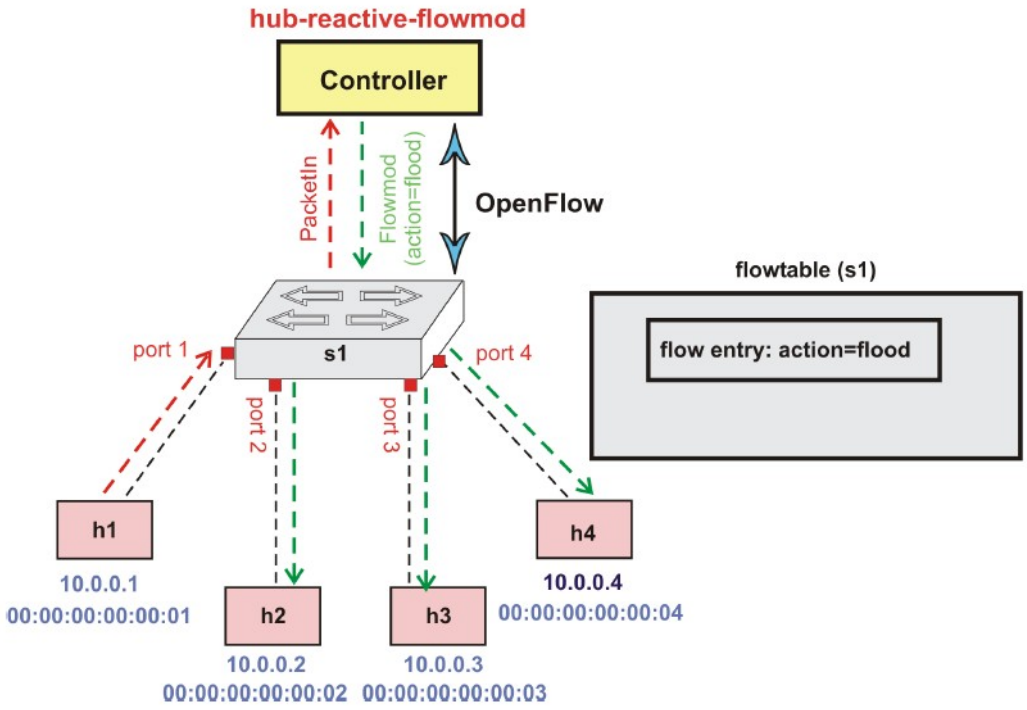
Διαχείριση Δικτύων Βασισμένων στο Λογισμικό
7ο εργαστήριο: "POX Controller"

ΟΝΟΜΑΤΕΠΩΝΥΜΟ:	Νικόλας Μαυρόπουλος
Α.Μ.:	21865

Part 1: Create a hub application.



A hub is used for connecting devices in LAN. When a packet arrives at hub, it is sent through all other ports (it means hubs works by flooding the packets) except the incoming port.
Perform the following steps and answer the questions in *blue*.
Step 1: Create the hub application in "/home/pox/pox/forwarding": Copy the file "hub-reactive-flowmod.py" in this directory and fill in the missing code as indicated in this file.

Write below the full code, highlighting with colour the code you have added:

```
from pox.core import core  # Central point for POX APIs
import pox.openflow.libopenflow_01 as of  # Import OpenFlow 1.0 module and rename it "of"

log = core.getLogger()  # Display messages
```

```
def _handle_PacketIn(event):  # Handler function which specifies what to do when PacketIn happens
    packet = event.parsed  # Extract Ethernet frame from event
    msg = of.ofp_flow_mod()  # Create an instruction "msg" which will be used for adding flow table entry into device later on
    # LAB: FILL THE PARENTHESIS IN THE NEXT LINE:
    action = of.ofp_action_output(port=of.OFPP_FLOOD)
    msg.actions.append(action)  # Specify flood action
    event.connection.send(msg)  # Send the instruction to device


def launch():  # Is used for initializing POX component
    core.openflow.addListenerByName("PacketIn", _handle_PacketIn)  # Specify handler function when PacketIn occurs
    log.info("Hub application is running.")  # Display messages
```

Step 2: Run the hub application. Be careful to be in the correct directory.

```
eirini@eirini-VirtualBox:~/pox$ ./pox.py pox.forwarding.hub-reactive-flowmod
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:forwarding.hub-reactive-flowmod:Hub application is running.
INFO:core:POX 0.5.0 (eel) is up.
```

Step 3: Create a topology of 4 hosts and 1 switch (use "- -controller remote option").

```
eirini@eirini-VirtualBox:~$ sudo mn --mac --topo single,4 --controller remote
[sudo] password for eirini:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Step 4: Test connectivity.

```
mininet> h1 ping -c4 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.654 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.097 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.087 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.071 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.071/0.227/0.654/0.246 ms
mininet>
```

***Question**: Why does the first packet take more time?*
The reason for first packet to take more time is, the routing decision happens only for first packet. Once the controller inserts the flow rule for the first packet, the switch buffers the flow rule in its flow table.

Step 5: Check flow table entry at the switch.

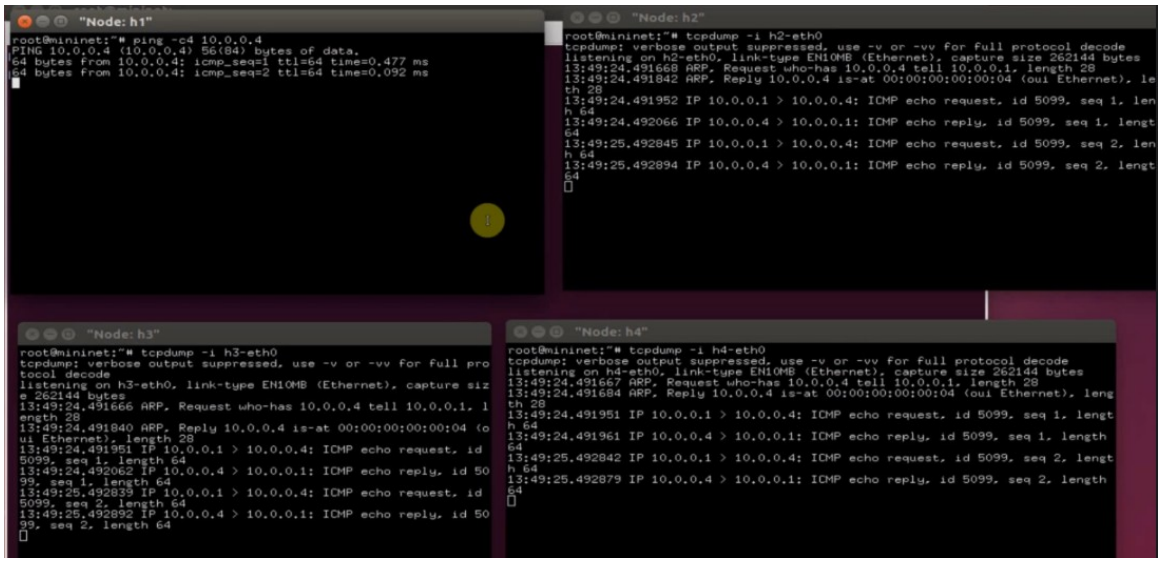
```
eirini@eirini-VirtualBox:~$ sudo ovs-ofctl dump-flows s1
```

***Question**: What do you see?*
I see 34 packets covering 2820 bytes have hit this rule. Also how long this flow entry has been in the table which is 11.683s.

```
mininet> sh ovs-ofctl dump-flows s1
 cookie=0x0, duration=11.683s, table=0, n_packets=34, n_bytes=2820, actions=FLOOD
```

Step 6: Ping from h1 to h4 and dump the output from h2, h3 and h4 to prove that flooding is indeed taking place (verify hub behavior). Use the command **xterm h1 h2 h3 h4** to open four terminals.

Provide the following four screenshots from your computer, to prove you have successfully created the hub application.



"Node: h1"

```
root@nikolas:~# ping -c4 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.333 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.063 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.061 ms

--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3067ms
rtt min/avg/max/mdev = 0.055/0.128/0.333/0.118 ms
root@nikolas:~#
```

"Node: h2"

```
root@nikolas:~# tcpdump -i h2-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:06:54.156592 IP 10.0.0.1 > 10.0.0.4: ICMP echo request, id 2259, seq 3, lengt
h 64
18:06:54.156624 IP 10.0.0.4 > 10.0.0.1: ICMP echo reply, id 2259, seq 3, length
64
18:06:55.171584 IP 10.0.0.1 > 10.0.0.4: ICMP echo request, id 2259, seq 4, lengt
h 64
18:06:55.171614 IP 10.0.0.4 > 10.0.0.1: ICMP echo reply, id 2259, seq 4, length
64
18:06:57.346424 ARP, Request who-has 10.0.0.4 tell 10.0.0.1, length 28
18:06:57.346446 ARP, Request who-has 10.0.0.1 tell 10.0.0.4, length 28
18:06:57.346451 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01 (oui Ethernet), leng
th 28
18:06:57.346490 ARP, Reply 10.0.0.4 is-at 00:00:00:00:00:04 (oui Ethernet), leng
th 28
```

"Node: h3"

```
root@nikolas:~# tcpdump -i h3-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h3-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:06:55.171585 IP 10.0.0.1 > 10.0.0.4: ICMP echo request, id 2259, seq 4, lengt
h 64
18:06:55.171614 IP 10.0.0.4 > 10.0.0.1: ICMP echo reply, id 2259, seq 4, length
64
18:06:57.346432 ARP, Request who-has 10.0.0.4 tell 10.0.0.1, length 28
18:06:57.346446 ARP, Request who-has 10.0.0.1 tell 10.0.0.4, length 28
18:06:57.346451 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01 (oui Ethernet), leng
th 28
18:06:57.346499 ARP, Reply 10.0.0.4 is-at 00:00:00:00:00:04 (oui Ethernet), leng
th 28
```

"Node: h4"

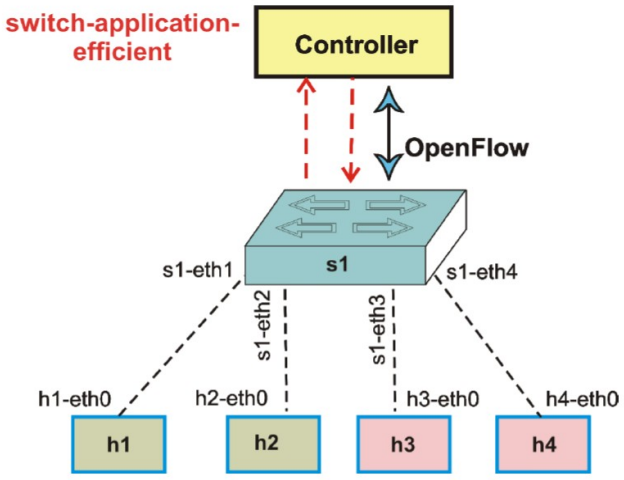
```
root@nikolas:~# tcpdump -i h4-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h4-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:06:57.346359 ARP, Request who-has 10.0.0.1 tell nikolas, length 28
18:06:57.346417 ARP, Request who-has nikolas tell 10.0.0.1, length 28
18:06:57.346422 ARP, Reply nikolas is-at 00:00:00:00:00:04 (oui Ethernet), lengt
h 28
18:06:57.346448 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01 (oui Ethernet), leng
th 28
```

***Question**: Replace "PacketIn" with "ConnectionUp" and "_handle_PacketIn" with "_handle_ConnectionUp". Do any other adjustments if needed and save the file as "hub-proactive-flowmod.py" and. What is different as compared to the previous reactive approach with respect to the establishment of the flow table entry?*

The duration, n_packets and n_bytes are different. They have all increased.

```
mininet> sh ovs-ofctl dump-flows s1
 cookie=0x0, duration=14.381s, table=0, n_packets=38, n_bytes=3140, actions=FLOOD
```

Part 2: Create a switch application.



When we will run our switch application "switch-application-efficient" on top of POX controller, two tables will be maintained. One table will be maintained at the controller and another table will be maintained at the switch. The table maintained at the controller will be "mac to port table" The table maintained at the switch will contain flow entries. Initially both tables will be empty.

Step 1: Create the switch application in "/home/pox/pox/forwarding": Copy the file "switch-application-efficient.py" in this directory and fill in the missing code as indicated in this file.

Write below the full code, highlighting with colour the code you have added:

```
# LAB: Add "eth.src" or "eth.dst" in the spaces _____ indicated and missing code in the
[------------]


from pox.core import core
from pox.lib.addresses import IPAddr, EthAddr  # Import IPAdrr & EthAddr class
import pox.openflow.libopenflow_01 as of



class Switch:
  def __init__(self, connection):
    self.connection = connection
    self.macToPort = {}  # Initialize macToPort dictionary

    connection.addListeners(self)

  def _handle_PacketIn(self, event):
    in_port = event.port  # Extract port from event & store it in variable
    dpid = event.dpid  # Extract dpid from event & store it in variable

    packet = event.parsed  # Extract frame from event & store it in variable
    eth = packet.find("ethernet")  # Extract ethernet header
    self.macToPort[eth.src] = in_port  # Insert entry based on source mac into dictionary
```

```
        # If destination mac is in dictionary, then send packet to corresponding port, otherwise flood
        if eth.dst in self.macToPort:
            out_port = self.macToPort[eth.dst]
        else:
            out_port = of.OFPP_FLOOD

        # Install flow entry into device:
        if out_port != of.OFPP_FLOOD:
            msg = of.ofp_flow_mod()
            msg.match = of.ofp_match()
            msg.match.dl_dst = eth.dst
            msg.match.in_port = event.port
            msg.idle_timeout = 10
            msg.hard_timeout = 30
            msg.actions.append(of.ofp_action_output(port=out_port))
            self.connection.send(msg)  # Create an instruction "msg" for sending packet out
        else:
            msg = of.ofp_packet_out()
            msg.actions.append(of.ofp_action_output(port=out_port))
            msg.data = event.ofp
            self.connection.send(msg)


def _handle_ConnectionUp(event):
    Switch(event.connection)  # Handler function which specifies what to do when ConnectionUp happens


def launch():  # Is used for initializing pox component
    core.openflow.addListenerByName("ConnectionUp", _handle_ConnectionUp)  # Specify handler function when ConnectionUp occurs
```

Provide screenshots for all the following steps:

Step 2: Run the switch application.

```
eirini@nikolas:~/pox$ ./pox.py pox.forwarding.switch-application-efficient
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
```

Step 3: Create a topology of 4 hosts and 1 switch (use “- -controller remote option”).

```
eirini@nikolas:~$ sudo mn --mac --topo single,4 --controller remote
[sudo] password for eirini:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Step 4: Test connectivity (ping to h4 from h1).

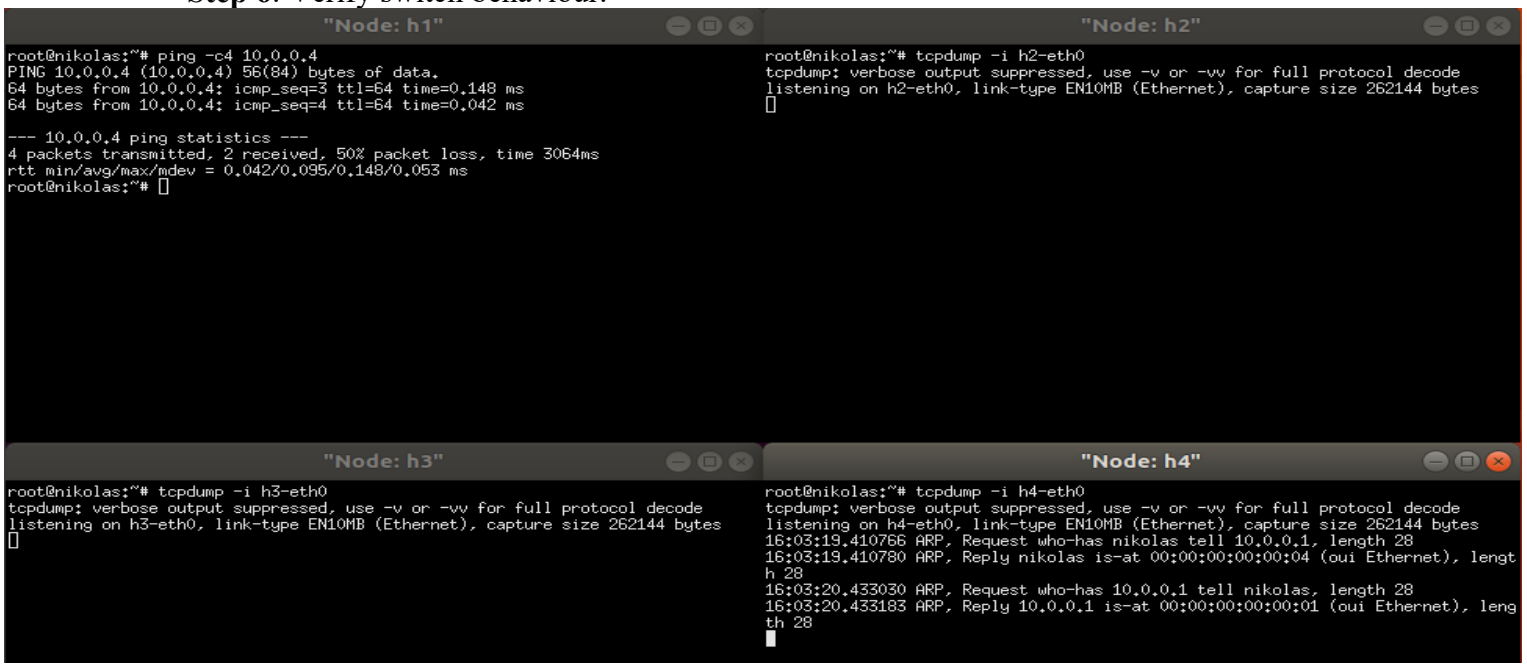
```
mininet> h1 ping -c4 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.042 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.041 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.040 ms

--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3055ms
rtt min/avg/max/mdev = 0.040/0.041/0.043/0.006 ms
mininet>
```

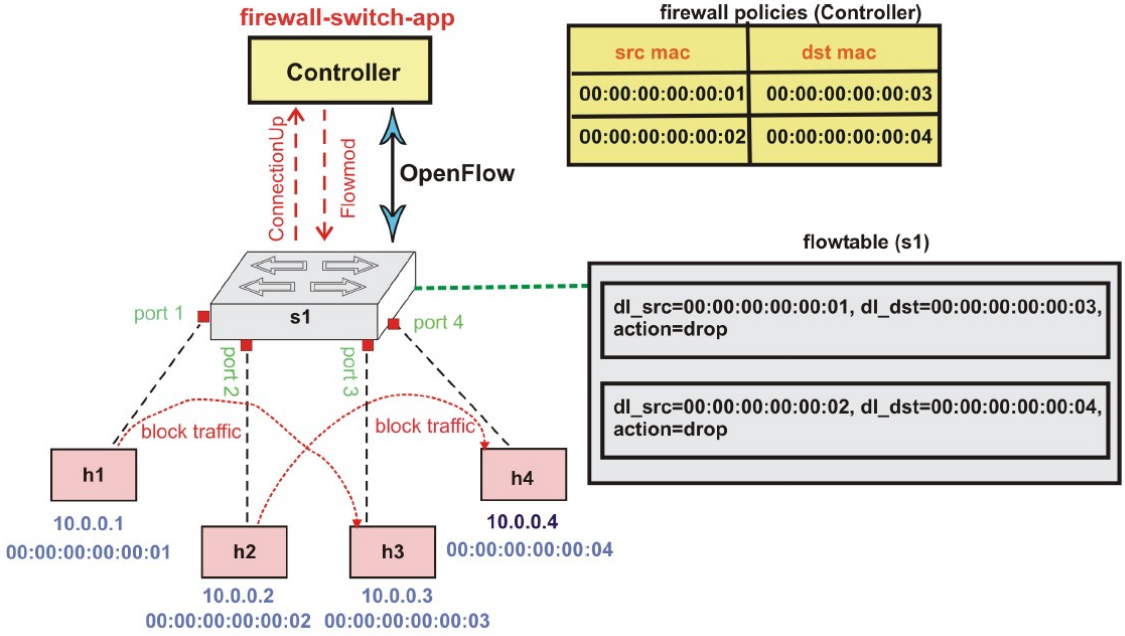
Step 5: Check flow table entry at the switch.

```
mininet> sh ovs-ofctl dump-flows s1
 cookie=0x0, duration=21.221s, table=0, n_packets=8, n_bytes=672, idle_timeout=10, hard_timeout=30,
in_port="s1-eth4",dl_dst=00:00:00:00:00:01 actions=output:"s1-eth1"
 cookie=0x0, duration=20.151s, table=0, n_packets=7, n_bytes=630, idle_timeout=10, hard_timeout=30,
in_port="s1-eth1",dl_dst=00:00:00:00:00:04 actions=output:"s1-eth4"
```

Step 6: Verify switch behaviour.



Part 3: Create a firewall application.



We want to block traffic from mac address "00:00:00:00:00:01" to "00:00:00:00:00:03" and from "00:00:00:00:00:02" to "00:00:00:00:00:04" (see "firewall-mac-policies" excel file).

Step 1: Create the firewall application in "/home/pox/pox/forwarding": Copy the file "firewall-switch-app.py" in this directory and fill in the missing code as indicated in this file.

Write below the full code, highlighting with colour the code you have added:

```
# LAB: Add "eth.src" or "eth.dst" in the spaces _____ indicated and missing code in the
[----------]

from pox.core import core
from pox.lib.addresses import IPAddr, EthAddr
import pox.openflow.libopenflow_01 as of
import os



class Switch:
  def __init__(self, connection):
    self.connection = connection
    self.macToPort = {}

    connection.addListeners(self)

  def _handle_PacketIn(self, event):
```

```
        in_port = event.port
        dpid = event.dpid
        packet = event.parsed
        eth = packet.find("ethernet")
        self.macToPort[eth.src] = in_port
        if eth.dst in self.macToPort:
            out_port = self.macToPort[eth.dst]
        else:
            out_port = of.OFPP_FLOOD

        if out_port != of.OFPP_FLOOD:
            msg = of.ofp_flow_mod()
            msg.match = of.ofp_match()
            msg.match.dl_dst = eth.dst
            msg.match.in_port = event.port
            msg.idle_timeout = 10
            msg.hard_timeout = 30
            msg.actions.append(of.ofp_action_output(port=out_port))
            msg.data = event.ofp
            self.connection.send(msg)

    else:
    msg = of.ofp_packet_out()
    msg.actions.append(of.ofp_action_output(port=out_port))
    msg.data = event.ofp
    self.connection.send(msg)



def _handle_ConnectionUp(event):
    policyFile = "%s/pox/pox/forwarding/firewall-mac-policies.csv" % os.environ['HOME']
    rules_file = open(policyFile, "r")  # Open file in read mode
    rules = [rule.strip() for rule in rules_file]  # Create list of rules from policy file

    # Install firewall rules when ConnectionUp event occurs
    for i in range(len(rules)):
        rule_list = rules[i].split(' ')
        fw_add_rule = of.ofp_flow_mod()
        fw_add_rule.match = of.ofp_match()
        fw_add_rule.match.dl_src = EthAddr(rule_list[0])
        fw_add_rule.match.dl_dst = EthAddr(rule_list[1])
        event.connection.send(fw_add_rule)

    Switch(event.connection)



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

Provide screenshots for all the following steps:

Step 2: Run the firewall application.

```
eirini@nikolas:~/pox$ ./pox.py pox.forwarding.firewall-switch-app
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
```

Step 3: Create a topology of 4 hosts and 1 switch (use "- -controller remote option").

```
eirini@nikolas:~$ sudo mn --mac --topo single,4 --controller remote
[sudo] password for eirini:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Step 4: Test connectivity (ping to h4 from h1).

```
mininet> h1 ping -c4 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=46.1 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.047 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.045 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.044 ms

--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3046ms
rtt min/avg/max/mdev = 0.044/11.581/46.189/19.980 ms
mininet>
```

Step 5: Check flow table entry at the switch.

```
mininet> sh ovs-ofctl dump-flows s1
 cookie=0x0, duration=180.427s, table=0, n_packets=0, n_bytes=0, dl_src=00:00:00:00:00:01,
dl_dst=00:00:00:00:00:03 actions=drop
 cookie=0x0, duration=180.424s, table=0, n_packets=0, n_bytes=0, dl_src=00:00:00:00:00:02,
dl_dst=00:00:00:00:00:04 actions=drop
```

Step 6: Verify firewall behaviour using ping commands among all pairs.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X h4
h2 -> h1 h3 X
h3 -> X h2 h4
h4 -> h1 X h3
*** Results: 33% dropped (8/12 received)
```