



---

## EE 555 FINAL PROJECT: OPENFLOW MININET

---



### **SUBMITTED BY:**

Amrutha Kumaraswamy (akumaras@usc.edu)  
Anjana Nair (anjanana@usc.edu)

**AIM:**

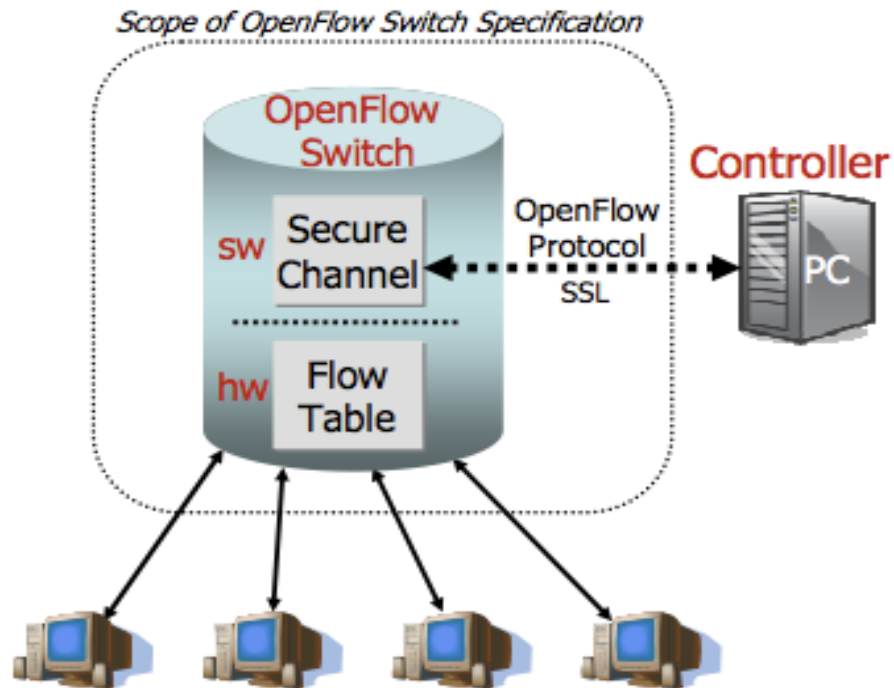
To use OpenFlow and Mininet tools to build a virtual SDN network. We learn the basics on using the tools and create a learning switch. After set up and observing how the hub works, we create a learning switch, then we build a static router implementation for a given topology using POX (a Python-based SDN controller platform).

**INTRODUCTION:**

Openflow is a communications protocol that gives access to the forwarding plane of a network switch or router over the network. It is used to govern the communication between a controller and the switch in a software defined network (SDN) environment. It is an open standard network protocol used to manage traffic between commercial Ethernet switches, routers and wireless access points.

OpenFlow enables software-defined networking (SDN) for programmable networks and is based on an Ethernet switch, with an internal flow-table and a standardized interface to add and remove flow entries. The controllers are distinct from the switches. This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols.

Compared to a conventional switch the OpenFlow switch separates the control path and the data path. The controller makes the routing decisions, while the data remains on the switch. It is through this protocol that the switch and the controller communicate, which is the basis of Software Define Networks.



## MININET

Mininet is a software emulator for prototyping a large network on a single machine. It runs a collection of end-hosts, switches, routers, and links on a single Linux kernel. Mininet can be used to quickly create a realistic virtual network running actual kernel, switch and software application code on a personal computer. Mininet allows the user to quickly create, interact with, customize and share a software-defined network (SDN) prototype to simulate a network topology that uses Openflow switches.

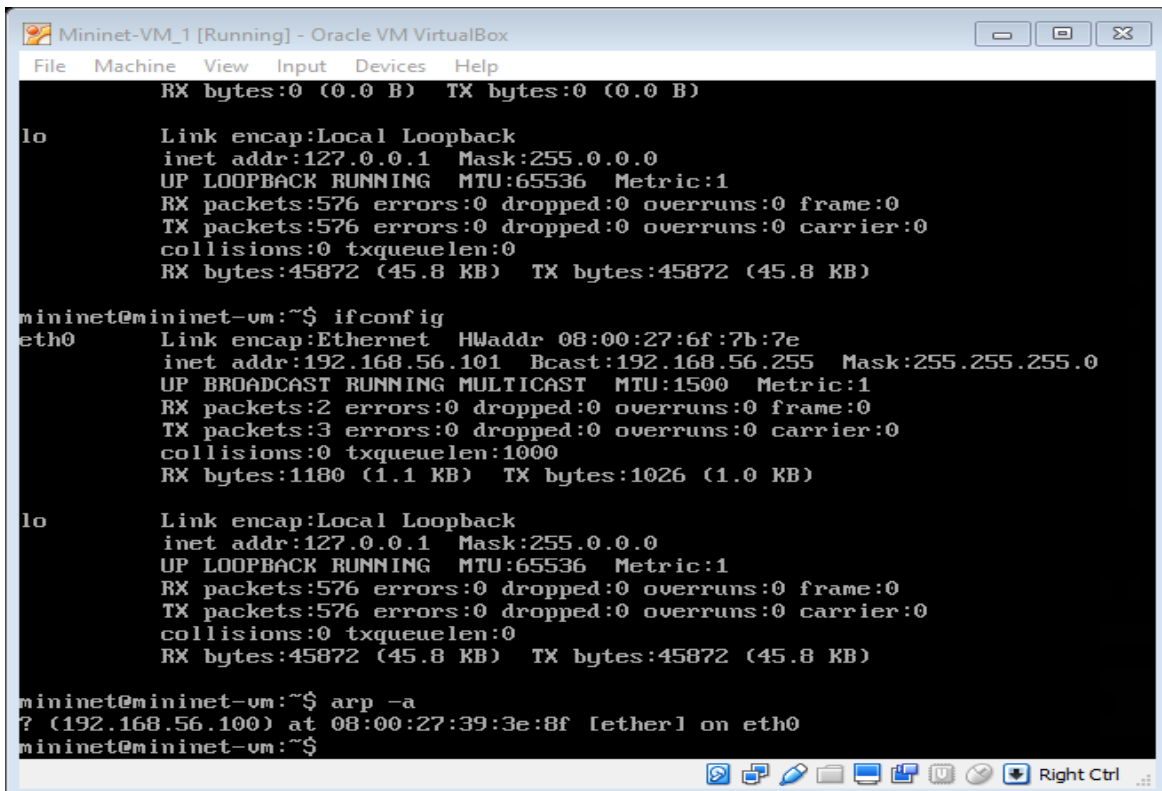
A Mininet host behaves just like a real machine; you can ssh into it (if you start up sshd and bridge the network to your host) and run arbitrary programs (including anything that is installed on the underlying Linux system). In short, Mininet's virtual hosts, switches, links, and controllers are the real thing – they are just created using software rather than hardware – and for the most part their behavior is like discrete hardware elements.

## IMPLEMENTATION

### INSTALLING REQUIRED SOFTWARE

- A virtual machine image (OVF) is downloaded in the beginning

- A virtual box is installed and the OVF image is imported into the virtual box
- Select the VM. Go to Settings tab. Go to Network ->Adapter 2. Select the Enable Adapter box and attach it to the “host-only-network”. Make sure the DHCP client is set
- Press the “**Start**” icon
- In the VM console window, log in with the user name and password both as “**mininet**”
- An X11 session can be started in the VM console window by typing “**startx**”
- Add a host-only interface to your VM by typing the command “**sudo dhclient eth1**”



```

Mininet-VM_1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
RX bytes:0 (0.0 B) TX bytes:0 (0.0 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:576 errors:0 dropped:0 overruns:0 frame:0
        TX packets:576 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:45872 (45.8 KB) TX bytes:45872 (45.8 KB)

mininet@mininet-vm:~$ ifconfig
eth0    Link encap:Ethernet HWaddr 08:00:27:6f:7b:7e
        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:2 errors:0 dropped:0 overruns:0 frame:0
        TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:1180 (1.1 KB) TX bytes:1026 (1.0 KB)

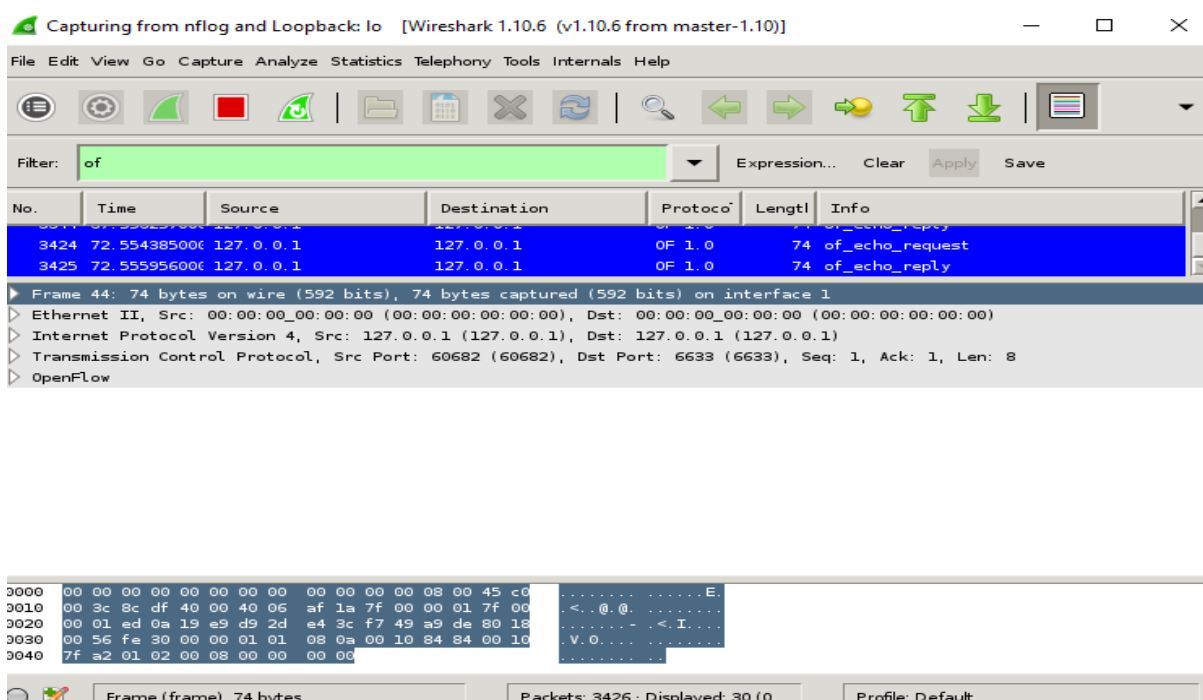
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:65536 Metric:1
        RX packets:576 errors:0 dropped:0 overruns:0 frame:0
        TX packets:576 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:45872 (45.8 KB) TX bytes:45872 (45.8 KB)

mininet@mininet-vm:~$ arp -a
? (192.168.56.100) at 08:00:27:39:3e:8f [ether] on eth0
mininet@mininet-vm:~$

```

```
mininet@mininet-vm: ~
mininet@192.168.56.101's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic i686)

 * Documentation:  https://help.ubuntu.com/
Last login: Sat Apr 22 20:24:08 2017 from 192.168.56.1
mininet@mininet-vm:~$ sudo wireshark &
[1] 5057
mininet@mininet-vm:~$ controller ptcp:
Apr 22 20:57:43|00001|vconn_tcp|ERR|ptcp:: bind: Address already in use
Apr 22 20:57:43|00002|controller|ERR|ptcp:: connect: Address already in use
controller: no active or passive switch connections
mininet@mininet-vm:~$ sudo controller ptcp:
Apr 22 20:58:18|00001|vconn_tcp|ERR|ptcp:: bind: Address already in use
Apr 22 20:58:18|00002|controller|ERR|ptcp:: connect: Address already in use
controller: no active or passive switch connections
mininet@mininet-vm:~$ controller ptcp:
Apr 22 21:00:05|00001|vconn_tcp|ERR|ptcp:: bind: Address already in use
Apr 22 21:00:05|00002|controller|ERR|ptcp:: connect: Address already in use
controller: no active or passive switch connections
mininet@mininet-vm:~$ controller ptcp:
Apr 22 21:01:11|00001|vconn_tcp|ERR|ptcp:: bind: Address already in use
Apr 22 21:01:11|00002|controller|ERR|ptcp:: connect: Address already in use
controller: no active or passive switch connections
mininet@mininet-vm:~$
```



## CONTROLLER USED: POX

POX is a Python-based SDN controller platform geared towards research and education. We are provided with starter code for a hub controller. After getting familiar with it, the provided hub was modified to act as an L2 learning switch. In this application, the switch will examine

each packet and learn the source-port mapping. Thereafter, the source MAC address will be associated with the port. If the destination of the packet is already associated with some port, the packet will be sent to the given port, else it will be flooded on all ports of the switch.

This is then converted into a flow-based switch, where seeing a packet with a known source and destination causes a flow entry to get pushed down. A mininet configuration below is loaded for both switch and hub using command:

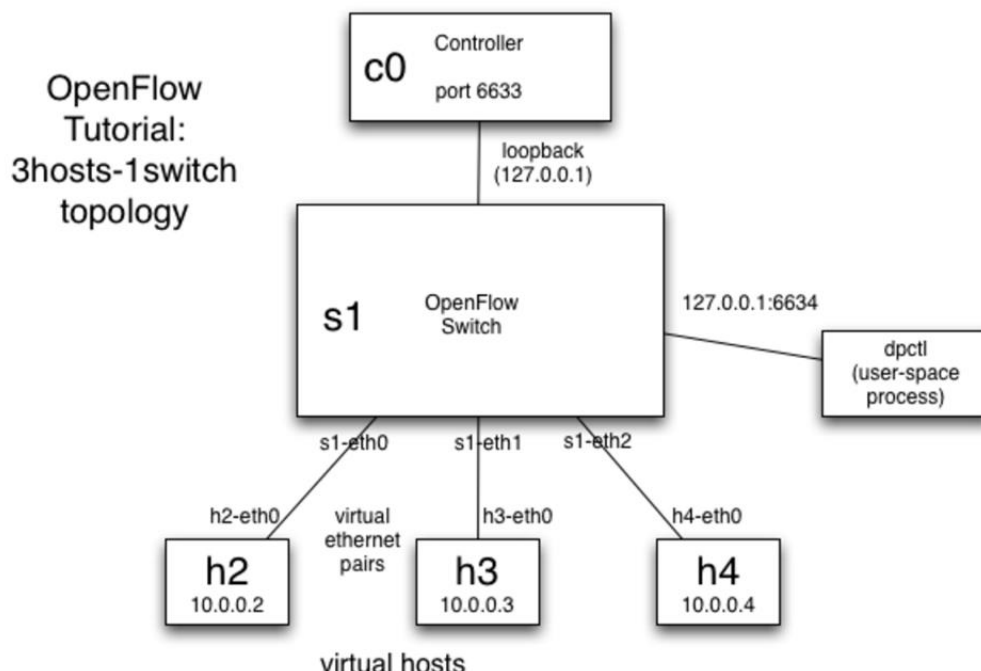
```
sudo mn --topo single,3 --mac --switch ovsk --controller remote
```

## PART 1: HUB & SWITCH IMPLEMENTATION

By running the below code the python file with each time functioning as hub or

a switch: (by commenting out the line which calls the each of the functionality)

```
mininet@mininet-vm:~/pox$ ./pox.py log.level --DEBUG misc.of_switch
```

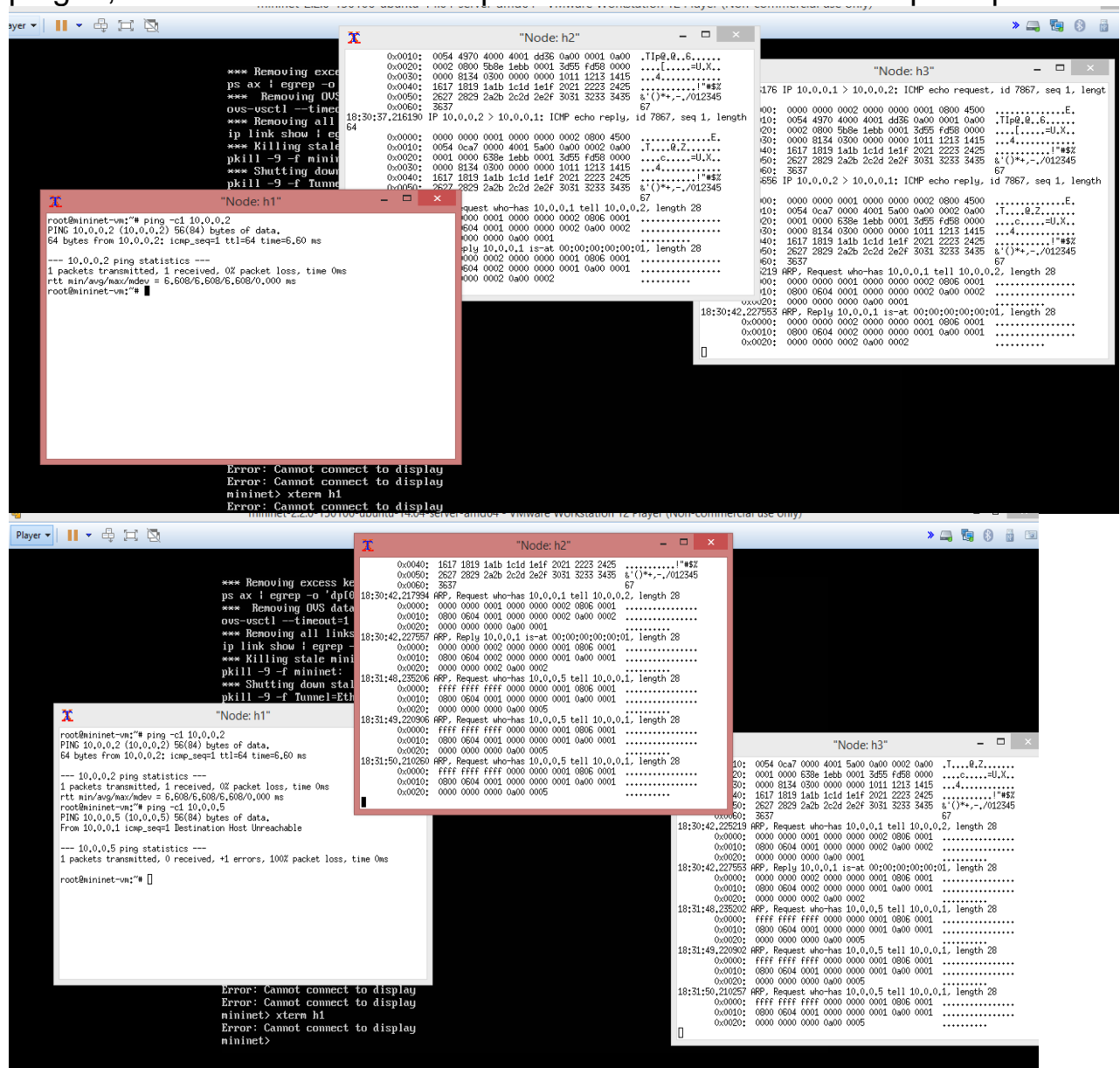


The controller performs the following actions:

1. It will construct ARP replies and forward them out the appropriate ports when Ethernet broadcasts are forwarded to it.
2. Once ARP has been handled, we need to handle routing for the static configuration.
3. It may receive ICMP echo (ping) requests for each switch, which will be responded to with ICMP network unreachable messages.

## Hub Implementation

On running the hub code and using the xterm and terminal to verify the behavior we get the following output. Here all the incoming packets are sent out to all the ports. This is verified by tcpdump. If unknown host is pinged, then there are 3 ARP requests seen on each host in tcpdump.



```
*** Removing excess ke
ps ax | egrep -o 'dpl0
*** Removing OUS data
ous-vscctl --timeout=1
*** Removing all links
ip link show | egrep -
*** Killing stale mini
pkill -9 -f mininet:
*** Shutting down stal
pkill -9 -f Tunnel-Eth

Node: h1
root@mininet-vms:~# ping -c1 10.0.0.2
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=6.50 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/ndev = 6.508/6.508/6.508/0.000 ms
root@mininet-vms:~#

Node: h2
0x0010: 0054 4970 4000 4001 d536 0a00 0001 0a00 .....T...0...6.....
0x0020: 0002 0800 5b8e 1ebb 0001 3d55 fd58 0000 .....[.....=U.X..
0x0030: 0000 8134 0300 0000 0000 1011 1213 1415 .....4.....
0x0040: 1517 1819 1a1b 1c1d 1e1f 2021 2223 2425 .....!*"#$%&'()*+,-./012345
0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 .....67
0x0060: 3637 .....89
18:30:37.216190 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 7867, seq 1, length 64
0x0000: 0000 0000 0001 0000 0000 0002 0800 4500 .....E.....
0x0010: 0054 0ca7 0000 4001 5a00 0a00 0002 0a00 .....T...0.Z.....
0x0020: 0001 0000 638e 1ebb 0001 3d55 fd58 0000 .....c.....=U.X..
0x0030: 0000 8134 0300 0000 0000 1011 1213 1415 .....4.....
0x0040: 1517 1819 1a1b 1c1d 1e1f 2021 2223 2425 .....!*"#$%&'()*+,-./012345
0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 .....67
0x0060: 3637 .....89
18:30:42.227553 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000: 0000 0000 0001 0000 0000 0002 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0002 0a00 0002 .....
0x0020: 0000 0000 0000 0a00 0001 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:30:42.227553 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000: 0000 0000 0000 0000 0000 0002 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0002 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0001 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....

Node: h3
176 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 7867, seq 1, length 64
0x0000: 0000 0000 0002 0000 0000 0001 0800 4500 .....E.....
0x0010: 0054 4970 4000 4001 d536 0a00 0001 0a00 .....T...0.Z.....
0x0020: 0002 0800 5b8e 1ebb 0001 3d55 fd58 0000 .....[.....=U.X..
0x0030: 0000 8134 0300 0000 0000 1011 1213 1415 .....4.....
0x0040: 1517 1819 1a1b 1c1d 1e1f 2021 2223 2425 .....!*"#$%&'()*+,-./012345
0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 .....67
0x0060: 3637 .....89
18:30:42.227553 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000: 0000 0000 0001 0000 0000 0002 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0002 0a00 0002 .....
0x0020: 0000 0000 0000 0a00 0001 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....

Node: h2
0x0040: 1517 1819 1a1b 1c1d 1e1f 2021 2223 2425 .....!*"#$%&'()*+,-./012345
0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 .....67
0x0060: 3637 .....89
18:30:42.217994 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000: 0000 0000 0001 0000 0000 0002 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0002 0a00 0002 .....
0x0020: 0000 0000 0000 0a00 0001 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:30:42.227557 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000: 0000 0000 0002 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0002 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0002 0a00 0002 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:31:48.235206 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0005 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:31:49.220906 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0005 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:31:50.210250 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0005 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....

Node: h1
root@mininet-vms:~# ping -c1 10.0.0.2
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=6.50 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/ndev = 6.508/6.508/6.508/0.000 ms
root@mininet-vms:~# ping -c1 10.0.0.5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data:
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable

--- 10.0.0.5 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
root@mininet-vms:~#

Node: h2
0x0040: 1517 1819 1a1b 1c1d 1e1f 2021 2223 2425 .....!*"#$%&'()*+,-./012345
0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 .....67
0x0060: 3637 .....89
18:30:42.227553 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000: 0000 0000 0001 0000 0000 0002 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0002 0a00 0002 .....
0x0020: 0000 0000 0000 0a00 0001 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:30:42.227553 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000: 0000 0000 0000 0000 0000 0002 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0002 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0001 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:31:48.235206 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0005 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:31:49.220906 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0005 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:31:50.210257 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0005 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....

Node: h3
10: 0054 0ca7 0000 4001 5a00 0a00 0002 0a00 .....T...0.Z.....
20: 0001 0000 638e 1ebb 0001 3d55 fd58 0000 .....c.....=U.X..
30: 0000 8134 0300 0000 0000 1011 1213 1415 .....4.....
40: 1517 1819 1a1b 1c1d 1e1f 2021 2223 2425 .....!*"#$%&'()*+,-./012345
50: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 .....67
60: 3637 .....89
18:30:42.225213 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000: 0000 0000 0001 0000 0000 0002 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0002 0a00 0002 .....
0x0020: 0000 0000 0000 0a00 0001 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:30:42.227553 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000: 0000 0000 0000 0000 0000 0002 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0002 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0001 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:31:48.235202 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0005 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:31:49.220902 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0005 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
18:31:50.210257 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0005 .....
0x0030: 0000 0000 0000 0000 0001 0806 0001 .....
0x0040: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0050: 0000 0000 0000 0000 0001 0a00 0001 .....
0x0060: 0000 0000 0000 0000 0001 0a00 0001 .....
```

## Flow Mods for s1 :

```
mininet@mininet-vm: ~  
state:      0  
current:    10GB-FD COPPER  
speed: 10000 Mbps now, 0 Mbps max  
LOCAL(s1):  addr:16:da:2b:98:b9:4f  
config:     0  
state:      0  
speed: 0 Mbps now, 0 Mbps max  
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0  
mininet@mininet-vm:~$ ovs-ofctl dump-flows s1  
ovs-ofctl: /var/run/openvswitch/s1.mgmt: failed to open socket (Permission denied)  
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1  
NXST_FLOW reply (xid=0x4):  
mininet@mininet-vm:~$ ovs-ofctl add-flow s1 in_port=1,actions=output:2  
ovs-ofctl: /var/run/openvswitch/s1.mgmt: failed to open socket (Permission denied)  
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=1,actions=output:2  
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=2,actions=output:1  
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1  
NXST_FLOW reply (xid=0x4):  
  cookie=0x0, duration=36.737s, table=0, n_packets=0, n_bytes=0, idle_age=36, in_port=  
1 actions=output:2  
  cookie=0x0, duration=17.738s, table=0, n_packets=0, n_bytes=0, idle_age=17, in_port=  
2 actions=output:1  
mininet@mininet-vm:~$
```

```
mininet@mininet-vm: ~/pox  
mininet@mininet-vm:~$ git clone http://github.com/noxrepo/pox  
fatal: destination path 'pox' already exists and is not an empty directory.  
mininet@mininet-vm:~$  
mininet@mininet-vm:~$ cd pox/  
mininet@mininet-vm:~/pox$ git branch  
* carp  
mininet@mininet-vm:~/pox$ git checkout betta  
Branch betta set up to track remote branch betta from origin.  
Switched to a new branch 'betta'  
mininet@mininet-vm:~/pox$ git branch  
* betta  
  carp  
mininet@mininet-vm:~/pox$ /pox.py log.level --DEBUG misc.of_tutorial  
-bash: /pox.py: No such file or directory  
mininet@mininet-vm:~/pox$ ./pox.py log.level --DEBUG misc.of_tutorial  
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.  
DEBUG:core:POX 0.1.0 (betta) going up...  
DEBUG:core:Running on CPython (2.7.6/Mar 22 2014 22:59:56)  
DEBUG:core:Platform is Linux-3.13.0-24-generic-x86_64-with-Ubuntu-14.04-trusty  
INFO:core:POX 0.1.0 (betta) is up.  
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633  
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected  
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 1]  
█
```

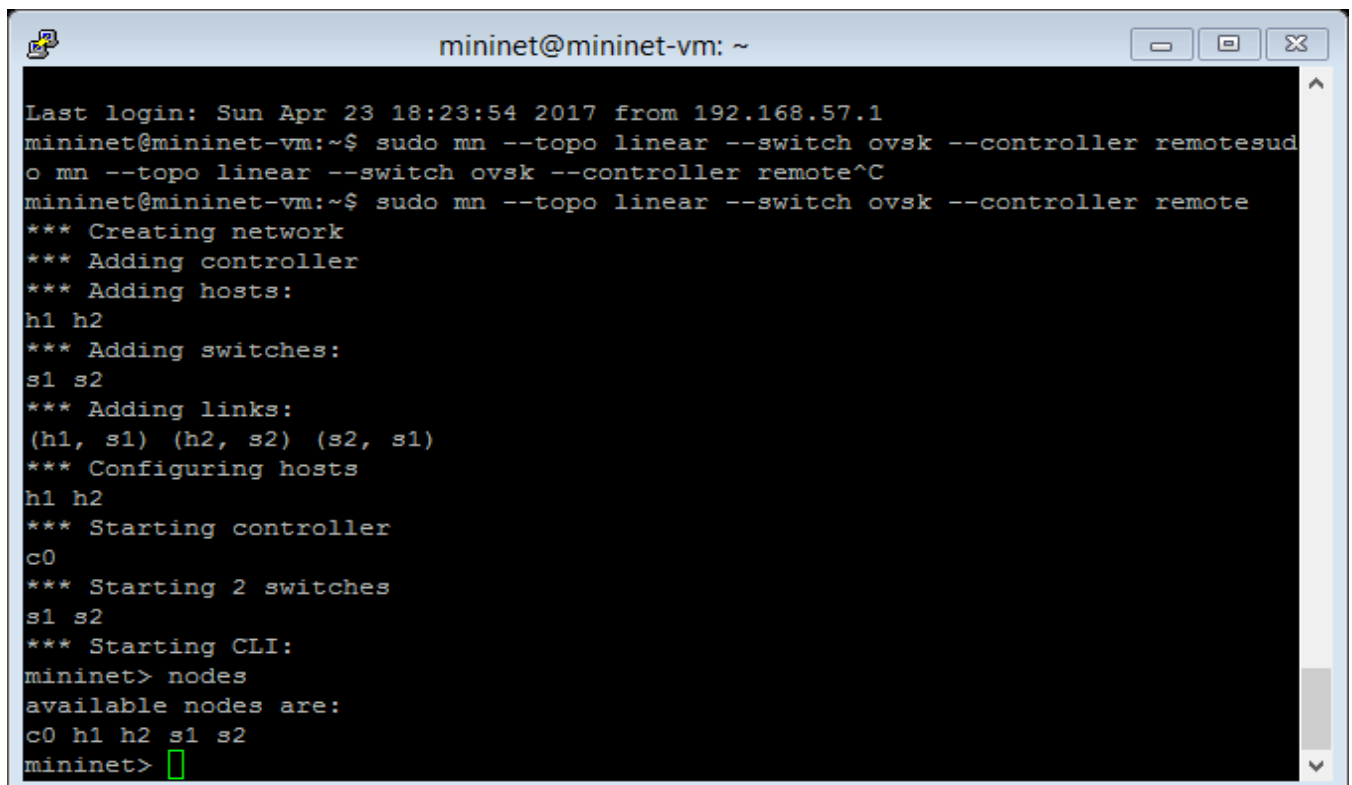


## CREATING A LEARNING 3 SWITCH

### SUPPORT MULTIPLE SWITCHES

Start mininet with a different topology.

In the Mininet console: **mininet> exit \$ sudo mn --topo linear --switch ovsk --controller remote**



```
mininet@mininet-vm: ~
Last login: Sun Apr 23 18:23:54 2017 from 192.168.57.1
mininet@mininet-vm:~$ sudo mn --topo linear --switch ovsk --controller remote
mininet@mininet-vm:~$ sudo mn --topo linear --switch ovsk --controller remote^C
mininet@mininet-vm:~$ sudo mn --topo linear --switch ovsk --controller remote
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s2) (s2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 s1 s2
mininet> 
```

## PART 1: ROUTER IMPLEMENTATION

In this exercise, we make a static layer-3 switch. It's not exactly a router, because it won't decrement the IP TTL and recomputed the checksum at each hop (so traceroute won't work). It will match on masked IP prefix ranges, just like a real router.

From RFC 1812:

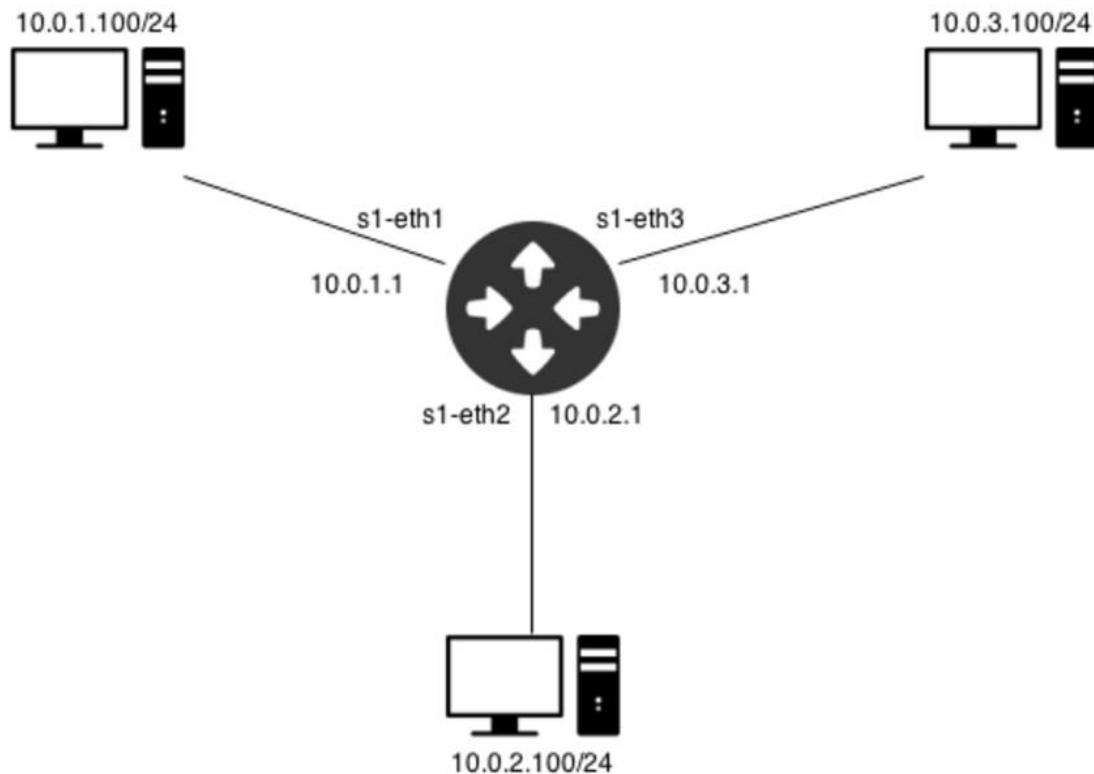
An IP router can be distinguished from other sorts of packet switching devices in that a router examines the IP protocol header as part of the switching process. It generally removes the Link Layer header a message was received with, modifies the IP header, and replaces the Link Layer header for retransmission.

To simplify this exercise, the "router" is a completely static one. With no BGP or OSPF, we don't have to send or receive route table updates.

Each network node will have a configured subnet. If a packet is destined for a host within that subnet, the node acts as a switch and forwards the packet with no changes, to a known port or broadcast, just like in the previous exercise. If a packet is destined for some IP address for which the router knows the next hop; it should modify the layer-2 destination and forward the packet to the correct port.

Since this is a Static Router the routing table is formulated as:  
`self.routingTable = [ ['10.0.1.100', '10.0.1.100', 's1-eth1', '10.0.1.1', 1], ['10.0.2.100', '10.0.2.100', 's1-eth2', '10.0.2.1', 2], ['10.0.3.100', '10.0.3.100', 's1-eth3', '10.0.3.1', 3]]`

The configuration is setup and "mytopo" file was modified as per the below image. 'addHost' and 'addSwitch' were used and the IP addresses and their respective network id's were set.



This code snippet explains how to add links and hosts:

```

# Add hosts and switches
leftHost = self.addHost( 'h1',ip="10.0.1.100",defaultRoute="via
10.0.1.1" )
rightHost = self.addHost( 'h2',ip="10.0.2.100",defaultRoute="via
10.0.2.1" )
bottomHost = self.addHost( 'h3',ip="10.0.3.100",defaultRoute="via
10.0.3.1" )
switch = self.addSwitch( 's1' )

# Add links
self.addLink( leftHost, switch )
self.addLink( rightHost, switch )
self.addLink( bottomHost, switch )

```

**sudo mn --custom mytopo1.py --topo mytopo --mac --switch ovsk --controller remote**

**Startting controller:**

```
mininet@mininet-vm: ~/pox
mininet@mininet-vm:~/pox$ ps -ef | grep controller
root      31801 31220  0 06:12 pts/10    00:00:00 sudo mn --custom mytopo.py --topo mytopo --mac --controller remote
root      31802 31801  0 06:12 pts/10    00:00:00 /usr/bin/python /usr/local/bin/mn --custom mytopo.py --topo mytopo --mac --controller remote
mininet   32127 31266  0 06:15 pts/11    00:00:00 grep --color=auto controller
mininet@mininet-vm:~/pox$ kill -9 31802
-bash: kill: (31802) - Operation not permitted
mininet@mininet-vm:~/pox$ sudo kill -9 31802
mininet@mininet-vm:~/pox$ kill -9 31802
-bash: kill: (31802) - No such process
mininet@mininet-vm:~/pox$ ps -ef | grep controller
mininet   32238 31266  0 06:16 pts/11    00:00:00 grep --color=auto controller
mininet@mininet-vm:~/pox$ ./pox.py log.level --DEBUG misc.of_tutorial_router
POX 0.1.0 (beta) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.1.0 (beta) going up...
DEBUG:core:Running on CPython (2.7.6/Mar 22 2014 22:59:56)
DEBUG:core:Platform is Linux-3.13.0-24-generic-x86_64-with-Ubuntu-14.04-trusty
DEBUG:misc.of_tutorial_router:Up...
INFO:core:POX 0.1.0 (beta) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-02 1] connected
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
```

This topology is placed in home/mininet and the executable is run in the misc folder by calling:

**\$ ./pox.py log.level --DEBUG misc.of\_router\_a  
misc.full\_payload**

Ping h2 from h1 :

```
mininet@mininet-vm: ~
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1
mininet> h1 ping -c1 h2
PING 10.0.2.100 (10.0.2.100) 56(84) bytes of data.
64 bytes from 10.0.2.100: icmp_seq=1 ttl=64 time=135 ms

--- 10.0.2.100 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 135.255/135.255/135.255/0.000 ms
mininet>

mininet@mininet-vm: ~/pox
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.2.100
DEBUG:misc.of_tutorial_router:1 1 learned 10.0.1.100
DEBUG:misc.of_tutorial_router:1 1 flooding ARP request 10.0.1.100 => 10.0.2.100
DEBUG:misc.of_tutorial_router:1 2 ARP reply 10.0.2.100 => 10.0.1.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.1.100
DEBUG:misc.of_tutorial_router:1 2 learned 10.0.2.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.1.100
DEBUG:misc.of_tutorial_router:1 2 flooding ARP reply 10.0.2.100 => 10.0.1.100
DEBUG:misc.of_tutorial_router:1 1 IP 10.0.1.100 => 10.0.2.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.2.100
DEBUG:misc.of_tutorial_router:Got Port Number : 2 from routing table
DEBUG:misc.of_tutorial_router:1 2 IP 10.0.2.100 => 10.0.1.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.1.100
DEBUG:misc.of_tutorial_router:Got Port Number : 1 from routing table
DEBUG:misc.of_tutorial_router:1 2 ARP request 10.0.2.100 => 10.0.1.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.1.100
DEBUG:misc.of_tutorial_router:1 2 answering ARP from 10.0.1.100 to 10.0.2.100
```

Pingall :

```
Configuring hosts
h3
Starting controller

Starting 1 switches

Starting CLI:
et> nodes
Available nodes are:
h2 h3 s1
et> h1 ping -c1 h2
10.0.2.100 (10.0.2.100) 56(84) bytes of data.
64 bytes from 10.0.2.100: icmp_seq=1 ttl=64 time=135 ms

--- 10.0.2.100 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 135.255/135.255/135.255/0.000 ms
et> pingall
Ping: testing ping reachability
h2 h3
h1 h3
h1 h2
Results: 0% dropped (6/6 received)
et>
```

```
DEBUG:misc.of_tutorial_router:1 3 answering ARP from 10.0.2.100 to 10.0.3.100
DEBUG:misc.of_tutorial_router:1 3 IP 10.0.3.100 => 10.0.2.100
DEBUG:misc.of_tutorial_router:Got port 2 from dictionary
DEBUG:misc.of_tutorial_router:1 3 IP 10.0.3.100 => 10.0.1.100
DEBUG:misc.of_tutorial_router:Got port 1 from dictionary
DEBUG:misc.of_tutorial_router:1 1 IP 10.0.1.100 => 10.0.3.100
DEBUG:misc.of_tutorial_router:Got port 3 from dictionary
DEBUG:misc.of_tutorial_router:1 3 IP 10.0.3.100 => 10.0.2.100
DEBUG:misc.of_tutorial_router:Got port 2 from dictionary
DEBUG:misc.of_tutorial_router:1 2 IP 10.0.2.100 => 10.0.3.100
DEBUG:misc.of_tutorial_router:Got port 3 from dictionary
DEBUG:misc.of_tutorial_router:1 1 ARP request 10.0.1.100 => 10.0.2.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.2.100
DEBUG:misc.of_tutorial_router:1 1 answering ARP from 10.0.2.100 to 10.0.1.100
DEBUG:misc.of_tutorial_router:1 3 ARP request 10.0.3.100 => 10.0.1.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.1.100
DEBUG:misc.of_tutorial_router:1 3 answering ARP from 10.0.1.100 to 10.0.3.100
DEBUG:misc.of_tutorial_router:1 2 ARP request 10.0.2.100 => 10.0.1.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.1.100
DEBUG:misc.of_tutorial_router:1 2 answering ARP from 10.0.1.100 to 10.0.2.100
```

Reachable host:

```
mininet> nodes
Available nodes are:
h0 h1 h2 h3 s1
mininet> h1 ping -c1 h2
PING 10.0.2.100 (10.0.2.100) 56(84) bytes of data.
64 bytes from 10.0.2.100: icmp_seq=1 ttl=64 time=135 ms

--- 10.0.2.100 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 135.255/135.255/135.255/0.000 ms
mininet> pingall
** Ping: testing ping reachability
1 -> h2 h3
2 -> h1 h3
3 -> h1 h2
** Results: 0% dropped (6/6 received)
mininet> h1 ping -c1 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=28.1 ms

--- 10.0.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 28.161/28.161/28.161/0.000 ms
mininet>
```

```
DEBUG:misc.of_tutorial_router:1 3 IP 10.0.3.100 => 10.0.1.100
DEBUG:misc.of_tutorial_router:Got port 1 from dictionary
DEBUG:misc.of_tutorial_router:1 1 IP 10.0.1.100 => 10.0.3.100
DEBUG:misc.of_tutorial_router:Got port 3 from dictionary
DEBUG:misc.of_tutorial_router:1 3 IP 10.0.3.100 => 10.0.2.100
DEBUG:misc.of_tutorial_router:Got port 2 from dictionary
DEBUG:misc.of_tutorial_router:1 2 IP 10.0.2.100 => 10.0.3.100
DEBUG:misc.of_tutorial_router:1 1 ARP request 10.0.1.100 => 10.0.2.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.2.100
DEBUG:misc.of_tutorial_router:1 1 answering ARP from 10.0.2.100 to 10.0.1.100
DEBUG:misc.of_tutorial_router:1 3 ARP request 10.0.3.100 => 10.0.1.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.1.100
DEBUG:misc.of_tutorial_router:1 3 answering ARP from 10.0.1.100 to 10.0.3.100
DEBUG:misc.of_tutorial_router:1 2 ARP request 10.0.2.100 => 10.0.1.100
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.1.100
DEBUG:misc.of_tutorial_router:1 2 answering ARP from 10.0.1.100 to 10.0.2.100
DEBUG:misc.of_tutorial_router:1 1 ARP request 10.0.1.100 => 10.0.1.1
DEBUG:misc.of_tutorial_router:1 1 answering ARP from 10.0.1.1 to 10.0.1.100
DEBUG:misc.of_tutorial_router:1 1 IP 10.0.1.100 => 10.0.1.1
```

Whenever a packet arrives, the packet is parsed to see if it is either of ARP or IP type.

It processes the packet(see the attached code) to categorize it and send the required reply back.

A pingall and iperf test was performed to see the working of this router:



## Unreachable host:

```
mininet@mininet-vm: ~  
--- 10.0.2.100 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 135.255/135.255/135.255/0.000 ms  
mininet> pingall  
*** Ping: testing ping reachability  
h1 -> h2 h3  
h2 -> h1 h3  
h3 -> h1 h2  
*** Results: 0% dropped (6/6 received)  
mininet> h1 ping -c1 10.0.1.1  
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.  
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=28.1 ms  
  
--- 10.0.1.1 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 28.161/28.161/28.161/0.000 ms  
mininet> h1 ping -c1 10.99.0.1  
PING 10.99.0.1 (10.99.0.1) 56(84) bytes of data.  
From 10.0.1.100 icmp_seq=1 Destination Host Unreachable  
  
--- 10.99.0.1 ping statistics ---  
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms  
  
mininet>
```

```
mininet@mininet-vm: ~/pox  
DEBUG:misc.of_tutorial_router:1 2 IP 10.0.2.100 => 10.0.3.100  
DEBUG:misc.of_tutorial_router:Got port 3 from dictionary  
DEBUG:misc.of_tutorial_router:1 1 ARP request 10.0.1.100 => 10.0.2.100  
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.2.100  
  
DEBUG:misc.of_tutorial_router:1 1 answering ARP from 10.0.2.100 to 10.0.1.100  
DEBUG:misc.of_tutorial_router:1 3 ARP request 10.0.3.100 => 10.0.1.100  
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.1.100  
  
DEBUG:misc.of_tutorial_router:1 3 answering ARP from 10.0.1.100 to 10.0.3.100  
DEBUG:misc.of_tutorial_router:1 2 ARP request 10.0.2.100 => 10.0.1.100  
DEBUG:misc.of_tutorial_router:Searching Routing Table for IP address: 10.0.1.100  
  
DEBUG:misc.of_tutorial_router:1 2 answering ARP from 10.0.1.100 to 10.0.2.100  
DEBUG:misc.of_tutorial_router:1 1 ARP request 10.0.1.100 => 10.0.1.1  
DEBUG:misc.of_tutorial_router:1 1 answering ARP from 10.0.1.1 to 10.0.1.100  
DEBUG:misc.of_tutorial_router:1 1 IP 10.0.1.100 => 10.0.1.1  
DEBUG:misc.of_tutorial_router:1 1 ARP request 10.0.1.100 => 10.99.0.1  
DEBUG:misc.of_tutorial_router:Unreachable IP Address : 10.99.0.1  
DEBUG:misc.of_tutorial_router:1 1 ARP request 10.0.1.100 => 10.99.0.1  
DEBUG:misc.of_tutorial_router:Unreachable IP Address : 10.99.0.1  
DEBUG:misc.of_tutorial_router:1 1 ARP request 10.0.1.100 => 10.99.0.1  
DEBUG:misc.of_tutorial_router:Unreachable IP Address : 10.99.0.1  
DEBUG:misc.of_tutorial_router:Unreachable IP Address : 10.99.0.1
```

```
login as: mininet  
mininet@192.168.56.100:~$  
Welcome to Ubuntu 16.04.2 LTS  
  
* Documentation: https://ubuntu.com/docs  
New release '16.04.2 LTS' available: 'do-release-upgrade' to upgrade to it or 'do-release-upgrade -d' to see the new version.  
Run 'do-release-upgrade' to upgrade to it or 'do-release-upgrade -d' to see the new version.  
  
Last login: Sun Aug 14 12:00:00 CEST 2016  
mininet@mininet-vm:~$
```

```
mininet@mininet-vm: ~  
*** Starting 1 switches  
s1  
*** Starting CLI:  
mininet> pingall  
*** Ping: testing ping reachability  
h1 -> h2 h3  
h2 -> h1 h3  
h3 -> h1 h2  
*** Results: 0% dropped (6/6 received)  
mininet> xterm h1 h2 h3  
mininet>
```

```
"Node: h2"  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0002 .....  
23:15:45.148249 ARP, Request who-has 10.0.0.2 tell 10.0.1.100, length 28  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0002 .....  
23:15:46.147442 ARP, Request who-has 10.0.0.2 tell 10.0.1.100, length 28  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0002 .....  
23:16:58.127876 ARP, Request who-has 10.0.0.5 tell 10.0.1.100, length 28  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0005 .....  
23:16:59.127566 ARP, Request who-has 10.0.0.5 tell 10.0.1.100, length 28  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0005 .....  
23:17:00.127302 ARP, Request who-has 10.0.0.5 tell 10.0.1.100, length 28  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0005 .....
```

```
"Node: h3"  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0002 .....  
23:15:45.148136 ARP, Request who-has 10.0.0.2 tell 10.0.1.100, length 28  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0002 .....  
23:15:46.147438 ARP, Request who-has 10.0.0.2 tell 10.0.1.100, length 28  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0002 .....  
23:16:58.127874 ARP, Request who-has 10.0.0.5 tell 10.0.1.100, length 28  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0005 .....  
23:16:59.127563 ARP, Request who-has 10.0.0.5 tell 10.0.1.100, length 28  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0005 .....  
23:17:00.127298 ARP, Request who-has 10.0.0.5 tell 10.0.1.100, length 28  
0x0000: ffff ffff ffff 0000 0000 0001 0806 0001 .....  
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0164 .....d  
0x0020: 0000 0000 0000 0a00 0005 .....
```

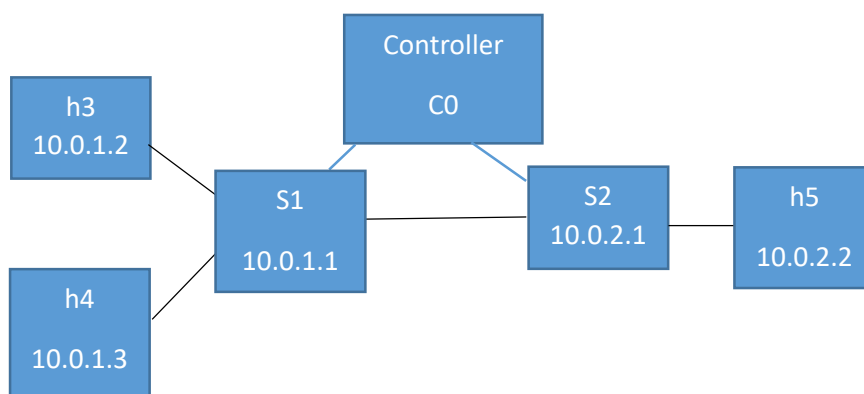
```
root@mininet-vm:~# ping -c1 10.0.0.2  
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.  
From 10.0.1.100 icmp_seq=1 Destination Host Unreachable  
  
--- 10.0.0.2 ping statistics ---  
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms  
  
root@mininet-vm:~# ping -c1 10.0.0.5  
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.  
From 10.0.1.100 icmp_seq=1 Destination Host Unreachable  
  
--- 10.0.0.5 ping statistics ---  
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms  
  
root@mininet-vm:~#
```

## PART 2 IMPLEMENTATION:

This part requires us to make the topology give to us and perform the actions.

The topology given has :

- 3 hosts
- 2 switches



Similar to part 1 implementation, same steps are followed except for a few changes are made in the topology and code. First we need to create the above mentioned topology by editing the topology file by adding the needed components and assigning IP addresses and establishing links.

The controller performs the similar actions in this stage too with ARP and ICMP. Now here each network node will have a configured subnet. If a packet is destined for a host within that subnet, the node acts as a switch and forwards the packet with no changes, to a known port or broadcast, just like in the previous part. If a packet is destined for some IP address for which the router knows the next hop, it should modify the layer-2 destination and forward the packet to the correct port.

**Creating the topology :**



```

mininet@mininet-vm: ~
from mininet.topo import Topo

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

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

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost1 = self.addHost( 'h3',ip="10.0.1.2",defaultRoute="via 10.0.1.1" )
        rightHost1 = self.addHost( 'h5',ip="10.0.2.2",defaultRoute="via 10.0.2.1" )
        leftHost2 = self.addHost( 'h4',ip="10.0.1.3",defaultRoute="via 10.0.1.1" )
        switch1 = self.addSwitch( 's1' )
        switch2 = self.addSwitch( 's2' )

        # Add links
        self.addLink( leftHost1, switch1 )
        self.addLink( rightHost1, switch2 )
        self.addLink( leftHost2, switch1 )
        self.addLink( switch1, switch2 )

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

```

When we create the topology, we see this on the console and finally get the “ mininet > ”

```

mininet@mininet-vm: ~
Caught exception. Cleaning up...

IndentationError: unexpected indent (advtopo.py, line 18)

mininet@mininet-vm:~$ sudo mn --custom advtopo.py --topo mytopo --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h3 h4 h5
*** Adding switches:
s1 s2
*** Adding links:
(h3, s1) (h4, s1) (h5, s2) (s1, s2)
*** Configuring hosts
h3 h4 h5
*** Starting controller
c0
*** Starting 2 switches
s1 s2
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h3 h4 h5 s1 s2
mininet>

```

We need to modify the controller and add the necessary code and run the controller. We can see the controller and mininet screens side by side in the screenhots attached for reference.

Now the **ping all** test yield viz.

```

mininet@mininet-vm: ~
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h3 h4 h5
*** Adding switches:
s1 s2
*** Adding links:
(h3, s1) (h4, s1) (h5, s2) (s1, s2)
*** Configuring hosts
h3 h4 h5
*** Starting controller
c0
*** Starting 2 switches
s1 s2
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h3 -> h4 h5
h4 -> h3 h5
h5 -> h3 h4
*** Results: 0% dropped (6/6 received)

```

```

mininet@mininet-vm: ~/pox
DEBUG:misc.adv_tut:The input 3 The input mac address is 00:00:00:00:00:03
DEBUG:misc.adv_tut:Sending 1 buffered packets to 10.0.2.2 from 00-00-00-00-00-01
DEBUG:misc.adv_tut:Searching Routing Table for IP address: 10.0.1.3
DEBUG:misc.adv_tut:1 3 flooding ARP reply 10.0.2.2 => 10.0.1.3
DEBUG:misc.adv_tut:2 2 IP 10.0.1.3 => 10.0.2.2
DEBUG:misc.adv_tut:2 2 learned 10.0.1.3
DEBUG:misc.adv_tut:Got port 3 from dictionary
DEBUG:misc.adv_tut:2 2 ARPing for 10.0.2.2 on behalf of 10.0.1.3
DEBUG:misc.adv_tut:2 1 ARP reply 10.0.2.2 => 10.0.1.3
DEBUG:misc.adv_tut:Searching Routing Table for IP address: 10.0.1.3
DEBUG:misc.adv_tut:The mac address is 00:00:00:00:00:02
DEBUG:misc.adv_tut:2 1 learned 10.0.2.2
DEBUG:misc.adv_tut:The input 1 The input mac address is 00:00:00:00:00:03
DEBUG:misc.adv_tut:Sending 1 buffered packets to 10.0.2.2 from 00-00-00-00-00-02
DEBUG:misc.adv_tut:Searching Routing Table for IP address: 10.0.1.3
DEBUG:misc.adv_tut:2 1 flooding ARP reply 10.0.2.2 => 10.0.1.3
DEBUG:misc.adv_tut:1 3 ARP reply 10.0.2.2 => 10.0.1.3
DEBUG:misc.adv_tut:Searching Routing Table for IP address: 10.0.1.3
DEBUG:misc.adv_tut:The mac address is 00:00:00:00:00:01
DEBUG:misc.adv_tut:1 3 learned 10.0.2.2
DEBUG:misc.adv_tut:The input 3 The input mac address is 00:00:00:00:00:03
DEBUG:misc.adv_tut:Searching Routing Table for IP address: 10.0.1.3
DEBUG:misc.adv_tut:1 3 flooding ARP reply 10.0.2.2 => 10.0.1.3

```

## Pinging a known host:

When the controller receives ICMP echo (ping) requests for the known, respond with the following messages:

```
mininet> xterm h1 h2 h3
Error: Cannot connect to display
mininet@mininet-vm: ~
*** Unknown command: h1 ping h2
mininet> h3 ping h5
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=157 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=64 time=123 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=64 time=102 ms
64 bytes from 10.0.2.2: icmp_seq=4 ttl=64 time=113 ms
64 bytes from 10.0.2.2: icmp_seq=5 ttl=64 time=139 ms
64 bytes from 10.0.2.2: icmp_seq=6 ttl=64 time=385 ms
64 bytes from 10.0.2.2: icmp_seq=7 ttl=64 time=141 ms
64 bytes from 10.0.2.2: icmp_seq=8 ttl=64 time=155 ms
64 bytes from 10.0.2.2: icmp_seq=9 ttl=64 time=176 ms
64 bytes from 10.0.2.2: icmp_seq=10 ttl=64 time=152 ms
64 bytes from 10.0.2.2: icmp_seq=11 ttl=64 time=152 ms
64 bytes from 10.0.2.2: icmp_seq=12 ttl=64 time=114 ms
64 bytes from 10.0.2.2: icmp_seq=13 ttl=64 time=208 ms
64 bytes from 10.0.2.2: icmp_seq=14 ttl=64 time=163 ms
64 bytes from 10.0.2.2: icmp_seq=15 ttl=64 time=131 ms
64 bytes from 10.0.2.2: icmp_seq=16 ttl=64 time=56.6 ms
64 bytes from 10.0.2.2: icmp_seq=17 ttl=64 time=172 ms
64 bytes from 10.0.2.2: icmp_seq=18 ttl=64 time=207 ms
64 bytes from 10.0.2.2: icmp_seq=19 ttl=64 time=116 ms
64 bytes from 10.0.2.2: icmp_seq=20 ttl=64 time=127 ms
^C
mininet@mininet-vm: ~
mininet@mininet-vm:~$
mininet@mininet-vm:~$
mininet@mininet-vm:~$
mininet@mininet-vm:~$
mininet@mininet-vm:~$
```

## Pinging an unknown host :

For packets for unreachable subnets (IP,'s that don't exists ) it responds with ICMP network unreachable messages: (Eg: trying to ping 10.0.1.10-non-existent )

```
mininet> xterm h1 h2 h3
Error: Cannot connect to display
mininet@mininet-vm: ~
64 bytes from 10.0.1.1: icmp_seq=13 ttl=64 time=43.4 ms
64 bytes from 10.0.1.1: icmp_seq=14 ttl=64 time=30.8 ms
64 bytes from 10.0.1.1: icmp_seq=15 ttl=64 time=25.5 ms
64 bytes from 10.0.1.1: icmp_seq=16 ttl=64 time=13.6 ms
64 bytes from 10.0.1.1: icmp_seq=17 ttl=64 time=22.7 ms
64 bytes from 10.0.1.1: icmp_seq=18 ttl=64 time=13.9 ms
64 bytes from 10.0.1.1: icmp_seq=19 ttl=64 time=56.6 ms
64 bytes from 10.0.1.1: icmp_seq=20 ttl=64 time=44.5 ms
64 bytes from 10.0.1.1: icmp_seq=21 ttl=64 time=32.2 ms
64 bytes from 10.0.1.1: icmp_seq=22 ttl=64 time=42.4 ms
64 bytes from 10.0.1.1: icmp_seq=23 ttl=64 time=30.3 ms
^C
--- 10.0.1.1 ping statistics ---
23 packets transmitted, 23 received, 0% packet loss, time 2204ms
rtt min/avg/max/mdev = 7.33/78.971/56.650/13.427 ms
mininet> h3 ping 10.0.1.10
PING 10.0.1.10 (10.0.1.10) 56(84) bytes of data.
From 10.0.1.2 icmp_seq=1 Destination Host Unreachable
From 10.0.1.2 icmp_seq=2 Destination Host Unreachable
From 10.0.1.2 icmp_seq=3 Destination Host Unreachable
From 10.0.1.2 icmp_seq=4 Destination Host Unreachable
From 10.0.1.2 icmp_seq=5 Destination Host Unreachable
From 10.0.1.2 icmp_seq=6 Destination Host Unreachable
^C
mininet@mininet-vm: ~
mininet@mininet-vm:~$
mininet@mininet-vm:~$
mininet@mininet-vm:~$
mininet@mininet-vm:~$
mininet@mininet-vm:~$
```

## Iperf to check the bandwidth :

```
Error: Cannot connect to display
mininet@mininet-vm: ~
From 10.0.1.2 icmp_seq=9 Destination Host Unreachable
From 10.0.1.2 icmp_seq=10 Destination Host Unreachable
From 10.0.1.2 icmp_seq=11 Destination Host Unreachable
From 10.0.1.2 icmp_seq=12 Destination Host Unreachable
From 10.0.1.2 icmp_seq=13 Destination Host Unreachable
From 10.0.1.2 icmp_seq=14 Destination Host Unreachable
From 10.0.1.2 icmp_seq=15 Destination Host Unreachable
From 10.0.1.2 icmp_seq=16 Destination Host Unreachable
From 10.0.1.2 icmp_seq=17 Destination Host Unreachable
From 10.0.1.2 icmp_seq=18 Destination Host Unreachable
From 10.0.1.2 icmp_seq=19 Destination Host Unreachable
From 10.0.1.2 icmp_seq=20 Destination Host Unreachable
From 10.0.1.2 icmp_seq=21 Destination Host Unreachable
From 10.0.1.2 icmp_seq=22 Destination Host Unreachable
From 10.0.1.2 icmp_seq=23 Destination Host Unreachable
From 10.0.1.2 icmp_seq=24 Destination Host Unreachable
^C
--- 10.0.1.10 ping statistics ---
25 packets transmitted, 0 received, 100% packet loss, time 2407ms
mininet> iperf
*** Iperf: testing TCP bandwidth between h3 and h5
*** Result: ['564 Kbits/sec', '1975 Kbits/sec']
mininet>
mininet@mininet-vm: ~
mininet@mininet-vm:~$
```

### Pinging the switch from host :

The switch is should pingable, and generates an ICMP echo reply in response to an ICMP echo request.

[illegible]

## Firewall Part:

We modified the switch to reject connection attempts to specific ports, just like a firewall.

The modification helps show how OpenFlow can even do basic layer-4 tasks. The modification blocks all flow entries with this particular port - 10.0.1.2 (h3).

### Firewall blocking the iperf request:

The image shows a Kali Linux desktop environment with three terminal windows open.

- Top-left window (Node: h3):** Displays the output of `ifconfig` for the `root@mininet-vm:~#` user. It shows the `eth0` interface with IP `10.0.0.1` and `10.0.0.2`, and the `lo` interface with IP `127.0.0.1`. The server is listening on TCP port 5001.
- Top-right window (Node: h4):** Displays the output of `ip route` for the `root@mininet-vm:~#` user. It shows the default route via `10.0.0.1` and `10.0.0.2`.
- Bottom window (mininet@mininet-vm: /gox):** Displays the output of `mininet` commands for the `mininet@mininet-vm: /gox` user. It shows a sequence of debug messages from the mininet controller, including ARP requests, flooding, and firewall blocking for IP `10.0.0.2`.

Ping gets blocked due to our firewall :

The image shows two terminal windows from a Mininet environment. The left window, titled 'mininet@mininet-vm: ~', shows a series of ping tests between hosts h1, h2, h3, and h4. The tests show successful connectivity with 0% packet loss. The right window, titled 'mininet@mininet-vm: ~/pox', displays OpenFlow logs from a switch. The logs show the switch receiving packets, searching its routing table, and flooding ARP requests to learn MAC addresses for specific IP addresses (10.0.1.3 and 10.0.1.2). It also shows the switch learning the MAC addresses and the IP addresses associated with them.

```
mininet@mininet-vm: ~  
mininet> ping -c1 h4  
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.  
bytes from 10.0.1.3: icmp_seq=1 ttl=64 time=68.2 ms  
--- 10.0.1.3 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
r min/avg/max/mdev = 68.241/68.241/68.241/0.000 ms  
mininet> xterm h3 h4 h5  
mininet> h4 ping -c1 h3  
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.  
--- 10.0.1.2 ping statistics ---  
1 packets transmitted, 0 received, 100% packet loss, time 0ms  
mininet> h3 ping -c1 h4  
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.  
--- 10.0.1.3 ping statistics ---  
1 packets transmitted, 0 received, 100% packet loss, time 0ms  
mininet> h3 ping -c1 h4  
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.  
--- 10.0.1.3 ping statistics ---  
1 packets transmitted, 0 received, 100% packet loss, time 0ms  
mininet@mininet-vm: ~  
mininet@mininet-vm: ~  
mininet@mininet-vm: ~  
mininet@mininet-vm: ~
```

```
mininet@mininet-vm: ~/pox  
DEBUG:misc.of_tutorial2_fw:Sending 1 buffered packets to 10.0.1.3 from 10.0.1.2  
DEBUG:misc.of_tutorial2_fw:Searching Routing Table for IP address: 10.0.1.3  
DEBUG:misc.of_tutorial2_fw:2 Flooding ARP reply 10.0.1.3 => 10.0.1.2  
DEBUG:misc.of_tutorial2_fw:2 ARP reply 10.0.1.3 => 10.0.1.2  
DEBUG:misc.of_tutorial2_fw:Searching Routing Table for IP address: 10.0.1.2  
DEBUG:misc.of_tutorial2_fw:The mac address is 00:00:00:00:00:02  
DEBUG:misc.of_tutorial2_fw:2 Learned 10.0.1.3  
DEBUG:misc.of_tutorial2_fw:The input 2 The input mac address is 00:00:00:00:00:02  
DEBUG:misc.of_tutorial2_fw:Searching Routing Table for IP address: 10.0.1.3  
DEBUG:misc.of_tutorial2_fw:2 Flooding ARP reply 10.0.1.3 => 10.0.1.2  
DEBUG:misc.of_tutorial2_fw:2 IF 10.0.1.3 => 10.0.1.2  
DEBUG:misc.of_tutorial2_fw:2 Learned 10.0.1.3  
DEBUG:misc.of_tutorial2_fw:FIREWALL BLOCKING THE IP
```

## CONCLUSION:

OpenFlow and Mininet tools were used to successfully create switches as a router and hub and the respective configurations.

We learnt to create different topologies and try the commands that ensure all the hosts are connected. The messages for successful and unsuccessful flows were studied.

Implementation of firewall was done and tested and studied that show how OpenFlow can even do basic layer-4 tasks

## REFERENCES:

<https://openflow.stanford.edu/display/ONL/POX+Wiki>

<http://mininet.org/walkthrough/>

<https://github.com/mininet/openflow-tutorial/wiki>