

Configuration Manual

MSc Research Project
Cloud Computing

Shivani Vikram More
Student ID: x19111606

School of Computing
National College of Ireland

Supervisor: Prof. Aqeel Kazmi

**National College of Ireland
Project Submission Sheet
School of Computing**



Student Name:	Shivani Vikram More
Student ID:	x19111606
Programme:	Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Prof. Aqeel Kazmi
Submission Due Date:	24/12/2020
Project Title:	Configuration Manual
Word Count:	1800
Page Count:	15

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	1st February 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shivani Vikram More
x19111606

1 Introduction

The purpose of this configuration manual is to enable the reader to execute the research project 'Fragmentation of Elephant Flows into Mice Flows to Optimize Routing Strategy in SDN'.

2 Objective

This document includes detailed information on how the execution of the research project is conducted. The use of various commands, codes, installation of software and hardware to get desired functioning of the proposed system are explained in further sections of the configuration manual.

3 System Configuration

In this section, an elaborated list of software and hardware is given along with its use for implementing the capstone research project.

3.1 Software Requirements

- Host Operating System: Linux Ubuntu LTS (Version: 20.04.1)
- Hypervisor: Oracle VirtualBox (Version: 6.1)
- Network Emulator: Mininet (Version: 2.3.0.d6)
- SDN Controller: POX Controller (0.5.0 eel branch)
- Programming Language: Python (Version: 2.7.18)

3.2 Hardware Requirements

Given below is the underlying configuration used while building the 2 virtual machines used for testing purpose.

- Processor: Intel Core i7 CPU, 2 core processor
- RAM: 2GB
- HDD: 16GB

4 Testbed Configuration for Research Project

This section provides steps conducted to setup the end-to-end emulated SDN environment that were identified during the initial phase of project implementation. These steps helps in setting up the environment on which the proposed system has been implemented.

4.1 Installation of Oracle VirtualBox

Oracle VirtualBox 6.1 is a type 2 hypervisor used to create two virtual machines for the proposed system. The software is available on the official website of Oracle VM VirtualBox website.¹ The virtual machines have host operating system as Linux Ubuntu LTS-20.04.1 installed using .iso image file.²

4.2 Installation of Mininet Network emulator

Mininet is a network emulator used in the proposed system to created emulated SDN fat-tree topology. The mininet version used is 2.3.0.d6. Given below are the steps to install mininet.³

- STEP 1: After successful installation of Oracle VirtualBox on the system, we have created two virtual machines in order to conduct evaluation of the proposed system. These machines help build comparative analysis of network behaviour where first system has proposed system implemented and the other is a unmodified implementation of mininet fat-tree topology.

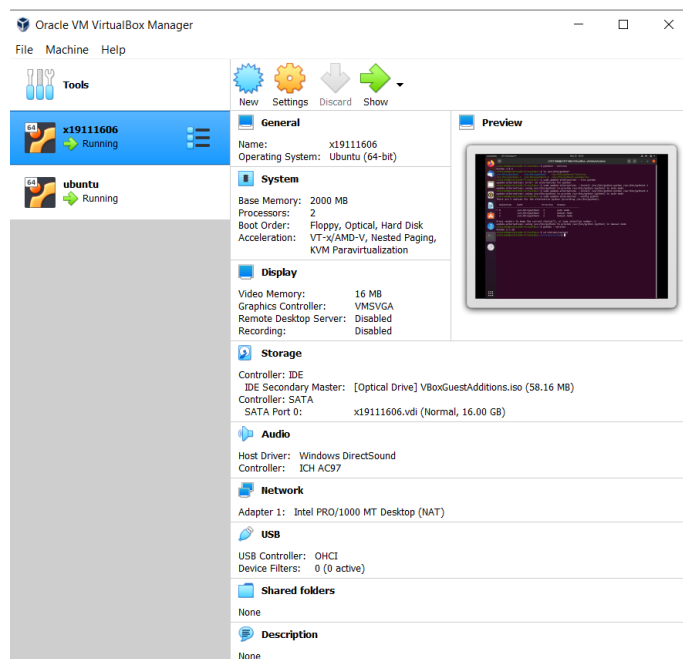


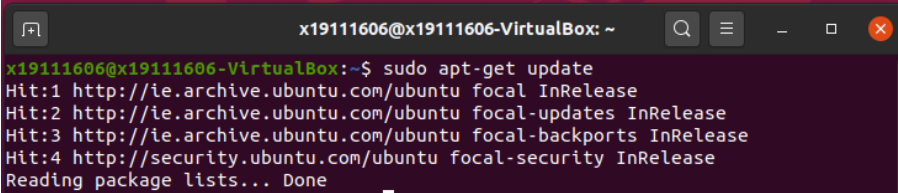
Figure 1: Oracle VirtualBox

¹Oracle VM VirtualBox: <https://www.virtualbox.org/>

²Ubuntu: <https://ubuntu.com/download/desktop>

³Mininet Installation: <http://www.brianlinkletter.com/how-to-install-mininet-sdn-network-simulator/>

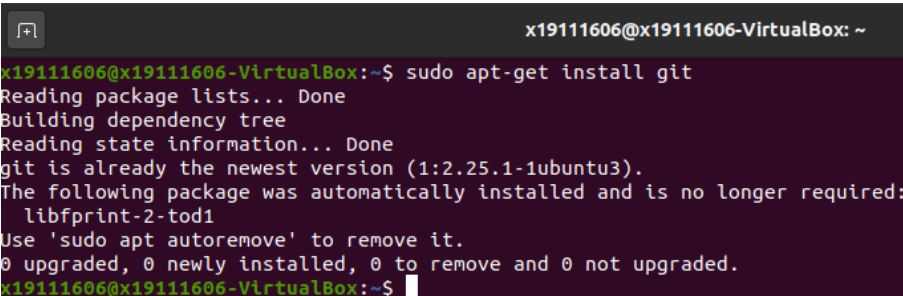
- Step 2: In the next step, we update the Ubuntu VMs to latest system update releases using command: `sudo apt-get update`



```
x19111606@x19111606-VirtualBox: ~
x19111606@x19111606-VirtualBox:~$ sudo apt-get update
Hit:1 http://ie.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://ie.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://ie.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease
Reading package lists... Done
```

Figure 2: System updated successfully


- Step 3: We need to install the software version control system called git for Mininet using the command: `sudo apt-get install git`



```
x19111606@x19111606-VirtualBox: ~
x19111606@x19111606-VirtualBox:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.25.1-1ubuntu3).
The following package was automatically installed and is no longer required:
  libfprint-2-tod1
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
x19111606@x19111606-VirtualBox:~$
```

Figure 3: Git installed successfully

- Step 4: We need to install Mininet using the git with command: `git clone git://github.com/mininet/mininet`



```
x19111606@x19111606-VirtualBox: ~
x19111606@x19111606-VirtualBox:~$ git clone git://github.com/mininet/mininet
Cloning into 'mininet'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9783 (delta 0), reused 3 (delta 0), pack-reused 9778
Receiving objects: 100% (9783/9783), 3.04 MiB | 1.42 MiB/s, done.
Resolving deltas: 100% (6483/6483), done.
x19111606@x19111606-VirtualBox:~$
```

Figure 4: Mininet installed

- Step 5: After successfully installing the mininet on the home directory, go to the Mininet directory. Once you are in the mininet directory, type the command: `'git tag'` which lists the available Mininet versions. Type command: `'git checkout -b 2.3.0d6'` which switches to the current Mininet version and then run the command `'/mininet/util/install.sh -a'` which is the install script for Mininet.

```
x19111606@x19111606-VirtualBox: ~/mininet
x19111606@x19111606-VirtualBox:~$ git clone git://github.com/mininet/mininet
Cloning into 'mininet'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9783 (delta 0), reused 3 (delta 0), pack-reused 9778
Receiving objects: 100% (9783/9783), 3.04 MiB | 1.42 MiB/s, done.
Resolving deltas: 100% (6483/6483), done.
x19111606@x19111606-VirtualBox:~$ cd mininet
x19111606@x19111606-VirtualBox:~/mininet$ git tag
1.0.0
2.0.0
2.1.0
2.1.0p1
2.1.0p2
2.2.0
2.2.1
2.2.2
2.3.0d3
2.3.0d4
2.3.0d5
2.3.0d6
cs244-spring-2012-final
x19111606@x19111606-VirtualBox:~/mininet$ git checkout -b 2.3.0d
2.3.0d3 2.3.0d4 2.3.0d5 2.3.0d6
x19111606@x19111606-VirtualBox:~/mininet$ git checkout -b 2.3.0d6
Switched to a new branch '2.3.0d6'
x19111606@x19111606-VirtualBox:~/mininet$
```

Figure 5: Mininet version selection

```
x19111606@x19111606-VirtualBox: ~/mininet
Switched to a new branch '2.3.0d6'
x19111606@x19111606-VirtualBox:~/mininet$ ~/mininet/util/install.sh -a
Detected Linux distribution: Ubuntu 20.04 focal amd64
sys.version_info(major=3, minor=8, micro=5, releaselevel='final', serial=0)
Detected Python (python3) version 3
Installing all packages except for -eix (doxypy, ivs, nox-classic)...
Install Mininet-compatible kernel if necessary
```

Figure 6: Mininet script installation

- Step 6: After successfully installing the Mininet on VM, you can check the version using command: `sudo mn - -version`

```
x19111606@x19111606-VirtualBox: ~/mininet
x19111606@x19111606-VirtualBox:~/mininet$ sudo mn --version
2.3.0d6
x19111606@x19111606-VirtualBox:~/mininet$
```

Figure 7: Version of mininet installed on the VM

5 Switching Ubuntu from Python3 to Python2

Since POX controller does not support Python 3, it is important to switch the default python3 usage of Linux Ubuntu LTS (Version: 20.04.1) to Python2 using the following commands

- Open Terminal
- `ls /usr/bin/python*` ...checks for available python versions
- `sudo update-alternatives - --list python` ...checks if Python-alternatives are configured
- `sudo update-alternatives - --install /usr/bin/python python /usr/bin/python2 1` ...configures alternative 1
- `sudo update-alternatives - --install /usr/bin/python python /usr/bin/python3 2` ...configures alternative 2
- `sudo update-alternatives - --config python` ...allows user to select Python2 from the configured Python alternatives
- run command: `python - --version`

```
x19111606@x19111606-VirtualBox: ~
x19111606@x19111606-VirtualBox:~$ python3 --version
Python 3.8.5
x19111606@x19111606-VirtualBox:~$ ls /usr/bin/python*
/usr/bin/python2  /usr/bin/python3  /usr/bin/python3-futurize
/usr/bin/python2.7 /usr/bin/python3.8  /usr/bin/python3-pasteurize
x19111606@x19111606-VirtualBox:~$ sudo update-alternatives --list python
update-alternatives: error: no alternatives for python
x19111606@x19111606-VirtualBox:~$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python2 1
update-alternatives: using /usr/bin/python2 to provide /usr/bin/python (python) in auto mode
x19111606@x19111606-VirtualBox:~$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 2
update-alternatives: using /usr/bin/python3 to provide /usr/bin/python (python) in auto mode
x19111606@x19111606-VirtualBox:~$ sudo update-alternatives --config python
There are 2 choices for the alternative python (providing /usr/bin/python).

  Selection    Path                        Priority  Status
  ----
*  0            /usr/bin/python3            2        auto mode
    1            /usr/bin/python2            1        manual mode
    2            /usr/bin/python3            2        manual mode

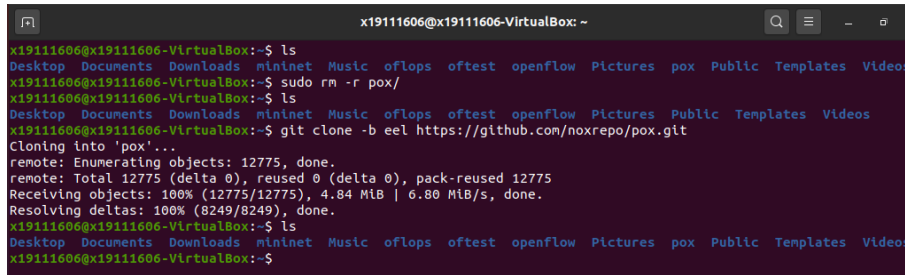
Press <enter> to keep the current choice[*], or type selection number: 1
update-alternatives: using /usr/bin/python2 to provide /usr/bin/python (python) in manual mode
x19111606@x19111606-VirtualBox:~$ python --version
Python 2.7.18
x19111606@x19111606-VirtualBox:~$
```

Figure 8: Switching from Python3 to Python2

6 POX Controller eel branch installation

Mininet by default installs the dart version of the POX controller which is the previous version of POX and the latest version is eel. In order to install latest POX controller, perform the given steps on your home directory.

- Remove existing pox directory using command: `sudo rm -r /pox`
- Install eel branch of POX controller using command:
`git clone -b eel https://github.com/noxrepo/pox.git`

A terminal window titled 'x19111606@x19111606-VirtualBox: ~' showing the execution of commands to remove the existing pox directory and clone the eel branch of the POX controller. The terminal output shows the directory listing, the removal of the pox directory, and the successful cloning of the eel branch from the GitHub repository.

```
x19111606@x19111606-VirtualBox:~$ ls
Desktop Documents Downloads mininet Music oflops oftest openflow Pictures pox Public Templates Videos
x19111606@x19111606-VirtualBox:~$ sudo rm -r pox/
x19111606@x19111606-VirtualBox:~$ ls
Desktop Documents Downloads mininet Music oflops oftest openflow Pictures Public Templates Videos
x19111606@x19111606-VirtualBox:~$ git clone -b eel https://github.com/noxrepo/pox.git
Cloning into 'pox'...
remote: Enumerating objects: 12775, done.
remote: Total 12775 (delta 0), reused 0 (delta 0), pack-reused 12775
Receiving objects: 100% (12775/12775), 4.84 MiB | 6.80 MiB/s, done.
Resolving deltas: 100% (8249/8249), done.
x19111606@x19111606-VirtualBox:~$ ls
Desktop Documents Downloads mininet Music oflops oftest openflow Pictures pox Public Templates Videos
x19111606@x19111606-VirtualBox:~$
```

Figure 9: Installation of POX eel branch

7 Fat Tree Topology Creation

Python programming language is used for creation of fat-tree topology. The python code is available in the file FatTreeTopo.py. This network topology has 20 OpenFlow vSwitches and 16 hosts. The topology built has value of $k=4$, where each switch within the topology will have four ports.

```
#!/usr/bin/env python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import Link, Intf, TCLink
from mininet.topo import Topo
from mininet.util import dumpNodeConnections
import logging
import os

logging.basicConfig(level=logging.DEBUG)
logger = logging.getLogger( __name__ )

#Creating fat tree topology with k=4 #

class ElephantMiceF(Topo):
    logger.debug("Class HugeTopo")
    CoreSwitchList = []
    AggSwitchList = []
    EdgeSwitchList = []
    HostList = []
    iNUMBER = 0
    def __init__(self):
        logger.debug("Class HugeTopo init")
        iNUMBER = 4

        self.iNUMBER = iNUMBER
        self.iCoreLayerSwitch = iNUMBER
        self.iAggLayerSwitch = iNUMBER * 2
        self.iEdgeLayerSwitch = iNUMBER * 2
        self.iHost = self.iEdgeLayerSwitch * 2

    #Initialize Topo
    Topo.__init__(self)

    def createTopo(self):
        logger.debug("Start creating Core Layer Swiches")
        self.createCoreLayerSwitch(self.iCoreLayerSwitch)
        logger.debug("Start creating Aggregation Layer Swiches ")
        self.createAggLayerSwitch(self.iAggLayerSwitch)
        logger.debug("Start creating Edge Layer Swiches ")
        self.createEdgeLayerSwitch(self.iEdgeLayerSwitch)
        logger.debug("Start creating Hosts")
        self.createHost(self.iHost)
```

Figure 10: Defining the topology

This section of the code helps in creating four core switches which are indicated from 1001 until 1004. The eight aggregation switches are named from 2002 until 2008. Followed by eight edge switches named as s3001 until s3008. Host connecting the edge layer switches are indicated as 4001 until 4016.

```
#Creating 4 Core switches,8 Aggregation switches,8 Edge switchs and 16 Hosts#

def createCoreLayerSwitch(self, NUMBER):
    logger.debug("Creating Core Layer Switches")
    for x in range(1, NUMBER+1):
        PREFIX = "100"
        if x >= int(10):
            PREFIX = "10"
        self.CoreSwitchList.append(self.addSwitch(PREFIX + str(x)))

def createAggLayerSwitch(self, NUMBER):
    logger.debug("Creating Aggregation Layer Switches")
    for x in range(1, NUMBER+1):
        PREFIX = "200"
        if x >= int(10):
            PREFIX = "20"
        self.AggSwitchList.append(self.addSwitch(PREFIX + str(x)))

def createEdgeLayerSwitch(self, NUMBER):
    logger.debug("Creating Edge Layer Switches")
    for x in range(1, NUMBER+1):
        PREFIX = "s300"
        if x >= int(10):
            PREFIX = "s30"
        self.EdgeSwitchList.append(self.addSwitch(PREFIX + str(x)))

def createHost(self, NUMBER):
    logger.debug("Creating Hosts")
    for x in range(1, NUMBER+1):
        PREFIX = "400"
        if x >= int(10):
            PREFIX = "40"
        self.HostList.append(self.addHost(PREFIX + str(x)))
```

Figure 11: Creating of 3-tier fat tree topology

This section of the code assigns links from Core Layer to Aggregation Layer switches. It then connects Aggregation Layer to Edge layer switches using links. Lastly, links are configured to connect Edge Layer switches with hosts. A bandwidth=100 is allocated to links connection Core Layer switches to Aggregation Layer switches while a bandwidth=50 is allocated to links connection Aggregation Layer switches to Edge Layer switches.

```
def createLink(self):
    logger.debug("Create Core to Agg")
    self.addLink(self.CoreSwitchList[0], self.AggSwitchList[0], port1=1, port2=1, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[0], self.AggSwitchList[2], port1=2, port2=1, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[0], self.AggSwitchList[4], port1=3, port2=1, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[0], self.AggSwitchList[6], port1=4, port2=1, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[1], self.AggSwitchList[0], port1=1, port2=2, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[1], self.AggSwitchList[2], port1=2, port2=2, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[1], self.AggSwitchList[4], port1=3, port2=2, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[1], self.AggSwitchList[6], port1=4, port2=2, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[2], self.AggSwitchList[1], port1=1, port2=1, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[2], self.AggSwitchList[3], port1=2, port2=1, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[2], self.AggSwitchList[5], port1=3, port2=1, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[2], self.AggSwitchList[7], port1=4, port2=1, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[3], self.AggSwitchList[1], port1=1, port2=2, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[3], self.AggSwitchList[3], port1=2, port2=2, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[3], self.AggSwitchList[5], port1=3, port2=2, bw=100, delay='1ms')
    self.addLink(self.CoreSwitchList[3], self.AggSwitchList[7], port1=4, port2=2, bw=100, delay='1ms')

    logger.debug("Create Agg to Edge")
    self.addLink(self.AggSwitchList[0], self.EdgeSwitchList[0], port1=3, port2=1, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[0], self.EdgeSwitchList[1], port1=4, port2=2, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[1], self.EdgeSwitchList[1], port1=3, port2=1, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[1], self.EdgeSwitchList[0], port1=4, port2=2, bw=50, delay='1ms')

    self.addLink(self.AggSwitchList[2], self.EdgeSwitchList[2], port1=3, port2=1, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[2], self.EdgeSwitchList[3], port1=4, port2=2, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[3], self.EdgeSwitchList[3], port1=3, port2=1, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[3], self.EdgeSwitchList[2], port1=4, port2=2, bw=50, delay='1ms')

    self.addLink(self.AggSwitchList[4], self.EdgeSwitchList[4], port1=3, port2=1, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[4], self.EdgeSwitchList[5], port1=4, port2=2, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[5], self.EdgeSwitchList[5], port1=3, port2=1, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[5], self.EdgeSwitchList[4], port1=4, port2=2, bw=50, delay='1ms')

    self.addLink(self.AggSwitchList[6], self.EdgeSwitchList[6], port1=3, port2=1, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[6], self.EdgeSwitchList[7], port1=4, port2=2, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[7], self.EdgeSwitchList[7], port1=3, port2=1, bw=50, delay='1ms')
    self.addLink(self.AggSwitchList[7], self.EdgeSwitchList[6], port1=4, port2=2, bw=50, delay='1ms')

    logger.debug("Create Edge to Host")
    for x in range(0, self.iEdgeLayerSwitch):
        ## limit = 2 * x + 1
        self.addLink(self.EdgeSwitchList[x], self.HostList[2 * x])
        self.addLink(self.EdgeSwitchList[x], self.HostList[2 * x + 1])
```

Figure 12: Adding network links

This section of the code shows integration on fat-tree topology with POX controller which runs on IP 127.0.0.1 with port number set to 6633.

```
def createTopo():
    logging.debug(" Create HugeTopo")
    topo = ElephantMiceF()
    topo.createTopo()
    topo.createLink()

    logging.debug("Start Mininet")
    #Configuration of Remote POX controller with default IP and PORT details#
    CONTROLLER_IP = "127.0.0.1"
    CONTROLLER_PORT = 6633
    net = Mininet(topo=topo, link=TCLink, controller=None, autoSetMacs=True)
    net.addController( 'controller',controller=RemoteController,ip=CONTROLLER_IP,port=CONTROLLER_PORT)
    net.start()

    logger.debug("dumpNode")
    enableSTP()
    dumpNodeConnections(net.hosts)

    pingTest(net)

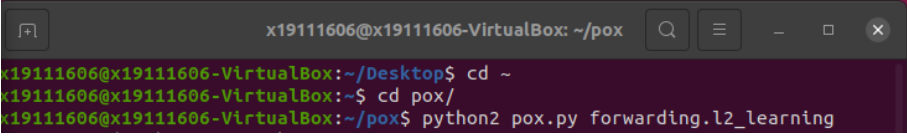
    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    if os.getuid() != 0:
        logger.debug("You are NOT root")
    elif os.getuid() == 0:
        createTopo()
```

Figure 13: POX configuration code

8 Running the Fat-tree topology to configure MTU for network interfaces

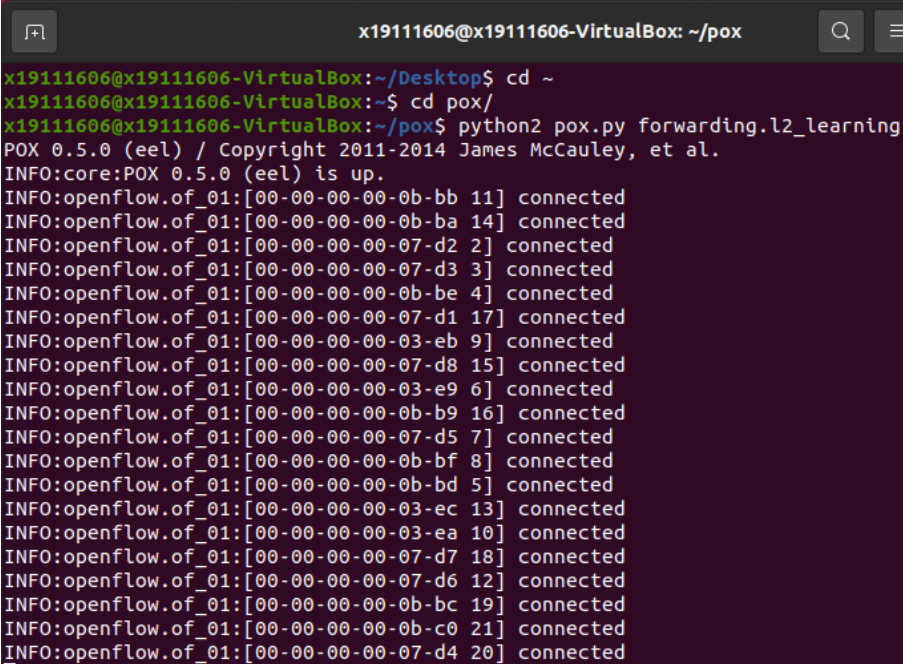
Now that we have created the fat tree topology using the mininet network emulator and configured it to connect to the remote POX controller, we will see the steps to run the topology along with the remote POX controller. Go to /pox directory present in the home directory using command: `cd /pox` and then run the command: `python2 pox.py forwarding.l2_learning`



```
x19111606@x19111606-VirtualBox: ~/pox
x19111606@x19111606-VirtualBox:~/Desktop$ cd ~
x19111606@x19111606-VirtualBox:~$ cd pox/
x19111606@x19111606-VirtualBox:~/pox$ python2 pox.py forwarding.l2_learning
```

Figure 14: Starting POX controller with forwarding.l2_learning routing algorithm

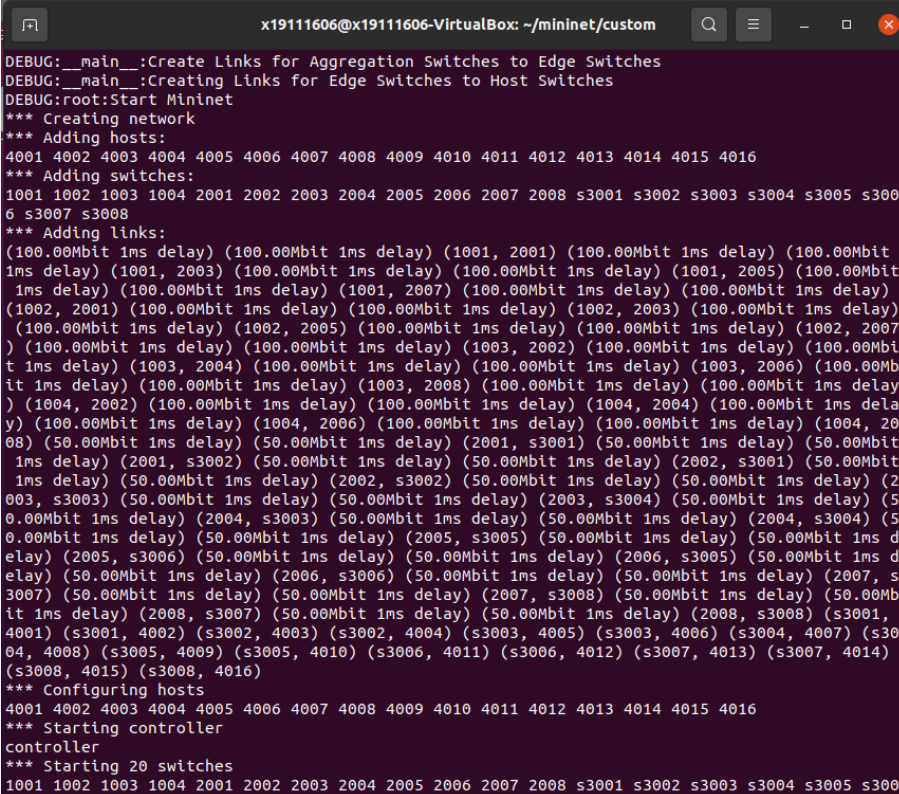
This command will start the remote POX controller and show the status as connected as eel POX controller.



```
x19111606@x19111606-VirtualBox: ~/pox
x19111606@x19111606-VirtualBox:~/Desktop$ cd ~
x19111606@x19111606-VirtualBox:~$ cd pox/
x19111606@x19111606-VirtualBox:~/pox$ python2 pox.py forwarding.l2_learning
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-0b-bb 11] connected
INFO:openflow.of_01:[00-00-00-00-0b-ba 14] connected
INFO:openflow.of_01:[00-00-00-00-07-d2 2] connected
INFO:openflow.of_01:[00-00-00-00-07-d3 3] connected
INFO:openflow.of_01:[00-00-00-00-0b-be 4] connected
INFO:openflow.of_01:[00-00-00-00-07-d1 17] connected
INFO:openflow.of_01:[00-00-00-00-03-eb 9] connected
INFO:openflow.of_01:[00-00-00-00-07-d8 15] connected
INFO:openflow.of_01:[00-00-00-00-03-e9 6] connected
INFO:openflow.of_01:[00-00-00-00-0b-b9 16] connected
INFO:openflow.of_01:[00-00-00-00-07-d5 7] connected
INFO:openflow.of_01:[00-00-00-00-0b-bf 8] connected
INFO:openflow.of_01:[00-00-00-00-0b-bd 5] connected
INFO:openflow.of_01:[00-00-00-00-03-ec 13] connected
INFO:openflow.of_01:[00-00-00-00-03-ea 10] connected
INFO:openflow.of_01:[00-00-00-00-07-d7 18] connected
INFO:openflow.of_01:[00-00-00-00-07-d6 12] connected
INFO:openflow.of_01:[00-00-00-00-0b-bc 19] connected
INFO:openflow.of_01:[00-00-00-00-0b-c0 21] connected
INFO:openflow.of_01:[00-00-00-00-07-d4 20] connected
```

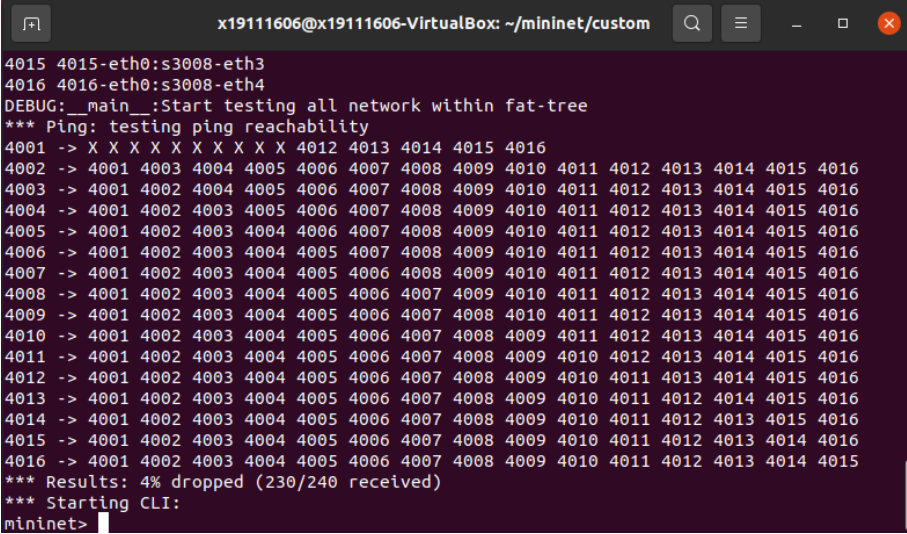
Figure 15: POX Controller connected

Now that the controller is up and running, run the fat-tree topology using the command: `sudo python FatTreeTopo.py` and enter the password to enable code execution. This topology code is stored in `/mininet/custom/` directory. The execution of the code successfully created the proposed OpenFlow vSwitches, hosts and links defined in the code file `FatTreeTopo.py` and then runs the ping test to check end to end host reachability.



```
x19111606@x19111606-VirtualBox: ~/mininet/custom
DEBUG:__main__:Create Links for Aggregation Switches to Edge Switches
DEBUG:__main__:Creating Links for Edge Switches to Host Switches
DEBUG:root:Start Mininet
*** Creating network
*** Adding hosts:
4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4012 4013 4014 4015 4016
*** Adding switches:
1001 1002 1003 1004 2001 2002 2003 2004 2005 2006 2007 2008 s3001 s3002 s3003 s3004 s3005 s300
6 s3007 s3008
*** Adding links:
(100.00Mbit 1ms delay) (100.00Mbit 1ms delay) (1001, 2001) (100.00Mbit 1ms delay) (100.00Mbit
1ms delay) (1001, 2003) (100.00Mbit 1ms delay) (100.00Mbit 1ms delay) (1001, 2005) (100.00Mbit
1ms delay) (100.00Mbit 1ms delay) (1001, 2007) (100.00Mbit 1ms delay) (100.00Mbit 1ms delay)
(1002, 2001) (100.00Mbit 1ms delay) (100.00Mbit 1ms delay) (1002, 2003) (100.00Mbit 1ms delay)
(100.00Mbit 1ms delay) (1002, 2005) (100.00Mbit 1ms delay) (100.00Mbit 1ms delay) (1002, 2007
) (100.00Mbit 1ms delay) (100.00Mbit 1ms delay) (1003, 2002) (100.00Mbit 1ms delay) (100.00Mbit
1ms delay) (1003, 2004) (100.00Mbit 1ms delay) (100.00Mbit 1ms delay) (1003, 2006) (100.00Mb
it 1ms delay) (100.00Mbit 1ms delay) (1003, 2008) (100.00Mbit 1ms delay) (100.00Mbit 1ms delay)
(1004, 2002) (100.00Mbit 1ms delay) (100.00Mbit 1ms delay) (1004, 2004) (100.00Mbit 1ms dela
y) (100.00Mbit 1ms delay) (1004, 2006) (100.00Mbit 1ms delay) (100.00Mbit 1ms delay) (1004, 20
08) (50.00Mbit 1ms delay) (50.00Mbit 1ms delay) (2001, s3001) (50.00Mbit 1ms delay) (50.00Mbit
1ms delay) (2001, s3002) (50.00Mbit 1ms delay) (50.00Mbit 1ms delay) (2002, s3001) (50.00Mbit
1ms delay) (50.00Mbit 1ms delay) (2002, s3002) (50.00Mbit 1ms delay) (50.00Mbit 1ms delay) (2
003, s3003) (50.00Mbit 1ms delay) (50.00Mbit 1ms delay) (2003, s3004) (50.00Mbit 1ms delay) (5
0.00Mbit 1ms delay) (2004, s3003) (50.00Mbit 1ms delay) (50.00Mbit 1ms delay) (2004, s3004) (5
0.00Mbit 1ms delay) (50.00Mbit 1ms delay) (2005, s3005) (50.00Mbit 1ms delay) (50.00Mbit 1ms d
elay) (2005, s3006) (50.00Mbit 1ms delay) (50.00Mbit 1ms delay) (2006, s3005) (50.00Mbit 1ms d
elay) (50.00Mbit 1ms delay) (2006, s3006) (50.00Mbit 1ms delay) (50.00Mbit 1ms delay) (2007, s
3007) (50.00Mbit 1ms delay) (50.00Mbit 1ms delay) (2007, s3008) (50.00Mbit 1ms delay) (50.00Mb
it 1ms delay) (2008, s3007) (50.00Mbit 1ms delay) (50.00Mbit 1ms delay) (2008, s3008) (s3001,
4001) (s3001, 4002) (s3002, 4003) (s3002, 4004) (s3003, 4005) (s3003, 4006) (s3004, 4007) (s30
04, 4008) (s3005, 4009) (s3005, 4010) (s3006, 4011) (s3006, 4012) (s3007, 4013) (s3007, 4014)
(s3008, 4015) (s3008, 4016)
*** Configuring hosts
4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4012 4013 4014 4015 4016
*** Starting controller
*** Starting 20 switches
1001 1002 1003 1004 2001 2002 2003 2004 2005 2006 2007 2008 s3001 s3002 s3003 s3004 s3005 s300
```

Figure 16: Creation of vSwitches, hosts and links



```
x19111606@x19111606-VirtualBox: ~/mininet/custom
4015 4015-eth0:s3008-eth3
4016 4016-eth0:s3008-eth4
DEBUG:__main__:Start testing all network within fat-tree
*** Ping: testing ping reachability
4001 -> X X X X X X X X X X 4012 4013 4014 4015 4016
4002 -> 4001 4003 4004 4005 4006 4007 4008 4009 4010 4011 4012 4013 4014 4015 4016
4003 -> 4001 4002 4004 4005 4006 4007 4008 4009 4010 4011 4012 4013 4014 4015 4016
4004 -> 4001 4002 4003 4005 4006 4007 4008 4009 4010 4011 4012 4013 4014 4015 4016
4005 -> 4001 4002 4003 4004 4006 4007 4008 4009 4010 4011 4012 4013 4014 4015 4016
4006 -> 4001 4002 4003 4004 4005 4007 4008 4009 4010 4011 4012 4013 4014 4015 4016
4007 -> 4001 4002 4003 4004 4005 4006 4008 4009 4010 4011 4012 4013 4014 4015 4016
4008 -> 4001 4002 4003 4004 4005 4006 4007 4009 4010 4011 4012 4013 4014 4015 4016
4009 -> 4001 4002 4003 4004 4005 4006 4007 4008 4010 4011 4012 4013 4014 4015 4016
4010 -> 4001 4002 4003 4004 4005 4006 4007 4008 4009 4011 4012 4013 4014 4015 4016
4011 -> 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4012 4013 4014 4015 4016
4012 -> 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4013 4014 4015 4016
4013 -> 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4012 4014 4015 4016
4014 -> 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4012 4013 4015 4016
4015 -> 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4012 4013 4014 4016
4016 -> 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4012 4013 4014 4015
*** Results: 4% dropped (230/240 received)
*** Starting CLI:
mininet>
```

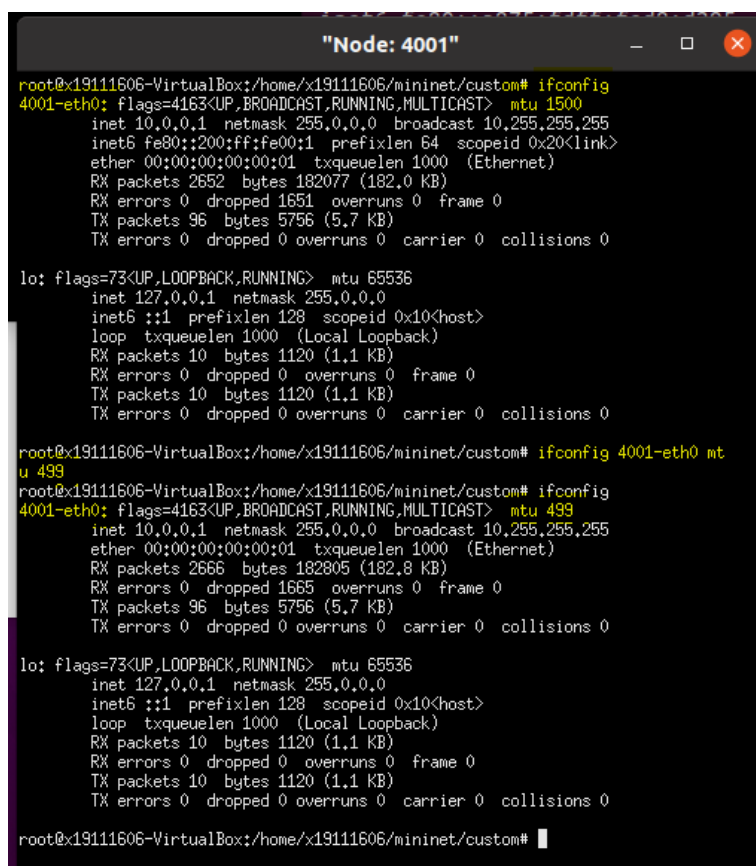
Figure 17: host reachability checked using ping utility

9 Modifying MTU value to set the threshold for proposed system

9.1 Modifying MTU value for Hosts in fat tree topology

After the execution of fat tree topology code, we enter the mininet terminal which allows us to access and modify the network components. In order to configure the threshold for the proposed system for detection of elephant flows, we modify the MTU value for all the network interfaces of vSwitches, Controller as well as the hosts to MTU=499. This step is performed only on one virtual machine. The other virtual machine is having unmodified MTU=1500. Given below are the steps to modify the MTU for host 4001 to 4016:

- on the 'mininet terminal type command: xterm 4001
- this will launch the CLI for host 4001
- type command: ifconfig ...this will display the network interface name of the host 4001.
- type command: ifconfig 4001-eth0 mtu 499 ... this command changes the MTU value of h4001 to 499 bytes
- Repeat these steps to manually change MTU value to 499 bytes for all the 16 hosts.



```
root@x19111606-VirtualBox:/home/x19111606/mininet/custom# ifconfig
4001-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::200:ff:fe00:1 prefixlen 64 scopeid 0x20<link>
    ether 00:00:00:00:00:01 txqueuelen 1000 (Ethernet)
    RX packets 2652 bytes 182077 (182.0 KB)
    RX errors 0 dropped 1651 overruns 0 frame 0
    TX packets 96 bytes 5756 (5.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 10 bytes 1120 (1.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 10 bytes 1120 (1.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@x19111606-VirtualBox:/home/x19111606/mininet/custom# ifconfig 4001-eth0 mt
u 499
root@x19111606-VirtualBox:/home/x19111606/mininet/custom# ifconfig
4001-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 499
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    ether 00:00:00:00:00:01 txqueuelen 1000 (Ethernet)
    RX packets 2666 bytes 182805 (182.8 KB)
    RX errors 0 dropped 1665 overruns 0 frame 0
    TX packets 96 bytes 5756 (5.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 10 bytes 1120 (1.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 10 bytes 1120 (1.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@x19111606-VirtualBox:/home/x19111606/mininet/custom#
```

Figure 18: MTU value for 4001 modified from 1500 to threshold value 499

9.2 Modifying MTU value for vSwitches in fat tree topology

In the proposed fat tree topology each vSwitch has four network interfaces. In order to change the MTU value of vSwitches, perform the steps mentioned below:

- on the 'mininet terminal type command: xterm 1001
- this will launch the CLI for host 1001
- type command: ifconfig ...this will display the network interface name of the host vSwitch 1001.
- type command: sudo ifconfig 1001-eth1 mtu 499 ... this command changes the MTU value of h4001 to 499 bytes
- repeat this step for all the four interfaces of vSwitch 1001
- Repeat these steps to manually change MTU value to 499 bytes for all the 20 hosts including the controller.

```
root@x19111606-VirtualBox:/home/x19111606/mininet/custom# sudo ifconfig 1001-eth1 mtu 499
root@x19111606-VirtualBox:/home/x19111606/mininet/custom# sudo ifconfig 1001-eth2 mtu 499
root@x19111606-VirtualBox:/home/x19111606/mininet/custom# sudo ifconfig 1001-eth3 mtu 499
root@x19111606-VirtualBox:/home/x19111606/mininet/custom# sudo ifconfig 1001-eth4 mtu 499
root@x19111606-VirtualBox:/home/x19111606/mininet/custom# ifconfig
1001-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 499
    ether ee:19:bc:b0:13:90 txqueuelen 1000 (Ethernet)
    RX packets 260 bytes 34540 (34.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4156 bytes 254671 (254.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

1001-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 499
    ether 86:93:b3:b7:aa:f0 txqueuelen 1000 (Ethernet)
    RX packets 234 bytes 30049 (30.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4168 bytes 257216 (257.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

1001-eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 499
    ether 8e:12:05:61:a7:26 txqueuelen 1000 (Ethernet)
    RX packets 301 bytes 38510 (38.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4148 bytes 255032 (255.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

1001-eth4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 499
    ether 8a:86:e4:ec:7c:11 txqueuelen 1000 (Ethernet)
    RX packets 4120 bytes 252004 (252.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 322 bytes 42023 (42.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 19: MTU value for vSwitch 1001 modified from 1500 to threshold value 499

10 Project Execution

After successful modification of MTU value to a threshold value of 499 for all the network interfaces within the fat-tree topology. We will be running the evaluation test using the iperf3 utility tool for Intra-POD as well as Inter-POD scenario. The same test is run on the other VM who runs the fat-tree implementation using unmodified MTU value. Both the systems uses forwarding.l2_learning routing algorithm.

10.1 Commands to run Intra-POD evaluation

- Stay logged into Mininet terminal
- type command: `xterm 4001 4004`
- here host 4001 acts as a client and host 4004 acts as a server
- on the terminal of host 4004, type command:
`iperf3 -s -p 7575 -i 1 -verbose`
- on the terminal of host 4001, type command:
`iperf3 -c 10.0.0.4 -p 7575 -t 10 -i 1 -b 100m`

10.2 Commands to run Inter-POD evaluation

- Stay logged into Mininet terminal
- type command: `xterm 4001 4016`
- here host 4001 acts as a client and host 4016 acts as a server
- on the terminal of host 4016, type command:
`iperf3 -s -p 5757 -i 1 -verbose`
- on the terminal of host 4001, type command:
`iperf3 -c 10.0.0.16 -p 5757 -t 10 -i 1 -b 100m`

The same steps can be performed on the other VM except for the step where the MTU value of network interfaces for nodes within fat-tree topology is changed.