網路實驗

Mininet

A Learning Switch

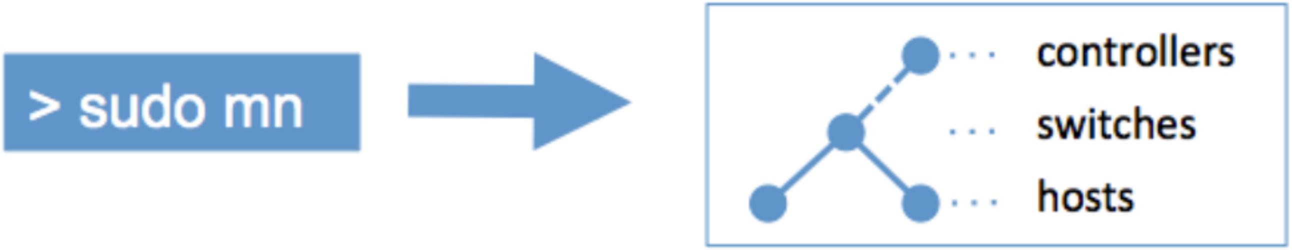
2022 Spring

T-C. Hou

Introduction

- **Mininet** creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native), in seconds, with a single command:

  $ sudo mn



- Because you can easily interact with your network using the **Mininet CLI (and API)**, customize it, share it with others, or deploy it on real hardware, Mininet is useful for development, teaching, and research.

Installation

- **Download & Install Mininet**

```
$ cd
$ git clone https://github.com/mininet/mininet
$ ./mininet/util/install.sh -a
```

- **Install pox controller**

```
$ git clone git://github.com/noxrepo/pox
```

- **Else**

```
$ sudo apt install git
```

Basic Mininet Usage

- To display a help message describing Mininet's startup options

```
$ sudo mn -h
```

- To view control traffic using the Wireshark

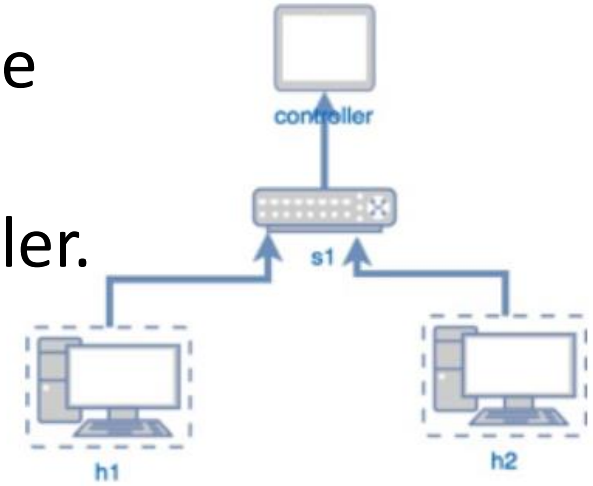
```
$ sudo wireshark &
```

  Filter : of (OpenFlow packet)

Basic Mininet Usage

- Start a minimal topology which includes one OpenFlow kernel switch connected to two hosts, plus the OpenFlow reference controller.

  $ sudo mn (sudo mn --topo=minimal)



mininet> help	Display Mininet CLI commands
mininet> nodes	Display nodes
mininet> net	Display links
mininet> dump	Dump information about all nodes
If the first string typed into the Mininet CLI is a host, switch or controller name, the command is executed on that node.	
mininet> s1 ifconfig –a	show the switch(s1) interfaces
mininet> h1 ifconfig –a	show the host(h1) interfaces

Basic Mininet Usage

- **Test connectivity between hosts**
  - Ping from host 1 to host 2

    ```
    mininet> h1 ping -c 1 h2
    ```

  - All-pairs ping

    ```
    mininet> pingall
    ```

- **Exit the CLI**

  ```
  mininet> exit
  ```

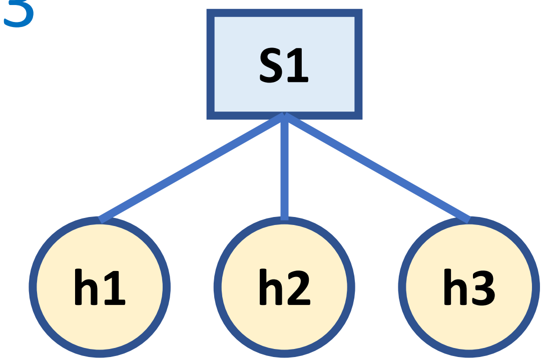
- **If mininet crashed for some reason, clean it up**

  ```
  $ sudo mn -c
  ```

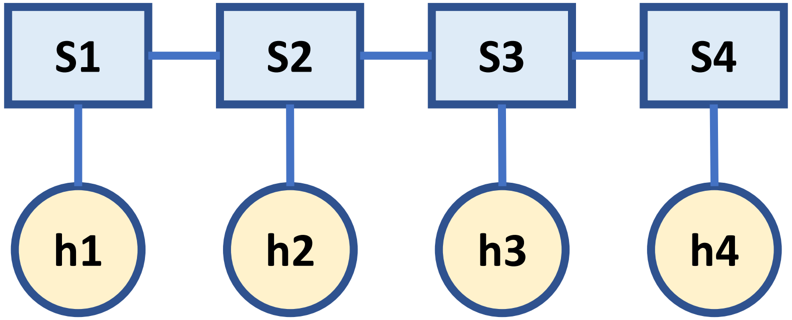
Advanced startup options

- **Run a Regression Test**
- **Changing topology type and size**

```
$ sudo mn --test pingall --topo single,3
```



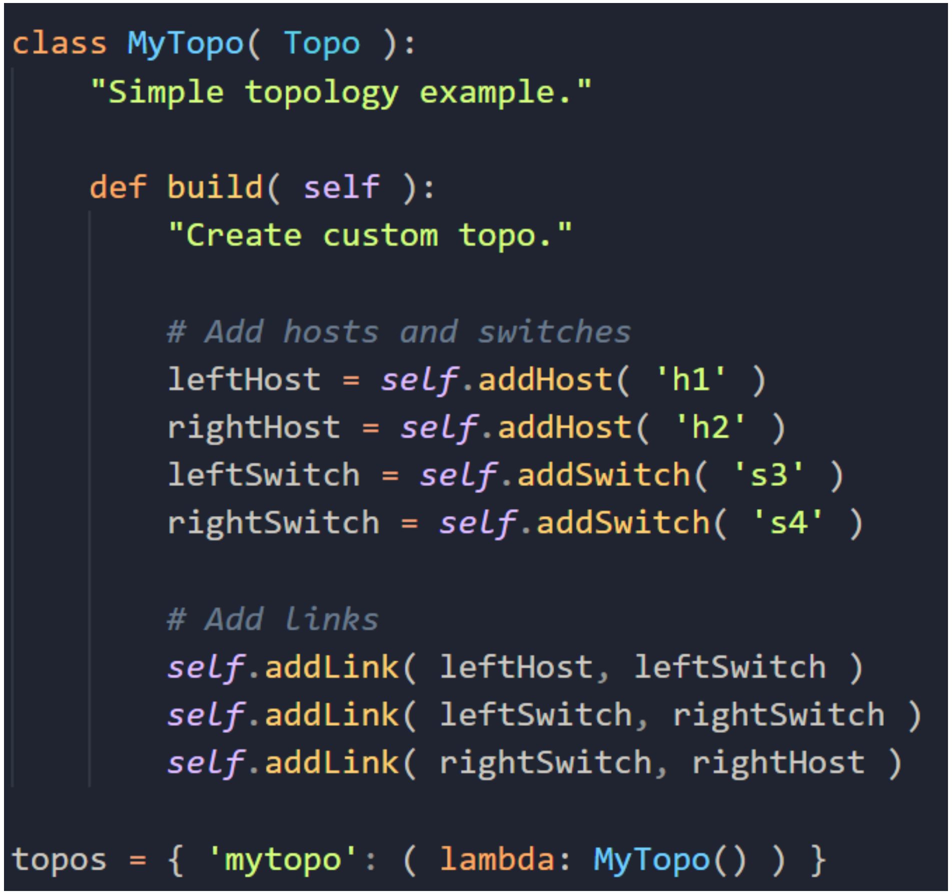
```
$ sudo mn --test pingall --topo linear,4
```



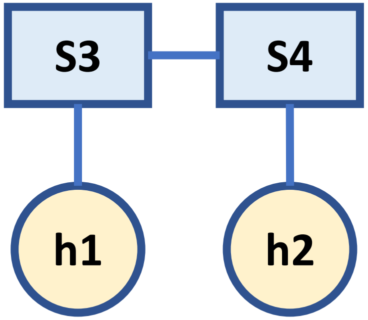
Advanced startup options

- **Custom topologies**

An example is provided in **custom/topo-2sw-2host.py**



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

    def build( self ):
        "Create custom topo."

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )

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



addSwitch()	adds a switch to a topology and returns the switch name.
addHost()	adds a host to a topology and returns the host name.
addLink()	adds a bidirectional link to a topology.

$ sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo

Openflow Lab

- Start a **remote controller** **in the first terminal**

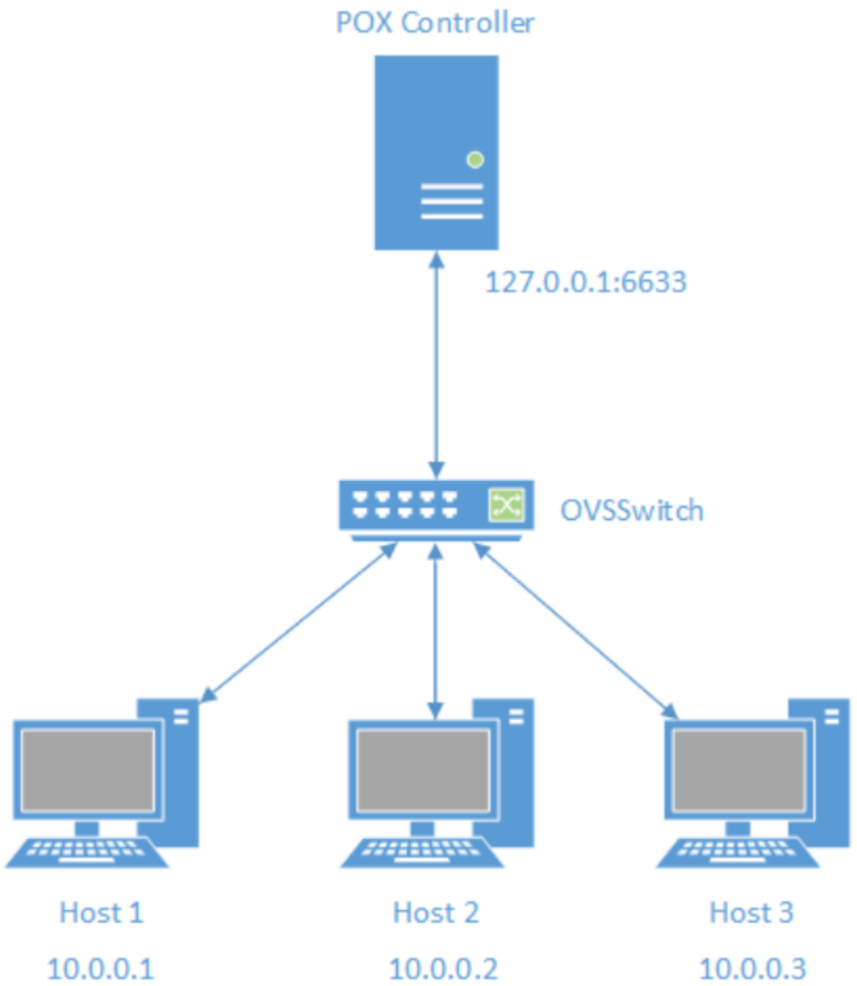
```
$ cd ~/pox
$ ./pox.py log.level --DEBUG misc.of_tutorial
```

- Start **mininet** **in the second terminal** (Create a topology)

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

Openflow Lab

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

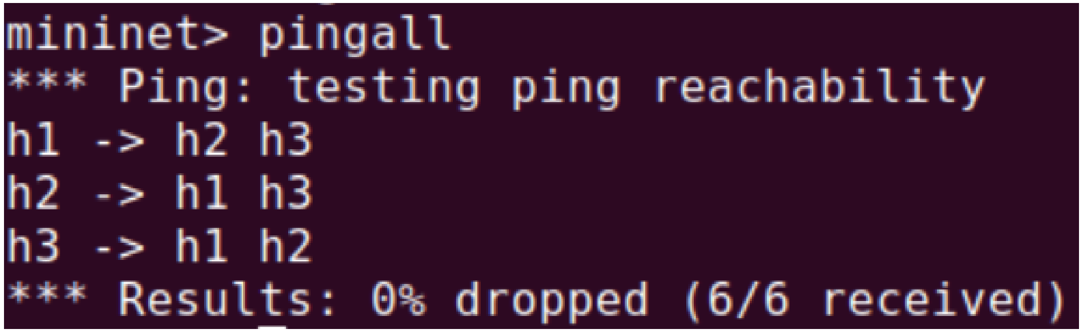


Openflow Lab

In the second terminal (mininet's terminal)

- check for connectivity

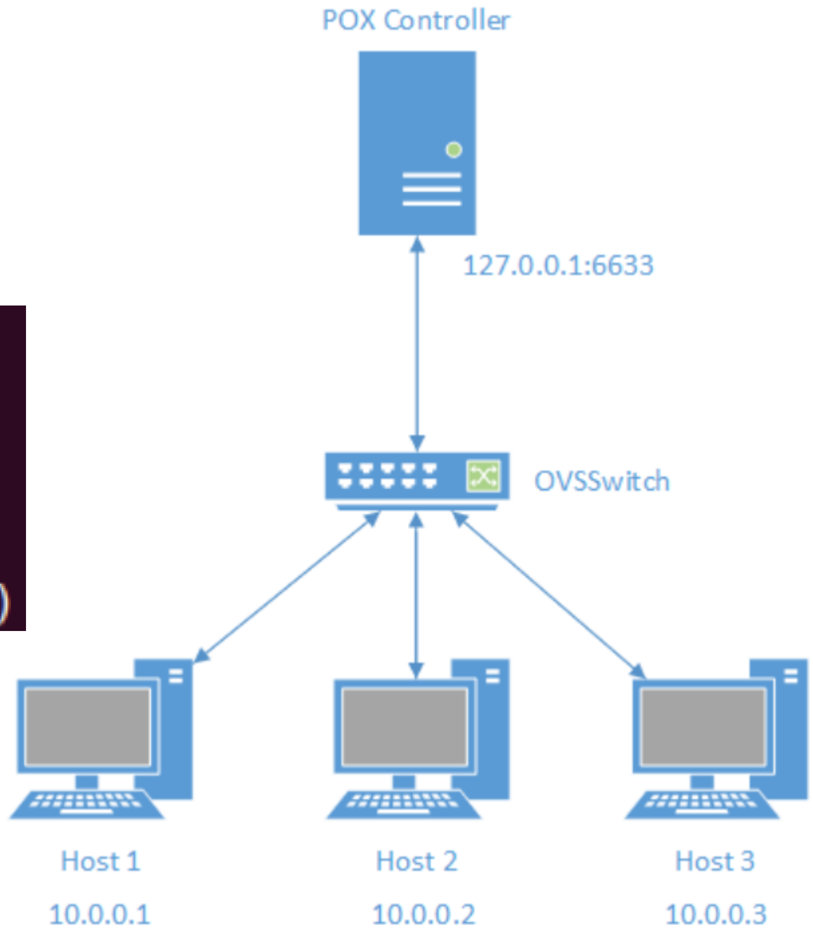
  mininet> pingall



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

- create xterm for each host

  mininet> xterm h1 h2 h3



Openflow Lab

- **In xterm for h2**

```
# tcpdump -X -n -i h2-eth0
```

- **In xterm for h3**

```
# tcpdump -X -n -i h3-eth0
```

- **In xterm for h1**

```
# ping -c1 10.0.0.2
```

(1) Make observations:

The ping packets are now going up to the controller, which then floods them out all interfaces except the sending one.

You should see identical ARP and ICMP packets corresponding to the ping in both xterms running tcpdump.

This is how a hub works;

it sends all packets to every port on the network.

tcpdump

基本選項

-n：以數字顯示，不對 IP 作反解，但仍顯示服務名稱

-nn：直接以 IP 及 port number 顯示，而非主機名與服務名稱

-i：指定要監控的網路介面，如 eth0、lo、any 等等

-c：監聽的封包數，如果沒有這個參數，tcpdump 會持續不斷的監聽，直到使用者輸入 ctrl+c 為止

-q：僅列出較為簡短的封包資訊，每一行的內容比較精簡

-X：可以列出十六進位 (hex) 以及 ASCII 的封包內容，對於監聽封包內容很有用

Openflow Lab

- **In xterm for h2**

```
# tcpdump -X -n -i h2-eth0
```

- **In xterm for h3**

```
# tcpdump -X -n -i h3-eth0
```

- **In xterm for h1**

```
# ping -c1 10.0.0.5
```

(2) Make observations:

You should see three unanswered ARP requests in the tcpdump xterms.

Openflow Lab

- Close all **xterm**

- Stop the **remote controller** **in the first terminal**

  Ctrl + c

- Close **mininet** **in the second terminal**

  ```
  mininet> exit
  $ sudo mn -c
  ```

Openflow Lab

Your job is to make a learning switch !

- **In the first terminal (controller's terminal)**

```
$ cd ~/pox/pox/misc
$ gedit of_tutorial.py
```

Install Gedit :

```
$ sudo apt-get install gedit
```

Openflow Lab

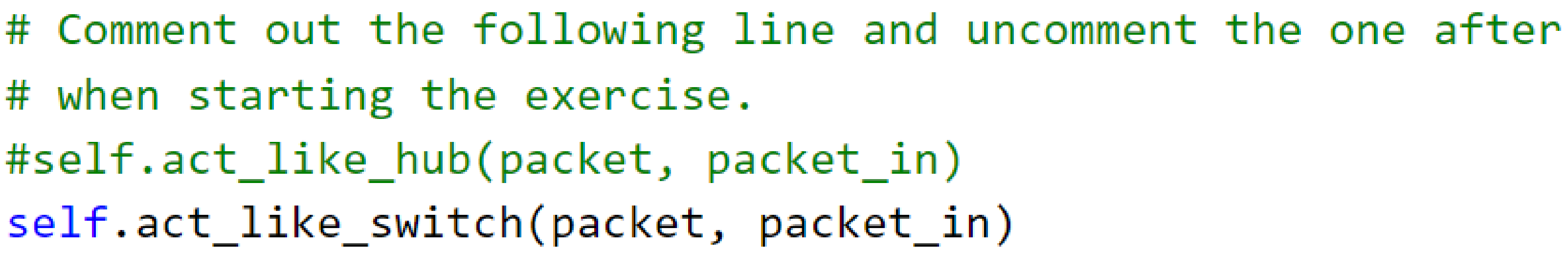
Step 1 : comment out act_like_hub()

uncomment act_like_switch()

- In function **_handle_PacketIn (self, event**)

  Comment self.act_like_hub(packet, packet_in)

  Uncomment self.act_like_switch(packet, packet_in)



```
# Comment out the following line and uncomment the one after
# when starting the exercise.
#self.act_like_hub(packet, packet_in)
self.act_like_switch(packet, packet_in)
```

Openflow Lab

Step 2 : edit act_like_switch() code (Make a learning switch)

(part A)

Send packet to switch output port from controller.

(part B)

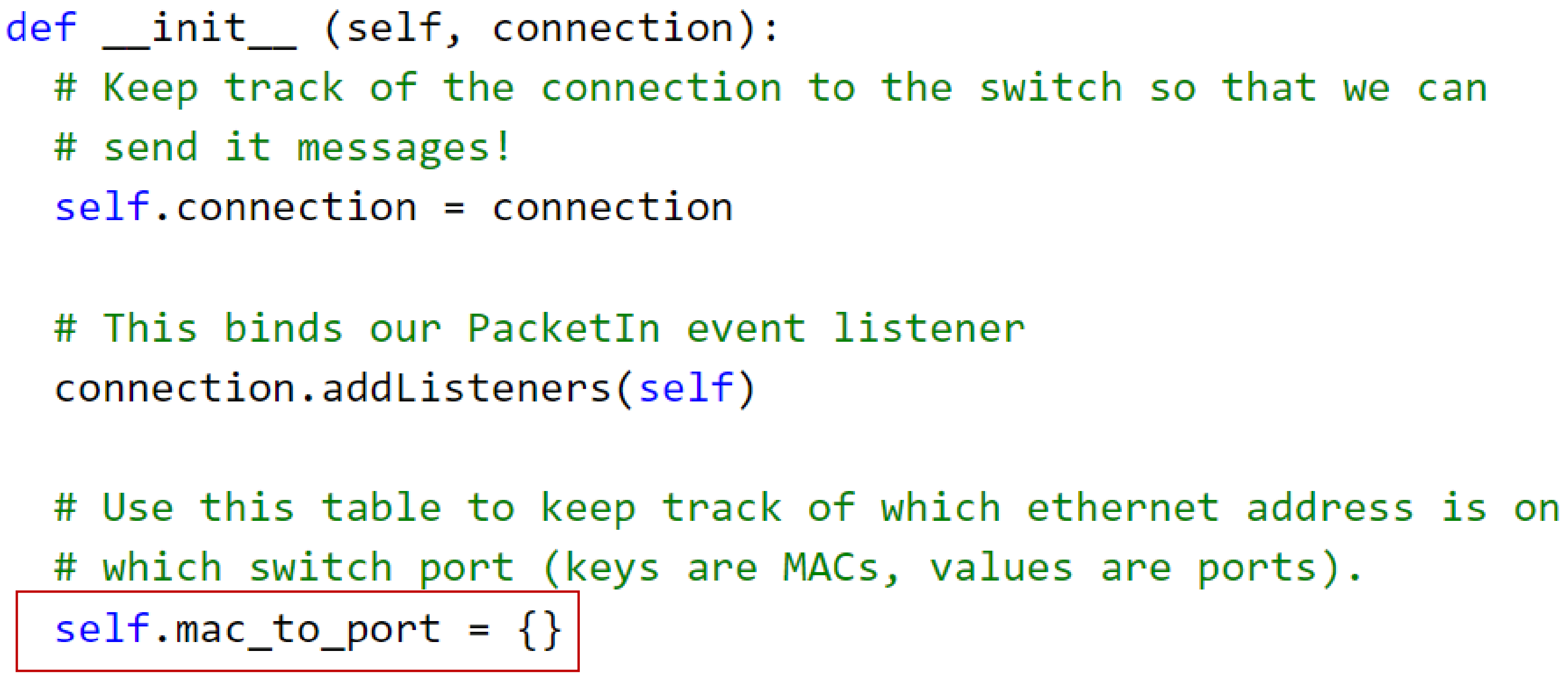
Write a flow entry in the openflow switch.

Learning Python

Python

- uses **indentation** to indicate a block of code.
- is dynamically typed. There is no need to pre-declare variables and types are automatically managed.
- has built-in hash tables, called **dictionaries**, and vectors, called **lists**.
- is object-oriented and introspective. You can easily **print the member variables and functions of an object** at runtime.

of_tutorial.py



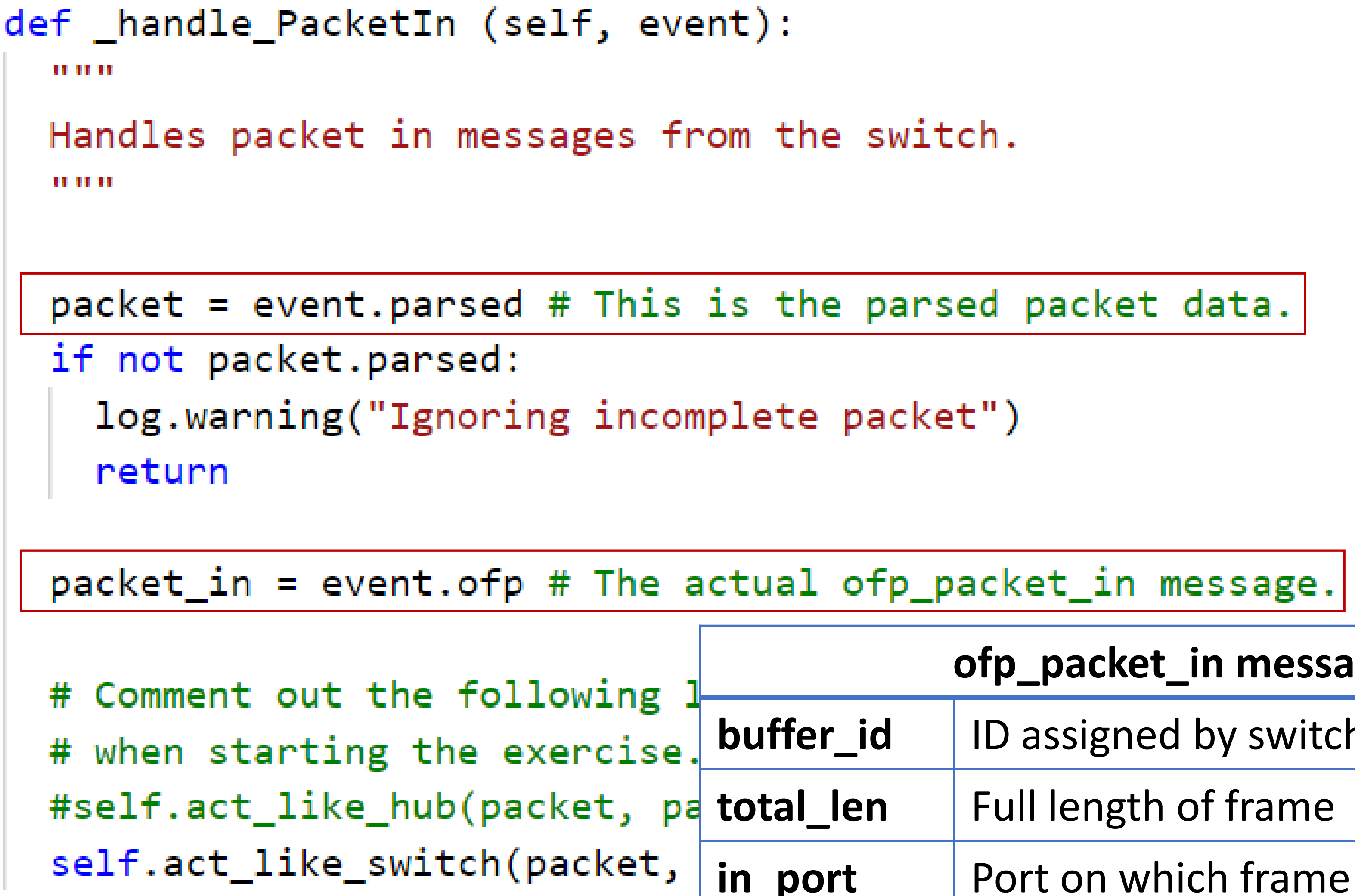
```
def __init__ (self, connection):
  # Keep track of the connection to the switch so that we can
  # send it messages!
  self.connection = connection

  # This binds our PacketIn event listener
  connection.addListeners(self)

  # Use this table to keep track of which ethernet address is on
  # which switch port (keys are MACs, values are ports).
  self.mac_to_port = {}
```

Create empty dictionary

d = {key1: value1, key2: value2, …}

mac_to_port = {mac1: port1, mac2: port2, …}

of_tutorial.py



```
def _handle_PacketIn (self, event):
  """
  Handles packet in messages from the switch.
  """

  packet = event.parsed # This is the parsed packet data.
  if not packet.parsed:
    log.warning("Ignoring incomplete packet")
    return

  packet_in = event.ofp # The actual ofp_packet_in message.

  # Comment out the following l
  # when starting the exercise.
  #self.act_like_hub(packet, pa
  self.act_like_switch(packet,
```

ofp_packet_in message	
buffer_id	ID assigned by switch
total_len	Full length of frame
in_port	Port on which frame was received
data	Ethernet frame

Learning Python

Common operations:

- To initialize a dictionary:

```
mac_table = { }
```

- To add an element to a dictionary:

```
mac_table[0x123] = 2
```

- To check for dictionary membership:

```
print(mac_table[0x123])  →  2
```

- To comment a line of code: #

Parsing Packets

- To see all members of a parsed packet object :

```
print dir(packet)
```

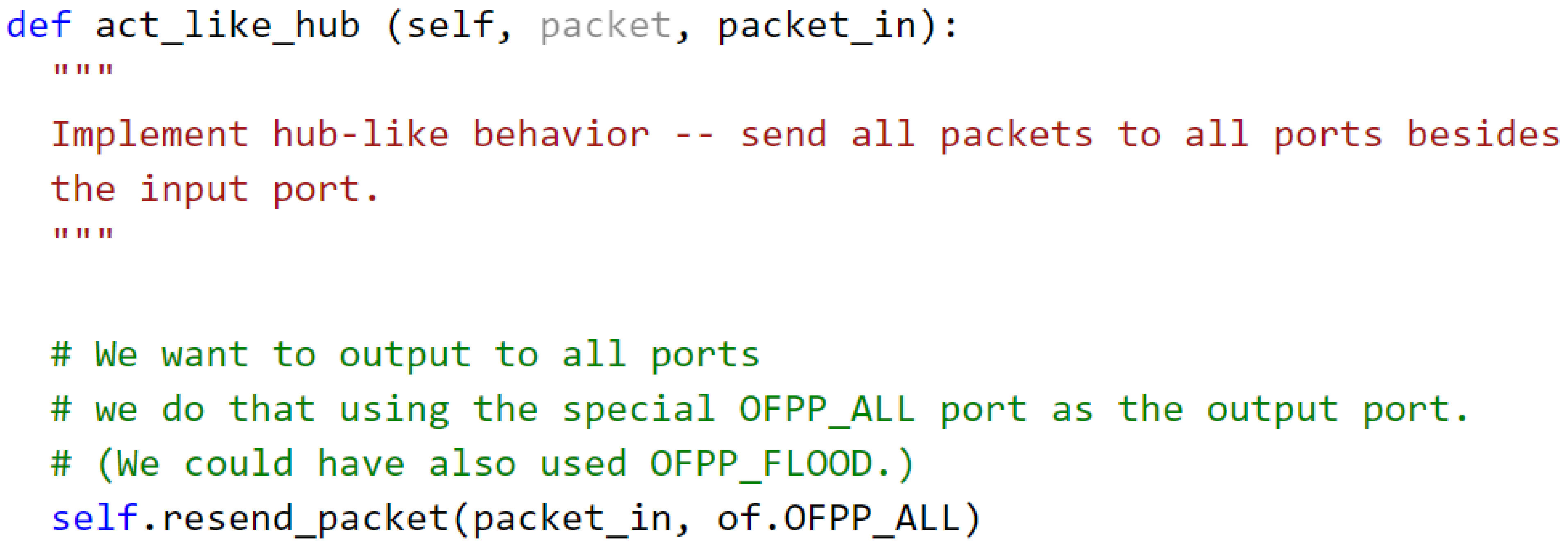
- To extract the source of a packet, use the dot notation :

```
packet.src
```

- To print a debug message :

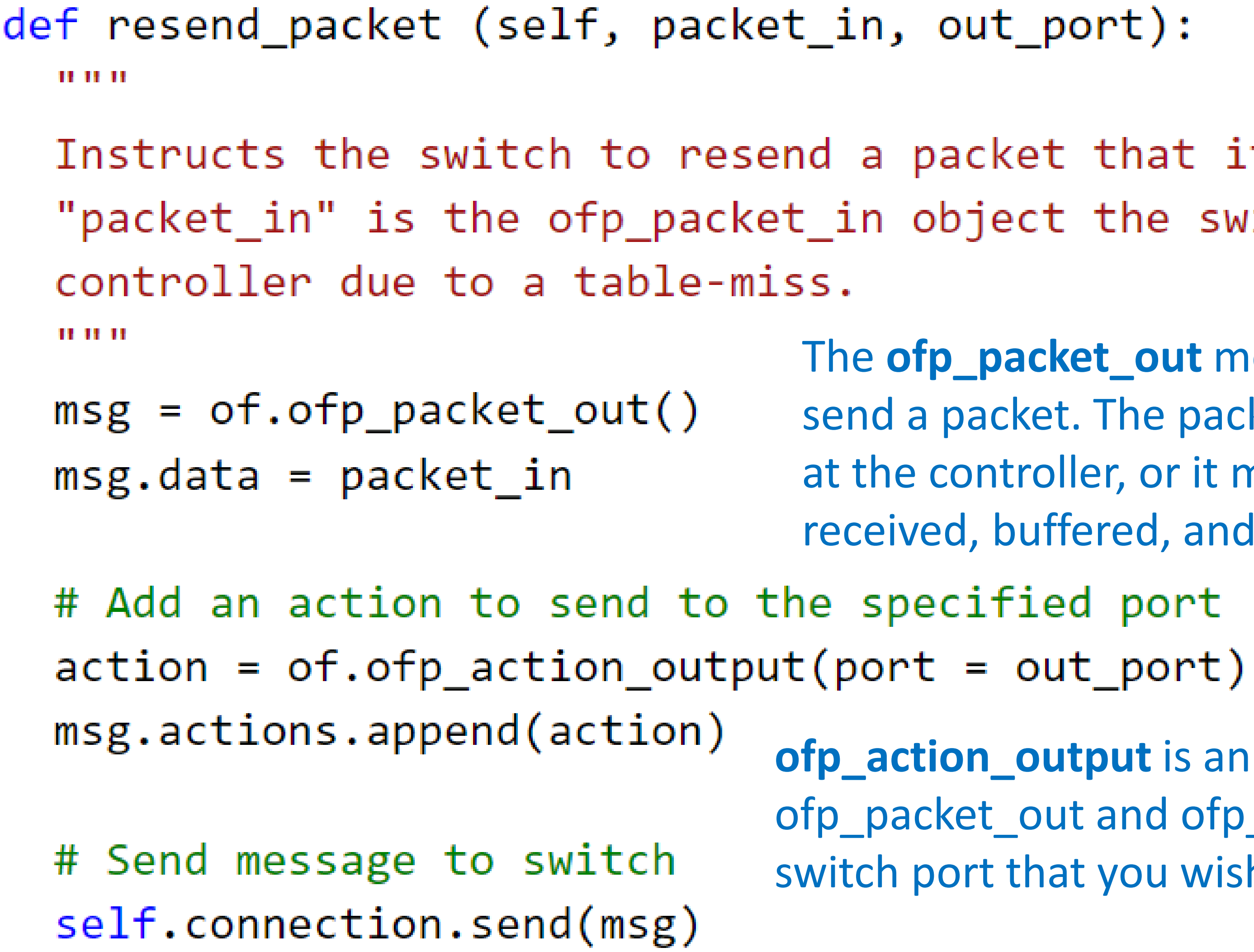
```
log.debug("Source MAC :  %s" % packet.src)
```

of_tutorial.py



```
def act_like_hub (self, packet, packet_in):
  """
  Implement hub-like behavior -- send all packets to all ports besides
  the input port.
  """

  # We want to output to all ports
  # we do that using the special OFPP_ALL port as the output port.
  # (We could have also used OFPP_FLOOD.)
  self.resend_packet(packet_in, of.OFPP_ALL)
```

of_tutorial.py



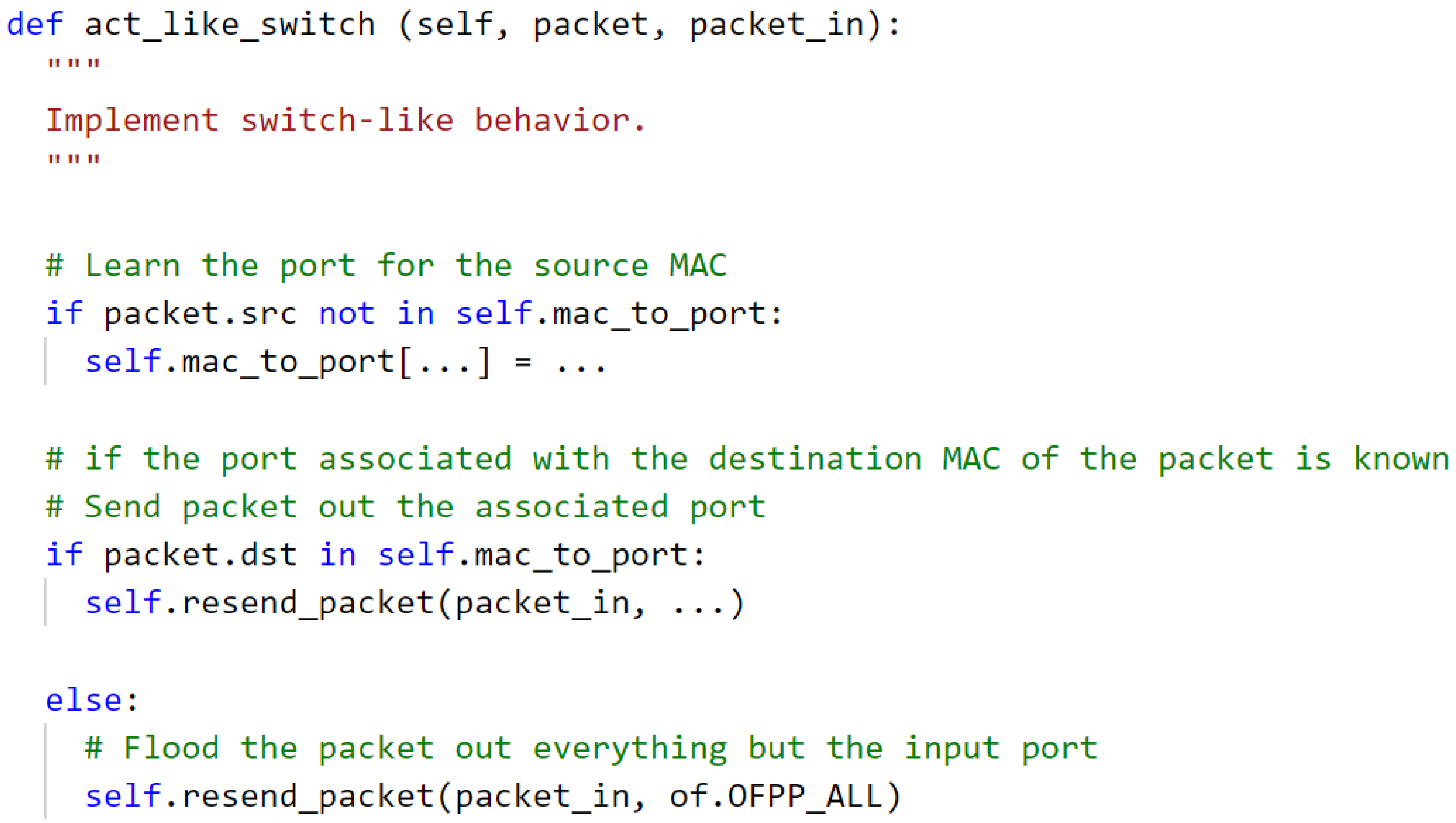
```
def resend_packet (self, packet_in, out_port):
    """
    Instructs the switch to resend a packet that it had sent to us.
    "packet_in" is the ofp_packet_in object the switch had sent to the
    controller due to a table-miss.
    """
    msg = of.ofp_packet_out()
    msg.data = packet_in

    # Add an action to send to the specified port
    action = of.ofp_action_output(port = out_port)
    msg.actions.append(action)

    # Send message to switch
    self.connection.send(msg)
```

The **ofp_packet_out** message instructs a switch to send a packet. The packet might be one constructed at the controller, or it might be one that the switch received, buffered, and forwarded to the controller.

ofp_action_output is an action for use with ofp_packet_out and ofp_flow_mod. It specifies a switch port that you wish to send the packet out of.

of_tutorial.py

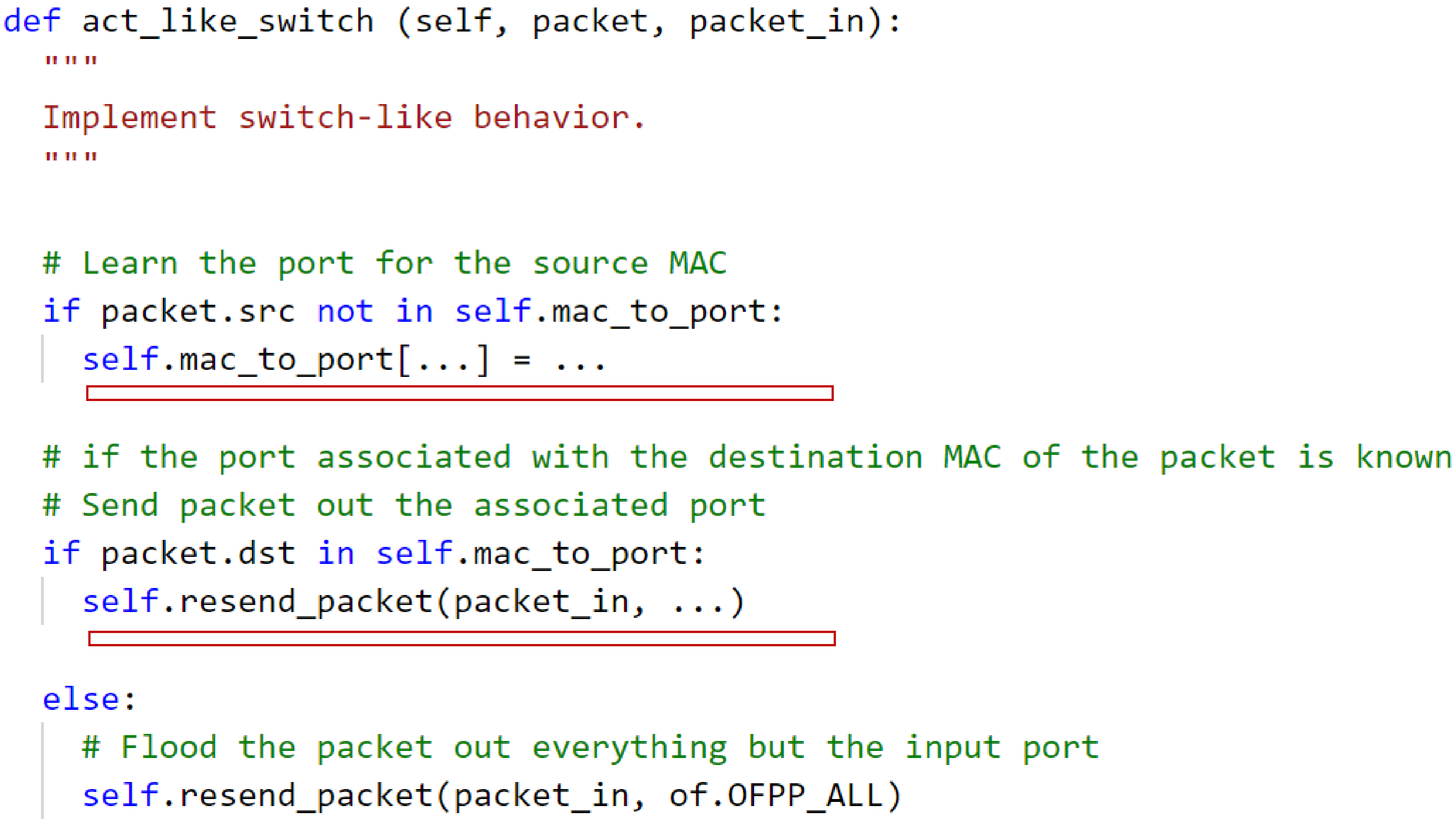
- **Part A :** Send packet to switch output port from controller.



```
def act_like_switch (self, packet, packet_in):
  """
  Implement switch-like behavior.
  """

  # Learn the port for the source MAC
  if packet.src not in self.mac_to_port:
    self.mac_to_port[...] = ...

  # if the port associated with the destination MAC of the packet is known
  # Send packet out the associated port
  if packet.dst in self.mac_to_port:
    self.resend_packet(packet_in, ...)

  else:
    # Flood the packet out everything but the input port
    self.resend_packet(packet_in, of.OFPP_ALL)
```

of_tutorial.py

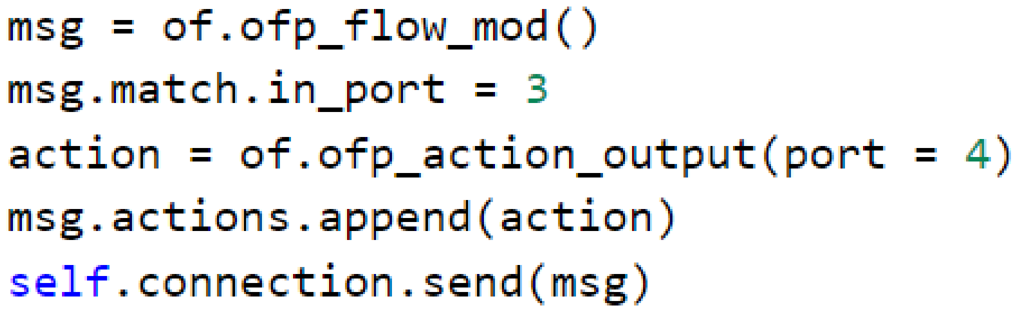
- **Part B :** Send packet to switch output port from controller and **write a flow entry** in the openflow switch.



```
def act_like_switch (self, packet, packet_in):
  """
  Implement switch-like behavior.
  """

  # Learn the port for the source MAC
  if packet.src not in self.mac_to_port:
    self.mac_to_port[...] = ...

  # if the port associated with the destination MAC of the packet is known
  # Send packet out the associated port
  if packet.dst in self.mac_to_port:
    self.resend_packet(packet_in, ...)

  else:
    # Flood the packet out everything but the input port
    self.resend_packet(packet_in, of.OFPP_ALL)
```

ofp_flow_mod

- This instructs a switch to **install a flow table entry**.
- Flow table entries match some fields of incoming packets, and executes some list of actions on matching packets.
- The match is described by an **ofp_match** object.
- Example : Create a flow_mod that sends packets from port 3 out of port 4.



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

ofp_match attribute	
dl_src	Ethernet source address
dl_dst	Ethernet destination address
in_port	Switch port number the packet arrived on

Openflow Lab

Step 3 : Restart POX and Mininet , check result

- mininet> h1 ping h2
- mininet> sh ovs-ofctl dump-flows s1 (show s1’s flow table)

(3) Make observations: what have changed?