Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2025-26

Class:	**TE**	**Semester:**	**V**
Course Code:	**CSC502**	**Course Name:**	**WC**

Name of Student:	**Mitesh Doulat Pawar**
Roll No. :	**73**
Experiment No.:	**10**
Title of the Experiment:	**Simulation of software defined network using mininet.**
Date of Performance:	**30/09/25**
Date of Submission:	**07/10/25**

Evaluation

Performance Indicator	**Max. Marks**	**Marks Obtained**
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	**Exceed Expectations (EE)**	**Meet Expectations (ME)**	**Below Expectations (BE)**
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Ms. Kshitija Gharat

Signature :

Date:

Aim: To simulate a Software Defined Network (SDN) environment using Mininet and observe communication between hosts.

Objective:

To understand the concept of Software Defined Networking
To simulate a virtual network topology using Mininet
To configure and test connectivity between hosts using ping command
To integrate a controller (such as POX/OVS) for centralized control of the SDN

Requirement:
Ubuntu Linux (or VM with Ubuntu installed)
Mininet installed (mininet.org)
Open vSwitch (default in Mininet)
Python support for running Mininet scripts

Theory:
Software Defined Networking (SDN) is a networking paradigm that separates the control plane from the data plane. In SDN, a central controller manages the flow of traffic in the network, while switches and routers only forward packets based on rules defined by the controller.

Mininet is a popular network emulator that can create a realistic virtual network with hosts, switches, and controllers on a single machine. It allows testing of SDN applications quickly and efficiently.

Key components:

Host: Represents end devices in the network

Switch: Open vSwitch used for packet forwarding

Controller: Centralized controller (like POX, Ryu, ONOS) that manages the network

Link: Virtual connections between hosts, switches, and controllers

Procedure:

Step 1: Launch Mininet
Open a terminal in Ubuntu and run:
```
sudo mn --topo single,3 --mac --switch ovsk --controller remote
```

This command creates a simple topology with 1 switch and 3 hosts.

Step 2: Test connectivity
Use the command:
```
pingall
```
This sends ICMP packets between all hosts to verify connectivity.

Step 3: Start Mininet CLI
Run commands inside Mininet CLI:

```
h1 ping h2
h1 ping h3
```

Step 4: Create custom topology using Python
Create a Python script (topo.py):

```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import RemoteController
from mininet.cli import CLI
class MyTopo(Topo):
def build(self):
h1 = self.addHost('h1')
h2 = self.addHost('h2')
s1 = self.addSwitch('s1')
self.addLink(h1, s1)
self.addLink(h2, s1)
topo = MyTopo()
net = Mininet(topo=topo, controller=RemoteController)
net.start()
CLI(net)
net.stop()
```

Run the script using:

```
sudo python3 topo.py
```

Step 5: Attach a controller
Install and run POX controller:

```
git clone https://github.com/noxrepo/pox.git
cd pox
./pox.py forwarding.l2_learning
```

Step 6: Connect Mininet to POX controller
Run Mininet with remote controller option:

```
sudo mn --controller=remote,ip=127.0.0.1,port=6633
```

Output:
Pingall shows 100% packet delivery between hosts

Hosts communicate via switch controlled by the SDN controller

Routing and forwarding decisions are handled dynamically by the controller

Conclusion:
Simulation of SDN using Mininet demonstrates how networks can be virtualized and centrally managed using controllers. This experiment shows host-to-host connectivity and highlights the role of the controller in defining packet forwarding behavior.