Mininet

Name: Samyak Rajesh Shah
Email: ss4604@rit.edu

1. Introduction

This report describes the implementation of a multi-router, multi-LAN network using **Mininet** for **CSCI-651 Homework 5**.
The project demonstrates:

- Designing subnets for three LANs with specific host requirements
- Building a functional Layer-3 Mininet topology
- Assigning correct IP addresses and subnet masks
- Testing intra-LAN connectivity
- Configuring static routing to enable inter-LAN communication
- Using ping and traceroute to verify routing paths

The system was implemented and tested inside a **Mininet Virtual Machine**, using Python to generate the topology and Mininet CLI commands to configure routing.

2. Project Structure

File Name	Description
layer3_network_code.py	Mininet topology builder (routers, hosts, subnets, switches) — Task 2
docs/	Sphinx-generated documentation for the topology code
Mininet Report.pdf	This report (Task 1–3 results)
requirements.txt	Python dependencies for documentation
README.md	Instructions for setup, running, and testing
revisions.txt	Git commit history

3. Environment Setup

- **Mininet VM** running Ubuntu Linux
- **Python Version:** 3.x inside the VM (Mininet's preinstalled environment)

- **Required Mininet Components:**
    - Mininet
    - OVSController
    - Linux namespaces for hosts/routers

Running the Topology

Inside the Mininet VM:

sudo python3 layer3_network_code.py

This launches the network and drops into the Mininet CLI for testing.

4. Subnet Design (Task 1)

The available address space is **20.10.172.0 – 20.10.172.255**.
Three LANs required:

LAN	Host Requirement	Subnet Chosen	Network Address	Lowest Usable	Highest Usable
LAN B	≥ 75 hosts	**20.10.172.0/25**	20.10.172.0	20.10.172.1	20.10.172.126
LAN A	≥ 50 hosts	**20.10.172.128/26**	20.10.172.128	20.10.172.129	20.10.172.190
LAN C	≥ 20 hosts	**20.10.172.192/27**	20.10.172.192	20.10.172.193	20.10.172.222

The 3 routers are also connected to a **core network**:
20.10.100.0/24

5. Layer-3 Network Construction (Task 2)

5.1 Description

The Python script builds:

- Three routers (ra, rb, rc)
- One core switch (s1)
- Three LAN switches (s2, s3, s4)
- Six hosts (hA1, hA2, hB1, hB2, hC1, hC2)
- IP forwarding enabled on all routers

- Correct subnet masks assigned to every interface

Each LAN connects to its router, and each router connects to the core switch.

5.2 LAN Connectivity Tests

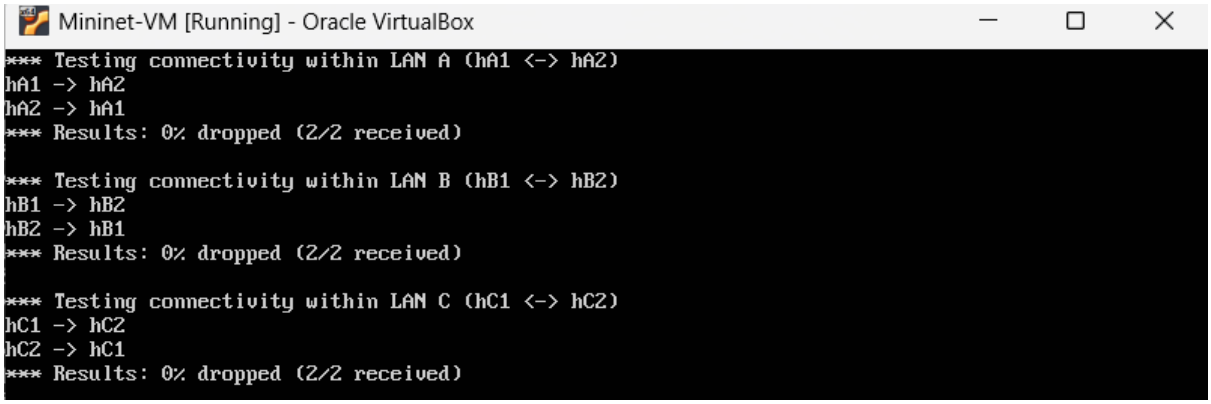
Using:

hA1 ping hA2

hB1 ping hB2

hC1 ping hC2

All LANs showed:



5.3 Cross-LAN Connectivity (Before Routing)

Using:

Pingall

Results:

- Only intra-LAN pairs were reachable
- All inter-LAN pairs failed
- Approximately **75% dropped**, which is expected before routing tables are added

This completes Task 2 requirements.

6. Static Routing Configuration (Task 3)

Static routes were added on:

- All three routers (ra, rb, rc)
- All six hosts (hA1–hC2)

These routes connect the LANs via the core network.

6.1 Router Routes

Example (Router A):

ra route add -net 20.10.172.0 netmask 255.255.255.128 gw 20.10.100.2

ra route add -net 20.10.172.192 netmask 255.255.255.224 gw 20.10.100.3

All routers received symmetric routes.

6.2 Host Routes

Example (LAN C):

hC2 route add -net 20.10.172.0 netmask 255.255.255.128 gw 20.10.172.193

hC2 route add -net 20.10.172.128 netmask 255.255.255.192 gw 20.10.172.193

All hosts now know how to reach all remote subnets.

```
mininet> ra route add -net 20.10.172.0 netmask 255.255.255.128 gw 20.10.100.2
mininet> ra route add -net 20.10.172.192 netmask 255.255.255.224 gw 20.10.100.3
mininet> rb route add -net 20.10.172.128 netmask 255.255.255.192 gw 20.10.100.1
mininet> rb route add -net 20.10.172.192 netmask 255.255.255.224 gw 20.10.100.3
mininet> rc route add -net 20.10.172.0 netmask 255.255.255.128 gw 20.10.100.2
mininet> rc route add -net 20.10.172.128 netmask 255.255.255.192 gw 20.10.100.1
mininet> hA1 route add -net 20.10.172.0 netmask 255.255.255.128 gw 20.10.172.129
mininet> hA1 route add -net 20.10.172.192 netmask 255.255.255.224 gw 20.10.172.129
mininet> hA2 route add -net 20.10.172.0 netmask 255.255.255.128 gw 20.10.172.129
mininet> hA2 route add -net 20.10.172.192 netmask 255.255.255.224 gw 20.10.172.129
mininet> hB1 route add -net 20.10.172.128 netmask 255.255.255.192 gw 20.10.172.1
mininet> hB1 route add -net 20.10.172.192 netmask 255.255.255.224 gw 20.10.172.1
mininet> hB2 route add -net 20.10.172.128 netmask 255.255.255.192 gw 20.10.172.1
mininet> hB2 route add -net 20.10.172.192 netmask 255.255.255.224 gw 20.10.172.1
mininet> hC1 route add -net 20.10.172.128 netmask 255.255.255.192 gw 20.10.172.193
mininet> hC1 route add -net 20.10.172.0 netmask 255.255.255.128 gw 20.10.172.193
mininet> hC2 route add -net 20.10.172.128 netmask 255.255.255.192 gw 20.10.172.193
mininet> hC2 route add -net 20.10.172.0 netmask 255.255.255.128 gw 20.10.172.193
```

7. Cross-LAN Communication Results (Task 3)

7.1 A → B Connectivity

Ping & Traceroute

```
mininet> hA1 ping -c 3 20.10.172.2
PING 20.10.172.2 (20.10.172.2) 56(84) bytes of data.
64 bytes from 20.10.172.2: icmp_seq=1 ttl=62 time=14.6 ms
64 bytes from 20.10.172.2: icmp_seq=2 ttl=62 time=0.751 ms
64 bytes from 20.10.172.2: icmp_seq=3 ttl=62 time=0.163 ms

--- 20.10.172.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2015ms
rtt min/avg/max/mdev = 0.163/5.181/14.631/6.685 ms
mininet> hA1 traceroute 20.10.172.2
traceroute to 20.10.172.2 (20.10.172.2), 30 hops max, 60 byte packets
 1  20.10.172.129 (20.10.172.129)  2.213 ms  12.937 ms  12.888 ms
 2  20.10.100.2 (20.10.100.2)  13.558 ms  13.300 ms  13.213 ms
 3  20.10.172.2 (20.10.172.2)  13.848 ms  13.856 ms  13.859 ms
mininet> _
```

7.2 C → A Connectivity

Ping & Traceroute

```
mininet> hC2 ping -c 3 20.10.172.130
PING 20.10.172.130 (20.10.172.130) 56(84) bytes of data.
64 bytes from 20.10.172.130: icmp_seq=1 ttl=62 time=2.61 ms
64 bytes from 20.10.172.130: icmp_seq=2 ttl=62 time=0.369 ms
64 bytes from 20.10.172.130: icmp_seq=3 ttl=62 time=0.152 ms

--- 20.10.172.130 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2020ms
rtt min/avg/max/mdev = 0.152/1.044/2.611/1.111 ms
mininet> hC2 traceroute 20.10.172.130
traceroute to 20.10.172.130 (20.10.172.130), 30 hops max, 60 byte packets
 1  20.10.172.193 (20.10.172.193)  6.927 ms  8.064 ms  8.203 ms
 2  20.10.100.1 (20.10.100.1)  10.024 ms  10.664 ms  10.849 ms
 3  20.10.172.130 (20.10.172.130)  12.189 ms  12.484 ms  12.542 ms
mininet>
```

These results confirm full network-wide connectivity and correct static routing.

8. Conclusion

This Mininet project successfully demonstrates:

- Subnetting with specific host constraints
- Building a complete multi-LAN, multi-router architecture
- IP addressing and forwarding on Linux routers
- Verification of intra-LAN isolation before routing
- Enabling full connectivity using static routes
- Using Mininet to emulate a real network topology

The implementation meets all requirements for Homework 5 and serves as a practical foundation for understanding routing, forwarding, and subnet design in virtualized environments.