.

- First, Start Xming. No window will appear. You can check Windows' task manager if you wish to verify.
- Second, make an ssh connection with X11 forwarding enabled.

Start PuTTY, enter your VM's IP address and enable X11 forwarding. To enable X11 forwarding from PuTTY's GUI, go to Connection -> SSH -> X11. Then click Enable X11 forwarding. Now click Open, a new terminal should appear.



Alternatively, you can create a login shortcut. In Windows, create a new shortcut and enter the full path to your PuTTY file as:

```
[Full Path] -X mininet @ [Guest IP] -pw mininet
```

```
e.g. C:\putty.exe -X mininet@192.168.56.101 -pw mininet
```

**Note:** *After this point, do not start Mininet directly from VirtualBox console. Open a terminal to make a SSH connection (i.e., use Putty) and start Mininet in your remote terminal. In fact, never use the VirtualBox VM console again. After starting the VirtualBox just minimize the console and do not use it.*

## Part 2: Basics of the POX Controller

### 4. The POX Controller

After setting up the network environment, a controller is needed to install flow entries to the switch so that packets can be forwarded among hosts. Go to the directory holding `pox.py` file in another remote SSH terminal:

```
$cd ~/pox
```

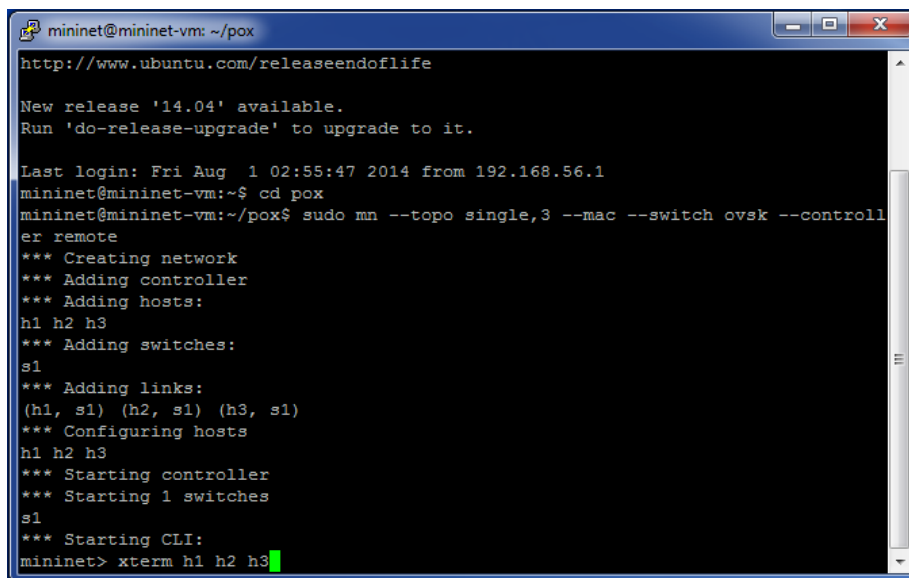
Then, start the hub:

```
$ pox.py log.level -DEBUG forwarding.hub
```

Notice that the source code of the hub example is located in the directory:

```
~/pox/pox/forwarding
```

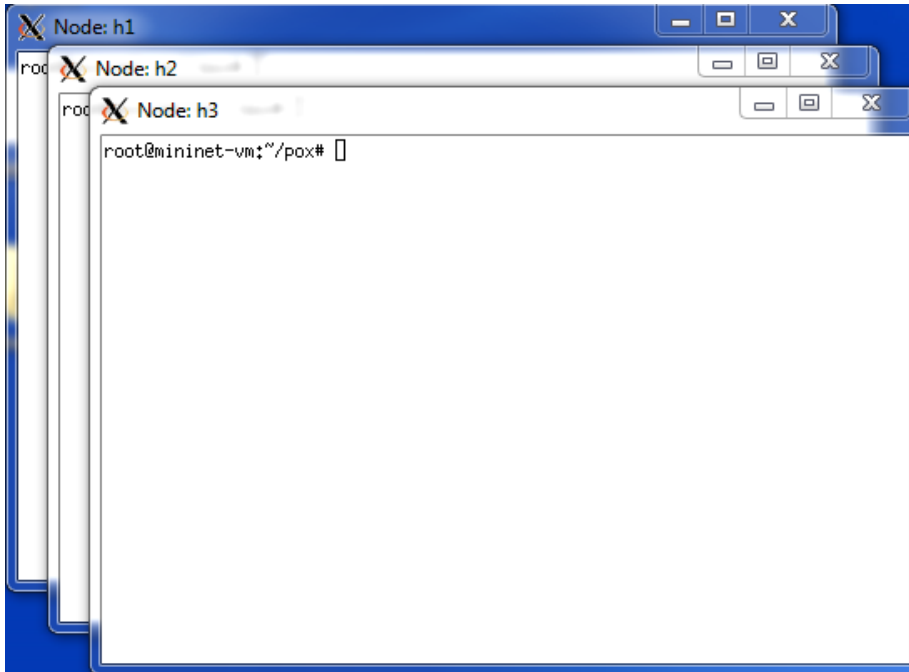
Then you can verify the connectivity of each host by either using `pingall` command or `xterm`. Here we will use `xterm` as an example:



```
mininet@mininet-vm: ~/pox
http://www.ubuntu.com/releaseendoflife
New release '14.04' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Aug  1 02:55:47 2014 from 192.168.56.1
mininet@mininet-vm:~$ cd pox
mininet@mininet-vm:~/pox$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
*** Starting 1 switches
s1
*** Starting CLI:
mininet> xterm h1 h2 h3
```

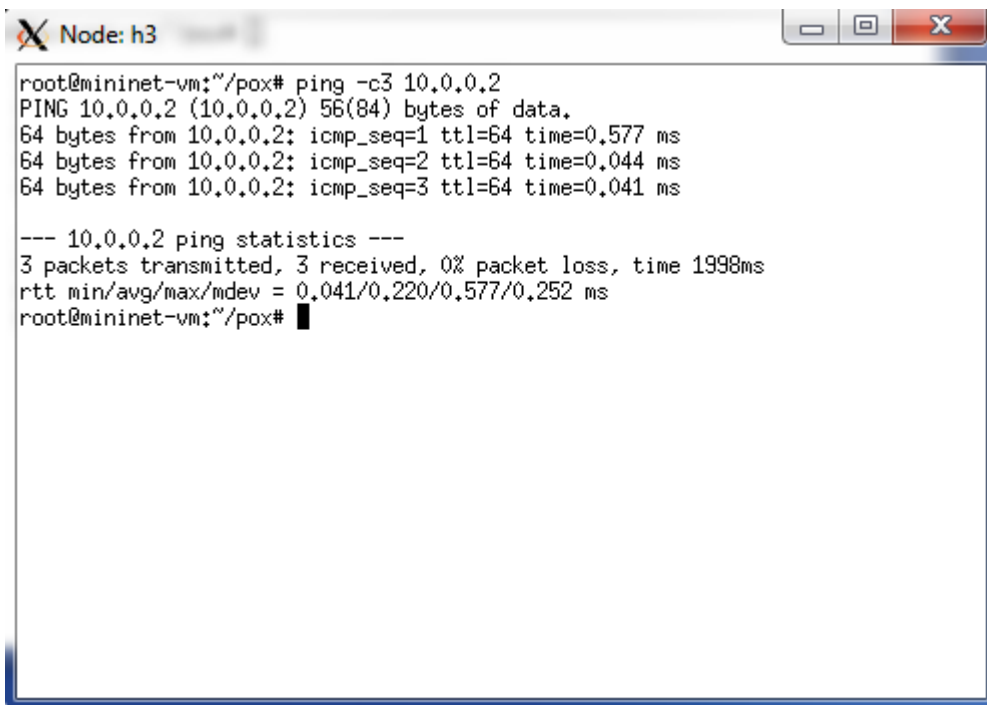
And you will see:



For example, from Node h3, we can test the connectivity to h2 by typing:

```
ping -c3 10.0.0.2
```

The result should be:



Now let's have a look at the hub code:



```

SDN-VM [Running] - Oracle VM VirtualBox
Machine View Devices Help
# You should have received a copy of the GNU General Public License
# along with POX. If not, see <http://www.gnu.org/licenses/>.
"""
Turns your complex OpenFlow switches into stupid hubs.
"""
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpidToStr

log = core.getLogger()

def _handle_ConnectionUp (event):
    msg = of.ofp_flow_mod()
    msg.actions.append(of.ofp_action_output(port = of.OFPP_FLOOD))
    event.connection.send(msg)
    log.info("Hubifying %s", dpidToStr(event.dpid))

def launch ():
    core.openflow.addListenerByName("ConnectionUp", _handle_ConnectionUp)

_ log.info("Hub running.")
38,1 Bot

```

In general a POX controller consists of three parts:

1. Listener
2. Control logic
3. Messenger

First you need to figure out the type of the event you want the controller to listen to (e.g., ConnectionUp, PacketIn, etc).

Then using some logic you can distinguish between different flows and attach a proper action for a specific flow.

Finally you send the message to the switch to add the new rule in the Openflow table.

### Listening to an event in POX

There are two common ways for an application to register with the controller for events.

1. Register callback functions for specific events thrown by either the OpenFlow handler module or specific modules like Topology Discovery
  - From the launch or from the init of a class, perform `core.openflow.addListenerByName("EVENTNAME", CALLBACK_FUNC, PRIORITY)`
  - For instance you already developed a switching function named `switch()`. If you want to trig the function when the controller starts to work, add a listener in lunch to provoke

*function switch when ConnectionUp handler is called:*  
*core.openflow.addListenerByName("ConnectionUp", switch).*

2. Register object with the OpenFlow handler module or specific modules like Topology Discovery
  - From typically the init of a class, perform `addListeners(self)`. Once this is added, the controller will look for a function with the name `_handle_EVENTNAME(self, event)`. This method is automatically registered as an event handler.

## Event handlers

As explained above you need to set a listener on your POX and when an event happens the relevant handler function will be activated. Your control logic should be placed in one of these handlers.

The two main handlers that you might need to modify in your program are:

- `ConnectionUp`, which is activated when a switch turns on (or connects to the controller). The name of the handler function for this event is: `_handle_ConnectionUp`. Be careful about timers when implementing this handler. `ConnectionUp` triggers only once when the switch starts to work.
- `PacketIn`, activated by arriving a packet into the controller. The name of the handler is `_handle_PacketIn`.

## Useful POX API

- *ofp\_flow\_mod OpenFlow message*  
 This composes a message which tells a switch to install a flow entry. It has a match attribute and a list of actions.  
 Notable fields are:
  - `actions` – A list of actions to perform on matching packets.
  - `priority` – This specifies the priority for overlapping matches. Higher values are of higher priority.
  - `match` – A `ofp_match` object. By default, none of the fields of this object is set. You may need to set some of its fields
- *ofp\_match class*  
 This class describes packet header fields and an input port to match on. Fields not specified are “wildcards” and will match on any value.  
 For example, create a match which matches packets arriving on port 3:  

```
match = of.ofp_match()
match.in_port = 3
```
- *ofp\_action\_output class*

This is an action which specifies a switch port that you want to send the packet out of. There is a variety of pre-defined “port numbers” such as OFPP\_FLOOD.

For example, create an action which sends packet out of port 2.

```
out_action = of.ofp_action_output(port = 2)
```

- *connection.send(...)*  
This function sends an OpenFlow message to a switch

*For example*, create a flow\_mod that sends packets arriving on port 3 out of port 4

```
msg = of.ofp_flow_mod()
msg.match.in_port = 3
msg.actions.append(of.ofp_action_output(port = 4))
connection.send(msg)
```

**Note: Many of these objects and associated attributes are not used in the hub example but they might be useful in the assignment.**

## Part 3: General Guidelines

So far you have learnt how to setup your system to start your project. Through this document you have got some hands on experience on how to use Mininet and how to modify the controller to add new rules on OpenFlow table.

Here as a quick survey we provide you all the essential steps require to start the system to work:

1. Start the VirtualBox
2. Use keyword “mininet” as both user name and password to login to the system.
3. Open a terminal (follow all the requirements explained in section 3) and create your network there.

To create the sample network of figure 1, enter:

```
$ sudo mn --topo single,3 --mac --controller remote --switch ovsk
```

After that the prompt will change from \$ to mininet>. From now on we will refer to this terminal as *mininet console*.

You can try out Mininet commands here. For instance try:

```
mininet> h1 ping h2
```

You will see that h1 fails to ping h2 as there is no controller installed yet.

4. So as the next step start a controller. You can start your own controller (file name controller.py) while doing your assignment or you can use some available controller in Mininet.

Open a new terminal and change the directory to pox (\$ cd pox) and try running a basic hub example:

```
$ pox.py log.level --DEBUG forwarding.hub
```

Now in the mininet console try to ping h2 from h1 again:

```
mininet> h1 ping h2
```

*Note: just remember to run your own controller when you are ready to start your assignment.*

5. The system is all set and ready to work. When you are working on your assignment, follow the steps bellow to reset mininet and controller both after any changes on the controller.py file.
  - a. Stop controller by press ctrl+c in the corresponding terminal.
  - b. Stop minnet by entering “exit” in mininet console.
  - c. Reset mininet in mininet console by:

```
sudo mn -c
```
  - d. Start mininet again.
  - e. Start controller again.

## More references to read:

### POX references:

- <https://openflow.stanford.edu/display/ONL/POX+Wiki> (recommend you read this if you want to fully understand POX)
- [http://archive.openflow.org/wk/index.php/OpenFlow\\_Tutorial#Sending\\_OpenFlow\\_messages\\_with\\_POX](http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial#Sending_OpenFlow_messages_with_POX) (very easy document to follow)
- <http://sdnhub.org/tutorials/pox/>

### Mininet References:

- <http://mininet.org/>
- <http://github.com/mininet/mininet/wiki/Introduction-to-Mininet>